Quantum Series

ENGINEERING

QUANTUM *Series*

Semester - 3 & 4 **Common to All Branches**

Python Programming

- Topic-wise coverage of entire syllabus in Question-Answer form.
- Short Questions (2 Marks)

Session 2023-24
Odd & Even Semester

Includes solution of following AKTU Question Papers
2019-20 • 2020-21 • 2021-22 • 2022-23

CONTENTS

BCC302 / BCC402 : Python Programming

1 UNIT

Introduction to Python

CONTENTS

PART-1
Python Variables.

Que 1.1. Define variable. Also discuss variable initialization.

Answer

Variables :

1. A variable holds a value that may change.
2. The process of writing the variable name is called declaring the variable.
3. In Python, variables do not need to be declared explicitly in order to reserve memory spaces as in other programming languages like C, Java, etc.
4. When we initialize the variable in Python, Python Interpreter automatically does the declaration process.

Initializing a variable :

1. The general format of assignment statement is as follows :
   Variable = Expression
2. The equal sign (=) is known as assignment operator.
3. An expression is any value, text or arithmetic expression where as variable is the name of variable.
4. The value of the expression will be stored in the variable.

Example of initializing a variable :

```
>>>year=2016
>>> name='Albert'
```

The two given statements reserve two memory spaces with variable names year and name. 2016 and Albert, are stored in these memory spaces.

Que 1.2. What do you understand by local and global variables in Python ? Give difference between them.

Answer

Local Variable :

A local variable is declared inside a specific function and can only be accessed by the function in which it is declared.

Example :

```
1. def f( ):
2.         # local variable
3.         S = "I love Python"
```



```
4.
5. # Driver code
6. f( )
```

In the above example, we have declared S inside the function. Hence it is local variable.

Global Variable :

A global variable is one that is "declared" outside of the function in a program and can, therefore, be accessed by any of the functions.

Example :

```
1. def f( ) :
2.         print("Inside function", S)
3. # Global scope
4. S = "I love Python"
5. f( )
6. print("Outside function", S)
```

In the above example, S is global variable.

Comparison :

S. No.	Basis	Global variable	Local variable
1.	Definition	Declared outside the functions.	Declared within the functions.
2.	Lifetime	They are created when the execution of the program begins and are lost when the program is ended.	They are created when the function starts its execution and are lost when the function ends.
3.	Data Sharing	Offers Data Sharing.	It doesn't offer Data Sharing.
4.	Scope	Can be access throughout the code.	Can access only inside the function.
5.	Parameters needed	Parameter passing is not necessary.	Parameter passing is necessary.
6.	Storage	A fixed location selected by the compiler.	They are kept on the stack.
7.	Value	Once the value changes it is reflected throughout the code.	Once changed the variable don't affect other functions of the program.

Que 1.3. **Write python program to swap two numbers without using Intermediate/Temporary variables. Prompt the user for input.**

AKTU 2021-22, Sem-4; Marks 10

Answer

```
# Prompt the user for input
num1 = int(input("Enter the first number: "))
num2 = int(input("Enter the second number: "))
# Swap the numbers without using a temporary variable
num1 = num1 + num2
num2 = num1 - num2
num1 = num1 - num2
# Print the swapped values
print(f"After swapping, the first number is: {num1}")
print(f"After swapping, the second number is: {num2}")
```

PART-2

Python Basic Operators.

Que 1.4. **Define the term operator.**

Answer

1. An operator is a symbol that represents an operation that may be performed on one or more operands.
2. Operators are used to modify the values of operands.
3. Operators that take one operand are called unary operators.
4. Operators that take two operands are called binary operators.
5. Based on functionality operators are categories into following seven types :
   - i. Arithmetic operators.
   - ii. Assignment operators.
   - iii. Bitwise operators.
   - iv. Comparison operators.
   - v. Identity operators.
   - vi. Logical operators.
   - vii. Membership operators.

Que 1.5. **Discuss various categories of operators in python. Find and explain stepwise solution of following expressions if a = 3, b = 5, c = 10.**

i. **a & b < 2 / 5 ** 2 + c^b**

ii. **b>>a**2 >>b**2^c**3**

AKTU 2022-23, Sem-4; Marks 10

Answer

Various categories of operators : Refer Q. 1.6 to Q. 1.10, Unit-1.

i. **a & b < 2/5**2 + c^b :**

Given values : a = 3, b = 5, c = 10

According to operators precedence we will evaluate the expression as follows :

Step 1 : 5**2 is 25

Now, the expression becomes a & b < 2/25 + c^b

Step 2 : 2/25 which is 0.08, now the expression becomes a & b < 0.08 + c^b

Step 3 : a & b will be bitwise AND of '3' and '5', which is '1'

Now, the expression becomes 1 < 0.08 + c^b

Step 4 : c^b is 10^5 which is 100000

Now, the expression becomes 1 < 0.08 + 100000

Step 5 : 0.08 + 100000 which is 100000.08

Step 6 : Finally 1 < 100000.08 which is true

ii. **b >> a**2 >> b**2^c**3 :**

According to operators precedence we will evaluate the expression as follows :

Step 1 : a**2 is 3**2 which is 9

So, the expression becomes b >> 9 >> b**2^c**3

Step 2 : b**2 is 5**2 which is 25

So, the expression becomes b >> 9 >> 25^c**3

Step 3 : c**3 is 10**3 which is 1000

So, the expression becomes b >> 9 >> 25^1000

Step 4 : b << 9 is a bitwise right shift of 5 by 9 position which results in 1.

So, the expression becomes 1 >> 25^1000

Step 5 : 1 >> 25 which is 1

Now, the expression becomes 1^1000

Step 6 : 1^1000 which is 1000.

Que 1.6. **Discuss arithmetic and comparison operator with example.**

Answer

Arithmetic operators : These operators are used to perform arithmetic operation such as addition, subtraction, multiplication and division.

Table 1.6.1. List of arithmetic operators.

Operator	Description	Example
+	Addition operator to add two operands.	10 + 20 = 30
–	Subtraction operator to subtract two operands.	10 – 20 = –10
×	Multiplication operator to multiply two operands.	10 × 20 = 200
/	Division operator to divide left hand by right hand operator.	5 / 2 = 2.5
**	Exponential operator to calculate power.	5 ** 2 = 25
%	Modulus operator to find remainder.	5 % 2 = 1
//	Floor division operator to find the quotient and remove the fractional part.	5 // 2 = 2

Comparison operators : These operators are used to compare values. It is also called relational operators. The result of these operator is always a Boolean value *i.e.*, true or false.

Table 1.6.2. List of comparison operators.

Operator	Description	Example
= =	Operator to check whether two operand are equal.	10 = = 20, false
! = or <>	Operator to check whether two operand are not equal.	10 ! = 20, true
>	Operator to check whether first operand is greater than second operand.	10 > 20, false
<	Operator to check whether first operand is smaller than second operand.	10 < 20, true
> =	Operator to check whether first operand is greater than or equal to second operand.	10 > = 20, false
< =	Operator to check whether first operand is smaller than or equal to second operand.	10 < = 20, true

Que 1.7. **Explain assignment operator with example.**

Answer

Assignment operators : This operator is used to store right side operands in the left side operand.

Table 1.7.1. List of assignment operators.

Operator	Description	Example
=	Store right side operand in left side operand.	$a = b + c$
+ =	Add right side operand to left side operand and store the result in left side operand	$a += b$ or $a = a + b$
– =	Subtract right side operand to left side operand and store the result in left side operand	$a -= b$ or $a = a - b$
* =	Multiply right side operand with left side operand and store the result in left side operand	$a *= b$ or $a = a * b$
/ =	Divide left side operand by right side operand and store the result in left side operand	$a /= b$ or $a = a / b$
% =	Find the modulus and store the remainder in left side operand	$a \%= b$ or $a = a \% b$
* * =	Find the exponential and store the result in left side operand	$a **= b$ or $a = a ** b$
// =	Find the floor division and store the result in left side operand	$a //= b$ or $a = a // b$

Que 1.8. **Define bitwise operator with example.**

Answer

Bitwise operators : These operators perform bit level operation on operands. Let us take two operand $x = 10$ and $y = 12$. In binary format this can be written as $x = 1010$ and $y = 1100$.

Table 1.8.1. List of bitwise operators.

Operator	Description	Example
& Bitwise AND	This operator performs AND operation between operands. Operator copies bit if it exists in both operands	x & y results 1000
| Bitwise OR	This operator performs OR operation between operands. Operator copies bit if it exists in either operand	x | y results 1 110
^ Bitwise XOR	This operator performs XOR operation between operands. Operator copies bit if it exists only in one operand.	x ^ y results 0110
~ bitwise inverse	This operator is a unary operator used to inverse the bits of operand.	~ x results 0101
<< left shift	This operator is used to shift the bits towards left	x << 2 results 101000
<<right shift	This operator is used to shift the bits towards right	x >> 2 results 0010

Que 1.9. **Discuss logical and identity operator with example.**

Answer

Logical operators : These operators are used to check two or more conditions. The resultant of this operator is always a Boolean value. Here x and y are two operands that store either true or false Boolean values.

Table 1.9.1. List of logical operators.

Operator	Description	Example
and logical AND	This operator perform AND operation between operands. When both operands are true, the resultant become true.	x and y results false
or logical OR	This operator perform OR operation between operands. When any operand is true, the resultant become true.	x or y results true
not logical NOT	This operator is used to reverse the operand state.	not x result false

Identity operators : These operator are used to check whether both operands are same or not. Suppose x stores a value 20 and y stores a value 40. Then x is y returns false and x not is y return true.

Table 1.9.2. List of identity operators.

Operator	Description	Example
is	Return true, if the operands are same. Return false, if the operands are not same.	x is y, results false
not is	Return false, if the operands are same. Return true, if the operands are not same.	x not is y, results true

Que 1.10. **Explain membership operator with example.**

Answer

Membership operators :

1. These operators are used to check an item or an element that is part of a string, a list or a tuple.
2. A membership operator reduces the effort of searching an element in the list.
3. Suppose x stores a value 20 and y is the list containing items 10, 20, 30, and 40. Then x is a part of the list y because the value 20 is in the list y.

Table 1.10.1. List of Membership operators.

Operator	Description	Example
in	Return true, if item is in list or in sequence. Return false, if item is not in list or in sequence.	x in y, results true
not in	Return false, if item is in list or in sequence. Return true, if item is not in list or in sequence.	x not in y, results true

Que 1.11. **What is short circuit evaluation ? What is printed by the following Python program ?**

```
a = 0
b = 2
c = 3
x = c or a
print (x)
```

AKTU 2020-21, Sem-3; Marks 10

Answer

1. When the evaluation of a logical expression stops because the overall value is already known, it is called short-circuit evaluation.
2. When Python detects that there is nothing to be gained by evaluating the rest of a logical expression, it stops its evaluation and does not do the computations in the rest of the logical expression.

   For Example :

   When Python is processing a logical expression such as '$x >= 2$ and $(x/y) > 2$', it evaluates the expression from left to right. Because of the definition of 'and', if x is less than 2, the expression '$x >= 2$' is False and so the whole expression is False regardless of whether $(x/y) > 2$ evaluates to True or False.

   Program :

```
a = 0
b = 2
c = 3
x = c or a
print(x)
```

   Output : 3

Que 1.12. What do you mean by operator precedence ?

Answer

1. When an expression has two or more operator, we need to identify the correct sequence to evaluate these operators. This is because result of the expression changes depending on the precedence.

   For example : Consider a mathematical expression :

   10 + 5 / 5

   When the given expression is evaluated left to right, the final answer becomes 3.
2. However, if the expression is evaluated right to left, the final answer becomes 11. This shows that changing the sequence in which the operators are evaluated in the given expression also changes the solution.
3. Precedence is the condition that specifies the importance of each operator relative to the other.

Table 1.12.1. Operator precedence from lower precedence to higher.

Operator	Description
NOT, OR AND	Logical operators
in , not in	Membership operator
is, not is	Identity operator
=, %=, /=, //=, –=, +=, *=, **==	Assignment operators.
<>, ==, !=	Equality comparison operator
<=, <, >, >=	Comparison operators
^, |	Bitwise XOR and OR operator
&	Bitwise AND operator
<<, >>	Bitwise left shift and right shift
+, –	Addition and subtraction
*, /, %, ??	Multiplication, Division, Modulus and floor division
**	Exponential operator

Que 1.13. Explain operator associativity.

Answer

Associativity :

1. Associativity decides the order in which the operators with same precedence are executed.
2. There are two types of associativity :
   a. **Left to right :** In left to right associativity, the operators of same precedence are executed from the left side first.
   b. **Right to left :** In right to left associativity, the operators of same precedence are executed from the right side first.
3. Most of the operators in Python have left to right associativity.
4. Left to right associative operators are multiplication, floor division, etc. and ** operator is right to left associative.
5. When two operators have the same precedence then operators are evaluated from left to right direction.

   For example :

```
>>> 3 * 4 // 6
2  # Output
```

```
>>> 3 * (4 / / 6)
0  # Output
>>> 3 * * 4 * * 2  # 3 ^ 16
43046721  # Output
>>> (3 * * 4) * * 2  # 81 ^ 2
6561  # Output
```

Que 1.14. **What do you mean by operator precedence and associativity ? Explain.** **AKTU 2021-22, Sem-4; Marks 10**

Answer

Operator precedence : Refer Q. 1.12, Page 1–10V, Unit-1.
Operator associativity : Refer Q. 1.13, Page 1–11V, Unit-1.

PART-3

Understanding Python Blocks.

Que 1.15. **Explain Python blocks. How do you define a block of code in Python ?**

Answer

1. A block in Python is a group of one or more statements that perform a specific task.
2. Blocks are defined by their indentation, which provides structure to the program.
3. Indentation in Python is important as it defines the scope of a block and helps to keep the code organized.
4. For example, consider the following code :

```
if x > 0 :
    print("x is positive")
x = x + 1
```

5. In this code, the block is defined by the indentation of the two statements under the if clause.
6. The block starts with the line print("x is positive") and ends with the line x = x + 1.
7. The statements within the block will only be executed if the condition specified in the if clause is met.

Que 1.16. **Differentiate between single-line block and multi-line block.**

Answer

S. No.		Single-Line block	Multi-line block
1.	Syntax	It contains a single statement and is typically written on the same line as the control flow statement.	It consists of multiple statements and is indicated by indentation, extending over several lines.
2.	Readability	Suitable for short and simple operations.	Preferred for complex or multi-step operations to enhance readability.
3.	Indentation	Minimal indentation is required.	Requires consistent indentation for all statements within the block.
4.	Maintainability	Can be harder to modify and debug when multiple statements are needed.	Easier to maintain and modify due to clearer structure.
5.	Clarity	Clear when the operation is concise and doesn't span multiple lines.	Offers a clearer visual separation of code blocks and steps.
6.	Error handling	Might be more challenging to identify and fix errors within a single line.	Easier to identify and correct errors due to the structured layout.
7.	Example	if condition: x = 5	if condition : x = 5 y = 10 In this example, both x = 5 and y = 10 are part of the same block because they are indented under the if statement.

PART-4
Python Data Types.

Que 1.17. What do you mean by data types ? Explain numeric and string data type with example.

Answer

Data types :

i. The data stored in the memory can be of many types. For example, a person's name is stored as an alphabetic value and his address is stored as an alphanumeric value.

ii. Python has six basic data types which are as follows :
   1. Numeric
   2. String
   3. List
   4. Tuple
   5. Dictionary
   6. Boolean

Numeric :

1. Numeric data can be broadly divided into integers and real numbers (i.e., fractional numbers). Integers can be positive or negative.
2. The real numbers or fractional numbers are called, floating point numbers in programming languages. Such floating point numbers contain a decimal and a fractional part.

For example :

```
>>> num1 = 2   # integer number
>>>num2 = 2.5   # real number (float)
>>>num1
2              # Output
>>>num2
2.5            # Output
>>>
```

String :

1. Single quotes or double quotes are used to represent strings.
2. A string in Python can be a series or a sequence of alphabets, numerals and special characters.

For example :

```
>>> sample_string = "Hello"   # store string value
>>> sample_string   # display string value
'Hello'            # Output
```

Que 1.18. Discuss list and tuple data types in detail.

Answer

List :

1. A list can contain the same type of items.
2. Alternatively, a list can also contain different types of items.
3. A list is an ordered and indexable sequence.
4. To declare a list in Python, we need to separate the items using commas and enclose them within square brackets ([ ]).
5. Operations such as concatenation, repetition and sub-list are done on list using plus (+), asterisk (*) and slicing ([]) operator.

For example :

```
>>>first = [1, "two", 3.0, "four"]   # 1st list
>>>second = ["five", 6]   # 2nd list
>>>first   # display 1st list
[1, 'two', 3.0, 'four']   # Output
```

Tuple :

1. A tuple is also used to store sequence of items.
2. Like a list, a tuple consists of items separated by commas.
3. Tuples are enclosed within parentheses rather than within square brackets.

For example :

```
>>>third = (7, "eight", 9, 10.0)
>>>third
(7, 'eight', 9, 10.0)   # Output
```

Que 1.19. Explain dictionary and Boolean data type.

Answer

Dictionary :

1. A Python dictionary is an unordered collection of key-value pairs.
2. When we have the large amount of data, the dictionary data type is used.
3. Keys and values can be of any type in a dictionary.

4. Items in dictionary are enclosed in the curly-braces {} and separated by the comma (,).
5. A colon (:) is used to separate key from value. A key inside the square bracket [ ] is used for accessing the dictionary items.

For example :

```
>>> dict1 = {1:"first line", "second" : 2}   # declare dictionary
>>> dict1[3] = "third line"   # add new item
>>> dict1 # display dictionary
{1 : 'first line', 'second' : 2, 3: 'third line'}   # Output
```

Boolean :

1. In a programming language, mostly data is stored in the form of alphanumeric but sometimes we need to store the data in the form of 'Yes' or 'No'.
2. In terms of programming language, Yes is similar to True and No is similar to False.
3. This True and False data is known as Boolean data and the data types which stores this Boolean data are known as Boolean data types.

For example :

```
>>> a = True
>>> type (a)
<type 'bool'>
```

Que 1.20. What do you mean by type conversion ?

Answer

1. The process of converting one data type into another data type is known as type conversion.
2. There are mainly two types of type conversion methods in Python :
   - **a. Implicit type conversion :**
     - i. When the data type conversion takes place during compilation or during the run time, then it called an implicit data type conversion.
     - ii. Python handles the implicit data type conversion, so we do not have to explicitly convert the data type into another data type.

For example :

```
a = 5
b = 5.5
sum = a + b
print (sum)
print (type (sum)) # type() is used to display the data type of a variable
```

Output :

```
10.5
<class 'float'>
```


- iii. In the given example, we have taken two variables of integer and float data types and added them.
- iv. Further, we have declared another variable named 'sum' and stored the result of the addition in it.
- v. When we checked the data type of the sum variable, we can see that the data type of the sum variable has been automatically converted into the float data type by the Python compiler. This is called implicit type conversion.

b. Explicit type conversion:

- i Explicit type conversion is also known as type casting.
- ii. Explicit type conversion takes place when the programmer clearly and explicitly defines the variables in the program.

For example :

```
# adding string and integer data types using explicit type conversion
a = 100
b = "200"
result1 = a + b
b = int(b)
result2 = a + b
print (result2)
```

Output :

```
Traceback (most recent call last):
File "", line 1, in
TypeError : unsupported operand type (s) for +: 'int' and 'str'
300
```

- iii. In the given example, the variable *a* is of the number data type and variable *b* is of the string data type.
- iv. When we try to add these two integers and store the value in a variable named result1, a TypeError occurs. So, in order to perform this operation, we have to use explicit type casting.
- v. We have converted the variable *b* into integer type and then added variable *a* and *b*. The sum is stored in the variable named result2, and when printed it displays 300 as output.

Que 1.21. **What do you mean by Boolean expression ?**

Answer

Boolean expression : A boolean expression may have only one of two values : True or False.

For example : In the given example comparison operator (==) is used which compares two operands and prints true if they are equal otherwise print false :

```
>>> 5 == 5
True  # Output
>>> 5 == 6
False  # Output
```

PART-5

Declaring and Using, Numeric Data Types : int, Float etc.

Que 1.22. **How do you declare numeric data type in python ?**

Answer

1. The numeric data type in Python represents the data that has a numeric value.
2. A numeric value can be an integer, a floating number, or even a complex number.

a. Integers :

i. This value is represented by int class.

ii. It contains positive or negative whole numbers (without fractions or decimals).

iii. In Python, there is no limit to how long an integer value can be.

For example :

```
a = 5
print("Type of a: ", type(a))
Output :
Type of a: <class "int'>
```

b. Float :

i. This value is represented by the float class.

ii. It is a real number with a floating-point representation.

iii. It is specified by a decimal point.

For example :

```
b = 5.0
print("\nType of b: ", type(b))
Output:
Type of b: <class 'float'>
```

c. Complex numbers :

i. Complex number is represented by a complex class.

ii. It is specified as (real part) + (imaginary part)j. For example - 2+3j

For example :

```
c = 2 + 4j
print("\nType of c: ", type(c))
Output :
Type of c: <class 'complex'>
```

Que 1.23. **What will be the output after the following statements ?**

```
x = 6
y = 3
print(x / y)
```

Answer

2.0

Que 1.24. **What will be the output after the following statements ?**

```
x = 8
y = 2
print(x // y)
```

Answer

4

Que 1.25. **What will be the output after the following statements ?**

```
x = 5
y = 4
print(x % y)
```

Answer

1

Que 1.26. What will be the output after the following statements ?

```
x = 30
y = 7
x %= y
print(x)
```

Answer

2

Que 1.27. What will be the output after the following statements ?

```
x = 8
y = 6
print(x != y)
```

Answer

True

Que 1.28. What will be the output after the following statements ?

```
x = 83
y = 57
print(x > y)
```

Answer

True

Que 1.29. What will be the output after the following statements ?

```
x = True
y = False
print(x and y)
```

Answer

False

Que 1.30. What will be the output after the following statements ?

```
x = True
y = False
print(x or y)
```

Answer

True

Que 1.31. What will be the output after the following statements ?

```
x = True
y = False
print(not x)
```

Answer

False

Que 1.32. What will be the output after the following statements ?

```
x = True
y = False
print(not y)
```

Answer

True

Que 1.33. What will be the output after the following statements ?

```
x = 20
y = 40
z = y if (y > x) else x
print(z)
```

Answer

40

Que 1.34. What will be the output after the following statements ?

```
x = 65
y = 53
z = y if (x % 2 == 0) else x
print(z)
```

Answer

65

Que 1.35. What will be the output after the following statements ?

```
x = 7 * (4 + 5)
print(x)
```

Answer

63

Que 1.36. **What will be the output after the following statements ?**

```
x = '24' + '16'
print(x)
```

Answer

2416

Que 1.37. **What will be the data type of *x* after the following statement if input entered is 18 ?**

```
x = input('Enter a number:')
```

Answer

String

Que 1.38. **What will be the data type of *y* after the following statements if input entered is 50 ?**

```
x = input('Enter a number:')
y = int(x)
```

Answer

Integer

Que 1.39. **What will be the data type of *y* after the following statements ?**

```
x = 48
y = str(x)
```

Answer

String

Que 1.40. **What will be the output after the following statements ?**

```
x = y = z = 8
print(y)
```

Answer

8

Que 1.41. **What will be the value of *x*, *y* and *z* after the following statement ?**


```
x, y, z = 3, 4, 5
```

Answer

x will have the value of 3, *y* will have the value 4 and *z* will have the value of 5.

Que 1.42. **In the order of precedence, which of the operation will be completed last in the following statement ?**

```
3 * 6 + 5 - 4 / 2
```

Answer

Subtraction

Que 1.43. **What will be the order of precedence of operations in the following statement ?**

```
10 * 4 - 1 + 8 / 5
```

Answer

Multiplication, Division, Addition, Subtraction

Que 1.44. **What will be the data type of *x* after the following statement if input entered is 64 ?**

```
x = float(input('Enter a number:'))
```

Answer

Float

Que 1.45. **What will be the output after the following statements ?**

```
a = 27 / 3 % 2 * 4**2
print(a)
```

Answer

16.0

Que 1.46. **What will be the output after the following statements ?**

```
a = 3 / 3 47 - 3*3
print(a)
```

Answer

20.0

☺☺☺

2 UNIT

Python Program Flow Control Conditional Blocks

CONTENTS

PART-1

if, else and else if.

Que 2.1. What are conditional statements in Python ?

Answer

1. Conditional statements help in making a decision based on certain conditions.
2. These conditions are specified by a set of conditional statements having Boolean expressions which are evaluated to true or false.
3. Conditional statements are also known as decision-making statements.
4. Python supports conditional execution using if-else statements.
5. In Python, we use different types of conditional statements :
   a. If statement
   b. If-else statement
   c. Nested-if statement
   d. Elif statement

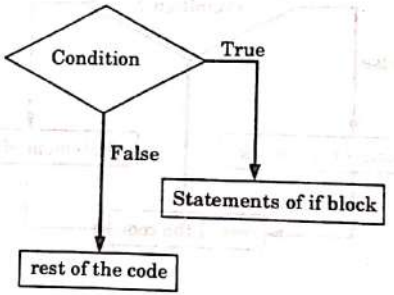
Que 2.2. Explain if statement with the help of an example.

Answer

1. An if statement consists of a Boolean expression followed by one or more statements.
2. With an if clause, a condition is provided; if the condition is true then the block of statement written in the if clause will be executed, otherwise not.
3. **Syntax :**

   If (Boolean expression) : Block of code #Set of statements to execute if the condition is true

4. **Flow chart :**



For example :

```
var = 100
if ( var == 100 ) : print "value of expression is 100"
print "Good bye !"
```

Output :

```
value of expression is 100
Good bye!
```

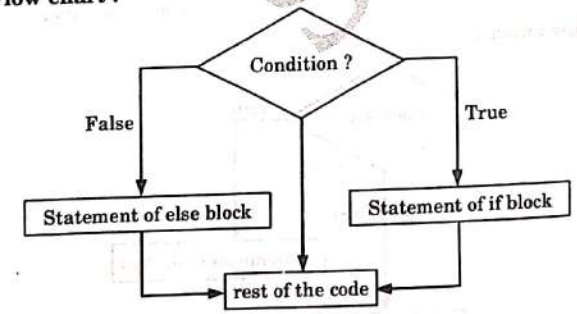
Que 2.3. What do you mean by if-else statement ? Explain with the help of example.

Answer

1. An if statement can be followed by an optional else statement, which executes when the Boolean expression is False.
2. The else condition is used when we have to judge one statement on the basis of other.
3. **Syntax :**

   If (Boolean expression): Block of code #Set of statements to execute if condition is true

   else : Block of code #Set of statements to execute if condition is false
4. **Working and execution :**
   a. The condition will be evaluated to a Boolean expression (true or false).
   b. If the condition is true then the statements or program present inside the if block will be executed
   c. If the condition is false then the statements or program present inside else block will be executed.
5. **Flow chart :**



For example :

```
num = 5
if (num > 10) :
    print ("Number is greater than 10")
else :
    print ("Number is less than 10")
print ("This statement will always be executed")
```

Output :

```
Number is less than 10.
```

Que 2.4. Discuss the Nested-if statement with the help of example.

Answer

1. Nested-if statements are nested inside other if statements. That is, a nested-if statement is the body of another if statement.
2. We use nested if statements when we need to check secondary conditions only if the fist condition executes as true.
3. **Syntax :**

```
if test expression 1 :
# executes when condition 1 is true
body of if statement
if test expression 2 :
# executes when condition 2 is true
Body of nested-if
else :
body of nested-if :
else :
body of if-else statement
```

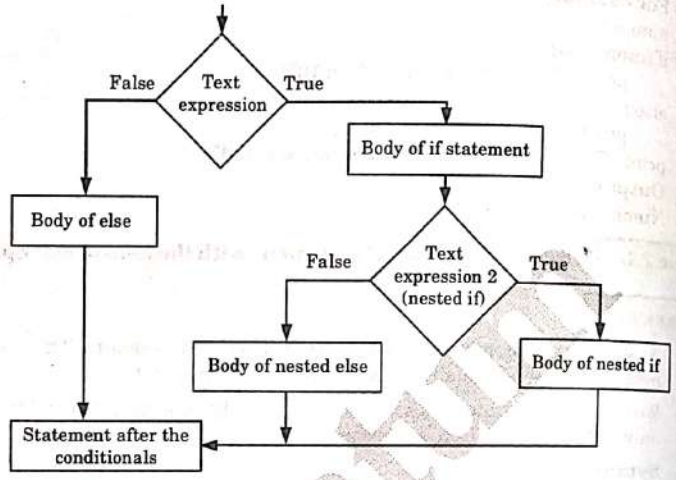
For example :

```
a = 20
if (a == 20) :
# First if statement
    if (a < 25) :
        print ("a is smaller than 25")
    else :
        print ("a is greater than 25")
else :
        print ("a is not equal to 20")
```

Output :

```
a is smaller than 25
```

4. Flow chart :



Que 2.5. Explain elif statement in Python.

OR

Explain all the conditional statement in Python using small code example. **AKTU 2019-20, Sem-3; Marks 10**

Answer

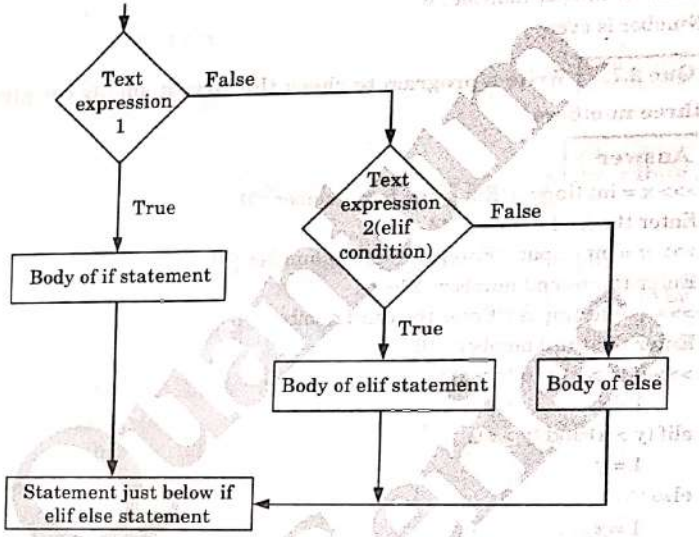
Different types of conditional statement are :

1. **If statement :** Refer Q. 2.2, Page 2–2V, Unit-2.
2. **If else statement :** Refer Q. 2.3, Page 2–3V, Unit-2.
3. **Nested-if statement :** Refer Q. 2.4, Page 2–4V, Unit-2.
4. **Elif statement :**
   a. Elif stands for else if in Python.
   b. We use elif statements when we need to check multiple conditions only if the given if condition executes as false.
   c. **Working and execution :**
      i. If the first if condition is true, the program will execute the body of the if statement. Otherwise, the program will go to the elif block (else if in Python) which basically checks for another if statement.
      ii. Again, if the condition is true, the program will execute the body of the elif statement, and if the condition is found to be false, the program will go to the next else block and execute the body of the else block.

   d. **Syntax :**

```
if test expression :
Body of if
elif test expression :
Body of elif
else :
Body of else
```

   e. **Flow chart :**



For example :

```
a = 50
if (a == 29) :
        print ("value of variable a is 20")
    elif (a == 30) :
        print ("value of variable a is 30")
    elif (a == 40) :
        print ("value of variable a is 40")
else :
    print ("value of variable a is greater than 40")
```

Output :

value of variable a is greater than 40

Que 2.6. Write a program to find whether a number is even or odd.

Answer

```
>>> number = int (input ("Enter an integer number :"))
>>> if (number % 2) == 0 :
    print "Number is even"
else :
    print "Number is odd"
Enter an integer number : 6
Number is even                    #Output
```

Que 2.7. **Write a program to check the largest among the given three numbers.**

Answer

```
>>> x = int (input ("Enter the first number :"))
Enter the first number : 14
>>> y = int (input ("Enter the second number :"))
Enter the second number : 21
>>> z = int (input ("Enter the third number :"))
Enter the third number : 10
>>> if (x > y) and (x > z) :
    1 = x
elif (y > x) and (y > z) :
    1 = y
else :
    1 = z
>>> print "The largest among the three is ", 1
The largest among the three is 21          #Output
```

Que 2.8. **Write a Python program to check if the input year is a leap year or not.** **AKTU 2021-22, Sem-4; Marks 10**

Answer

```
>>> year = int (input ("Enter year :"))
>>> if (year % 4) == 0 :
        if (year % 100) == 0 :
            if (year % 400) == 0 :
                print (year, 'is leap year')
            else :
                print (year, 'is not leap year')
        else :
            print (year, 'is leap year')
    else :
        print (year, 'is leap year')
```

Output :

```
Enter a year : 2016
2016 is leap year
```

Output :

```
Enter a year : 1985
1985 is not leap year
```


Que 2.9. **Write a Python program to display the Fibonacci sequence for *n* terms.**

OR

Write a program to produce Fibonacci series in Python. **AKTU 2021-22, Sem-3; Marks 10**

Answer

```
>>> number = int (input ("Enter the value for x (where x > 2) ?"))
# first the terms
>>> x1 = 0
>>> x2 = 1
>>> count = 2
# check if the number of terms is valid
>>> if numbers < = 0 :
    print ("Please enter positive integer")
elif numbers == 1 :
    print ("Fibonacci sequence is :")
    print (x1)
else :
    print ("Fibonacci sequence is :")
    print (x1, ",", x2)
    while count < numbers :
        xth = x1 + x2
        print (xth)
```

```
    # update values
    x1 = x2
    x2 = xth
    count + = 1
```

Output :

Enter the value for n (where n > 2) ? 10

Fibonacci sequence :

0, 1, 1, 2, 3, 5, 8, 13, 21, 34

Que 2.10. **What will be the output after the following statements ?**

```
x = 70
if x <= 30 or x >= 100:
    print('true')
elif x <= 50 and x == 50:
    print('not true')
elif x >= 150 or x <= 75:
    print('false')
else:
    print('not false')
```

Answer

false

Que 2.11. **What will be the output after the following statements ?**

```
x = 40
y = 25
if x + y >= 100:
    print('true')
elif x + y == 50:
    print('not true')
elif x + y <= 90:
    print('false')
else:
    print('not false')
```

Answer

false

Que 2.12. **What will be the output after the following statements ?**

```
x = 15
if x > 15:
    print(0)
elif x == 15:
```



```
    print(1)
else:
    print(2)
```

Answer

1

Que 2.13. **What will be the output after the following statements ?**

```
x = 5
if x > 15:
    print('yes')
elif x == 15:
    print('equal')
else:
    print('no')
```

Answer

no

Que 2.14. **What will be the output after the following statements ?**

```
x = 50
if x > 10 and x < 15:
    print('true')
elif x > 15 and x < 25:
    print('not true')
elif x > 25 and x < 35:
    print('false')
else:
    print('not false')
```

Answer

not false

Que 2.15. **What will be the output after the following statements ?**

```
x = 25
if x > 10 and x < 15:
    print('true')
elif x > 15 and x < 25:
    print('not true')
elif x > 25 and x < 35:
    print('false')
else:
    print('not false')
```

Answer

not false

Que 2.16. **What will be the output after the following statements?**

```
x = 15
if x > 10 and x <= 15:
    print('true')
elif x > 15 and x < 25:
    print('not true')
elif x > 25 and x < 35:
    print('false')
else:
    print('not false')
```

Answer

true

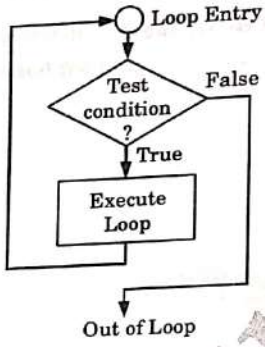
PART-2

Simple for Loops in Python, For Loop Using Ranges.

Que 2.17. **Define loop. Also, discuss the purpose and working of loops.**

Answer

1. A loop is a programming structure that repeats a sequence of instructions until a specific condition is met.
2. A loop statement allows us to execute a statement or group of statements multiple times.
3. Python programming language provides following types of loops to handle looping requirements:
   a. For
   b. While
   c. Nested
4. **Purpose :** The purpose of loops is to repeat the same, or similar, code a number of times. This number of times could be specified to a certain number, or the number of times could be dictated by a certain condition being met.
5. **Working :** Consider the flow chart for a loop execution :
   a. In the flow chart if the test condition is true, then the loop is executed, and if it is false then the execution breaks out of the loop.
   b. After the loop is successfully executed the execution again starts from the loop entry and again checks for the test condition, and this keeps on repeating until the condition is false.



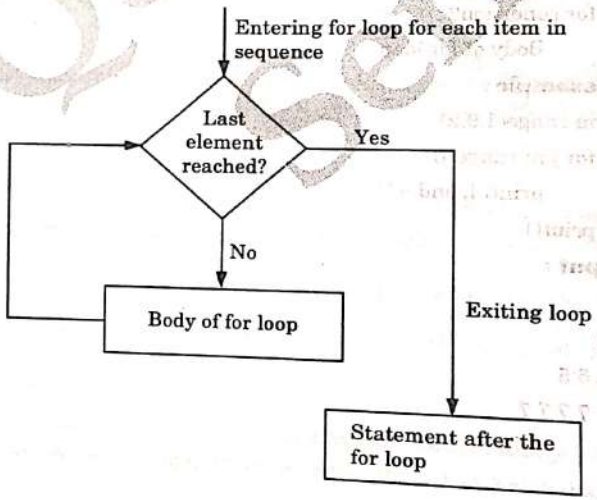
Que 2.18. **Explain for loop with the help of example.**

Answer

1. For loop in python is used to execute a block of statements or code several times until the given condition becomes false.
2. We use for loop when we know the number of times to iterate.
3. **Syntax :**

   for variable in sequence:

       Body of for loop
4. **Flow chart :**



For example :

```
i = 1
for i in range(1, 8) :
    print 2*i
```

Output :

2
4
6
8
10
12
14

Que 2.19. What is nested loop ? Explain.

Answer

1. Loop defined within another loop is known as nested loops.
2. Nested loops are the loops that are nested inside an existing loop, that is, nested loops are the body of another loop.
3. **Syntax :**

```
for condition1 :
    for condition2 :
        Body of for loop
```

4. **For example :**

```
for i in range(1,9,2) :
    for j in range(i) :
        print( i, end = ' ')
    print()
```

Output :

1
3 3 3
5 5 5 5 5
7 7 7 7 7 7 7

Que 2.20. Write a Python program to construct the following pattern, using a nested for loop.

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

AKTU 2021-22, Sem-3; Marks 10

Answer

Program :

```
n = 5
for i in range(n):
for j in range(i):
print("*",end="")
print()
for i in range(n):
for j in range(i,n):
print("*",end="")
print()
```

Que 2.21. Write and explain an algorithm through python code to generate prime numbers. **AKTU 2022-23, Sem-4; Marks 10**

Answer

A prime number is a natural number which is greater than 1 and has no positive divisor other than 1 and itself, such as 2, 3, 5, 7, 11, 13, and so on.

Algorithm :

Step 1 : Loop through all the elements in the given range.

Step 2 : Check for each number if it has any factor between 1 and itself.

Step 3 : If yes, then the number is not prime, and it will move to the next number.

Step 4 : If no, it is the prime number, and the program will print it and check for the next number.

Step 5 : The loop will break when it is reached to the upper value.

Code :

```
# First, we will take the input :
lower_value = int(input ("Please, Enter the Lowest Range Value: "))
upper_value = int(input ("Please, Enter the Upper Range Value: "))
```

```
4.  print ("The Prime Numbers in the range are: ")
5.  for number in range (lower_value, upper_value + 1) :
6.    if number > 1 :
7.      for i in range (2, number):
8.        if (number % i) == 0:
9.          break
10.     else :
11.       print (number)
```

Que 2.22. Explain range () function. How do you use a for loop with range () function in Python ?

Answer

1. The Python range() function returns a sequence of numbers, in a given range. The most common use of it is to iterate sequences on a sequence of numbers using Python loops.
2. **Syntax :** range(start, stop, step)

Parameter :

i. Start : [optional] start value of the sequence

ii. Stop : next value after the end value of the sequence

iii. Step : [optional] integer value, denoting the difference between any two numbers in the sequence

Return : Returns an object that represents a sequence of numbers

3. for loops repeat a block of code for all of the values in a range().
4. **Example :**

```
for x in range(6) :
    print(x)
```

5. The range() function defaults to 0 as a starting value, however it is possible to specify the starting value by adding a parameter: range(2, 6), which means values from 2 to 6 (but not including 6)

PART-3

String, List and Dictionaries.

Que 2.23. What do you mean by the term strings ?

Answer

1. Strings are created by enclosing various characters within quotes. Python does not distinguish between single quotes and double quotes.
2. Strings are of literal or scalar type. The Python interpreter treats them as a single value.
3. Strings, in Python, can be used as a single data type, or, alternatively, can be accessed in parts. This makes strings really useful and easier to handle in Python.

For example :

```
>>> var1 = 'Hello Python!'
>>> var2 = "Welcome to Python Programming!"
>>> print var1
Hello Python!  #  Output
>>> print var2
Welcome to Python Programming!  #  Output
```

Que 2.24. Write short note on list.

Answer

1. A list is also a series of values in Python.
2. In a list, all the values are of any type.
3. The values in a list are called elements or items.
4. A list is a collection of items or elements.
5. The sequence of data in a list is ordered and can be accessed by their positions, *i.e.*, indices.

For example :

```
>>> list = [1, 2, 3, 4]
>>> list [1]
2  #  Output
>>> list [3]
4  #  Output
```

Que 2.25. What is dictionary in Python ?

Answer

1. The Python dictionary is an unordered collection of items or elements.
2. All other compound data types in Python have only values as their elements or items whereas the dictionary has a key : value pair. Each value is associated with a key.

3. In dictionary, we have keys and they can be of any type.
4. Dictionary is said to be a mapping between some set of keys and values.
5. The mapping of a key and value is called as a key-value pair and together they are called one item or element.
6. A key and its value are separated by a colon (:) between them.
7. The items or elements in a dictionary are separated by commas and all the elements must be enclosed in curly braces.
8. A pair of curly braces with no values in between is known as an empty dictionary.
9. The values in a dictionary can be duplicated, but the keys in the dictionary are unique.

PART-4

Use of While Loop in Python, Loop Manipulation Using Pass, Continue, Break and Else.

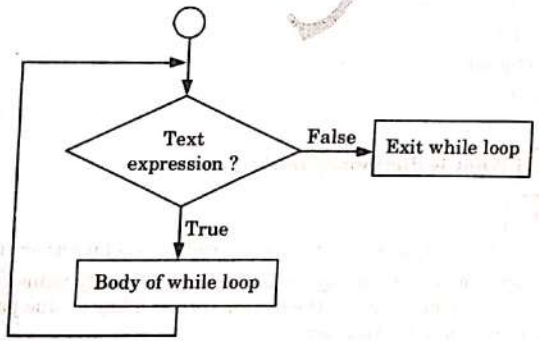
Que 2.26. Discuss while loop in brief.

Answer

1. While loop statements in Python are used to repeatedly execute a certain statement as long as the condition provided in the while loop statement is true.
2. While loops let the program control to iterate over a block of code.
3. **Syntax :**

```
while test_expression:
    body of while
```

4. **Flow chart :**



5. **Working :**
   a. The program first evaluates the while loop condition.
   b. If it is true, then the program enters the loop and executes the body of the while loop.
   c. It continues to execute the body of the while loop as long as the condition is true.
   d. When it is false, the program comes out of the loop and stops repeating the body of the while loop.

For example :

```
>>>count = 0
while (count < 9) :
    print 'The count is :', count
    count = count + 1
```

Que 2.27. Explain the following loops with a flow diagram, syntax, and suitable examples.

i. For

ii. While

AKTU 2022-23, Sem-3; Marks 10

Answer

i. **For loop :** Refer Q. 2.18, Page 2–12V, Unit-2.

ii. **While loop :** Refer Q. 2.26, Page 2–17V, Unit-2.

Que 2.28. What is break statement in Python ?

Answer

1. The break keyword terminates the loop and transfers the control to the end of the loop.
2. While loops, for loops can also be prematurely terminated using the break statement.
3. The break statement exits from the loop and transfers the execution from the loop to the statement that is immediately following the loop.

For example :

```
>>> count = 2
>>> while True :
    print count
    count = count + 2
    if count > = 12 :
        break  # breaks the loop
```

Output :

2
4
6
8
10

Que 2.29. Explain continue statement with example.

Answer

1. The continue statement causes execution to immediately continue at the start of the loop, it skips the execution of the remaining body part of the loop.
2. The continue keyword terminates the ongoing iteration and transfers the control to the top of the loop and the loop condition is evaluated again. If the condition is true, then the next iteration takes place.
3. Just as with while loops, the continue statement can also be used in Python for loops

For example :

```
>>> for i in range (1, 10) :
        if i % 2! = 0 :
            continue # if condition becomes true, it skips the print part
            print i
```

Output :

2
4
6
8

Que 2.30. Explain the purpose and working of loops. Discuss break and continue with example. Write a Python program to convert time from 12 hour to 24-hour format.

AKTU 2019-20, Sem-3; Marks 10

Answer

Purpose and working of loops : Refer Q. 2.17, Page 2–11V, Unit-2.

Break statement : Refer Q. 2.28, Page 2–18V, Unit-2.

Continue statement : Refer Q. 2.29, Page 2–19V, Unit-2.

Program :

```
# Function to convert the date format
def convert24(str1):
# Checking if last two elements of time # is AM and first two elements are
12
    if str1[– 2:] == "AM" and str1[:2] == "12":
    return "00" + str1[2:– 2]
# remove the AM
    elif str1[-2:] == "AM":
        return str1[:– 2]
# Checking if last two elements of time is PM and first two elements are 12
    elif str1[– 2:] == "PM" and str1[:2] == "12":
        return str1[:– 2]
    else:
# add 12 to hours and remove PM
        return str(int(str1[:2]) + 12) + str1[2:8]
# Driver Code
print(convert24("08:05:45 PM"))
```

Que 2.31. Write a program to demonstrate while loop with else.

Answer

```
>>> count = 0
>>> while count < 3 :
        print ("Inside the while loop")
        print (count)
        counter = count + 1
  else :
        print ("Inside the else statement")
```

Que 2.32. Differentiate between for and while loop.

Answer

Properties	For	While
Format	Initialization, condition checking, iteration statement are written at the top of the loop.	Only initialization and condition checking is done at top of the loop.
Use	The 'for' loop is used only when we already knew the number of iterations.	The 'while' loop is used only when the number of iteration are not exactly known.
Condition	If the condition is not given in 'for' loop, then loop iterates infinite times.	If the condition is not given in 'while' loop, it provides compilation error.
Initialization	In 'for' loop the initialization once done is never repeated.	In while loop if initialization is done during condition checking then initialization is done each time the loop iterate.

Que 2.33. **Explain the continue, break, and pass statements with a suitable example.** **AKTU 2022-23, Sem-3; Marks 10**

OR

Explain the use of break and continue with a suitable example. **AKTU 2021-22, Sem-4; Marks 10**

Answer

Continue : Refer Q. 2.29, Page 2–19V, Unit-2.

Break : Refer Q. 2.28, Page 2–18V, Unit-2.

Pass :

1. The pass statement is used as a placeholder for future code.
2. When the pass statement is executed, nothing happens, but you avoid getting an error when empty code is not allowed.
3. Empty code is not allowed in loops, function definitions, class definitions, or in if statements.

For example :

```
n = 10
# use pass inside if statement
if n > 10 :
    pass
print('Hello')
```


Que 2.34. **What will be the output after the following statements ?**

```
x, y = 0, 1
while y < 10:
    print(y, end=' ')
    x, y = y, x + y
```

Answer

1 1 2 3 5 8

Que 2.35. **What will be the output after the following statements ?**

```
x = 1
while x < 4:
    x += 1
    y = 1
    while y < 3:
        print(y, end=' ')
        y += 1
```

Answer

1 2 1 2 1 2

Que 2.36. **What will be the output after the following statements ?**

```
x, y = 2, 5
while y - x < 5:
    print(x*y, end=' ')
    x += 3
    y += 4
```

Answer

10 45

PART-5

Programming Using Python Conditional and Loop Blocks.

Que 2.37. **Write a program to check an input number is prime or not.** **AKTU 2021-22, Sem-4; Marks 10**

Answer

```
defPrimeChecker(a):
# Checking that given number is more than 1
if a > 1:
# Iterating over the given number with for loop
for j in range(2, int(a/2) + 1):
# If the given number is divisible or not
if (a % j) == 0:
print(a, "is not a prime number")
break
# Else it is a prime number
else:
print(a, "is a prime number")
# If the given number is 1
else:
print(a, "is not a prime number")
# Taking an input number from the user
a = int(input("Enter an input number:"))
# Printing result
PrimeChecker(a)
```

Output :

```
Enter an input number:17
17 is a prime number
```

Que 2.38. **Develop a program to calculate the reverse of any entered number.** **AKTU 2022-23, Sem-3; Marks 10**

Answer

```
# Ask for enter the number from the use
```



```
number = int(input("Enter the integer number : "))
# Initiate value to null
revs_number = 0
# reverse the integer number using the while loop
while (number > 0) :
remainder = number % 10
revs_number = (revs_number * 10) + remainder
number = number // 10
# Display the result
print("The reverse number is : {}".format(revs_number))
```

Output :

```
Enter the integer number : 12345
The reverse number is : 54321
```

Que 2.39. **Explain expression evaluation and float representation with example. Write a Python program for how to check if a given number is Fibonacci number.** **AKTU 2019-20, Sem-3; Marks 10**

Answer

Expression evaluation : Out of syllabus from session (2023-24).

Float representation : Out of syllabus from session (2023-24).

Program :

```
import math
# A utility function that returns true if x is perfect square
def isPerfectSquare(x):
    s = int(math.sqrt(x))
    return s*s == x
# Returns true if n is a Fibonacci number, else false
def isFibonacci(n):
    return isPerfectSquare(5*n*n + 4) or isPerfectSquare(5*n*n – 4)
# A utility function to test above functions
for i in range(1,6):
    if (isFibonacci(i) == True):
```

```
        print i,"is a Fibonacci Number"
    else:
        print i,"is a not Fibonacci Number"
```

Output :

1 is a Fibonacci Number
2 is a Fibonacci Number
3 is a Fibonacci Number
4 is a not Fibonacci Number
5 is a Fibonacci Number

☺☺☺

3 UNIT

Python Complex Data Types

CONTENTS

PART-1

Using String Data Types and String Operations.

Que 3.1. What is len( ) function ?

OR

Explain length of the string.

Answer

1. len ( ) is a built-in function in Python. When used with a string, len returns the length or the number of character in the string.
2. Blank symbol and special characters are considered in the length of the string.

For example :

```
>>> var = "Hello Python!"
>>> len(var)
13  # Output
```

Here, we took a string 'Hello Python!' and used the len function with it. The len function returned the value 13 because not only characters, but also the blank space and exclamation mark in our string will also be counted as elements.

Que 3.2. Discuss the concatenation and repeat operation in Python with example.

Answer

Concatenation :

1. Concatenation means joining two operands by linking them end-to-end.
2. In list concatenation, + operator concatenate two lists with each other and produce a third list.
3. For example, to concatenate two lists :

```
>>> x = [1, 2, 3]
>>> y = [4, 5, 6]
>>> z = x + y                    # concatenate two lists
>>> print z
[1, 2, 3, 4, 5, 6]
```

Repeat / replicate :

1. Lists can be replicated or repeated or repeatedly concatenated with the asterisk (*) operator.

2. For example,

```
>>> aList = [1, 2, 3]
>>> print aList*3                # list aList repeats three times
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

Here, the list aList multiplied by 3 and printed three times.

Que 3.3. What do you mean by string slices ?

Answer

1. A piece or subset of a string is known as slice.
2. Slice operator is applied to a string with the use of square braces ([]).
3. Operator $[n : m]$ will give a substring which consists of letters between n and m indices, including letter at index n but excluding that at m, *i.e.*, letter from n^{th} index to $(m-1)^{th}$ index.

For example :

```
>>> var = 'Hello Python'
>>> print var [0 : 4]
Hell # Output
```

In the above example, we can see that in the first case the slice is [0 : 4], which means that it will take the 0^{th} element and will extend to the 3^{rd} element, while excluding the 4^{th} element.

4. Similarly, operator $[n : m : s]$ will give a substring which consists of letters from n^{th} index to $(m-1)^{th}$ index, where s is called the step value, *i.e.*, after letter at n, that at $n+s$ will be included, then $n+2s$, $n+3s$, etc.

For example :

```
>>> alphabet = "abcdefghij"
>>> print alphabet [1 : 8 : 3]
beh # Output
```

In the given example, the slice is [1 : 8 : 3], which means it will take the element at 1^{st} index which is *b* and will extend till 7^{th} element. Since step is 3, it will print 1^{st} element, then 4^{th} element and then 7^{th} element *i.e.*, beh.

Que 3.4. Explain the term indexing in Python.

Answer

1. Indexing means referring to an element of an iterable by its position within the iterable.
2. **For example :** Create a list using a list comprehension :

```
my_list = [_for_in 'abcdefghi']
my_list
```

['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']

3. Now to retrieve an element of the list, we use the index operator ([ ]).
   For example :
   my_list[0]
   'a'
4. In Python, lists are "zero indexed", so [0] returns the zeroth (*i.e.*, the left-most) item in the list.
5. In the given example there are 9 elements in list ([0] to [8]) and if we want to access my_list[9] then it throws an indexError : list index out of range.
6. Python also allows to index from the end of the list using a negative number. In negative indices, we start counting from the right instead from the left.
7. Negative indices start from – 1.

Que 3.5. Demonstrate five different built in functions used in the string. Write a program to check whether a string is a palindrome or not. **AKTU 2022-23, Sem-3; Marks 10**

Answer

Python provides a variety of built-in string methods that allow you to perform various operations on strings. Here are some common ones :

S. No.	Method	Description
1.	str.capitalize()	Converts the first character of the string to uppercase.
2.	str.lower()	Converts all characters in the string to lowercase.
3.	str.upper()	Converts all characters in the string to uppercase.
4.	str.title()	Converts the first character of each word to uppercase.
5.	str.strip([chars])	Removes any leading (spaces at the beginning) and trailing (spaces at the end) characters.
6.	str.replace (old, new[, count])	Replaces occurrences of a substring with another substring. You can also specify a maximum number of replacements.
7.	str.isdigit()	True if all characters in the string are digits.

8.	str.isnumeric():	Returns True if all characters in the string are numeric.
9.	str.isalnum():	Returns True if all characters in the string are alphanumeric.
10.	str.islower():	Returns True if all characters in the string are lowercase.
11.	str.isupper():	Returns True if all characters in the string are uppercase.

Program :

```
def isPalindrome(s):
    return s == s[::-1]
s = "malayalam"
ans = isPalindrome(s)
if ans:
    print("Yes")
else:
    print("No")
```

Output :

Yes

PART-2

Defining List and List Slicing.

Que 3.6. Explain the copying method in list.

Answer

We can make a duplicate or copy of an existing list.

There are two ways to make copy of a list.

1. **Using [:] Operator :**

```
>>> list_original = [1, 2, 3, 4]
>>> list_copy = list_original [:]   #   Using [:] operator
>>> print list_copy
[1, 2, 3, 4]
```

2. **Using built-in function :** Python has a built-in copy function which can be used to make copy of an existing list. In order to use the copy function, first we have to import it.
   For example :

```
>>> from copy import copy   #   Import library
```

```
>>> list_original = [1, 2, 3, 4]
>>> list_copy = copy(list_original)  #  Copying list
>>> print list_copy
[1, 2, 3, 4]
```

Que 3.7. **How elements are deleted from a list ?**

Answer

1. Python provides many ways in which the elements in a list can be deleted.

Methods of deleting element from a list :

1. **pop operator :**
   a. If we know the index of the element that we want to delete, then we can use the pop operator.

   For example :

```
>>> list = [10, 20, 30, 40,]
>>> a = list.pop (2)
>>> print list
[10, 20, 40]  #  Output
```

   b. The pop operator deletes the element on the provided index and stores that element in a variable for further use.

2. **del operator :** The del operator deletes the value on the provided index, but it does not store the value for further use.

   For example :

```
>>> list = ['w', 'x', 'y', 'z',]
>>> del list (1)
>>> print list
['w', 'y', 'z']  #  Output
```

3. **Remove operator :**
   a. We use the remove operator if we know the item that we want to remove or delete from the list.

   For example :

```
>>> list = [10, 20, 30, 40,]
>>> list.remove (10)
>>> print list
[20, 30, 40]  #  Output
```

   b. In order to delete more than one value from a list, del operator with slicing is used.

   For example :

```
>>>list = [1, 2, 3, 4, 5, 6, 7, 8]
>>>del list [1 : 3]
>>>print list
[1, 4, 5, 6, 7, 8] # Output
```


Que 3.8. **Discuss built-in list operators in detail.**

Answer

Built-in list operators are as follows :

1. **Concatenation :** The concatenation operator is used to concatenate two lists. This is done by the + operator in Python.

   For example :

```
>>>list1 = [10, 20, 30, 40]
>>>list2 = [50, 60, 70]
>>>list3 = list1 + list2
>>>print list3
[10, 20, 30, 40, 50, 60, 70]  #  Output
```

2. **Repetition :** The repetition operator repeats the list for a given number of times. Repetition is performed by the * operator.

   For example :

```
>>>list1 = [1, 2, 3]
>>>list 1 * 4
[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]  # Output
```

3. **In operator :**
   a. The in operator tells the user whether the given string exists in the list or not.
   b. It gives a Boolean output *i.e.*, true or false.
   c. If the given input exists in the string, it given true as output, otherwise, false.

   For example :

```
>>>list = ['Hello', 'Python', 'Program']
>>> 'Hello' in list
True  #  Output
>>>'World' in list
False  #  Output
```

Que 3.9. **Write various built-in list methods. Explain any three with example.**

Answer

Python includes many built-in methods for use with list as shown in Table 3.9.1.

Table 3.8.1. List of built-in list methods.

S. No.	Method	Description
1.	cmp (list1, list2)	It compares the elements of both lists, list1 and list2.
2.	max (list)	It returns the item that has the maximum value in a list.
3.	min (list)	It returns the item that has the minimum value in a list.
4.	list (seq)	It converts a tuple into a list.
5.	list.append (item)	It adds the item to the end of the list.
6.	list.count (item)	It returns number of times the item occurs in the list.
7.	list.extend (seq)	It adds the elements of the sequence at the end of the list.
8.	list.remove (item)	It deletes the given item from the list.
9.	list.reverse ( )	It reverses the position (index number) of the items in the list.
10.	list.sort ([func])	It sorts the elements inside the list and uses compare function if provided.

1. **Append method :** This method can add a new element or item to an existing list.

   For example :

```
>>>list = [1, 2, 3, 4]
>>>list.append (0)
[1, 2, 3, 4, 0]                    # Output
```

2. **Extend method :** This method works like concatenation. It takes a list as an argument and adds it to the end of another list.

   For example :

```
>>>list1 = ['x', 'y', 'z']
>>>list2 = [1, 2, 3]
>>>list1.extend (list2)
>>>print list1
['x', 'y', 'z', 1, 2, 3]           # Output
```

3. **Sort method :** This method arranges the list in ascending order.

   For example :

```
>>>list = [4, 2, 5, 8, 1, 9]
```



```
>>>list.sort ( )
>>>print list
[1, 2, 4, 5, 8, 9]                 # Output
```

Que 3.10. **Discuss list data structure of python. Explain various inbuilt methods of list with suitable example of each.**

AKTU 2022-23, Sem-4; Marks 10

Answer

List data structure : Refer Q. 2.24, Page 2–16V, Unit-2.

Built-in list methods : Refer Q. 3.9, Page 3–7V, Unit-3.

Que 3.11. **Define the term list comprehension.**

OR

Explain the list comprehension with any suitable example.

AKTU 2022-23, Sem-3; Marks 10

Answer

1. List comprehension is used to create a new list from existing sequences.
2. It is a tool for transforming a given list into another list.
3. Using list comprehension, we can replace the loop with a single expression that produces the same result.
4. The syntax of list comprehension is based on set builder notation in mathematics.
5. Set builder notation is a notation is a mathematical notation for describing a set by stating the property that its members should satisfy. The syntax is

   [<expression> for <element> in <sequence> if <conditional>]

   The syntax is read as "Compute the expression for each element in the sequence, if the conditional is true".

   For example :

```
>>> List1 = [10, 20, 30, 40, 50]
>>> List1
    [10, 20, 30, 40, 50]
>>> for i in range (0, len(List1)) :
        List1 [i] = List1[i] + 10
>>> List1
[20, 30, 40, 50, 60]
```

Without list comprehension

```
>>> List1 = [10, 20, 30, 40, 50]
>>> List1= [x + 10 for x in List1]
>>> List1
[20, 30, 40, 50, 60]
```

Using list comprehension

5. In the given example, the output for both without list comprehension and using list comprehension is the same.

6. The use of list comprehension requires lesser code and also runs faster.
7. From above example we can say that list comprehension contains :
   a. An input sequence
   b. A variable referencing the input sequence
   c. An optional expression
   d. An output expression or output variable

Que 3.12. **Write a function lessthan (lst, *k*) to return list of numbers less than *k* from a list lst. The function must use list comprehension.**
Example :
lessthan ([1, – 2, 0, 5, – 3], 0) returns [– 2, – 3]

AKTU 2020-21, Sem-3; Marks 10

Answer

```
def lessthan (lst,k) :
        return [i for i in lst if i < k]
print(lessthan([1,–2,0,5,–3],0))
# output
[–2,–3]
```

Que 3.13. **Define list slicing and illustrate it with an example.**

Answer

List slicing :

1. List slicing in Python refers to the technique of extracting a portion of a list by specifying a range of indices.
2. It allows you to create a new list containing elements from the original list, starting from a specified index to an ending index (exclusive). This is achieved by using the colon (:) operator.
3. **Syntax :**

The format for list slicing is of Python List Slicing is as follows:

Lst[Initial : End : IndexJump]

If Lst is a list, then the above expression returns the portion of the list from index Initial to index End, at a step size IndexJump.

Example :

```
# Initialize list
List = [1, 2, 3, 4, 5, 6, 7, 8, 9]
# Show original list
print("Original List:\n", List)
print("\nSliced Lists: ")
# Display sliced list
print(List[3:9:2])
# Display sliced list
print(List[::2])
# Display sliced list
print(List[::])
```

4. Leaving any argument like Initial, End, or IndexJump blank will lead to the use of default values *i.e.*, 0 as Initial, length of the list as End, and 1 as IndexJump.

PART-3

Use of Tuple Data Type, String,List and Dictionary.

Que 3.14. **Define tuples. How are tuples created in Python ?**

Answer

1. Tuples are the sequence or series values of different types separated by commas (,).
2. Values in tuples can also be accessed by their index values, which are integers starting from 0.

For example :

The names of the months in a year can be defined in a tuple :

```
>>> months = ('January', 'February', 'March', 'April', 'May', 'June',
'July', 'August', 'September', 'October', 'November', 'December')
```

Creating tuples in Python :

1. To create a tuple, all the items or elements are placed inside parentheses separated by commas and assigned to a variable.
2. Tuples can have any number of different data items (that is, integer, float, string, list, etc.).

For examples :

1. **A tuple with integer data items :**

```
>>> tuple = (4, 2, 9, 1)
>>> print tuple
(4, 2, 9, 1) # Output
```

2. **A tuple with items of different data types :**

```
>>>tuple_mix = (2, 30, "Python", 5.8, "Program")
>>>print tuple_mix
(2, 30, 'Python', 5.8, 'Program')  #  Output
```

3. **Nested tuple :**

```
>>>nested_tuple = ("Python", [1, 4, 2], ["john", 3.9])
```

```
>>> print nested_tuple
('Python', [1, 4, 2], ['john', 3.9])  # Output
```

4. **Tuple can also be created without parenthesis :**

```
>>>tuple = 4.9, 6, 'house'
>>>print tuple
(4.9, 6, 'house')  # Output
```

Que 3.15. **Write a Python program to add an item in a tuple.**

AKTU 2021-22, Sem-3; Marks 10

Answer

Code :

```
t = (10, 40, 50, 70, 90)
print(t)
t = t + (20, )
print(t)
l = list(t)
l.append (3)
t = tuple(l)
print(t)
```

Que 3.16. **How are the values in a tuple accessed ?**

Answer

Accessing values in tuples :

1. In order to access the values in a tuple, it is necessary to use the inde number enclosed in square brackets along with the name of the tuple.

   For example : Using square brackets

```
>>>tup1 = ('Physics', 'Chemistry', 'Mathematics')
>>>tup2 = (10, 20, 30, 40, 50)
>>>print tup1 [1]
Chemistry  # Output
>>>print tup2 [4]
50  # Output
```

2. We can also use slicing in order to print the continuous values in a tuple

   For example :

```
>>>tup1 = ('Physics', 'Chemistry', 'Mathematics')
>>>tup2 = (10, 20, 30, 40, 50)
>>>tup2 [1 : 4]
(20, 30, 40)  # Output
```



```
>>>tup1[ : 1]
('Physics',)  # Output
```

Que 3.17. **Explain tuples as return values and variable-length arguments tuples with the help of example.**

Answer

Tuple as return values :

1. Tuples can also be returned by the function as return values.
2. Generally, the function returns only one value but by returning tuple, a function can return more than one value.

   For example : If we want to compute a division with two integers and want to know the quotient and the remainder, both the quotient and the remainder can be computed at the same time. Two values will be returned, *i.e.*, quotient and remainder, by using the tuple as the return value of the function.

```
>>>def div_mod(a, b) :  # defining function
    quotient = a/b
    remainder = a%b
    return quotient, remainder  # function returning two values
# function calling
>>>x = 10
>>>y = 3
>>>t = div_mod(x, y)
>>>print t
(3, 1)  # Output
>>>type(t)
<type 'tuple'>  # Output
```

Variable length argument tuples :

1. Variable number of arguments can also be passed to a function.
2. A variable name that is preceded by an asterisk (*) collects the arguments into a tuple.

   For example : In the given example, we have defined a function traverse with argument *t*, which means it can take any number of arguments and will print each of them one by one.

```
>>>def traverse (* t) :
...         i = 0
...     while i<len(t) :
...         print t[i]
...         i = i + 1
>>>traverse(1, 2, 3, 4, 5)
```

Que 3.18. **Explain the basic operation of tuples with example.**

Answer

Basic operations of tuples are :

1. **Concatenation :** The concatenation in tuples in used to concatenates two tuples. This is done by using + operator in Python.

   For example :

```
>>>t1 = (1, 2, 3, 4)
>>>t2 = (5, 6, 7, 8)
>>>t3 = tl + t2
>>>print t3
(1, 2, 3, 4, 5, 6, 7, 8)  #  Output
```

2. **Repetition :** The repetition operator repeats the tuples a given number of times. Repetition is performed by the * operator in Python.

   For example :

```
>>>tuple = ('ok',)
>>>tuple * 5
('ok', 'ok', 'ok', 'ok', 'ok')  # Output
```

3. **In operator :**
   i. The in operator also works on tuples. It tells user that the given element exists in the tuple or not.
   ii. It gives a Boolean output, that is, true or false.
   iii. If the given input exists in the tuple, it gives the true as output, otherwise false.

   For example :

```
>>>tuple = (10, 20, 30, 40)
>>>20 in tuple
True  #  Output
>>>50 in tuple
False  #  Output
```

4. **Interation :** It can be done in tuples using for loop. It helps in traversing the tuple.

   For example :

```
>>>tuple = (1, 2, 3, 4, 5, 6)
>>> for x in tuple :
... print x
```

Que 3.19. **How do we create a dictionary in Python ?**

Answer

1. Creating a dictionary is simple in Python.
2. The values in a dictionary can be of any data type, but the keys must be of immutable data types (such as string, number or tuple).

For example :

1. **Empty dictionary :**

```
>>> dict1 = {}
>>>print dict1
{}  #  Output
```

2. **Dictionary with integer keys :**

```
>>> dict1 = {1 : 'red', 2 : 'yellow', 3 : 'green'}
>>>print dict1
{1 : 'red', 2 : 'yellow', 3 : 'green'}  #  Output
```

3. **Dictionary with mixed keys :**

```
>>> dict1 = {'name' : 'Rahul', 3 : ['Hello', 2, 3]}
>>>print dict1
{3 : ['Hello', 2, 3], 'name' : 'Rahul'}  #  Output
```

Que 3.20. **Explain the operations performed on dictionary with example.**

Answer

Operations in dictionary :

1. **Traversing :**
   a. Traversing in dictionary is done on the basis of keys.
   b. For traversing, for loop is used, which iterates over the keys in the dictionary and prints the corresponding values using keys.

   For example :

```
>>>defprint_dict(d):
...     for c in d:
...         print c,d[c]
>>> dictl = {1 : 'a', 2 : 'b', 3 : 'c' 4 : 'd'}
>>> print_dict(dict1)
```

   This example prints the key: value pairs in the dictionary dict.

2. **Membership :**
   i. Using the membership operator (in and not in), we can test whether a key is in the dictionary or not.
   ii. In operator takes an input key and finds the key in the dictionary.
   iii. If the key is found, then it returns true, otherwise, false.

   For example :

```
>>>cubes = {1 : 1, 2 : 8, 3 : 27, 4 : 64, 5 : 125, 6 : 216}
>>> 3 in cubes
True                    # Output
>>> 17 not in cubes
True                    # Output
```

Que 3.21. **Write some built in dictionary methods used in Python with example.**

Answer

There are some built-in dictionary methods which are included in Python given in Table 3.21.1.

Table 3.21.1. Built-in dictionary methods.

S. No.	Function	Description
1.	all (dict)	It is a Boolean type function, which returns true if all keys of dictionary are true (or the dictionary is empty).
2.	any (dict)	It is also a Boolean type function, which returns true if any key of the dictionary is true. It returns false if the dictionary is empty.
3.	len (dict)	It returns the number of items (length) in the dictionary.
4.	cmp (dict1, dict2)	It compares the items of two dictionaries.
5.	sorted (dict)	It returns the sorted list of keys.
6.	dict.clear ( )	It deletes all the items in a dictionary at once.
7.	dict. copy ( )	It returns a copy of the dictionary.
8.	dict.get (key, default = None)	For key key, returns value or default if key not in dictionary.
9.	dict.items ( )	It returns a list of entire key : value pair of dictionary.
10.	dict.keys ( )	It returns the list of all the keys in dictionary.
11.	dict.update (dict2)	It adds the items from dict2 to dict.
12.	dict.values ( )	It returns all the values in the dictionary.

For example :

```
>>>cubes = {1 : 1, 2 : 8, 3 : 27, 4 : 64, 5 : 125, 6 : 216}
>>>all (cubes)
True                                    # Output
>>>any (cubes)
True                                    # Output
>>>len (cubes)
6                                       # Output
>>>sorted (cubes)
[1, 2, 3, 4, 5, 6]                      # Output
>>>str (cubes)
```



```
'{1 : 1, 2 : 8, 3 : 27, 4 : 64, 5 : 125, 6 : 216}'      # Output
```

Que 3.22. **Differentiate between list, tuple set and dictionary.**

Answer

S. No.	List	Tuple	Set	Dictionary
1.	A list is a non-homogeneous data structure that stores the elements in columns of a single row or multiple rows.	A Tuple is also a non-homogeneous data structure that stores elements in columns of a single row or multiple rows.	The set data structure is also a non homogeneous data structure but stores the elements in a single row.	A dictionary is also a non-homogeneous data structure that stores key-value pairs.
2.	The list can be represented by [ ]	Tuple can be represented by ( )	The set can be represented by { }	The dictionary can be represented by { }
3.	The list allows duplicate elements	Tuple allows duplicate elements	The Set will not allow duplicate elements	The dictionary doesn't allow duplicate keys.
4.	The list can use nested among all	Tuple can use nested among all	The set can use nested among all	The dictionary can use nested among all
5.	Example: [1, 2, 3, 4, 5]	Example: (1, 2, 3, 4, 5)	Example: {1, 2, 3, 4, 5}	Example: {1: "a", 2: "b", 3: "c", 4: "d", 5: "e"}
6.	A list can be created using the list()	Tuple can be created using the tuple() function.	A setA dictionary can be created using the set() function	A dictionary can be created using the dict() function.
7.	A list is mutable *i.e.* we can make any changes in the list.	A tuple is immutable *i.e.* we can not make any changes in the tuple.	A set is mutable *i.e.*, we can make any changes in the set, but elements are not duplicated.	A dictionary is mutable, but Keys are not duplicated.
8.	List is ordered	Tuple is ordered	Set is unordered	Dictionary is ordered (Python 3.7 and above)

Que 3.23. **Compare list and tuple data structure with suitable examples. Explain the concept of list comprehension.**

AKTU 2022-23, Sem-4; Marks 10

Answer

Compare list and tuple : Refer Q. 3.22, Page 3–17V, Unit-3.

List comprehension : Refer Q. 3.11, Page 3–9V, Unit-3.

Que 3.24. **Compare list and dictionary data structure. Explain various dictionary methods with suitable examples of each.**

AKTU 2022-23, Sem-4; Marks 10

Answer

Compare list and dictionary data structure : Refer Q. 3.22, Page 3–17V, Unit-3.

Dictionary methods : Refer Q. 3.21, Page 3–15V, Unit-3.

PART-4

Manipulations Building Blocks of Python Programs, String Manipulation Methods, List Manipulation, Dictionary Manipulation.

Que 3.25. **What are the fundamental building blocks in Python programming language ?**

Answer

The fundamental building blocks in Python programming are :

1. **Variables and Data Types :** Variables are used to store information. Data types specify the type of data a variable can hold, such as integers, floats, strings, lists, dictionaries, etc.
2. **Lists :** Lists are ordered collections of elements. They allow you to store and manipulate sets of related data.
3. **Dictionaries :** Dictionaries store data in key-value pairs. They are useful for associating information.
4. **Conditionals :** Conditionals like if, elif, and else statements allow your program to make decisions based on conditions.
5. **Loops :** Loops allow you to repeat a block of code multiple times. Python supports for and while loops.

6. **Functions :** Functions are blocks of reusable code. They allow you to organize code into manageable chunks.
7. **Modules and Libraries :** Modules are files that contain Python definitions and statements. Libraries are collections of modules. They allow you to use pre-written code to perform specific tasks.

Que 3.26. **Explain string manipulation. What are the various methods used to manipulate strings in Python ?**

Answer

String manipulation : String manipulation refers to the process of modifying or manipulating text data in a program. This can involve tasks like combining strings, extracting specific portions of text, changing the case of letters, replacing certain characters, and much more.

Here are some common string manipulation methods in Python:

1. **Concatenation (+) :** You can combine two or more strings together.
   Example:
   ```
   greeting = "Hello"
   name = "Alice"
   full_greeting = greeting + " " + name  # Result: "Hello Alice"
   ```
2. **Length (len()) :** This function returns the length of a string.
   Example:
   ```
   message = "Hello, world!"
   length = len(message)  # Result: 13
   ```
3. **Uppercase and Lowercase (str.upper() and str.lower()) :** These methods change the case of the string.
   Example:
   ```
   text = "Hello"
   uppercase_text = text.upper()  # Result: "HELLO"
   lowercase_text = text.lower()  # Result: "hello"
   ```
4. **Replace (str.replace()) :** This method replaces a specified substring with another substring.
   Example:
   ```
   sentence = "Python is fun"
   new_sentence = sentence.replace("fun", "awesome")  # Result: "Python is awesome"
   ```
5. **Split (str.split()) :** This method divides a string into a list of substrings based on a delimiter.
   Example:
   ```
   sentence = "Python is amazing"
   words = sentence.split()  # Result: ['Python', 'is', 'amazing']
   ```

6. **Join (str.join()) :** This method combines a list of strings into a single string using a specified delimiter.

   Example:

   ```
   words = ['Python', 'is', 'awesome']
   sentence = ' '.join(words)  # Result: "Python is awesome"
   ```

7. **Strip (str.strip()) :** This method removes leading and trailing whitespace from a string.

   Example:

   ```
   text = "  Hello  "
   stripped_text = text.strip()  # Result: "Hello"
   ```

8. **Checking for Substrings (in keyword) :** You can use the in keyword to check if a substring is present in a string.

   Example:

   ```
   sentence = "Python is powerful"
   if "Python" in sentence:
       print("Python is in the sentence")
   ```

Que 3.27. Explain list manipulation. What are the various methods used to manipulate list in Python ?

Answer

List manipulation refers to the process of modifying or manipulating lists in a program. Lists are a fundamental data structure in Python, and they allow you to store and organize collections of items. List manipulation involves tasks like adding or removing elements, rearranging the order of items, extracting specific elements, and more. Following are some common list manipulation methods in Python :

1. **Append (list.append()) :** This method adds an element to the end of a list.

   Example:

   ```
   my_list = [1, 2, 3]
   my_list.append(4)  # Result: [1, 2, 3, 4]
   ```

2. **Extend (list.extend()) :** This method appends the elements of another list to the end of the current list.

   Example:

   ```
   list1 = [1, 2, 3]
   list2 = [4, 5, 6]
   list1.extend(list2)  # Result: [1, 2, 3, 4, 5, 6]
   ```

3. **Insert (list.insert()) :** This method inserts an element at a specified position in the list.

   Example:

   ```
   my_list = [1, 2, 3]
   ```



   ```
   my_list.insert(1, 4)  # Result: [1, 4, 2, 3]
   ```

4. **Remove (list.remove()) :** This method removes the first occurrence of a specified element from the list.

   Example:

   ```
   my_list = [1, 2, 3, 2, 4]
   my_list.remove(2)  # Result: [1, 3, 2, 4]
   ```

5. **Pop (list.pop()) :** This method removes and returns the element at a specified index.

   Example:

   ```
   my_list = [1, 2, 3]
   popped_element = my_list.pop(1)  # Result: popped_element = 2, my_list = [1, 3]
   ```

6. **Clear (list.clear()) :** This method removes all elements from the list, leaving it empty.

   Example:

   ```
   my_list = [1, 2, 3]
   my_list.clear()  # Result: my_list = []
   ```

7. **Index (list.index()) :** This method returns the index of the first occurrence of a specified element.

   Example:

   ```
   my_list = [1, 2, 3, 2, 4]
   index = my_list.index(2)  # Result: index = 1
   ```

8. **Count (list.count()) :** This method returns the number of occurrences of a specified element in the list.

   Example:

   ```
   my_list = [1, 2, 3, 2, 4, 2]
   count = my_list.count(2)  # Result: count = 3
   ```

9. **Reverse (list.reverse()) :** This method reverses the order of elements in the list.

   Example:

   ```
   my_list = [1, 2, 3]
   my_list.reverse()  # Result: [3, 2, 1]
   ```

10. **Sort (list.sort()) :** This method sorts the elements of the list in ascending order.

    Example:

    ```
    my_list = [3, 1, 2]
    my_list.sort()  # Result: [1, 2, 3]
    ```

11. **Copy (list.copy()) :** This method creates a shallow copy of the list.

    Example:

    ```
    my_list = [1, 2, 3]
    ```

```
copy_list = my_list.copy()  # copy_list is a separate list with the same elements
```

Que 3.28. Explain dictionary manipulation. List various methods to manipulate dictionary in python.

Answer

Dictionary manipulation : Dictionary manipulation refers to the process of performing operations or making changes to a dictionary in Python. This can include tasks like adding or removing key-value pairs, updating existing values, checking for the presence of keys, iterating through the elements, merging dictionaries, and more.

You can manipulate dictionaries using various methods. Here are some common ones :

1. **Accessing Elements :**
   my_dict[key]: Retrieves the value associated with the specified key.
   my_dict.get(key, default): Retrieves the value for a given key, with an optional default value if the key is not found.
2. **Adding or Updating Elements :**
   my_dict[key] = value: Adds a new key-value pair or updates the value if the key already exists.
   my_dict.update(another_dict): Merges another dictionary into the current one.
3. **Removing Elements :**
   del my_dict[key]: Deletes the key-value pair associated with the specified key.
   my_dict.pop(key, default): Removes and returns the value for the given key, with an optional default value if the key is not found.
   my_dict.clear(): Removes all key-value pairs from the dictionary.
4. **Iterating through a Dictionary :**
   for key in my_dict: ...: Iterates through the keys.
   for key, value in my_dict.items(): ...: Iterates through both keys and values.
5. **Checking for Key Existence :**
   key in my_dict: Checks if a key exists in the dictionary.
   my_dict.get(key) is not None: Checks if a key exists and its value is not None.
6. **Getting Keys and Values :**
   my_dict.keys(): Returns a view of all keys in the dictionary.
   my_dict.values(): Returns a view of all values in the dictionary.
   my_dict.items(): Returns a view of key-value tuples.

7. **Copying a Dictionary :**
   new_dict = my_dict.copy(): Creates a shallow copy of the dictionary.
   new_dict = dict(my_dict): Another way to create a shallow copy.

PART-5

Programming Using String, List and Dictionary in-built Functions.

Que 3.29. Write a program that accepts a sentence and calculate the number of digits, uppercase and lowercase letters.

AKTU 2021-22, Sem-4; Marks 10

Answer

```
defstring_test(s):
    d={"UPPER_CASE":0,"LOWER_CASE":0 , "DIGITS":0}
    for c in s:
    if c.isupper():
    d["UPPER_CASE"]+=1
    elifc.islower():
    d["LOWER_CASE"]+=1
    elif c in range(0,10):
    d["DIGITS"]+=1
    else:
    pass
    print("Original String : ", s)
    print("No. of Upper case characters : ", d["UPPER_CASE"])
    print("No. of Lower case Characters : ", d["LOWER_CASE"])
    print("No. of Digits : ", d["NUMBERS"])
    string_test('The quick Brown Fox 12')
```

Que 3.30. Write a Python program to change a given string to a new string where the first and last chars have been exchanged.

AKTU 2021-22, Sem-3; Marks 10

Answer

```
defchange_sring(str1):
return str1[-1:] + str1[1:-1] + str1[:1]
print(change_sring('abcd'))
```

Output : dbca

Que 3.31. **Write a Python program to count the vowels present in given input string. Explain the output of program through example.**

AKTU 2022-23, Sem-4; Marks 10

Answer

```
def vowel_count(str):
    count = 0
    vowel = set("aeiouAEIOU")
    for alphabet in str:
        if alphabet in vowel:
            count = count + 1
    print("No. of vowels :", count)
str = "namasteLondon"
vowel_count(str)
```

Output:

No. of vowels : 5

Que 3.32. **How can you randomize the items of a list in place in Python ?**

AKTU 2021-22, Sem-3; Marks 10

Answer

The method shuffle() can be used to randomize the items of a list in place. However, this function is not accessible directly and therefore we need to import or call this function using random static object.

Syntax: shuffle (list_name)

Here, 'list_name' is passed as a parameter which could be a list or tuple. The shuffle() returns a reshuffled list of items.

For example :

```
import random
list = [20, 16, 10, 5];
random.shuffle(list)
print "Reshuffled list : ", list
random.shuffle(list)
print "Reshuffled list : ", list
```

Que 3.33. **Write a program that accepts sequence of lines as input and prints the lines after making all characters in the sentence capitalized.**

e.g., If Input :

Hello world
Practice makes perfect
Then, Output :
HELLO WORLD
PRACTICE MAKES PERFECT

AKTU 2022-23, Sem-4; Marks 10

Answer

```
# Initialize an empty list to store the lines
lines = []
# Continuously take input lines until an empty line is entered
while True:
    line = input("Enter a line (or press Enter to finish): ")
    # If the input line is empty, exit the loop
    if not line:
        break
    # Append the input line to the list
    lines.append(line)
# Capitalize each line and print them
for line in lines:
    print(line.upper())
```

Que 3.34. **Write a Python program to find permutations of a given string.**

Answer

```
def permute(s, answer):
    if (len(s) == 0):
        print(answer, end = " ")
        return
    for i in range(len(s)):
        ch = s[i]
        left_substr = s[0:i]
        right_substr = s[i + 1:]
        rest = left_substr + right_substr
        permute(rest, answer + ch)
# Driver Code
answer = ""
s = input("Enter the string : ")
print("All possible strings are : ")
permute(s, answer)
```

Output :

Enter the string : abc

All possible strings are : abc acb bac bca cab cba

PART-6

Python Functions, Organizing Python Codes Using Functions.

Que 3.35. Define function and write its advantages.

Answer

1. Functions are self-contained programs that perform some particular tasks.
2. Once a function is created by the programmer for a specific task, this function can be called anytime to perform that task.
3. Each function is given a name, using which we call it. A function may or may not return a value.
4. There are many built-in functions provided by Python such as dir (), len ( ), abs ( ), etc.
5. Users can also build their own functions, which are called user-defined functions.

Advantages of using functions :

1. They reduce duplication of code in a program.
2. They break the large complex problems into small parts.
3. They help in improving the clarity of code (*i.e.*, make the code easy to understand).
4. A piece of code can be reused as many times as we want with the help of functions.

Que 3.36. How to define and call function in Python ? Explain different parts of a function.

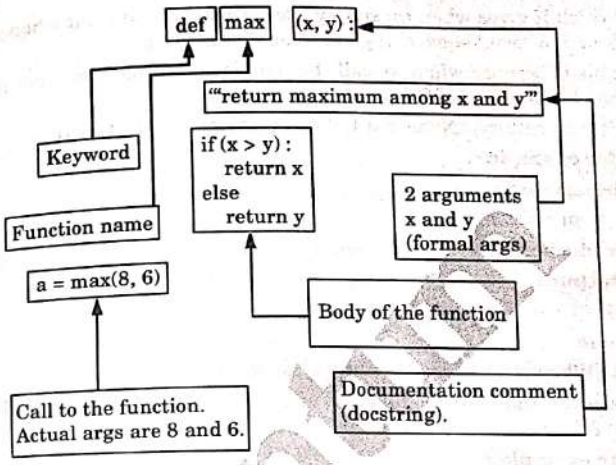
Answer

Function is defined by "def" keyword following by function name and parentheses.

Syntax of function definition : def function_name ( ) :

Syntax of functional call : function_name ( )

For example :



1. **Keyword :** The keyword 'def' is used to define a function header.
2. **Function name :** We define the function name for identification or to uniquely identify the function. In the given example, the function name is max. Function naming follows the same rules of writing identifiers in Python.
3. A colon (:) to mark the end of function header.
4. **Arguments :** Arguments are the values passed to the functions between parentheses. In the given example, two arguments are used, x and y. These are called formal arguments.
5. **Body of the function :** The body processes the arguments to do something useful. In the given example, body of the function is intended w.r.t. the def keyword.
6. **Documentation comment (docstring) :** A documentation string (docstring) to describe what the function does. In the given example, "return maximum among x and y" is the docstring.
7. An optional return statement to return a value from the function.
8. **Function call :** To execute a function, we have to call it. In the given example, a = max (8, 6) is calling function with 8 and 6 as arguments.

Que 3.37. Discuss the execution of a function with the help of example.

Answer

Let consider an example to understand the execution of function :

Step 1 : When function without "return" :

a. For example, we want the square of 4, and it should give answer "16" when the code is executed.

b. Which it gives when we simply use "print x*x" code, but when we call function "print square" it gives "None" as an output.

c. This is because when we call the function, recursion does not happen and leads to end of the function.

d. Python returns "None" for failing off the end of the function.

For example :

```
def square(x) :
    print(x*x)
print(square(4))
```

Output :

16

None

Step 2 : When we retrieve the output using "return" command :

When we use the "return" function and execute the code, it will give the output "16".

For example :

```
def square(x) :
    return(x*x)
print(square(4))
```

Output :

16

Step 3 : When function is treated as an object :

a. Functions in Python are themselves an object, and an object has some value.

b. When we run the command "print square" it returns the value of the object.

c. Since we have not passed any argument, we do not have any specific function to run hence it returns a default value (0x021B2D30) which is the location of the object.

For example :

```
def square(x) :
    return(x*x)
print(square)
```

Output :

<function square at 0x021B2D30>

Que 3.38. Discuss the different types of argument-passing methods in Python. Explain the variable length argument with any suitable example. **AKTU 2022-23, Sem-3; Marks 10**

Answer

Argument-passing methods : There are three main types of argument-passing methods in Python :

1. **Default argument :**
   i. Default arguments are assigned a default value in the function definition. We assign default values to the argument using the '=' (assignment) operator at the time of function definition.
   ii. We can define a function with any number of default arguments.
   iii. If a corresponding argument is not provided in the function call, the default value is used.
2. **Keyword arguments (named arguments) :**
   i. Keyword arguments are those arguments where values get assigned to the arguments by their keyword (name) when the function is called.
   ii. It is preceded by the variable name and an (=) assignment operator.
3. **Positional arguments :**
   i. Positional arguments are those arguments where values get assigned to the arguments by their position when the function is called.
   ii. For example, the 1st positional argument must be 1^{st} when the function is called. The 2^{nd} positional argument needs to be 2^{nd} when the function is called.

Variable-length arguments : Python also supports variable-length arguments, allowing a function to accept a variable number of arguments. There are two types of variable-length arguments :

1. **arbitrary positional arguments (*args) :**
   i. We can declare a variable-length argument with the * (asterisk) symbol.
   ii. The *args parameter allows a function to accept any number of positional arguments.
   iii. The arguments are collected as a tuple within the function.

For Example :

```
def sum_numbers(*args):
    total = 0
    for num in args:
        total += num
    return total
print(sum_numbers(1, 2, 3, 4)) # Output: 10
print(sum_numbers(5, 10, 15, 20, 25)) # Output: 75
```

1. In the example above, the sum_numbers() function accepts any number of positional arguments using *args.

2. The arguments are received as a tuple named args, and the function calculates the sum of all the numbers.

ii. arbitrary keyword arguments (kwargs) :**

i. The **kwargs parameter allows a function to accept any number of keyword arguments.

ii. The arguments are collected as a dictionary within the function.

For example :

```
def print_info(**kwargs):
    for key, value in kwargs.items():
        print(key, ":", value)
print_info(name="Alice", age=25, city="London")
```

1. In the example above, the print_info() function accepts any number of keyword arguments using **kwargs.
2. The arguments are received as a dictionary named kwargs, and the function prints each.

Que 3.39. Show an example where both Keyword arguments and Default arguments are used for the same function in a call. Show both the definition of the function and its call.

AKTU 2020-21, Sem-3; Marks 10

Answer

1. **Keyword arguments :** Refer Q. 3.38, Page 3–28V, Unit-3.

Definition of function and function call with keyword arguments :

```
def function_name(par1,par2,par3=defaultvalue1,
par4=defaultvalue2, ...):
return statements
function_name(par1,par2=value1, par3=value2,par4=value3)
```

For example :

```
def func(a, b, c=10):
print('a is', a, 'and b is', b, 'and c is', c)
func(3, 7) # output:  a is 3 and b is 7 and c is 10
func(5,25, c=24)     # output:   a is 5 and b is 25 and c is 24
func(c=50, a=100,b=10))  # output:   a is 100 and b is 10 and c is 50
```

2. **Default arguments :** Refer Q. 3.38, Page 3–28V, Unit-3.

For example :

In the below example, the default value is given to argument 'b' and 'c'

```
def add(a,b=5,c=10):
    return (a+b+c)
```


Definition of function with default arguments and keyword :

```
def function_name(par1,par2,par3,
par4=defaultvalue1,par5=defaultvalue2,...):
    return statements
```

Function can be called in three ways :

1. **Giving only the mandatory argument :** It means only pass the value of non-default parameter to the function.

For example : In this example we only pass value of 'a' to the function and rest value are assigned as default value.

```
def add(a,b=5,c=10):
    return (a+b+c)
print(add(3))
```

Output : 18

2. **Giving one of the optional arguments :** It means passing value to any one optional default parameter and all non-default parameter.

For example : Here, 3 is assigned to a, 4 is assigned to b.

```
def add(a,b=5,c=10):
    return (a+b+c)
print(add(3,4))
```

Output : 17

3. **Giving all the arguments :** It means to pass values to all the parameter in the function during function call.

For example : Here, 2 is assigned to a, 3 is assigned to b,4 is assigned to c.

```
def add(a,b=5,c=10):
    return (a+b+c)
print(add(2,3,4))
```

Output : 9

Que 3.40. Discuss the scope rules in Python.

Answer

1. Scope of a name is the part of the program in which the name can be used.
2. Two variables can have the same name only if they are declared in separate scopes.
3. A variable cannot be used outside its scopes.
4. Fig. 3.40.1 illustrates Python's four scopes.

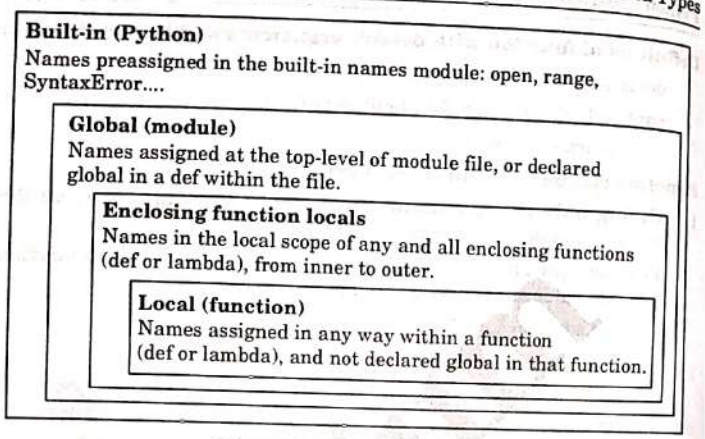


Fig. 3.40.1. The LEGB scope.

5. The LEGB rule refers to local scope, enclosing scope, global scope, and built-in scope.
6. Local scope extends for the body of a function and refers to anything indented in the function definition.
7. Variables, including the parameter, that are defined in the body of a function are local to that function and cannot be accessed outside the function. They are local variables.
8. The enclosing scope refers to variables that are defined outside a function definition.
9. If a function is defined within the scope of other variables, then those variables are available inside the function definition. The variables in the enclosing scope are available to statements within a function.

Que 3.41. Discuss function in Python with its parts and scope. Explain with example. (Take simple calculator with add, subtract, division and multiplication). **AKTU 2019-20, Sem-3; Marks 10**

Answer

Function : Refer Q. 3.35, Page 3–26V, Unit-3.

Parts of function : Refer Q. 3.36, Page 3–26V, Unit-3.

Scope : Refer Q. 3.40, Page 3–31V, Unit-3.

For example : Simple calculator using Python :

```
# This function adds two numbers
def add(x, y) :
    return x + y
```



```
# This function subtracts two numbers
def subtract(x, y) :
    return x – y
# This function multiplies two numbers
def multiply(x, y) :
    return x * y
# This function divides two numbers
def divide(x, y) :
    return x / y
print("Select operation.")
print("1.Add")
print("2.Subtract")
print("3.Multiply")
print("4.Divide")
# Take input from the user
choice = input("Enter choice(1/2/3/4) : ")
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
if choice == '1' :
print(num1,"+",num2,"=", add(num1,num2))
elif choice == '2' :
print(num1,"–",num2,"=", subtract(num1,num2))
elif choice == '3' :
print(num1,"*",num2,"=", multiply(num1,num2))
elif choice == '4':
print(num1,"/",num2,"=", divide(num1,num2))
else :
print("Invalid input")
```

Que 3.42. Write a function in find the HCF of some given numbers.

Answer

```
>>>def hcf (a, b) :
    if a > b :
        small = b
    else :
        small = a
    for i in range (1, small + 1) :
        if (a % i == 0) and (b % i == 0)) :
            hcf = i
```

```
    return hcf
>>> hcf (20, 40)
20                                  #Output
>>>hcf (529, 456)
1                                   #Output
```

Que 3.43. **Write a function to find the ASCII value of the character.**

Answer

```
>>>def ascii_val_of (a) :
    print ("The ASCII value of '" + a + "' is", ord (a))
>>>ascii_val_of ('A')
("The ASCII value of 'A' is", 65)      #Output
>>>ascii_val_of (' ')                  #Finding ASCII value of space
("The ASCII value of ' ' is", 32)       #Output
```

Que 3.44. **Write a function to convert a decimal number to its binary, octal and hexadecimal equivalents.**

Answer

```
>>>def bin_oct_hex (a) :
    print (bin (a), "binary equivalent")
    print (oct (a), "octal equivalent")
    print (hex (a), "hexadecimal equivalent")
>>>bin_oct_hex (10)                   #Finding binary, octal, hex value of 10
('0b1010', 'binary equivalent')        #Output
('012', 'octal equivalent')
('0xa', 'hexadecimal equivalent')
```

Que 3.45. **Write a Python function removekth (s, k) that takes as input a string a and an integer $k >= 0$ and removes the character at index k. If k is beyond the length of s, the whole of s is returned. For example,**

removekth ("PYTHON", 1) returns "PTHON"
removekth ("PYTHON", 3) returns "PYTON"
removekth ("PYTHON", 0) returns "YTHON"
removekth ("PYTHON", 20) returns "PYTHON"

AKTU 2020-21, Sem-3; Marks 10

Answer

```
def removekth(s, k):
```



```
    str=""
    for i in range(len(s)):
    ifi !=k :
    str += s[i]
    return str
    removekth("PYTHON", 1)  #output PTHON
    removekth("PYTHON", 3)  #output PYTON
    removekth("PYTHON", 0)  #output YTHON
    removekth("PYTHON", 20) #output  PYTHON
```

Que 3.46. **Write a program factors (N) that returns a list of all positive divisors of N ($N > = 1$). For example :**

factors (6) returns [1, 2, 3, 6]
factors (1) returns [1]
factors (13) returns [1, 13]

AKTU 2020-21, Sem-3; Marks 10

Answer

```
def factor(num):
    flist=[]
    if num>=1:
        for i in range(1,num+1):
        ifnum % i==0:
            flist.append(i)
        return flist
        print(factor(6))   # [1, 2, 3, 6]
        print(factor(1))   # [1]
        print(factor(13))  # [1, 13]
```

Que 3.47. **Write a function makePairs that takes as input two lists of equal length and returns a single list of same length where k-th element is the pair of k-th elements from the input lists. For example,**

makePairs ([1, 3, 5, 7], [2, 4, 6, 8])
returns [(1, 2), (3, 4), (5, 6), (7, 8)]
makePairs ([], [])
returns []

AKTU 2020-21, Sem-3; Marks 10

Answer

```
def makePairs(list1,list2):
    x = len(list1)
```

```
    y = len(list2)
    if x==y:
        pairlist =[(list1[i], list2[i]) for i in range(0, x)]
    return pairlist
print(makePairs([1,3,5,7],[2,4,6,8]))
```

Output :

```
[(1,2),(3,4),(5,6),(7,8)]
```

Que 3.48. **Write a Python program, countSquares (*N*), that returns the count of perfect squares less than or equal to *N* (*N* > 1). For example :**

countSquares (1) returns 1
 # Only 1 is a perfect square < = 1
countSquares (5) returns 2
 # 1, 4 are perfect squares < = 5
countSquares (55) returns 7
 # 1, 4, 9, 16, 25, 36, 49 < = 55

AKTU 2020-21, Sem-3; Marks 10

Answer

```
def countSquares(N):
    cnt=0
    fori  in range(1,N+1):
        j = 1;
        while j * j <= i:
        if j * j == i:
        cnt = cnt + 1
        j = j + 1
        i = i + 1
        returncnt
    b = 55
    print (countSquares(b)) #output 7
```

Que 3.49. **Write a Python function, alternating (lst), that takes as argument a sequence lst. The function returns True if the elements in lst are alternately odd and even, starting with an even number. Otherwise it returns False. For example :**

alternating ([10, 9, 6]) returns False
alternating ([10, 15, 8]) returns True
alternating ([10]) returns True
alternating ([ ]) returns True

alternating ([15, 10, 9]) returns False

AKTU 2020-21, Sem-3; Marks 10

Answer

```
def alternating(lst):
    if len(lst) == 0:
        return True
    if len(lst) == 1:
        if lst[0]%2==0:
            return True
        return False
    if lst[0]%2 ==0:
        for i in range(len(lst)-1):
            if lst[i]%2 == lst[i+1]%2:
        return False
        return True
    else:
        return False
        # lst = [10,9,9,6]
        print(alternating([10,9,9,6])) # False
        print(alternating([10,15,8,7,90,11])) # True
        print(alternating([10])) # True
        print(alternating([]))  # True
        print(alternating([15,10,9])) # False
```

Que 3.50. **Write a Python function, searchMany(*s*, *x*, *k*), that takes as argument a sequence *s* and integers *x*, *k* (*k* > 0). The function returns True if there are at most *k* occurrences of *x* in *s*. Otherwise it returns False. For example :**

searchMany ([10, 17, 15, 12], 15, 1) returns True
searchMany ([10, 12, 12, 12], 12, 2) returns False
searchMany ([10, 12, 15, 11], 17, 18) returns True

AKTU 2020-21, Sem-3; Marks 10

Answer

```
defsearchMany(s, x, k):
    count=0
    for i in s :
```

```
        if i == x :
            count += 1
            if count == k :
            return True
    else:
    return False
    print (searchMany([10, 17, 15, 12],15,1)) # output True
    print (searchMany([10, 12, 12, 12],12,2)) # output True
    print (searchMany([10, 17, 15, 11],17,18)) # output False
```

Que 3.51. **Explain higher order function with respect to lambda expression. Write a Python code to count occurrences of an element in a list.**

AKTU 2019-20, Sem-3; Marks 10

Answer

Higher order function with respect to lambda expression : Out of syllabus from session (2023-24).

Program to count occurrences of an element in a list :

```
# vowels list
vowels = ['a', 'e', 'i', 'o', 'i', 'u']
# count element 'i'
count = vowels.count('i')
# print count
print('The count of i is:', count)
# count element 'p'
count = vowels.count('p')
# print count
print('The count of p is:', count)
```

Que 3.52. **Write a Python program, triangle (*N*), that prints a right triangle having base and height consisting of *N* * symbols as shown in these examples :**

triangle (3) prints :

```
*
* *
* * *
```


triangle (5) prints

```
*
* *
* * *
* * * *
* * * * *
```

AKTU 2020-21, Sem-3; Marks 10

Answer

```
def triangle(N):
  for i in range(1, N + 1):
    for j in range(i):
      print('*', end=' ')
    print()
# Get user input for the number of rows
N = int(input("Enter the number of rows: "))
triangle(N)
```

[Note : In above program we can provide the value of N as input, and it will print the right triangle with N rows, where each row consists of N asterisks.]

☺☺☺

4 UNIT

Python File Operations

CONTENTS

PART-1

Reading Files, Writing Files in Python.

Que 4.1. What are files ? How are they useful ?

Answer

1. A file in a computer is a location for storing some related data.
2. It has a specific name.
3. The files are used to store data permanently on to a non-volatile memory (such as hard disks).
4. As we know, the Random Access Memory (RAM) is a volatile memory type because the data in it is lost when we turn off the computer. Hence, we use files for storing of useful information or data for future reference.

Que 4.2. Describe the opening a file function in Python.

Answer

1. Python has a built-in open () function to open files from the directory.
2. Two arguments that are mainly needed by the open () function are :
   a. **File name :** It contains a string type value containing the name of the file which we want to access.
   b. **Access_mode :** The value of access_mode specifies the mode in which we want to open the file, *i.e.*, read, write, append etc.
3. **Syntax :**

```
file_object = open(file_name [, access_mode])
```

For example :

```
>>>f = open ("test.txt")                #Opening file current directory
>>>f = open ("C:/Python27/README.txt")
                                        #Specifying full path      #Output
>>>f
<open file 'C:/Python27/README.txt', mode 'r' at 0x02BC5128>
                                                                   #Output
```

Que 4.3. Explain the closing a file method in Python.

Answer

1. When the operations that are to be performed on an opened file are finished, we have to close the file in order to release the resources.
2. Python comes with a garbage collector responsible for cleaning up the unreferenced objects from the memory; we must not rely on it to close a file.
3. Proper closing of a file frees up the resources held with the file.
4. The closing of file is done with a built-in function close ().
5. **Syntax :**

   fileObject. close ()

   For example :

```
# open a file
>>> f = open ("test. txt", "wb")
# perform file operations
>>> f. close()  # close the file
```

Que 4.4. **Discuss writing to a file operation.**

Answer

1. After opening a file, we have to perform some operations on the file. Here we will perform the write operation.
2. In order to write into a file, we have to open it with *w* mode or *a* mode, on any writing-enabling mode.
3. We should be careful when using the *w* mode because in this mode overwriting persists in case the file already exists.

   For example :

```
# open the file with w mode
>>> f = open ("C :/Python27/test.txt", "w")
# perform write operation
>>>f. write ('writing to the file line 1/n')
>>>f. write ('writing to the file line 2/n')
>>>f. write ('writing to the file line 3/n')
>>>f. write ('writing to the file line 4')
# close the file after writing
>>> f.close ()
```

The given example creates a file named test.txt if it does not exist, and overwrites into it if it exists. If we open the file, we will find the following content in it.


```
Output :
Writing to the file line 1
Writing to the file line 2
Writing to the file line 3
Writing to the file line 4
```

Que 4.5. **Explain reading from a file operation with example.**

Answer

1. In order to read from a file, we must open the file in the reading mode (*r* mode).
2. We can use read (size) method to read the data specified by size.
3. If no size is provided, it will end up reading to the end of the file.
4. The read() method enables us to read the strings from an opened file.
5. **Syntax :**

   file object. read ([size])

   For example :

```
# open the file
>>> f = open ("C :/Python27/test.txt", "r")
>>>f . read (7)  #  read from starting 7 bytes of data
'writing'  #  Output
>>>f. read (6)  #  read next 6 bytes of data
'to the'  #  Output
```

Que 4.6. **Discuss file I/O in Python. How to perform open, read, write, and close into a file ? Write a Python program to read a file line-by-line store it into a variable.**

AKTU 2019-20, Sem-3; Marks 10

OR

What are file input and output operations in Python Programming ?

AKTU 2021-22, Sem-3; Marks 10

Answer

File I/O : Refer Q. 4.1, Page 4–2V, Unit-4.

Open, read, write and close into a file : Refer Q. 4.2, Page 4–2V, Refer Q. 4.5, Page 4–4V, Refer Q. 4.4, Page 4–3V and Refer Q. 4.3, Page 4–2V; Unit-4.

Program :

```
L = ["Quantum\n", "for\n", "Students\n"]
# writing to file
file1 = open('myfile.txt', 'w')
file1.writelines(L)
file1.close()
# Using readlines()
```

```
file1 = open('myfile.txt', 'r')
Lines = file1.readlines()
count = 0
# Strips the newline character
for line in Lines:
print(line.strip())
print("Line{}: {}".format(count, line.strip()))
```

PART-2

Understanding Read Functions, Read (), Readline (), Readlines ().

Que 4.7. What is the purpose of reading data from file in Python ?

Answer

The purpose of reading data from file in Python is :

1. **Data Persistence :** Files allow data to be stored persistently. This means the data can be saved even after the program terminates.
2. **Data Retrieval :** Reading from a file allows a program to retrieve and use data that was previously saved.
3. **Input Sources :** Files can serve as input sources for a program, providing a way to feed external data into the program's logic.
4. **Configuration Files :** Files are commonly used to store configuration parameters for a program. This allows users to customize the behavior of the program without modifying the source code.
5. **Data Analysis :** When dealing with large datasets, it's often more efficient to read data from a file in chunks rather than loading it all into memory at once.
6. **Interoperability :** Files provide a universal way to exchange data between different programs or systems. For example, data files can be shared between different programming languages or software applications.
7. **Logs and Records :** Files are commonly used to log events, errors, or other information generated by a program. These logs can be crucial for debugging and monitoring.
8. **Archiving and Backups :** Files are a primary means for creating backups or archiving data for future use or reference.

Que 4.8. What are the different file access modes in Python ?

Answer

There are 6 access modes in python.

1. **Read Only ('r') :** Open text file for reading. The handle is positioned at the beginning of the file. If the file does not exist, raises the I/O error. This is also the default mode in which a file is opened.
2. **Read and Write ('r+') :** Open the file for reading and writing. The handle is positioned at the beginning of the file. Raises I/O error if the file does not exist.
3. **Write Only ('w') :** Open the file for writing. For the existing files, the data is truncated and over-written. The handle is positioned at the beginning of the file. Creates the file if the file does not exist.
4. **Write and Read ('w+') :** Open the file for reading and writing. For an existing file, data is truncated and over-written. The handle is positioned at the beginning of the file.
5. **Append Only ('a') :** Open the file for writing. The file is created if it does not exist. The handle is positioned at the end of the file. The data being written will be inserted at the end, after the existing data.
6. **Append and Read ('a+') :** Open the file for reading and writing. The file is created if it does not exist. The handle is positioned at the end of the file. The data being written will be inserted at the end, after the existing data.

Que 4.9. Explain the purpose of read ( ) function in python. How does it operate ?

Answer

1. In Python, the read() function is used to read a specified number of characters from a file or, if no argument is provided, it reads the entire content of the file.
2. Syntax: file.read()
3. Parameter Values

Parameter	Description
size	Optional. The number of bytes to return. Default -1, which means the whole file.

4. It operates in different ways depending on the mode in which the file is opened :
   a. **Text Mode ('r' or 'rt') :** When a file is opened in text mode, read() reads a specified number of characters (or the entire file if no

argument is given) and returns them as a string. If the end of the file is reached, it returns an empty string.

Example :

```
with open('example.txt', 'r') as file:
content = file.read(10) # Reads the first 10 characters
    print(content)
```

b. **Binary Mode ('rb') :** In binary mode, read() reads a specified number of bytes (or the entire file if no argument is given) and returns them as bytes.

Example :

```
with open('example.jpg', 'rb') as file:
    data = file.read(1024) # Reads the first 1024 bytes
print(data)
```

Que 4.10. What is the purpose of readline () in Python ? What does it return ?

Answer

1. The readline() function in Python is used to read a single line from a file.
2. Syntax

file.readline(size)

3. Parameter Values

Parameter	Description
size	Optional. The number of bytes from the line to return. Default -1, which means the whole line.

4. It reads characters from the current position of the file pointer until it encounters a newline character ('\n') or reaches the end of the file.

   Here's an example :

```
with open('example.txt', 'r') as file:
line1 = file.readline() # Reads the first line
line2 = file.readline() # Reads the second line
print(line1)
print(line2)
```

In this example, if example.txt contains:


```
Hello, this is line 1.
And this is line 2.
```

The output would be:

```
Hello, this is line 1.
And this is line 2.
```

5. As you can see, readline() returns a string containing the characters up to and including the newline character.
6. If the end of the file is reached, it returns an empty string (''). This behavior allows you to loop through a file line by line until you've processed the entire content.

Que 4.11. What is purpose of readlines () function ? What type of object it return ?

Answer

1. The readlines() function in Python is used to read all the lines from a file and return them as a list of strings.
2. Syntax

file.readlines(hint)

3. Parameter Values

Parameter	Description
hint	Optional. If the number of bytes returned exceeds the hint number, no more lines will be returned.

Default value is – 1, which means all lines will be returned.

4. Each element in the list corresponds to a line in the file, including the newline character ('\n') at the end of each line.

   Here's an example:

```
with open('example.txt', 'r') as file:
lines = file.readlines()
    print(lines)
```

If example.txt contains :

```
Line 1
Line 2
Line 3
```

The output would be:

```
['Line 1\n', 'Line 2\n', 'Line 3\n']
```

5. As you can see, readlines() returns a list where each element is a string representing a line from the file.

PART-3

Understanding Write Functions, Write () and Writelines ().

Que 4.12. Explain the purpose of write () function in python. How does it operate ?

Answer

1. The write() function in Python is used to write data to a file.
2. Syntax

file.write(byte)

3. Parameter Values

Parameter	Description
byte	The text or byte object that will be inserted.

4. It operates in different ways depending on the mode in which the file is opened:

a. **Text Mode ('w' or 'wt') :** When a file is opened in text mode, write() is used to write a string of characters to the file. If the file already exists, it will be truncated (emptied) before writing. If the file doesn't exist, a new file will be created.

Example:

```
with open('example.txt', 'w') as file:
file.write('This is line 1.\n')
file.write('This is line 2.\n')
```

This will create a file named example.txt with the content :

This is line 1.

This is line 2.

b. **Binary Mode ('wb') :** In binary mode, write() is used to write a sequence of bytes to the file.

Example :

```
with open('example.bin', 'wb') as file:
file.write(b'\x48\x65\x6c\x6c\x6f') # Writes ASCII values for "Hello"
```

This will create a binary file named example.bin.

5. It's important to note that the write() function does not automatically add a newline character ('\n') at the end of the string. If you want to start a new line, you need to include the newline character explicitly.
6. Also, when using write(), be careful with the mode in which the file is opened. If the file is opened in 'w' mode, it will overwrite the existing content. If you want to append to an existing file, you should open it in 'a' mode (append mode) instead.

Que 4.13. Describe the behaviour of writelines () function in python. What kind of input does it expect ?

Answer

1. The writelines function in Python is a method used to write a list of strings to a file. It is typically used with text files.
2. Syntax : file.writelines(list)
3. Parameter Values

Parameter	Description
list	The list of texts or byte objects that will be inserted.

4. This function expects a list of strings as input, where each string represents a line of text that you want to write to the file.

For example, if you have a list of strings like this:

```
lines = ['First line\n', 'Second line\n', 'Third line\n']
```

You can use writelines to write these lines to a file:

```
with open('example.txt', 'w') as file:
    file.writelines(lines)
```

In this example, the writelines function would write each string from the list as a separate line in the file 'example.txt'. The strings in the list should already contain the necessary newline characters ('\n') if you want them to be on separate lines in the file.

Que 4.14. How can you use 'writelines ()' to write a list of strings to a file ?

Answer

1. To use the writelines method to write a list of strings to a file in Python, you can follow these steps :

```
# Open a file in write mode ('w')
```

```
with open('myfile.txt', 'w') as file :
    # Create a list of strings
    lines = ['This is the first line\n', 'This is the second line\n', 'This is the third line\n']
    # Use writelines to write the list of strings to the file
    file.writelines(lines)
```

2. In this example, we :
   a. Open a file named 'myfile.txt' in write mode ('w') using a with statement.
   b. Create a list of strings called lines.
   c. Use the writelines method to write the list of strings to the file. This method takes a list of strings as an argument and writes each string to the file.
3. Remember to include newline characters ('\n') if you want to separate the lines in the file. This is important because writelines does not automatically add newline characters.

Que 4.15. **There is a file named input. Txt. Enter some positive numbers into the file named Input.Txt. Read the contents of the file and if it is an odd number write it to ODD.TXT and if the number is even, write it to EVEN.TXT.** **AKTU 2021-22, Sem-4; Marks 10**

Answer

Enter some contents into Input.txt file :

```
with open("Input.txt",'w',encoding = 'utf-8') as f :
    f.write("18\n")
    f.write("11\n")
    f.write("2\n")
    f.write("8\n")
    f.write("48\n")
    f.write("0\n")
    f.write("5\n")
    f.write("9\n")
    f.write("7\n")
    f.write("22\n")
```

Reading a file and saving into odd.txt and even.txt based on the condition :

```
file = open("Input.txt","rt")
for i in file:
    if i.strip:
        num = int(i)
        if (num % 2 == 0):
            even = open("even.txt","a")
            even.write(str(num))
            even.write("\n")
        else:
            odd = open("odd.txt","a")
            odd.write(str(num))
            odd.write("\n")
```


Que 4.16. **Discuss the concept of Iterators in Python. Explain, how we can create text file in Python ? Describe Python program to write the number of letters and digits in given input string into a File object.** **AKTU 2022-23, Sem-4; Marks 10**

Answer

Concept of iterators : Out of Syllabus from session (2023-24).

Creating a text file in Python :

1. To create a text file in Python, you can use the open() function. Here's an example code snippet :

```
# Open a file in write mode ('w' stands for write)
file = open('example.txt', 'w')
# Write content to the file
file.write('This is some text.')
# Close the file
file.close()
```

2. In this example, we :
   Use the open() function to create a file named 'example.txt' in write mode ('w').
3. Write the desired content to the file using the write() method.
4. Finally, remember to close the file using close() to free up system resources.
5. Keep in mind that if a file with the same name already exists, it will be overwritten. If you want to append content to an existing file, you can open it in append mode ('a') instead of write mode.

Python program to write the number of letters and digits in given input string into a File object :

```
def count_letters_digits(input_string):
```

```
    letters = 0
    digits = 0
for char in input_string:
    if char.isalpha():
        letters += 1
    elif char.isdigit():
    digits += 1
    return letters, digits
input_string = "Hello123"
file_name = "output.txt"
letters, digits = count_letters_digits(input_string)
with open(file_name, "w") as file:
    file.write(f"Number of letters: {letters}\n")
    file.write(f"Number of digits: {digits}\n")
print("Count of letters and digits written to the file successfully!")
```

PART-4

Manipulating File Pointer Using Seek Programming, Using File Operations.

Que 4.17. Explain the purpose of file handlers in python and why are they important in file.

Answer

File handlers, often referred to as file objects, are essential components in Python for working with files. They serve as an interface between your Python program and external files on your computer's storage.

Here are a few key purposes and reasons why file handlers are important in Python :

1. **Reading and Writing Files :** File handlers provide methods to read from and write to files. This enables you to interact with external data, such as text files, CSVs, JSON, and more.
2. **Resource Management :** File handlers help manage system resources efficiently. They automatically handle the opening and closing of files. The with statement is commonly used with file handlers to ensure that files are properly closed after use, even in the event of an error.

3. **Mode Selection :** When you open a file, you can specify a mode (e.g., 'r' for reading, 'w' for writing, 'a' for appending). File handlers ensure that the file is accessed according to the specified mode.
4. **File Position :** File handlers keep track of the current position within the file. This allows you to read or write at specific locations, or to move the pointer to a different position within the file.
5. **Buffering :** They manage buffering, which helps optimize I/O operations. This means that data may be read or written in chunks rather than one character at a time, improving performance.
6. **Iterating Over Lines :** File handlers provide methods to iterate over the lines of a file. This is very useful when dealing with large files or logs.
7. **Error Handling :** They handle errors that may occur during file operations. For example, if a file doesn't exist or there are permission issues, the file handler raises appropriate exceptions.
8. **Compatibility Across Platforms :** File handlers provide a consistent way to work with files, regardless of the underlying operating system. This ensures that your code can be easily ported to different environments.

Que 4.18. Demonstrate the file handling procedure in detail. Write a Python code to create file with 'P.txt' name and write your name and father's name in this file and then read this file to print it.

AKTU 2022-23, Sem-3; Marks 10

Answer

File handling procedure : Refer Q. 4.2, Page 4–2V, Refer Q. 4.5, Page 4–4V, Refer Q. 4.4, Page 4–3V and Refer Q. 4.3, Page 4–2V, Unit-4.

Program :

```
file_path = "P.txt"
# Writing to the file
with open(file_path, "w") as file:
    file.write("Your Name: Aditya Kumar\n")
    file.write("Father's Name: Shailesh Kumar")
# Reading from the file and printing its contents
with open(file_path, "r") as file:
content = file.read()
print("Contents of the file:")
print(content)
```

Que 4.19. What is the purpose of manipulating the file pointer using seek () function.

Answer

1. The purpose of manipulating the file pointer using the seek function in programming is to change the current position within a file. This allows you to read or write data at specific locations within the file, rather than just sequentially from the beginning.
2. Syntax: f.seek(offset, from_what), where f is file pointer
3. **Parameters :**

   Offset : Number of positions to move forward

   from_what : It defines point of reference.

   Returns : Return the new absolute position.
4. **The reference point is selected by the from_what argument. It accepts three values :**
   - **0 :** sets the reference point at the beginning of the file
   - **1 :** sets the reference point at the current file position
   - **2 :** sets the reference point at the end of the file

By default from_what argument is set to 0.

Que 4.20. How can you use seek () function to move read position to the beginning of file ?

Answer

To be able to reset the read position to the beginning of the file, follow the steps given below :

- **Step 1 :** Start by opening the file in read mode using the open() function.
- **Step 2 :** Then go on to assign the returned file object to a variable.
- **Step 3 :** Use the seek() method with an offset of 0 as the argument to reset the read position to the beginning of the file.
- **Step 4 :** Finally, can read the content from the new position using the read() method.

Example

Assuming there is a file myfile.txt with following content

```
#myfile.txt
This is a test file
# Open the file in read mode
file = open('myfile.txt', 'r')
```



```
# Reset the read position to the beginning of the file
file.seek(0)
# Read the content from the new position
content = file.read()
# Print the content
print("Content:", content)
# Close the file
file.close()
```

When above code is run, and the file myfile.txt is opened we get the following

Output

This is a test file

Que 4.21. How can you use seek () function to move write position to the beginning of file ?

Answer

By following the steps given below, you will be able to reset the write position to the beginning of the file.

- **Step 1 :** Begin by opening the file in write mode using the open() function.
- **Step 2 :** Go on to assign the returned file object to a variable.
- **Step 3 :** Use the seek() method with an offset of 0 as the argument to reset the write position to the beginning of the file.
- **Step 4 :** Lastly, write the desired content using the write() method.

Example

Assuming there is a file myfile.txt with following content

```
#myfile.txt
This is a test file
# Open the file in write mode
file = open('myfile.txt', 'w')
# Reset the write position to the beginning of the file
file.seek(0)
# Write new content at the write position
file.write("This is new content.")
# Close the file
file.close()
```

When above code is run, and the file myfile.txt is opened we get the following

Output

This is new content.

Que 4.22. How can you use seek () function to reset the read/write position to specific location.

Answer

You can reset the read/write position to a specific location within the file by providing the desired offset value to the seek() method. Let us look at an example :

Example

Assuming there is a file myfile.txt with the following content

```
#myfile.txt
This is a test file
# Open the file in read and write mode
file = open('myfile.txt', 'r+')
# Reset the read/write position to a specific location (e.g., 15th character)
file.seek(15)
# Perform read or write operations at the new position
content = file.read()
print("Content:", content)
# Close the file
file.close()
```

When the above code is run, and the file myfile.txt is opened we get the following

Output

Content: file.

Que 4.23. How can you use seek () function to reset the read position relative to current position.

Answer

- **Step 1 :** Start by opening the file in read mode using the open() function.
- **Step 2 :** Move on to assign the returned file object to a variable.
- **Step 3 :** Next make use of the seek() method with a positive or negative offset value to move the read position forward or backward from the current position.

- **Step 4 :** Finally, you read the content from the new position using the read() method.

Example

Assuming there is a file myfile.txt with following content

```
#myfile.txt
This is a test file
# Open the file in read mode
file = open('myfile.txt', 'r')
# Move the read position 5 characters forward from the start position
file.seek(5, 0)
# Read the content from the new position
content = file.read()
# Print the content
print("Content:", content)
# Close the file
file.close()
```

Output

Content: is a test file.

Que 4.24. How can you use seek () function to reset the write position relative to current position.

Answer

1. We begin by first opening the file in read mode and reading its entire content into the content variable.
2. Then we go on to close the file.
3. We modify the content of the file by inserting the desired new content at the desired position (in this case, 10 bytes from the beginning).
4. At last, we open the file in write mode, write the modified content back into the file, and close it.

Assuming there is a file myfile.txt with the following content :

```
#myfile.txt
This is a test file
# Open the file in read mode
file = open('myfile.txt', 'r')
content = file.read()
```

```
file.close()

# Modify the content

new_content = content[:10] + "New content" + content[10:]

# Open the file in write mode

file = open('myfile.txt', 'w')

file.write(new_content)

file.close()
```

When above code is run, and the file myfile.txt is opened we get the following

Que 4.25. How does the 'whence' parameter affect to positioning of the file pointer using seek () ?

Answer

1. The whence parameter in the seek function determines the reference point from which the offset is measured when repositioning the file pointer.
2. It takes one of the following three values :

a. **0 (SEEK_SET) :** This sets the offset from the beginning of the file. In other words, if you pass 0 as the whence parameter, the offset is interpreted as an absolute position from the start of the file.

b. **1 (SEEK_CUR) :** This sets the offset relative to the current position of the file pointer. If you pass 1 as the whence parameter, the offset is interpreted as a relative position from the current position.

c. **2 (SEEK_END) :** This sets the offset from the end of the file. If you pass 2 as the whence parameter, the offset is interpreted as an offset from the end of the file.

3. Here are some examples to illustrate how whence affects the positioning of the file pointer :

```
# Example 1 : Set the file pointer to the beginning of the file

file.seek(0, 0)  # Equivalent to file.seek(0)

# Example 2 : Move the file pointer 10 bytes forward from the current position

file.seek(10, 1)

# Example 3: Set the file pointer to the end of the file, then move 5 bytes back

file.seek(-5, 2)
```

4. In Example 3, if the file is 100 bytes long, seek(-5, 2) sets the file pointer to 5 bytes before the end of the file. The whence parameter (2) indicates that the offset is measured from the end.

☺☺☺

5
UNIT

Python Packages

CONTENTS

PART-1

Simple Programs using the Built-in Functions of Packages Matplotlib, Numpy, Pandas Etc.

Que 5.1. What is a Python package ? List various steps to create a package in Python.

Answer

Python modules may contain several classes, functions, variables, etc. whereas Python packages contain several modules. In simple terms, Package in Python is a folder that contains various modules as files.

Creating package :

Let's create a package in Python named mypckg that will contain two modules mod1 and mod2. To create this module follow the below steps :

- Create a folder named mypckg.
- Inside this folder create an empty Python file *i.e.*, __init__.py
- Then create two modules mod1 and mod2 in this folder.

Mod1.py

```
def gfg():
    print("Welcome to GFG")
```

Mod2.py

```
def sum(a, b):
    return a + b
```

Que 5.2. What is use of __init__.py in Python ?

Answer

1. __init__.py helps the Python interpreter recognize the folder as a package.
2. It also specifies the resources to be imported from the modules.
3. If the __init__.py is empty this means that all the functions of the modules will be imported. We can also specify the functions from each module to be made available.

   For example, we can also create the __init__.py file for the above module (Q. 5.1.) as :

   __init__.py

```
from .mod1 import vehicle
from .mod2 import car
```

This __init__.py will only allow the vehicle and car functions from the mod1 and mod2 modules to be imported.

4. Without an __init__.py file, the directory it is in cannot be treated as a package, and any attempt to import the package will fail.

Que 5.3. **What is Python module ? How to create and import a module in Python ?**

OR

How to create and import a module in Python ?

AKTU 2021-22, Sem-3, Marks 10

Answer

Python module :

1. A module is a file containing Python definitions and statements. A module can define functions, classes and variables.
2. It allows us to logically organize our Python code.
3. The file name is the module name with the suffix .py appended.
4. A module can also include runnable code. Grouping related code into a module makes the code easier to understand and use.
5. Definitions from a module can be imported into other modules or into the main module.

For example :

Here is an example of a simple module named as support.py

```
def print_func( par ):
    print "Hello : ", par
    return
```

Creating and importing a module in Python :

To create a module just save the code we want in a file with the file extension .py.

For example :

Save the given code in a file named mymodule.py

```
def greeting(name):
    print("Hello," + name)
```

Importing a module : Now we can use the newly created module or built-in module, by using the import statement:

For example :

Import the module named mymodule, and call the greeting function:

```
importmymodule
mymodule.greeting("Aditya Kumar")
```


Que 5.4. **What is the significance of module in Python ? What are the methods of importing a module ? Explain with suitable example.**

AKTU 2021-22, Sem-4, Marks 10

Answer

Significance of module in Python :

1. **Reusability :** Working with modules makes the code reusable.
2. **Simplicity :** Module focuses on a small proportion of the problem, rather than focusing on the entire problem.
3. **Scoping :** A separate namespace is defined by a module that helps to avoid collisions between identifiers.

Four ways to import a module :

1. Import the whole module using its original name: pycon import random.
2. Import specific things from the module: pycon from random import choice, randint.
3. Import the whole module and rename it, usually using a shorter variable name: pycon import pandas as pd.
4. Import specific things from the module and rename them as you're importing them: pycon from os.path import join as join_path.

Example :

Using import statement :

```
>>>importmath
```

You can also use the as syntax when importing a whole module.

```
>>>importmathasm
```

Que 5.5. **What is the Python libraries ? List some of the important Python libraries.**

Answer

A Python library is simply a collection of codes or modules of codes that we can use in a program for specific operations. We use libraries so that we don't need to write the code again in our program that is already available.

Let's have a look at some of the commonly used libraries :

1. **TensorFlow :** This library was developed by Google in collaboration with the Brain Team. It is an open-source library used for high-level

computations. It is also used in machine learning and deep learning algorithms. It contains a large number of tensor operations. Researchers also use this Python library to solve complex computations in Mathematics and Physics.

2. **Matplotlib :** This library is responsible for plotting numerical data. And that's why it is used in data analysis. It is also an open-source library and plots high-defined figures like pie charts, histograms, scatterplots, graphs, etc.

3. **Pandas :** Pandas are an important library for data scientists. It is an open-source machine learning library that provides flexible high-level data structures and a variety of analysis tools.

4. **Numpy :** The name "Numpy" stands for "Numerical Python". It is the commonly used library. It is a popular machine learning library that supports large matrices and multi-dimensional data.

5. **SciPy :** The name "SciPy" stands for "Scientific Python". It is an open-source library used for high-level scientific computations. This library is built over an extension of Numpy. It works with Numpy to handle complex computations. While Numpy allows sorting and indexing of array data, the numerical data code is stored in SciPy.

6. **Scrapy :** It is an open-source library that is used for extracting data from websites. It provides very fast web crawling and high-level screen scraping. It can also be used for data mining and automated testing of data.

7. **Scikit-learn :** It is a famous Python library to work with complex data. Scikit-learn is an open-source library that supports machine learning. It supports variously supervised and unsupervised algorithms like linear regression, classification, clustering, etc. This library works in association with Numpy and SciPy.

8. **PyGame :** This library provides an easy interface to the Standard Directmedia Library (SDL) platform-independent graphics, audio, and input libraries. It is used for developing video games using computer graphics and audio libraries along with Python programming language.

9. **PyTorch :** PyTorch is the largest machine learning library that optimizes tensor computations. It has rich APIs to perform tensor computations with strong GPU acceleration. It also helps to solve application issues related to neural networks.

10. **PyBrain :** The name "PyBrain" stands for Python Based Reinforcement Learning, Artificial Intelligence, and Neural Networks library. It is an open-source library built for beginners in the field of Machine Learning. It provides fast and easy-to-use algorithms for machine learning tasks.

Que 5.6. **How can you create Python file that can be imported as a library as well as run as a standalone script ?**

AKTU 2020-21, Sem-3, Marks 10

Answer

1. Module/library are units that store code and data.
2. It provides code-reuse to Python projects, and also useful in partitioning the system's namespaces in self-contained packages.
3. They are self-contained because we can only access attributes of a module/library after importing it.

Steps for to import Python file as a library :

Step 1 : Create a file and name it test.py

Step 2 : Inside test.py create a function called display_message().

For Example :

```
def display_message(name):
    return("Welcome "+ name +" to Quantum Page!")
```

Step 3 : Now create another file display.py.

Step 4 : Inside display.py import the module test.py file, as shown below :

```
import test
```

While importing, we do not have to mention the test.py but just the name of the file.

Step 5 : Then we can call the function display_message() from test.py inside display.py, we need to make use of module_name.function_name.

For example :

```
import test
    print(test.display_message("Aditya"))
```

Step 6 :

When we execute display.py, we will get the following output:

Welcome Aditya to Quantum Page!

Que 5.7. **Describe the difference between**

import library
and
from library import *
when used in a Python program. Here library is some Python library.

AKTU 2020-21, Sem-3, Marks 10

Answer

S. No.	Import library	From library import*
1.	Python's "import" loads a Python module/library into its own namespace.	"from" loads a Python module into the current namespace.
2.	By using import, we include all the code inside the module or library.	We include all the code or only the required function, classes present inside the module or library.
3.	Dot operator is required.	Dot operator is not required.
4.	It is required to mention the module name followed by dot to access any names in the module.	It is not needed to mention the module name to access any names in the module.
5.	**For example :** Here we import a library name feathers and then by using dot we call the methods which is included in the library. import feathers duster = feathers. ostrich("South Africa")	**For example :** Here we import library name feathers and then we call the method included in library without using dot operator. from feathers import * duster = ostrich("South Africa")

Que 5.8. Define Matplotlib. List some built-in functions in Matplotlib.

Answer

Matplotlib :

1. Matplotlib is a low level graph plotting library in python that serves as a visualization utility.
2. It Matplotlib is open source and we can use it freely.

3. It Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.
4. It provides various built-in functions for creating different types of plots. Some of the key functions include :
   i. matplotlib.pyplot.plot(): Used to create line plots.
   ii. matplotlib.pyplot.scatter(): Creates scatter plots.
   iii. matplotlib.pyplot.bar() and matplotlib.pyplot.barh(): Used to create vertical and horizontal bar plots respectively.
   iv. matplotlib.pyplot.hist(): Generates histograms.
   v. matplotlib.pyplot.pie(): Produces pie charts.
   vi. matplotlib.pyplot.imshow(): Displays images.
   vii. matplotlib.pyplot.plot_date(): Plots dates.
   viii. matplotlib.pyplot.text(): Adds text to the plot.
   ix. matplotlib.pyplot.grid(): Adds a grid to the plot.
   x. matplotlib.pyplot.axis(): Sets the axis properties.

Que 5.9. Write a program to plot a simple line graph using matplotlib.

Answer

First import Matplotlib.pyplot library for plotting functions. Also, import the Numpy library as per requirement. Then define data values x and y.

```
# importing the required libraries
import matplotlib.pyplot as plt
import numpy as np
# define data values
x = np.array([1, 2, 3, 4]) # X-axis points
y = x*2 # Y-axis points
plt.plot(x, y) # Plot the chart
plt.show() # display
```

Output :

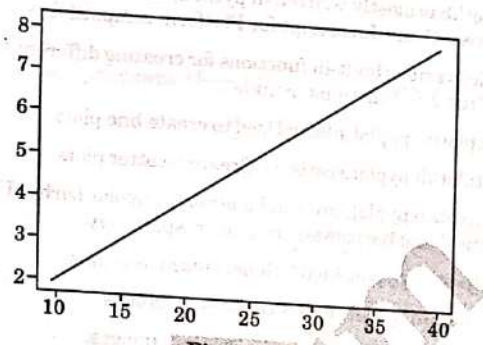


Fig. 5.9.1.

Simple line plot between X and Y data

Que 5.10. **Create a bar chart using Matplotlib to represent data.**

Answer

With Pyplot, you can use the bar() function to draw bar graphs :

Example

Draw 4 bars :

```
import matplotlib.pyplot as plt
import numpy as np
x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])
plt.bar(x,y)
plt.show()
```

Result :

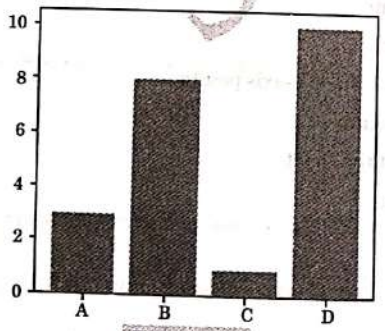


Fig. 5.10.1.

Que 5.11. **Create a histogram using Matplotlib.**

Answer

In Matplotlib, we use the hist() function to create histograms.

The hist() function will use an array of numbers to create a histogram, the array is sent into the function as an argument.

For simplicity we use NumPy to randomly generate an array with 250 values, where the values will concentrate around 170, and the standard deviation is 10.

Example :

A simple histogram :

```
import matplotlib.pyplot as plt
import numpy as np
x = np.random.normal(170, 10, 250)
plt.hist(x)
plt.show()
```

Result :

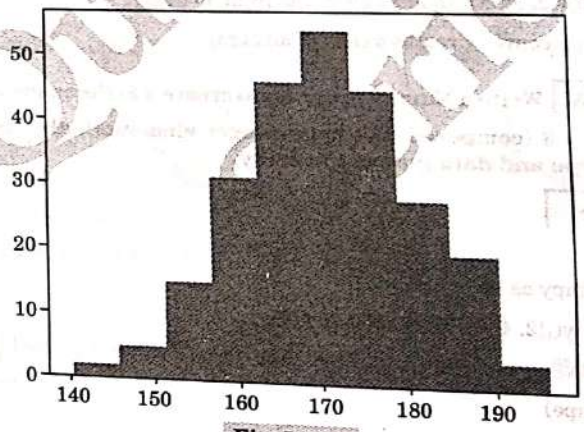


Fig. 5.11.1.

Que 5.12. **Define Numpy. List some built-in functions in Numpy.**

Answer

Numpy is a powerful library in Python for numerical computations. It provides a high-performance multidimensional array object and tools for working with these arrays. Here are some of the built-in functions in Numpy :

1. numpy.array(): Creates a new ndarray (n-dimensional array).
2. numpy.transpose(): Transposes the array.
3. numpy.sum(): Calculates the sum of array elements.
4. numpy.mean(): Calculates the mean of array elements.
5. numpy.max(): Returns the maximum value in an array.
6. numpy.min(): Returns the minimum value in an array.
7. numpy.std(): Computes the standard deviation of array elements.
8. numpy.var(): Computes the variance of array elements.
9. numpy.dot(): Computes the dot product of two arrays.
10. numpy.concatenate(): Joins a sequence of arrays along an existing axis.
11. numpy.split(): Splits an array into multiple sub-arrays.
12. numpy.append(): Appends values to the end of an array.
13. numpy.delete(): Deletes elements from an array.
14. numpy.copy(): Creates a copy of an array.

Que 5.13. Write a Numpy program to create a 2-dimensional array of size 2 × 3 (composed of 4 byte integer elements), also print the shape, type and data type of the array.

Answer

Code :

```
import numpy as np
x = np.array([[2, 4, 6], [6, 8, 10]], np.int32)
print(type(x))
print(x.shape)
print(x.dtype)
```

Output :

```
<class 'numpy.ndarray'>
(2, 3)
int32
```


Que 5.14. Compute the mean, standard deviation, and variance of a given Numpy array.

Answer

Code :

```
import numpy as np
# Original array
array = np.arange(10)
print(array)
r1 = np.mean(array)
print("\nMean: ", r1)
r2 = np.std(array)
print("\nstd:", r2)
r3 = np.var(array)
print("\nvariance:", r3)
```

Que 5.15. How do you concatenate two numpy arrays ?

Answer

```
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
arr = np.concatenate((arr1, arr2))
print(arr)
```

Que 5.16. Define Pandas. List some built-in functions in Pandas.

Answer

Pandas is a popular Python library for data manipulation and analysis. It provides data structures like Series and DataFrame, which allow for efficient handling of structured data.

Here are some built-in functions commonly used in pandas :

1. pd.DataFrame(): Creates a pandas DataFrame, which is a two-dimensional labeled data structure.
2. df.head(): Returns the first n rows of a DataFrame (default is 5).
3. df.tail(): Returns the last n rows of a DataFrame (default is 5).
4. df.info(): Provides a concise summary of a DataFrame, including column data types and non-null counts.
5. df.columns: Returns the column labels of the DataFrame.
6. df.index: Returns the row labels of the DataFrame.
7. df['column_name']: Retrieves a specific column from the DataFrame.
8. df[['column1', 'column2']]: Retrieves multiple columns from the DataFrame.
9. df.iloc[]: Allows for selection of specific rows and columns by integer location.
10. df.loc[]: Allows for selection of specific rows and columns by label.
11. df.groupby(): Groups the DataFrame using a particular column.
12. df.merge(): Merges two DataFrames on a specific column.

Que 5.17. Define Pandas DataFrame. How do you create a Pandas DataFrame.

Answer

Pandas DataFrame is a two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). Pandas DataFrame consists of three principal components, the data, rows, and columns.

Creating a Pandas DataFrame :

Dataframe can be created in different ways. Following are some ways by which we create a dataframe :

Example 1 : DataFrame can be created using a single list or a list of lists.

```
# import pandas as pd
import pandas as pd
# list of strings
lst = ['Quantum Series', 'For', 'Engineers', 'is',
    'help book', 'for', 'Engineers']
```



```
# Calling DataFrame constructor on list
df = pd.DataFrame(lst)
display(df)
```

Example 2 : Creating DataFrame from dict of ndarray/lists.

To create DataFrame from dict of narray/list, all the narray must be of same length. If index is passed then the length index should be equal to the length of arrays. If no index is passed, then by default, index will be range(n) where n is the array length.

```
# Python code demonstrate creating
# DataFrame from dict narray / lists
# By default addresses.
import pandas as pd
# initialise data of lists.
data = {'Name':['Tom', 'nick', 'krish', 'jack'],
    'Age':[20, 21, 19, 18]}
# Create DataFrame
df = pd.DataFrame(data)
# Print the output.
display(df)
```

Que 5.18. How do you select rows and columns from Pandas DataFrame ?

Answer

Example 1: Selecting rows.

pandas.DataFrame.loc is a function used to select rows from Pandas DataFrame based on the condition provided.

Syntax: df.loc[df['cname'] 'condition']

Parameters :

- df: represents data frame
- cname: represents column name
- condition: represents condition on which rows has to be selected

```
# Importing pandas as pd
```

```
from pandas import DataFrame
# Creating a data frame
Data = {'Name': ['Mohe', 'Shyni', 'Parul', 'Sam'],
    'ID': [12, 43, 54, 32],
    'Place': ['Delhi', 'Kochi', 'Pune', 'Patna']
    }
df = DataFrame(Data, columns = ['Name', 'ID', 'Place'])
# Print original data frame
print("Original data frame:\n")
display(df)
# Selecting the product of Electronic Type
select_prod = df.loc[df['Name'] == 'Mohe']
print("\n")
# Print selected rows based on the condition
print("Selecting rows:\n")
display (select_prod)
```

Output :

Original data frame :

	Name	ID	Place
0	Mohe	12	Delhi
1	Shyni	43	Kochi
2	Parul	54	Pune
3	Sam	32	Patna

Selecting row :

	Name	ID	Place
0	Mohe	12	Delhi

Example 2 : Selecting column.

```
# Importing pandas as pd
from pandas import DataFrame
```



```
# Creating a data frame
Data = {'Name': ['Mohe', 'Shyni', 'Parul', 'Sam'],
    'ID': [12, 43, 54, 32],
    'Place': ['Delhi', 'Kochi', 'Pune', 'Patna']
    }
df = DataFrame(Data, columns = ['Name', 'ID', 'Place'])
# Print original data frame
print("Original data frame:")
display(df)
print("Selected column: ")
display(df[['Name', 'ID']] )
```

Output :

Original data frame :

	Name	ID	Place
0	Mohe	12	Delhi
1	Shyni	43	Kochi
2	Parul	54	Pune
3	Sam	32	Patna

Selected column :

	Name	ID
0	Mohe	12
1	Shyni	43
2	Parul	54
3	Sam	32

PART-2

GIU Programming : Tkinter Introduction, Tkinter and Python Programming, Tk Widgets, Tkinter Examples.

Que 5.19. What is GUI in Python with example ?

Answer

1. GUI stands for Graphical User Interface.
2. It refers to the visual components like windows, buttons, menus, etc., that enable users to interact with a computer program.
3. In Python, you can create GUI applications using libraries like Tkinter, PyQt, Kivy, and more. Here's a basic example using Tkinter to create a simple GUI window:

```
import tkinter as tk
# Create the main window
root = tk.Tk()
root.title("Simple GUI Example")
# Create a label widget
label = tk.Label(root, text="Hello, GUI World!")
label.pack(pady=10)
# Run the GUI application
root.mainloop()
```

4. In this code, Tkinter is used to create a window with a label displaying the text "Hello, GUI World!".

Que 5.20. How to write a Python program using GUI ?

Answer

Python provides various options for developing graphical user interfaces (GUIs). The most important features are listed below :

- **Tkinter :** Tkinter is the Python interface to the Tk GUI toolkit shipped with Python.
- **wxPython :** This is an open-source Python interface for wxWidgets GUI toolkit.
- **PyQt :** This is also a Python interface for a popular cross-platform Qt GUI library.
- **PyGTK :** PyGTK is a set of wrappers written in Python and C for GTK + GUI library.

- **PySimpleGUI :** PySimpleGui is an open source, cross-platform GUI library for Python. It aims to provide a uniform API for creating desktop GUIs based on Python's Tkinter, PySide and WxPython toolkits.
- **Pygame :** Pygame is a popular Python library used for developing video games. It is free, open source and cross-platform wrapper around Simple DirectMedia Library (SDL).
- **Jython :** Jython is a Python port for Java, which gives Python scripts seamless access to the Java class libraries on the local machine.

Que 5.21. What is Tkinter and why it is used in Python programming ?

Answer

1. Tkinter is a Python library that provides a set of tools for creating graphical user interfaces (GUIs).
2. It's a standard Python interface to the Tk GUI toolkit and is included with most Python installations.
3. Tkinter allows you to create windows, dialogs, buttons, labels, and other GUI elements, making it easier to build interactive applications with a graphical interface.
4. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.
5. The tkinter package includes following modules :
   - **Tkinter :** Main Tkinter module.
   - **tkinter.colorchooser :** Dialog to let the user choose a color.
   - **tkinter.commondialog :** Base class for the dialogs defined in the other modules listed here.
   - **tkinter.filedialog :** Common dialogs to allow the user to specify a file to open or save.
   - **tkinter.font :** Utilities to help work with fonts.
   - **tkinter.messagebox :** Access to standard Tk dialog boxes.
   - **tkinter.scrolledtext :** Text widget with a vertical scroll bar built in.
   - **tkinter.simpledialog :** Basic dialogs and convenience functions.

- **tkinter.ttk :** Themed widget set introduced in Tk 8.5, providing modern alternatives for many of the classic widgets in the main tkinter module.

Que 5.22. How do you create basic window using Tkinter ?

Answer

Creating a GUI using tkinter is an easy task.

To create a tkinter app :

1. Importing the module - tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets.

Example :

```
# note that module name has changed from Tkinter in Python 2
# to tkinter in Python 3
import tkinter
top = tkinter.Tk()
# Code to add widgets will go here...
top.mainloop()
```

Que 5.23. What are widgets in Tkinter and can you give example of some commonly used widgets ?

Answer

1. In general, Widget is an element of Graphical User Interface (GUI) that displays/illustrates information or gives a way for the user to interact with the OS.
2. In Tkinter, Widgets are objects; instances of classes that represent buttons, frames, and so on.
3. There are currently 15 types of widgets in Tkinter.
4. Some of the major widgets are explained below :

i. **Button :** To add a button in your application, this widget is used.

The general syntax is :


```
w=Button(master, option=value)
```

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the Buttons. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- activebackground: to set the background color when button is under the cursor.
- activeforeground: to set the foreground color when button is under the cursor.
- bg: to set the normal background color.
- command: to call a function.
- font: to set the font on the button label.
- image: to set the image on the button.
- width: to set the width of the button.
- height: to set the height of the button.

```
import tkinter as tk
r = tk.Tk()
r.title('Counting Seconds')
button = tk.Button(r, text='Stop', width=25, command=r.destroy)
button.pack()
r.mainloop()
```

ii. **Canvas :** It is used to draw pictures and other complex layout like graphics, text and widgets.

The general syntax is :

```
w = Canvas(master, option=value)
```

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below :

- bd: to set the border width in pixels.
- bg: to set the normal background color.
- cursor: to set the cursor used in the canvas.

- highlightcolor: to set the color shown in the focus highlight.
- width: to set the width of the widget.
- height: to set the height of the widget.

```
from tkinter import *
master = Tk()
w = Canvas(master, width=40, height=60)
w.pack()
canvas_height=20
canvas_width=200
y = int(canvas_height / 2)
w.create_line(0, y, canvas_width, y )
mainloop()
```

iii. **CheckButton :** To select any number of options by displaying a number of options to a user as toggle buttons. The general syntax is :

```
w = CheckButton(master, option=value)
```

There are number of options which are used to change the format of this widget. Number of options can be passed as parameters separated by commas. Some of them are listed below :

- Title: To set the title of the widget.
- activebackground: to set the background color when widget is under the cursor.
- activeforeground: to set the foreground color when widget is under the cursor.
- bg: to set the normal background color.
- command: to call a function.
- font: to set the font on the button label.
- image: to set the image on the widget.

```
from tkinter import *
master = Tk()
var1 = IntVar()
```



```
Checkbutton(master, text='male', variable=var1).grid(row=0, sticky=W)
var2 = IntVar()
Checkbutton(master, text='female', variable=var2).grid(row=1, sticky=W)
mainloop()
```

Que 5.24. **Write a program to print "Hello World" using Tkinter.**

Answer

```
# Python tkinter hello world program
from tkinter import *
root = Tk()
a = Label(root, text ="Hello World")
a.pack()
root.mainloop()
```

PART-3

Python Programming with IDE.

Que 5.25. **What is Python ? How Python is interpreted ? What are the tools that help to find bugs or perform static analysis ? What are Python decorators ?** **AKTU 2019-20, Sem-3, Marks 10**

OR

Explain why Python is considered an interpreted language. **AKTU 2020-21, Sem-3, Marks 10** **AKTU 2021-22, Sem-3, Marks 10**

OR

Discuss why python is interpreted language. Explain history and features of python while comparing python version 2 and 3. **AKTU 2022-23, Sem-4, Marks 10**

Answer

History and features of Python while comparing Python version 2 and 3 : Out of syllabus from session (2023-24).

Python : Python is a high-level, interpreted, interactive and object-oriented scripting language. It is a highly readable language. Unlike other programming languages, Python provides an interactive mode similar to that of a calculator.

Interpretation of Python :

1. An interpreter is a kind of program that executes other programs.
2. When we write Python programs, it converts source code written by the developer into intermediate language which is again translated into the machine language that is executed.
3. The python code we write is compiled into python bytecode, which creates file with extension .pyc.
4. The bytecode compilation happened internally and almost completely hidden from developer.
5. Compilation is simply a translation step, and byte code is a lower-level, and platform-independent, representation of source code.
6. Each of the source statements is translated into a group of bytecode instructions. This bytecode translation is performed to speed execution. Bytecode can be run much quicker than the original source code statements.
7. The .pyc file, created in compilation step, is then executed by appropriate virtual machines.
8. The Virtual Machine iterates through bytecode instructions, one by one, to carry out their operations.
9. The Virtual Machine is the runtime engine of Python and it is always present as part of the Python system, and is the component that actually runs the Python scripts.
10. It is the last step of Python interpreter.

Following tools are the static analysis tools that help to find bugs in Python :

1. **Pychecker :** Pychecker is an open source tool for static analysis that detects the bugs from source code and warns about the style and complexity of the bug.

2. **Pylint :**
   a. Pylint is highly configurable and it acts like special programs to control warnings and errors, it is an extensive configuration file.
   b. It is an open source tool for static code analysis and it looks for programming errors and is used for coding standard.
   c. It also integrates with Python IDEs such as Pycharm, Spyder, Eclipse, and Jupyter.

Python decorators :

1. Decorators are very powerful and useful tool in Python since it allows programmers to modify the behavior of function or class.
2. Decorators allow us to wrap another function in order to extend the behavior of wrapped function, without permanently modifying it.
3. In decorators, functions are taken as the argument into another function and then called inside the wrapper function.
4. **Syntax :**

```
@gfg_decorator
def hello_decorator():
    print("Gfg")
```

5. gfg_decorator is a callable function, will add some code on the top of some another callable function, hello_decorator function and return the wrapper function.

Que 5.26. What is IDE ? Discuss some Python IDE.

OR

What do you mean by Python IDE ? Explain in detail.

AKTU 2021-22, Sem-3, Marks 10

Answer

1. IDE is a software package that consists of several tools for developing and testing the software.
2. An IDE helps the developer by automating the process.
3. IDEs integrate many tools that are designed for SDLC.
4. IDEs were introduced to diminish the coding and typing errors.
5. Some of the Python IDEs are :

a. **PyCharm :** PyCharm assists the developers to be more productive and provides smart suggestions. It saves time by taking care of routine tasks, hence increases productivity.

Features of PyCharm :

i. It has smart code navigation, good code editor, a function for quick refactoring.

ii. The integrated activities with PyCharm are profiling, testing, debugging, remote development, and deployments.

iii. PyCharm supports Python web development frameworks, Angular JS, JavaScript, CSS, HTML and live editing functions.

b. **Spyder :** Spyder is widely used for data science works. It is mostly used to create a secure and scientific environment for Python. Spyder Python uses PyQt (Python plug-in) which a developer can add as an extension.

Features of Spyder :

i. It has good syntax highlighting and auto code completion features.

ii. Spyder Python explores and edits variables directly from GUI.

iii. It performs very well in multi-language editor.

c. **PyDev :** It is an external plug-in for Eclipse and is very popular as Python interpreter.

Features of PyDev :

i. PyDev has strong parameters like refactoring, debugging, type hinting, code analysis, and code coverage function.

ii. PyDev supports tokens browser, PyLint integration, interactive console, remote debugger, Unittest integration, etc.

d. **IDLE :** IDLE is a basic IDE mainly used by beginner level developer.

i. IDLE Python is a cross-platform IDE, hence it increases the flexibility for users.

ii. It is developed only in Python in collaboration with Tkinter GUI toolkit.

iii. The feature of multi-window text editor in IDLE has some great functions like smart indentation, call tips, Python colorizing, and undo option.

iv. It also comes with a strong debugger along with continuous breakpoints, local spaces, and global view.

v. It supports browsers, editable configurations, and dialog boxes.

e. **Visual studio :** It enables development for various platforms and has its own marketplace for extensions.

Features of visual studio :

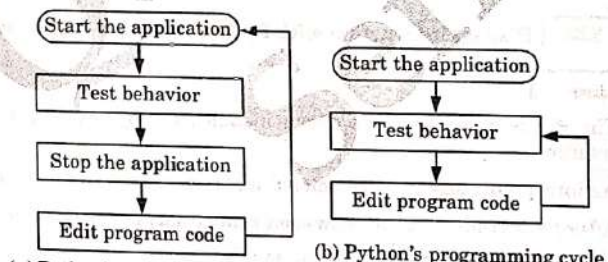
i. It supports Python coding in visual studio, debugging, and other activities.

ii. It has both paid and free versions in the market with great features.

Que 5.27. Explain the programming cycle for Python.

Answer

1. Python's programming cycle is dramatically shorter than that of traditional programming cycle.
2. In Python, there are no compile or link steps.
3. Python programs simply import modules at runtime and use the objects they contain. Because of this, Python programs run immediately after changes are made.
4. In cases where dynamic module reloading can be used, it is even possible to change and reload parts of a running program without stopping it at all.
5. Fig. 5.27.1 shows Python's impact on the programming cycle.



(a) Python's programming cycle (b) Python's programming cycle with module reloading

Fig. 5.27.1.

6. Since Python is interpreted, there is a rapid turnaround after program changes. And because Python's parser is embedded in Python-based systems, it is easy to modify programs at runtime.

Que 5.28. **Write short notes with example : The programming cycle for Python, elements of Python, type conversion in Python, operator precedence, and Boolean expression.**

AKTU 2019-20, Sem-3, Marks 10

Answer

Programming cycle for Python : Refer Q. 5.27, Page 5–26V, Unit-5.

Elements of Python : Refer Q. 1.17, Page 1–14V, Refer Q. 1.18, Page 1–15V, and Refer Q. 1.19, Page 1–15V; Unit-1.

Type conversion in Python : Refer Q. 1.20, Page 1–16V, Unit-1.

Operator precedence : Refer Q. 1.12, Page 1–10V, Unit-1.

Boolean expression : Refer Q. 1.21, Page 1–18V, Unit-1.

Que 5.29. **Write short notes on :**

i. The programming cycle for Python.

ii. Type conversion in Python. **AKTU 2021-22, Sem-4, Marks 10**

Answer

i. The programming cycle for Python : Refer Q. 5.27, Page 5–26V, Unit-5.

ii. Type conversion in Python : Refer Q. 1.20, Page 1–16V, Unit-1.

Que 5.30. **Discuss interaction with Python program with example.**

Answer

1. The Python program that we have installed will by default act as an interpreter.
2. An interpreter takes text commands and runs them as we enter text.
3. After Python opens, it will show some contextual information like this :

   Python 3.5.0 (default, dec 20 2019, 11 : 28 : 25)

   [GCC 5.2.0] on Linux

   Type "help", "copyright", "credits" or "license" for more information

   >>>
4. The prompt >>> defines that we are now in an interactive Python interpreter session, also called the Python shell.

5. Now enter some code for Python to run such as print("Hello World!") and press the Enter key.
6. The interpreter's response should appear on the next line like this :

   >>> print ("Hello World!")

   Hello World!
7. After showing the results, Python will bring you back to the interactive prompt, where we could enter another command or code.
8. Python program communicates its result to user using print statement.

Que 5.31. **What do you mean by comments in Python ?**

Answer

Comments :

1. Python allows us to add comments in the code.
2. Comments are used by the programmer to explain the piece of code to be understood by other programmer in a simple language. Every programming language makes use of some character for commenting.
3. Python uses the hash character (#) for comments. Putting # before a text ensures that the text will not be parsed by the interpreter.
4. Comments do not affect the programming part and the Python interpreter does not display any error message for comments.

   For example : Commenting using hash mark (#)

```
>>> 8 + 9          # addition
17                 # Output
>>>
```

   In this example, 'addition' is written with a hash mark. Hence the interpreter understands it as a comment and does not display any more messages.

Que 5.32. **How memory is managed in Python ? Explain PEP 8. Write a Python program to print even length words in a string.**

AKTU 2019-20, Sem-3, Marks 10

Answer

Memory management :

1. Memory management in Python involves a private heap containing all Python objects and data structures.
2. The management of this private heap is ensured internally by the Python memory manager.
3. The Python memory manager has different components which deal with various dynamic storage management aspects, like sharing, segmentation, preallocation or caching.
4. At the lowest level, a raw memory allocator ensures that there is enough room in the private heap for storing all Python-related data by interacting with the memory manager of the operating system.
5. On top of the raw memory allocator, several object-specific allocators operate on the same heap and implement distinct memory management policies adapted to the peculiarities of every object type.
6. For example, integer objects are managed differently within the heap than strings, tuples or dictionaries because integers imply different storage requirements and speed/space tradeoffs.
7. Python memory manager thus delegates some of the work to the object-specific allocators, but ensures that the latter operate within the bounds of the private heap.

PEP 8 :

1. A PEP is a design document providing information to the Python community, or describing a new feature for Python or its processes or environment.
2. The PEP should provide a concise technical specification of the feature.
3. PEP is actually an acronym that stands for Python Enhancement Proposal.
4. PEP 8 is Python's style guide. It is a set of rules for how to format the Python code to maximize its readability.
5. A PEP is a design document providing information to the Python community, or describing a new feature for Python or its processes or environment.

Program to print even length words in a string :

```
def printWords(s) :
```



```
    # split the string
    s = s.split(' ')
    # iterate in words of string
    for word in s:
    # if length is even
    if len(word)%2==0:
        print(word)
# Driver Code
s = "i am quantum"
printWords(s)
```

1 UNIT

Introduction to Python (2 Marks Questions)

1.1. What is Python ?

Ans. Python is a high-level, interpreted, interactive and object-oriented scripting language. It is a highly readable language. Unlike other programming languages, Python provides an interactive mode similar to that of a calculator.

1.2. What are the difference between Java and Python ?

Ans.

Features	Java	Python
Syntax	The syntax of Java is complex than Python.	The syntax of Python is easier than Java.
Speed	Java is statically typed programming, makes it faster.	Python is manually typed, makes it slower.
Code	Longer lines of code than Python.	Shorter lines of code than Java.

1.3. What are the features of Python ?

Ans. **Features of Python :**

1. The code written in Python is automatically compiled to byte code and executed.
2. Python can be used as a scripting language, as a language for implementing web applications, etc.
3. Python supports many features such as nested code blocks, functions, classes, modules and packages.
4. Python makes use of an object oriented programming approach.
5. It has many built-in data types such as strings, lists, tuples, dictionaries, etc.

1.4. What are the different ways of starting Python ?

Ans. There are three different ways of starting Python :

1. Running a script written in Python.

2. Using a graphical user interface (GUI) from an Integrated Development Environment (IDE),
3. Employing an interactive approach.

1.5. Which character is used for commenting in Python ?

Ans. Hash mark (#) is used for commenting in Python.

1.6. How is Python an interpreted language ?

AKTU 2019-20, Marks 02

Ans. Python is called an interpreted language because it goes through an interpreter, which turns the Python code into the language understood by processor of the computer.

1.7. What type of language is Python ?

AKTU 2019-20, Marks 02

Ans. Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.

1.8. Define the type () function.

Ans. Type () function in Python programming language is a built-in function which return the datatype of any arbitrary object. The object is passed as an argument to the type() function. Type() function can take anything as an argument and return its datatype, such as integers, strings, dictionaries, lists, classes, module, tuple, functions, etc.

1.9. Define the unary operator.

Ans. Unary operators are operators with only one operand. These operators are basically used to provide sign to the operand.
+, –, ~ are some unary operators.

1.10. What do you mean by binary operator ?

Ans. Binary operators are operators with two operands that are manipulated to get the result. They are also used to compare numeric values and string values.
**, %, <<, >>, &, |, ^, <, >, <=, >=, ==, !=, <> are some binary operators.

1.11. What is the purpose of PYTHONPATH environment variable ?

Ans. Pythonpath has a role similar to PATH. This variable tells Python Interpreter where to locate the module files imported into a program.

1.12. Can we make multiline comments in Python ?

Ans: Python does not have a specific syntax for including multiline comments like other programming languages. However, programmers can use triple-quoted strings (docstrings) for making multiline comments.

1.13. Do we need to declare variables with data types in Python ?

Ans: No, Python is a dynamically typed language, *i.e.*, Python Interpreter automatically identifies the data type of a variable based on the type of value assigned to the variable.

1.14. In some languages, every statement ends with a semi-colon (;). What happens if you put a semi-colon at the end of a Python statement ? **AKTU 2019-20, Sem-3; Marks 02**

Ans: Python allows semicolon to use as a line terminator. So no error will occur if we put a semi-colon; at the end of python statement.

1.15. Mention five benefits of using Python.

AKTU 2019-20, Sem-3; Marks 02

Ans: **Benefits of Python :**

1. Python is easy to learn.
2. Most automation, data mining, and big data platforms depend on Python. This is because it is the ideal language to work with for general purpose tasks.
3. Python provides productive coding environment.
4. It supports extensive libraries.

1.16. Define floor division with example.

AKTU 2019-20, Sem-3; Marks 02

AKTU 2022-23, Sem-3; Marks 02

Ans: Floor division returns the quotient in which the digits after the decimal point are removed. But if one of the operands (dividend and divisor) is negative, then the result is floored, i.e., rounded away from zero (means, towards the negative of infinity). It is denoted by "//".

For example :

```
5.0 // 2
2.0
```


1.17. Mention some of the reserved keyword in Python.

Ans:

1. and
2. false
3. is
4. pass
5. return
6. def

1.18. Give an example of assigning one variable value to another.

Ans:

```
>>> name1 = 'Albert'
>>> name2 = name1
>>> name2
'Albert' # Output
>>>
```

1.19. Give an example of different types of values to the same variable.

Ans:

```
>>> amount = 50
>>> amount
50 # Output
>>> amount = 'Fifty'
>>> amount
'Fifty' # Output
>>>
```

1.20. What are the types of type conversion ?

Ans: **Two types of type conversion are :**

1. Implicit type conversion
2. Explicit type conversion

1.21. List the categories of operators.

Ans: **Following are the seven categories of operators :**

1. Arithmetic operators.
2. Assignment operators.
3. Bitwise operators.
4. Comparison operators.
5. Identity operators.
6. Logical operators.
7. Membership operators.

1.22. What are the different data types used in Python ?

Ans: **Python has six basic data types which are as follows :**

1. Numeric
2. String

3. List
4. Tuple
5. Dictionary
6. Boolean

1.23. Define operator associativity with its type.

Ans.

1. Associativity decides the order in which the operators with same precedence are executed.
2. There are two types of associativity :
   a. **Left to right :** In left to right associativity, the operator of same precedence are executed from the left side first.
   b. **Right to left :** In right to left associativity, the operator of same precedence are executed from the right side first.

1.24. What are rules for naming variables in Python ?

Ans. **Rules for Python variables :**

1. A variable name must start with a letter or the underscore character
2. A variable name cannot start with a number
3. A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
4. Variable names are case-sensitive (age, Age and AGE are three different variables)
5. A variable name cannot be any of the Python keywords.

1.25. What is role of (:) in Python blocks ?

Ans.

1. In Python, the colon (:) is used to indicate the start of an indented block of code. This is particularly important in control flow statements (like if-else, loops, functions, etc.) and when defining classes or functions.
2. It helps Python understand the structure and organization of your program.

1.26. Explain the role of precedence with an example.

AKTU 2020-21, Sem-3; Marks 02

Ans. Precedence guides the order in which the operations are carried out Operator precedence affects how an expression is evaluated. For example, $x = 7 + 3 * 2$; here, x is assigned 13, not 20 because operator * has higher precedence than +, so it first multiplies $3 * 2$ and then adds into 7.

1.27. How do you read an input from a user in Python to be used as an integer in the rest of the program ? Explain with an example.

AKTU 2020-21, Sem-3; Marks 02

Ans. We use int() function explicitly to read inputs from the user into integers.

For example :

```
a = int(input("Enter a number: "))
```

1.28. Explain the use of __lt__ function in a class Python ?

AKTU 2020-21, Sem-3; Marks 02

Ans. '__lt__' is a magic methods in Python used to define overloaded behaviour of operators. It overload the 'less than' comparison operators.

☺☺☺

2 UNIT

Python Program Flow Control Conditionals Blocks (2 Marks Questions)

2.1. What is range () function ?

Ans. The range () function is a built-in function in Python that helps us to iterate over a sequence of numbers. It produces an iterator that follows arithmetic progression.

2.2. Give an example of range () function.

Ans. >>> range (8)
[0, 1, 2, 3, 4, 5, 6, 7]
range (8) provides a sequence of number 0-7. That is to say range (n) generates a sequence of number that starts with 0 and end with $(n-1)$.

2.3. Explain begin and end arguments passed by range () function.

Ans. >>> range (3, 9)
[3, 4, 5, 6, 7, 8]
We provided the begin index with 3 and the end index with 9. Hence, the range function generates a sequence iterator of number that starts from 3 and ends at 8.

2.4. Define the term alternative execution.

Ans. The alternative execution provides two possibilities and the condition determines which one is to be executed. This is the second form of the if statement.

2.5. What do you mean by block ?

Ans. The intended statements that follow the conditional statements are called block. The first unintended statement marks the end of the block.

2.6. What will be the output of the following code :
Str [0 : 4] if str="Hello"

Ans. 'Hello'

2.7. What are control statements ?

Ans. A control statement is a statement that determines the control flow of a set of instructions. There are three fundamental forms of control that programming languages provide: sequential control, selection control, and iterative control.

2.8. What is short-circuit evaluation ?

Ans. In short-circuit (lazy) evaluation, the second operand of Boolean operators AND and OR is not evaluated if the value of the Boolean expression can be determined from the first operand alone.

2.9. Define the terms : header, suite and clause.

Ans. A header in Python starts with a keyword and ends with a colon. The group of statements following a header is called a suite. A header and its associated suite are together referred to as a clause.

2.10. What do you mean by iterative control ?

Ans. An iterative control statement is a control statement providing the repeated execution of a set of instructions. An iterative control structure is a set of instructions and the iterative control statement(s) controlling their execution.

2.11. What do you mean by definite loop ?

Ans. A definite loop is a program loop in which the number of times the loop will iterate can be determined before the loop is executed.

2.12. What do you mean by indefinite loop ?

Ans. An indefinite loop is a program loop in which the number of times that the loop will iterate cannot be determined before the loop is executed.

2.13. Is indentation optional in Python ?

Ans. No indentation in Python is compulsory and is part of its syntax. Indentation is a way of defining the scope and extent of the block of codes. Indentation provides better readability to the code.

2.14. What happen if break statement is used in for loop ?

Ans. If the break statement in a for loop is executed then the else part of that for loop is skipped.

2.15. What is raw_input ( ) function ?

Ans. Raw_input ( ) takes the input from the user but it does not interpret the input and also it returns the input of the user without doing any changes.

2.16. Differentiate fruitful functions and void functions.

AKTU 2019-20, Sem-3; Marks 02

Ans. The main difference between void and fruitful function in python is :

1. Void does not return any value
2. Fruitful function returns some value

2.17. What will be the output of the following Python code ?

```
i = 0
while i< 3 :
    print(i)
i+= 1
else:
print(0)
```

AKTU 2021-22, Sem-3; Marks 02

Ans. Syntax error at else.

Correction in question :

```
i = 0
while i < 3:
print(i)
i = i + 1
else:
print(0)
```

Output :

```
1
2
0
```

2.18. Write a for loop that prints numbers from 0 to 57, using range function.

AKTU 2021-22, Sem-4; Marks 02

Ans.

```
for i in range(58):
print(i)
```

2.19. What happens if you omit the starting value in the range() function within a for loop ?

Ans.

1. Syntax :
   range(start, stop, step)
2. If we don't specify the start index, the default start index of 0 is used and if we don't specify the step value, the default step size of 1 is used.

2.20. What are some common use cases for using a for loop with ranges in Python ?

Ans. The range() function can be applied in numerous scenarios to facilitate iteration through number sequences within a for loop. Let's explore some common use cases :

1. Counting up or down.
2. Iterating over elements in a list.
3. We can use the range() function to create nested loops.

2.21. Is it possible to use a for loop with a decreasing range of numbers ? If so, how ?

Ans.

1. Yes, it's possible to use a for loop with a decreasing range of numbers. In many programming languages, you can achieve this by using the range() function with three arguments: start, stop, and step.
2. Example :

```
for i in range(10, 0, -1):
print(i)
```

In this example, the loop will start at 10, and decrement by 1 in each iteration until it reaches 1.

2.22. Discuss the purpose of the break and continue statement in loops.

AKTU 2022-23, Sem-4; Marks 02

Ans. **Break :** Refer Q. 2.28, Page 2–18V, Unit-2.
Continue : Refer Q. 2.29, Page 2–19V, Unit-2.

2.23. Describe the behavior of "range (s, e)" in Python.

AKTU 2020-21, Sem-3; Marks 02

Ans. Refer Q. 2.22, Page 2–15V, Unit-2.

☺☺☺

3 UNIT

Python Complex Data Types (2 Marks Questions)

3.1. Define traversing of string. Give an example.

Ans. Traversal is a process in which we access all the elements of the string one by one using some conditional statements such as for loop, while loop, etc.

For example :

```
>>> var = 'jack john'
>>> i = 0
>>> while i < len (var) :
...     x = var [i]
...     print x
...     i = i + 1
```

3.2. What are escape characters ?

Ans. The backslash character (/) is used to escape characters. It converts difficult-to-type characters into a string. For example, we need the escaping character concept when we want to print a string with double quotes or single quotes. When single or double quotes are used with the string, Python normally neglects them and prints only the string.

3.3. What do you mean by tuple assignment ?

Ans. Tuple assignment allows the assignment of values to a tuple of variables on the left side of assignment from the tuple of values on the right side of the assignment.

3.4. What do you mean by "Lists are mutable" ?

Ans. Lists are mutable means that we can change the value of any elements inside the list at any point of time. The elements inside the list are accessible with their index value. The index will always start with 0 and end with $n - 1$, if the list contains n elements.

3.5. What do you understand by traversing a list ?

Ans. Traversing of the list refers to accessing all the elements or items of the list. Traversing can be done using any conditional statement of Python, but it is preferable to use for loop.

3.6. What are the different methods used in deleting elements from dictionary ?

Ans. **Methods used in deleting elements from dictionary are :**

1. **pop( ) :** pop() method removes that item from the dictionary for which the key is provided. It also returns the value of the item.
2. **popitem( ) :** popitem() method is used to remove or delete and return an arbitrary item from the dictionary.
3. **clear( ) :** clear() method removes all the items or elements from a dictionary at the same time.

3.7. What are the two properties of key in the dictionary ?

Ans. **Properties of key :**

1. One key in a dictionary cannot have two values, *i.e.*, duplicate keys are not allowed in the dictionary; they must be unique.
2. Keys are immutable, *i.e.*, we can use string, integers or tuples for dictionary keys, but cannot use something like ['key'].

3.8. Why we use functions ?

Ans.

1. Break up complex problem into small sub-programs.
2. Solve each of the sub-problems separately as a function, and combine them together in another function.
3. Hide the details and shows the functionality.

3.9. What are mathematical functions ? How are they used in Python ?

Ans. Python provides us a math module containing most of the familiar and important mathematical functions. A module is a file that contain some predefine Python codes. A module can define functions, classes and variables. It is a collection of related functions grouped together.

Before using a module in Python, we have to import it

For example, to import the math module, we use :

```
>>> import math
```

3.10. What are user-defined functions ? Give the syntax.

Ans. Python also allows users to define their own functions. To use their own functions in Python, users have to define the functions first; this is known as function definition. In a function definition, users have to define a name for the new function and also the list of the statements that will execute when the function will be called.

Syntax :

```
def functionname (parameters) :
    "function_docstring"
    statement (s)
    return (expression)
```

3.11. Define the return statement in a function. Give the syntax.

Ans: The return statement is used to exit a function. A function may or may not return a value. If a function returns a value, it is passed back by the return statement as argument to the caller. If it does not return a value, we simply write return with no arguments.

Syntax :

```
return [expression]
```

3.12. Define anonymous function.

Ans: The anonymous functions are the functions created using a lambda keyword.

3.13. You have been given a string 'I live in Cochin. I love pets.' Divide this string in such a very that the two sentences in it are separated and stored in different variables. Print them.

Ans:

```
>>> var = 'I live in Cochin, I love pets.'
>>> var1 = var[: 17]
>>> var2 = var[18 : 30]
>>> print var1
I live in Cochin,        # Output
>>> print var2
I love pets,             # Output
```

3.14. An email address is provided : hello@python. org. Using tuple assignment, split the username and domain from the email address.

Ans:

```
>>> addr = 'hello@python. org'
>>> usrname, domain = addr. split ('@')
>>> print usrname
Hello                    # Output
>>> print domain
python.org               # Output
```

3.15. Write a function called sumall that takes any number of arguments and returns their sum.

Ans:

```
>>> def sumall (*t) :
        i = 0
        sum = 0
        while i < len(t) :
            sum = sum + t[i]
            i = i + 1
        return sum
>>>sumall(1, 2, 3, 4, 5, 6, 7)
28                       # Output
```


3.16. Write a function called circleinfo which takes the radius of circle as argument and returns the area and circumference of the circle.

Ans:

```
>>>def circleinfo(r) :
        c = 2 * 3.14159 * r
        a = 3.14159 * r * r
        return (c, a)
>>> circleinfo(10)
(62.8318, 314.159)  # Output
```

3.17. Give examples for len, max and min methods.

Ans:

```
>>> list = [789, 'abcd', 'jinnie', 1234]
>>> len(list)
4                        # Output
>>>  max(list)
'jinnie'                 # Output
>>>min(list)
789                      # Output
```

3.18. Give examples for all, any, len and sorted methods in dictionary.

Ans:

```
>>> dict1 = {8 : 'a', 3 : 'b', 5 : 'c', 7 : 'd'}
>>> all (dict1)
True                     # Output
>>> any(dict1)
True                     # Output
>>> len (dict1)
4                        # Output
>>> sorted(dict1)
[3, 5, 7, 8]             # Output
```

3.19. Give the syntax required to convert an integer number into string and a float to an integer.

Ans:

```
# integer to string
>>>str (5)
'5'                      # Output
                         # float to integer
>>>float (5.50)
5                        # Output
```

3.20. Write a program to print the calendar for the month of March 1991.

Ans:
```
>>> import calendar
>>> c = calender.month(1991, 3)
>>> print c
March 1991
```

Mo	Tu	We	Th	Fr	Sa	Su
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

3.21. Write a function which accepts two numbers and returns their sum.

Ans:
```
>>> def sum (arg1, arg2) :
        sum = arg1 + arg2
        return sum
                        # Now calling the function here
>>> a = 4
>>> b = 3
>>> total = sum(a, b)    # calling the sum function
>>> print(total)
7                       # Output
```

3.22. What are the types of arguments used for calling a function?

Ans: Four types of arguments used for calling a function :

i. Required argument
ii. Keyword argument
iii. Default argument
iv. Variable length argument

3.23. Give one-one example for zip, max and min methods.

Ans:
```
>>> tuple1 = ('a', 'b', 'c')
>>> tuple1 = (1, 2, 3)
>>> max (tuple 2)
3                       # Output
>>>min(tuple 1)
'a'                     # Output
>>>zip(tuple 1, tuple 2)
[('a', 1), ('b', 2), ('c', 3)]   # Output
```

3.24. What is the output of print list[2] when list = ['abcd', 2.23, 'john'] ?

Ans: john

3.25. What is slicing ?

Ans: Slicing in Python refers to the technique of extracting a portion or a subset of elements from a sequence-like data structure, such as a string, list, tuple, or other iterable objects.

3.26. What is the use of docstring ?

Ans: Docstring command is use to know about a function using triple-quoted string.

3.27. What is the difference between list and tuples in Python ?

AKTU 2019-20, Marks 02

Ans:

S. No.	List	Tuples
1.	Lists are mutable, *i.e.*, they can be edited.	Tuples are immutable (they are lists that cannot be edited).
2.	Lists are usually slower than tuples.	Tuples are faster than lists.
3.	**Syntax :** list_1 = [10, 'Quantum', 20]	**Syntax :** tup_1 = (10, 'Quantum', 20)

3.28. What is the difference between Python arrays and lists ?

AKTU 2021-22, Sem-4; Marks 02

AKTU 2019-20, Sem-3; Marks 02

OR

Differentiate between Python arrays and lists ?

AKTU 2022-23, Sem-3; Marks 02

Ans:

S. No.	Arrays	Lists
1.	Arrays can only store homogeneous data (data of the same type).	Lists can store heterogeneous and arbitrary data.
2.	Arrays use less memory to store data.	Lists require more memory to store data.
3.	The length of an array is pre-fixed while creating it, so more elements cannot be added.	The length of a list is not fixed, so more elements can be added.

3.29. Which of the following statements produce an error in Python ?

```
x, y, z = 1, 2, 3 # s1
a, b = 4, 5, 6 # s2
u = 7, 8, 9 # s3
```

(List all the statements that have error)

AKTU 2020-21, Sem-3; Marks 02

Ans. In $s1$, no error. $x = 1$, $y = 2$ and $z = 3$ In $s2$, it gives error. $a = 4$, $b = 5$ but 6 is not assigned to any variable.

In $s3$, it gives error, $u = 7$ but 8 and 9 are not assigned to any variable.

Therefore,

$s2$ and $s3$ has error because number of variable is less than the assigned value.

3.30. Consider the program :

```
x = ['12', 'hello', 456]
x [0] * = 3
x [1] [1] = 'bye'
```

Explain why this program generates an error.

AKTU 2020-21, Sem-3; Marks 02

Ans. Given, x = ['12', 'hello', 456]

$x[0] * = 3$,It does not give any error. If we print($x[0]$) then its output is '121212'

$x[1][1]$ = 'bye', This code will give error.

Value of $x[1][1]$ is 'e'. Here in the given code we are assigning 'bye' to 'e' which cannot be possible because string is immutable means once we declare string we cannot change it.

3.31. What will be the output of the following Python code ?

```
def cube(x):
    return x * x * x
x = cube(3)
print x
```

AKTU 2021-22, Sem-3; Marks 02

Ans. **Output :** 27

3.32. Explain the output of following function.

```
defprintalpha (abc_list, num_list) :
    for char in abc_list :
        for num in num_list :
            print(char, num)
    return
printalpha ('a', 'b', 'c'], [1, 2, 3]
```

AKTU 2022-23, Sem-4; Marks 02

Ans.

1. The code will generate syntax error at Line 1. The correct syntax will be def printalpha (abc_list, num_list) :
2. After correcting the syntax output will be

a1
a2
a3
b1
b2
b3
c1
c2
c3

3.33. Describe the concept of List Slicing with a suitable example.

AKTU 2022-23, Sem-3; Marks 02

Ans. If L is a list, the expression L [start : stop : step] returns the portion of the list from index start to index stop, at a step size step.

Syntax : L [start : stop : step]

start : Start position, stop : End position, step : increment

For example :

```
L = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']
print(L[2 : 7])
# Output : ['c', 'd', 'e', 'f', 'g']
```

3.34. Explain the difference between 'append' and 'extend' in Python ?

AKTU 2022-23, Sem-3; Marks 02

Ans.

S. No.		'append()'	'extend()'
1.	Function	Appends a single element at the end of a list.	Appends multiple elements from an iterable at the end of a list.
2.	Input	Accepts a single element as an argument.	Accepts an iterable (e.g., list, tuple, string) as an argument.

3.35. What will be the output of the following Python code

```
def count1 (s) :
    vowels = "AEIOUaeiou"
    count = 0
    for c in s :
        if c in vowels :
        count + = 1
    return count
print (count1 ('I love India'))
```

AKTU 2022-23, Sem-3; Marks 02

Ans. **Output :** 6

3.36. What will be the output of the following code ?

```
list1 = ['M', 'n', 'k', 'y']
print ("@".join(list1))
```

AKTU 2022-23, Sem-3; Marks 02

Ans. **Output :** M@n@k@y

3.37. Compare list and Dictionary data structure.

Ans.

S. No.	List	Dictionary
1.	The list is a collection of index value pairs like that of the array in C++.	The dictionary is a hashed structure of the key and value pairs.
2.	The list is created by placing elements in [ ] separated by commas ","	The dictionary is created by placing elements in { } as "key":"value", each key-value pair is separated by commas ","
3.	The indices of the list are integers starting from 0.	The keys of the dictionary can be of any data type.

3.38. What is a dictionary in Python ?

AKTU 2022-23, Sem-3; Marks 02

Ans. The Python dictionary is an unordered collection of items or elements.

3.39. What are local variables and global variables in Python ?

AKTU 2019-20, Sem-3; Marks 02

Ans. **Local Variable :** A local variable is declared inside a specific function and can only be accessed by the function in which it is declared.
Global Variable : A global variable is one that is "declared" outside of the function in a program and can, therefore, be accessed by any of the functions.

☺☺☺

4 UNIT

Python File Operations (2 Marks Questions)

4.1. List the order of file operations in Python.

Ans: The order is as follows :

1. Opening a file
2. Perform read or write operation
3. Closing a file

4.2. Explain any four modes of opening the file.

OR

Discuss various file opening modes of python.

AKTU 2022-23, Sem-4; Marks 02

Ans: Modes of opening the file :

i. ***r* :** It opens a file in reading mode. The file pointer is placed at the starting of the file.

ii. ***r +* :** It opens the file in both reading and writing mode. The file pointer is placed at the starting of the file.

iii. ***w* :** It opens the file in writing mode. If a file exists, it overwrites the existing file; otherwise, it creates a new file.

iv. ***w +* :** It opens the file in both reading and writing mode. If a file exists, it overwrites the existing file; otherwise, it creates a new file.

4.3. Explain the file object attributes in detail.

Ans:

Attribute	Description
file.closed	It will return true if the file is closed ; it will otherwise return false.
file.mode	It will return the access mode with which the file is opened.
file.name	It will return name of the file
file.softspace	It will return false if space explicitly required with print; otherwise it will return true.

4.4. Give the syntax for reading from a file. What is the work of the readline() function ?

Ans: Syntax :

fileobject.read([size])

The count parameter size gives the number of bytes to be read from an opened file. It starts reading from the beginning of the file until the size given. If no size is provided, it ends up reading until the end of the file.

4.5. How are renaming and deleting performed on a file ? Give the syntax for each.

Ans: Renaming a file : Renaming a file in Python is done with the help of the rename() method. The rename() method is passed with two argument, the current filename and the new filename.

Syntax :

os.rename(current_filename, new_filename)

Deleting a file : Deleting a file in Python is done with the help of the remove() method. The remove() method takes the filename as an argument to be deleted.

Syntax :

os.remove(filename)

4.6. What are the various file positions methods ?

Ans: In Python, the tell() method tells us about the current position of the pointer. The current position tells us where reading will starts from at present.

We can also change the position of the pointer with the help of the seek() method. We pass the number of bytes to be moved by the pointer as arguments to the seek() method.

4.7. Differentiate between reading file in text mode and binary mode.

Ans:

S. No.	Text mode	Binary mode
1.	This mode is used for reading files that contain text, such as plain text files, CSV files, etc.	This mode is used for reading files that contain non-text data, like images, audio files, executables, etc.
2.	When you read a file in text mode, Python handles encoding and decoding of the characters. It automatically converts bytes into strings using a specified character encoding.	When you read a file in binary mode, Python does not perform any encoding/decoding. It treats the file content as a stream of bytes.

4.8. How does read function behave when called without an argument ?

Ans. When the read() function is called without an argument in Python, it will attempt to read the entire contents of the file from the current position of the file pointer until the end of the file. It will then return the content as a single string (in text mode) or a bytes object (in binary mode).

4.9. In which situation might you choose to use 'readline()' instead of 'readlines()' ?

Ans. We might choose to use readline() instead of readlines() in Python when you want to read a file line by line and process each line individually, especially if the file is very large and you want to conserve memory.

4.10. What is primary difference between writing to a file in text mode and binary mode ?

Ans.

S. No.	Text mode	Binary mode
1.	In text mode, when you write data to a file, Python handles encoding and decoding automatically. It converts strings into bytes using a specified character encoding.	In binary mode, Python treats the data as a stream of bytes and does not perform any encoding or decoding. It preserves the exact byte values you write to the file.
2.	Text mode is used for writing textual data, like strings, to a file. It's suitable for plain text files, CSV files, and other text-based formats.	Binary mode is used for writing non-text data, such as images, audio files, executables, etc.

4.11. How does write () function handle different data types such as strings and bytes ?

Ans. It operates in different ways depending on the mode in which the file is opened :

a. **Text Mode (w' or 'wt') :** When a file is opened in text mode, write() is used to write a string of characters to the file. If the file already exists, it will be truncated (emptied) before writing. If the file doesn't exist, a new file will be created.

b. **Binary Mode ('wb') :** In binary mode, write() is used to write a sequence of bytes to the file.

4.12. Can you give an example of using the write () function to add content to existing file ?

Ans. To write to an existing file, you must add a parameter to the open () function :
"a" - Append - will append to the end of the file
"w" - Write - will overwrite any existing content

Example :
Open the file "demofile2.txt" and append content to the file :

```
f = open("demofile2.txt", "a")
f.write("Now the file has more content!")
f.close()
#open and read the file after the appending :
f = open("demofile2.txt", "r")
print(f.read())
```

4.13. What if you try to use the 'write ()' function on a file opened in read mode ?

Ans. If we attempt to use the write function on a file that has been opened in read mode ('r'), we will encounter a unsupported operation error. This is because attempting to write to a file that is open in read mode is not allowed by the Python language.
To write to a file, you need to open it in a mode that supports writing, such as 'w' for write mode or 'a' for append mode.

4.14. Explain role of flush () method in file writing operation ?

Ans. The flush() method in Python file handling clears the internal buffer of the file. In Python, files are automatically flushed while closing them. However, a programmer can flush a file before closing it by using the flush() method.

4.15. When is it appropriate to use the 'a' mode in conjunction with 'write ()' function ?

Ans. The append mode ('a') is appropriate when you want to add content to the end of an existing file without overwriting the existing content. This is useful when you want to extend or update a file without losing its original contents.
For example, if you have a log file and you want to add new log entries to it without erasing the previous ones, you would open the file in append mode :

```
with open('logfile.txt', 'a') as file:
    file.write('New log entry\n')
```

In this example, the 'a' mode allows you to append the new log entry to the end of the file.

4.16. Describe the purpose of 'with' statement when working with files in Python ?

Ans. The 'with' statement in Python is used for automatic resource management. When working with files, it ensures that a file is

properly opened and closed, even if an error occurs during the execution of the code.

Here's how the with statement works with files :

```
with open('example.txt', 'r') as file:
    content = file.read()
    # Do operations with the file
# Once the block is exited (even if an error occurs), the file is automatically closed.
```

4.17. Explain the significance of offset parameter in the seek () function.

Ans. The seek function in Python is used to change the current position of the file pointer within an open file. The significance of the offset parameter in the seek function is that it allows you to navigate to specific positions within a file. This is crucial when you want to read or write data at a particular location rather than just sequentially from the beginning.

4.18. What happens if you use a negative offset in the seek () function with 'whence' set to 0 ?

Ans. If we use a negative offset with whence set to 0 (which corresponds to SEEK_SET, indicating an absolute position from the beginning of the file), Python will raise a ValueError. This is because using a negative value in this context is not meaningful.

4.19. What happens if you attempt to seek beyond the end of a file ?

Ans.

1. If you attempt to seek beyond the end of a file using the seek function, the file's size is extended with null bytes (zeros) to accommodate the new position. This means that the file is effectively padded with zeros up to the specified position.
2. For example, if you have a file that is 100 bytes long and you attempt to seek to position 200, the file will be extended to 200 bytes with null bytes filling the space in between.

4.20. What is the default behaviour of seek () functions if you don't provide any arguments ?

Ans. The seek method requires at least the first parameter (offset) be met. The second (from_what/whence) is optional. If no arguments are passed to the method then you will receive: TypeError: seek() takes at least 1 argument (0 given)

4.21. How can you check if file exists in Python ?

Ans. You can check if a file exists in Python using the os.path.exists() function. Here's an example :


```
import os
file_path = 'example.txt'
if os.path.exists(file_path):
    print(f'The file {file_path} exists.')
else:
    print(f'The file {file_path} does not exist.')
```

4.22. What are directories ?

Ans. If there is a large number of file, then related files are placed in different directories. Directory can be said to be a collection of files and sub directories. The module 'os' in Python enables us to use various methods to work with directories.

4.23. What are the basic methods performed on directories ?

Ans. Following are the four basic methods that are performed on directories :

i. mkdir () method (Creating a directory)
ii. chdir() method (Changing the current directory)
iii. getcwd () method (Displaying the current directory)
iv. rmdir () method (Deleting the directory).

☺☺☺

5 UNIT

Python Packages (2 Marks Questions)

5.1. What is pip ?

Ans. Python PIP is the package manager for Python packages. We can use PIP to install packages that do not come with Python. The basic syntax of PIP commands in the command prompt is: pip 'arguments'

5.2. Differentiate between package and module.

Ans.

S. No.	Modules	Packages
1.	Modules come with a .py extension carrying a collection of functions and variables.	A package is a container that contains different modules along with with an _init_.py file.
2.	The purpose of modules is code organization.	The purpose of packages is code reuse and code distribution.

5.3. Differentiate between built-in packages and user defined packages.

Ans.

S. No.		Built-in packages	User-defined packages
1.	Availability	Pre-installed with Python	Created by users
2.	Purpose	Provide fundamental functionalities	Organize user's code into logical units
3.	Examples	os, sys, math, random, etc.	Any package created by a user
4.	Importing	Directly import using 'import' statement.	Import using the package name and module name.

5.4. List some Python IDEs.

Ans. Some Python IDEs are :

1. PyCharm
2. Spyder
3. PyDev

5.5. What is the Python package index (PyPI) ?

Ans. The Python Package Index (PyPI) is a repository of software for the Python programming language. PyPI helps you find and install software developed and shared by the Python community.

5.6. How can you uninstall a Python package ?

Ans.

1. Open the command prompt.
2. With the help of "PIP uninstall module_name" command, uninstall the module.
3. The flask package will be deleted.
4. In the Python 2.7 version, flask uninstall through pip.
5. This would be "pip3.6 uninstall --user flask" for Python 3.6.
6. Following a list of the files that need to be deleted, the command will request your approval. Enter the Enter key after typing "y" to confirm this action.

5.7. How do you import a module from a package in Python ?

Ans. In Python, we can import modules from packages using the dot (.) operator.

For example, if we want to import the start module, it can be done as follows :

import Game.Level.start

5.8. Differentiate between 'plt.plot()' and 'plt.Scatter()'.

Ans.

S. No.	plt.plot	plt.Scatter
1.	plt.plot is primarily used for creating line for creating line plots or connecting data points with lines.	plt.Scatter is used for creating scatter plots, where individual data points are plotted as distinct markers.
2.	It is suitable for showing trends, patterns, or relationships between continuous data points.	It is useful for visualizing the distribution and relationships between two continuous variables.

5.9. What is broadcasting in Numpy ?

Ans: The term broadcasting describes how Numpy treats arrays with different shapes during arithmetic operations. Subject to certain constraints, the smaller array is "broadcast" across the larger array so that they have compatible shapes.

5.10. What is purpose of np.zeros() function in Numpy ?

Ans: Python numpy. zeros() function returns a new array of given shape and type, where the element's value as 0.

5.11. What is the purpose of np.random module in Numpy ?

Ans: Random() is one of the function for doing random sampling in numpy. It returns an array of specified shape and fills it with random floats in the half-open interval [0.0, 1.0).

5.12. What are two main data structures in pandas ?

Ans: The two primary data structures of pandas, Series (1-dimensional) and DataFrame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering.

5.13. What is data frame in pandas ?

Ans: DataFrame is a 2-dimensional labeled data structure with columns of potentially different types. It is generally the most commonly used pandas object.

5.14. What does GUI stands for ? Name one popular library for creating GUI applications in Python ?

Ans: A graphical user interface (GUI) is a desktop interface that allows you to communicate with computers.

Tkinter is a Python library that can be used to construct basic graphical user interface (GUI) applications. In Python, it is the most widely used module for GUI applications.

5.15. What is the key difference between GUI and a command line interface (CLI) ?

Ans:

S. No.	GUI	CLI
1.	GUI is graphical user interface.	CUI is the character user interface.
2.	Windows is characterized as GUI.	DOS is the characterized as CUI.
3.	A GUI use pictures, symbols, word that control with mouse.	A CUI use characters on screen that control with keyboard.
4.	GUI is very friendly and easy to remember.	CUI can be confusing and difficult to remember.

5.16. What is purpose of callback function in GUI programming ?

Ans: When one function is called from another function it is known as a callback. The main purposes of callback functions in GUI programming are as follows :

1. Event handling
2. Separation of concerns
3. User interaction
4. Customization

5.17. How does an IDE differ from a text editor in terms of functionality ?

Ans: IDEs are designed to test and preview code projects. A text editor can only write code.

5.18. Explain the programming cycle for Python in detail.

AKTU 2021-22, Sem-3; Marks 02

AKTU 2022-23, Sem-3; Marks 02

OR

Define the programming cycle for Python.

AKTU 2022-23, Sem-4; Marks 02

Ans: The programming cycle for Python typically follows a series of steps from writing code to running and testing the program. Here's a general outline of the Python programming cycle :

1. Problem identification and requirements gathering.
2. Planning and design.
3. Writing code.
4. Testing.
5. Documentation.

6. Integration and deployment.
The specific steps and their order may vary depending on the project and development methodology being used.

5.19. Show the way to import the module in Python.

AKTU 2022-23, Sem-3; Marks 02

Ans. You can import an entire module using the 'import' statement followed by the module name. For example :

```
import module_name
```

5.20. Explain features of any two Python IDEs.

AKTU 2022-23, Sem-4; Marks 02

Ans.

A. PyCharm : Features of PyCharm :

1. It has smart code navigation, good code editor, a function for quick refactoring.
2. The integrated activities with PyCharm are profiling, testing, debugging, remote development, and deployments.

B. Spyder : Features of Spyder :

1. It has good syntax highlighting and auto code completion features.
2. Spyder Python explores and edits variables directly from GUI.

5.21. Discuss why Python is called as dynamic and strongly typed language ?

AKTU 2021-22, Sem-4; Marks 02

Ans. Python is both a strongly typed and a dynamic typed language, strongly typed because the interpreter keeps track of all variable types and dynamic as the type of the variable is determined only during runtime.

☺☺☺

B.Tech.
(SEM. III) ODD SEMESTER THEORY EXAMINATION, 2019-20
PYTHON PROGRAMMING

Time : 3 Hours **Max. Marks : 100**

Note : 1. Attempt **all** Section.

Section-A

1. Answer **all** questions in brief. **(2 × 10 = 20)**

a. What is the difference between list and tuples in Python?

Ans. Refer Q. 3.27, Page SQ–16V, Unit-3, Two Marks Questions.

b. In some languages, every statement ends with a semi-colon (;). What happens if you put a semi-colon at the end of a Python statement?

Ans. Refer Q. 1.14, Page SQ–3V, Unit-1, Two Marks Questions.

c. Mention five benefits of using Python.

Ans. Refer Q. 1.15, Page SQ–3V, Unit-1, Two Marks Questions.

d. How is Python an interpreted language ?

Ans. Refer Q. 1.6, Page SQ–2V, Unit-1, Two Marks Questions.

e. What type of language is Python ?

Ans. Refer Q. 1.7, Page SQ–2V, Unit-1, Two Marks Questions.

f. What are local variables and global variables in Python ?

Ans. Refer Q. 3.39, Page SQ–20V, Unit-3, Two Marks Questions.

g. What is the difference between Python Arrays and lists ?

Ans. Refer Q. 3.28, Page SQ–16V, Unit-3, Two Marks Questions.

h. Define ADT interface.

Ans. This question is out of syllabus from session (2023-24).

i. Define floor division with example.

Ans. Refer Q. 1.16, Page SQ–3V, Unit-1, Two Marks Questions.

j. Differentiate fruitful functions and void functions.

Ans. Refer Q. 2.16, Page SQ–9V, Unit-2, Two Marks Questions.

Section-B

2. Answer any **three** of the following : (3 × 10 = 30)

a. Explain iterator. Write a program to demonstrate the Tower of Hanoi using function.

Ans. This question is out of syllabus from session (2023-24).

b. Discuss function in python with its parts and scope. Explain with example. (Take Simple calculator with add, subtract, Division and Multiplication).

Ans. Refer Q. 3.41, Page 3–32V, Unit-3.

c. Discuss ADT in Python. How to define ADT ? Write code for a student information.

Ans. This question is out of syllabus from session (2023-24).

d. Explain all the conditional statement in Python using small code example.

Ans. Refer Q. 2.5, Page 2–5V, Unit-2.

e. What is Python? How Python is interpreted? What are the tools that help to find bugs or perform static analysis ? What are Python decorators ?

Ans. Refer Q. 5.25, Page 5–22V, Unit-5.

Section-C

3. Answer any **one** part of the following : (1 × 10 = 10)

a. Write short notes with example : The programming cycle for Python, elements of Python, type conversion in Python, operator precedence, and Boolean expression.

Ans. Refer Q. 5.28, Page 5–27V, Unit-5.

b. How memory is managed in Python? Explain PEP 8. Write a Python program to print even length words in a string.

Ans. Refer Q. 5.32, Page 5–28V, Unit-5.

4. Answer any **one** part of the following : (1 × 10 = 10)

a. Explain expression evaluation and float representation with example. Write a python program for how to check if a given number is Fibonacci number.

Ans. Refer Q. 2.39, Page 2–24V, Unit-2.

b. Explain the purpose and working of loops. Discuss break and continue with example. Write a Python program to convert time from 12 hour to 24-hour format.

Ans. Refer Q. 2.30, Page 2–19V, Unit-2.

5. Answer any **one** part of the following : (10 × 1 = 10)

a. Explain higher order function with respect to lambda expression. Write a Python code to count occurrences of an element in a list.

Ans. Refer Q. 3.51, Page 3–38V, Unit-3.

b. Explain unpacking sequences, mutable sequences, and list comprehension with example. Write a program to sort list of dictionaries by values in Python – Using lambda function.

Ans. This question is out of syllabus from session (2023-24).

6. Answer any **one** part of the following : (1 × 10 = 10)

a. Discuss File I/O in Python. How to perform open, read, write, and close into a file ? Write a Python program to read a file line-by-line store it into a variable.

Ans. Refer Q. 4.6, Page 4–4V, Unit-4.

b. Discuss exceptions and assertions in Python. How to handle exceptions with try-finally ? Explain five built-in exceptions with example.

Ans. This question is out of syllabus from session (2023-24).

7. Answer any **one** part of the following : (7 × 1 = 7)

a. Discuss and differentiate iterators and recursion. Write a program for recursive Fibonacci series.

Ans. This question is out of syllabus from session (2023-24).

b. Discuss sorting and merging. Explain different types of sorting with example. Write a Python program for Sieve of Eratosthenes.

Ans. This question is out of syllabus from session (2023-24).

☺☺☺

B. Tech.
(SEM. III) ODD SEMESTER THEORY EXAMINATION, 2020-21
PYTHON PROGRAMMING

Time : 3 Hours **Max. Marks : 100**

Note : 1. Attempt **all** sections. If require any missing data; then choose suitably.

Section-A

1. Attempt **all** questions in brief. (2 × 10 = 20)

a. What is the use of "raise" statement ? Describe with an example.

Ans. This question is out of syllabus from session (2023-24).

b. Write a recursive Python function "rprint" to print all elements in a list in reverse.
rprint ([ ]) prints nothing
rprint ([1, 2, 3]) prints 3 2 1

Ans. This question is out of syllabus from session (2023-24).

c. Describe the behavior of "range (s, e)" in Python.

Ans. Refer Q. 2.23, Page SQ–10V, Unit-2, Two Marks Questions.

d. Explain the use of "with" construct in Python with an example program.

Ans. This question is out of syllabus from session (2023-24).

e. Which of the following statements produce an error in Python ?

$x, y, z = 1, 2, 3$ # s1
$a, b = 4, 5, 6$ # s2
$u = 7, 8, 9$ # s3

(List all the statements that have error)

Ans. Refer Q. 3.29, Page SQ–17V, Unit-3, Two Marks Questions.

f. Explain the role of precedence with an example.

Ans. Refer Q. 1.26, Page SQ–5V, Unit-1, Two Marks Questions.

g. How do you read an input from a user in Python to be used as an integer in the rest of the program ? Explain with an example.

Ans. Refer Q. 1.27, Page SQ–6V, Unit-1, Two Marks Questions.

h. Consider the program :

```
x = ['12', 'hello', 456]
x [0] * = 3
x [1] [1] = 'bye'
```

Explain why this program generates an error.

Ans. Refer Q. 3.30, Page SQ–17V, Unit-3, Two Marks Questions.

i. What is the output of the following program ?
(lambda x, y : y – 2*x) (1, 11)

Ans. This question is out of syllabus from session (2023-24).

j. Explain the use of __lt__ function in a class Python ?

Ans. Refer Q. 1.28, Page SQ–6V, Unit-1, Two Marks Questions.

Section-B

2. Attempt any **three** of the following : (10 × 3 = 30)

a. Write a Python function removekth (s, k) that takes as input a string a and an integer k > = 0 and removes the character at index k. If k is beyond the length of s, the whole of s is returned. For example,
removekth ("PYTHON", 1) returns "PTHON"
removekth ("PYTHON", 3) returns "PYTON"
removekth ("PYTHON", 0) returns "YTHON"
removekth ("PYTHON", 20) returns "PYTHON"

Ans. Refer Q. 3.45, Page 3–34V, Unit-3.

b. Write a Python function average to compute the average of a list of numbers. The function must use try-except to handle the case where the input list is empty. Further, in the case the average for the empty list should be set to 0.0 using the except block.

Ans. This question is out of syllabus from session (2023-24).

c. Describe the differences between a linear search and a binary search ?

Ans. This question is out of syllabus from session (2023-24).

d. Write a function lessthan (lst, k) to return list of numbers less than k from a list lst. The function must use list comprehension.
Example :

lessthan ([1, – 2, 0, 5, – 3], 0) returns [– 2, – 3]

Ans. Refer Q. 3.12, Page 3–10V, Unit-3.

e. Write a program factors (N) that returns a list of all positive divisors of N ($N >= 1$). For example :
factors (6) returns [1, 2, 3, 6]
factors (1) returns [1]
factors (13) returns [1, 13]

Ans. Refer Q. 3.46, Page 3–35V, Unit-3.

Section-C

3. Attempt any **one** part of the following : **(10 × 1 = 10)**

a. How can you create Python file that can be imported as a library as well as run as a standalone script ?

Ans. Refer Q. 5.6, Page 5–6V, Unit-5.

b. Describe the difference between
import library
and
from library import *
when used in a python program. Here library is some python library.

Ans. Refer Q. 5.7, Page 5–6V, Unit-5.

4. Attempt any **one** part of the following : **(10 × 1 = 10)**

a. Write a function makePairs that takes as input two lists of equal length and returns a single list of same length where k-th element is the pair of k-th elements from the input lists. For example,
makePairs ([1, 3, 5, 7], [2, 4, 6, 8])
returns [(1, 2), (3, 4), (5, 6), (7, 8)]
makePairs ([], [])
returns []

Ans. Refer Q. 3.47, Page 3–35V, Unit-3.

b. Show an example where both Keyword arguments and Default arguments are used for the same function in a call. Show both the definition of the function and its call.

Ans. Refer Q. 3.39, Page 3–30V, Unit-3.

5. Attempt any **one** part of the following : **(10 × 1 = 10)**

a. Explain why Python is considered an interpreted language.

Ans. Refer Q. 5.25, Page 5–22V, Unit-5.

b. What is short circuit evaluation ? What is printed by the following Python program ?


```
a = 0
b = 2
c = 3
x = c or a
print (x)
```

Ans. Refer Q. 1.11, Page 1–9V, Unit-1.

6. Attempt any **one** part of the following : **(10 × 1 = 10)**

a. Write a Python program, triangle (N), that prints a right triangle having base and height consisting of N * symbols as shown in these examples :
triangle (3) prints :

```
*
* *
* * *
```

triangle (5) prints :

```
*
* *
* * *
* * * *
* * * * *
```

Ans. Refer Q. 3.52, Page 3–38V, Unit-3.

b. Write a Python program, countSquares (N), that returns the count of perfect squares less than or equal to N ($N > 1$). For example :
countSquares (1) returns 1
 # Only 1 is a perfect square < = 1
countSquares (5) returns 2
 # 1, 4 are perfect squares < = 5
countSquares (55) returns 7
 # 1, 4, 9, 16, 25, 36, 49 < = 55

Ans. Refer Q. 3.48, Page 3–36V, Unit-3.

7. Attempt any **one** part of the following : **(10 × 1 = 10)**

a. Write a Python function, alternating (lst), that takes as argument a sequence lst. The function returns True if the elements in lst are alternately odd and even, starting with an even number. Otherwise it returns False. For example :
alternating ([10, 9, 6]) returns False
alternating ([10, 15, 8]) returns True
alternating ([10]) returns True
alternating ([ ]) returns True
alternating ([15, 10, 9]) returns False

Ans. Refer Q. 3.49, Page 3–36V, Unit-3.

b. Write a Python function, searchMany(s, x, k), that takes as argument a sequence s and integers x, k ($k > 0$). The function returns True if there are at most k occurrences of x in s. Otherwise it returns False. For example :
searchMany ([10, 17, 15, 12], 15, 1) returns True
searchMany ([10, 12, 12, 12], 12, 2) returns False
searchMany ([10, 12, 15, 11], 17, 18) returns True

Ans. Refer Q. 3.50, Page 3–37V, Unit-3.

☺☺☺

B.Tech.
(SEM. III) ODD SEMESTER THEORY EXAMINATION, 2021-22
PYTHON PROGRAMMING

Time : 3 Hours **Max. Marks : 50**

Note : Attempt **all** Section. If you require any missing data, then choose suitably.

Section-A

1. Answer **all** questions in brief. **(2 × 5 = 10)**

a. Explain the Programming Cycle for Python in detail.

Ans. Refer Q. 5.18, Page SQ–30V, Unit-5, Two Marks Questions.

b What will be the output of the following Python code ?

```
i = 0
while i< 3 :
    print(i)
i+= 1
else:
print(0)
```

Ans. Refer Q. 2.17, Page SQ–9V, Unit-2, Two Marks Questions.

c. What will be the output of the following Python code ?

```
def cube(x):
    return x * x * x
x = cube(3)
print x
```

Ans. Refer Q. 3.31, Page SQ–17V, Unit-3, Two Marks Questions.

d. How do we define an Interface for an ADT ?

Ans. This question is out of syllabus from session (2023-24).

e. How do you perform a search in Python ?

Ans. This question is out of syllabus from session (2023-24).

Section-B

2. Attempt any **three** of the following : **(5 × 3 = 15)**

a. What do you mean by Python IDE ? Explain in detail.

Ans. Refer Q. 5.26, Page 5–24V, Unit-5.

b. How can you randomize the items of a list in place in Python ?

Ans. Refer Q. 3.32, Page 3–24V, Unit-3.

c. Explain Tuples and Unpacking Sequences in Python Data Structure.

Ans. This question is out of syllabus from session (2023-24).

d. What are file input and output operations in Python Programming ?

Ans. Refer Q. 4.6, Page 4–4V, Unit-4.

e. Solve the Tower of Hanoi problem for n= 3 disk and show all the steps.

Ans. This question is out of syllabus from session (2023-24).

Section-C

3. Attempt any **one** part of the following : **(5 × 1 = 5)**

a. Write a program in Python to execute the Selection sort algorithm.

Ans. This question is out of syllabus from session (2023-24).

b. Explain why Python is considered an interpreted language.

Ans. Refer Q. 5.25, Page 5–22V, Unit-5.

4. Attempt any **one** part of the following : **(5 × 1 = 5)**

a. Write a Python program to construct the following pattern, using a nested for loop.

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

Ans. Refer Q. 2.20, Page 2–14V, Unit-2.

b. Write a program to produce Fibonacci series in Python.

Ans. Refer Q. 2.9, Page 2–8V, Unit-2.

5. Attempt any **one** part of the following : **(5 × 1 = 5)**

a. Write a Python program to change a given string to a new string where the first and last chars have been exchanged.

Ans. Refer Q. 3.30, Page 3–23V, Unit-3.

b. Write a Python program to add an item in a tuple.

Ans. Refer Q. 3.15, Page 3–12V, Unit-3.

6. Attempt any **one** part of the following : **(5 × 1 = 5)**

a. How to create and import a module in Python ?

Ans. Refer Q. 5.3, Page 5–3V, Unit-5.

b. Explain the algorithm Sieve of Eratosthenes used in Python Programming.

Ans. This question is out of syllabus from session (2023-24).

7. Attempt any **one** part of the following : **(5 × 1 = 5)**

a. Write a Recursive function in python Binary Search (Arr, l, R, X) to search the given element X to be searched from the List Arr having R elements, where 1 represent lower bound and R represents the upper bound.

Ans. This question is out of syllabus from session (2023-24).

b. Explain the terms Merge List and Merge Sort in Python Programming.

Ans. This question is out of syllabus from session (2023-24).

☺☺☺

B. Tech.
(SEM. IV) EVEN SEMESTER THEORY EXAMINATION, 2021-22
PYTHON PROGRAMMING

Time : 3 Hours **Max. Marks : 50**

Note : Attempt **all** Sections. If you require any missing data, then choose suitably.

Section-A

1. Attempt **all** questions in brief. **(4 × 2.5 = 20)**

a. Discuss why Python is called as dynamic and strongly typed language ?

Ans. Refer Q. 5.21, Page SQ–31V, Unit-5, Two Marks Questions.

b. How pass statement is different from a comment ?

Ans. This question is out of syllabus from session (2023-24).

c. Write a for loop that prints numbers from 0 to 57, using range function.

Ans. Refer Q. 2.18, Page SQ–9V, Unit-2, Two Marks Questions.

d. What is the difference between Python Arrays and Lists ?

Ans. Refer Q. 3.28, Page SQ–16V, Unit-3, Two Marks Questions.

Section-B

2. Attempt any **two** of the following : **(2 × 4 = 8)**

a. Write a python function named Compute Average to find average of a list of numbers. It should handle the exception if the list is empty and return 0 in that case.

Ans. This question is out of syllabus from session (2023-24).

b. Implement the binary search technique.

Ans. This question is out of syllabus from session (2023-24).

c. Explain the use of break and continue with a suitable example.

Ans. Refer Q. 2.33, Page 2–21V, Unit-2.

Section-C

3. Attempt any **two** part of the following : **(4 × 2 = 8)**

a. What do you mean by operator precedence and associativity ? Explain.

Ans. Refer Q. 1.14, Page 1–12V, Unit-1.

b. Write short notes on :

i. The programming cycle for python.

ii. Type conversion in python.

Ans. Refer Q. 5.29, Page 5–27V, Unit-5.

c. Write python program to swap two numbers without using Intermediate/Temporary variables. Prompt the user for input.

Ans. Refer Q. 1.3, Page 1–4V, Unit-1.

4. Attempt any **two** part of the following : **(4 × 2 = 8)**

a. Write a program to check an input number is prime or not.

Ans. Refer Q. 2.37, Page 2–23V, Unit-2.

b. Write a program that accepts a sentence and calculate the number of digits, uppercase and lowercase letters.

Ans. Refer Q. 3.29, Page 3–23V, Unit-3.

c. Write a program to check an input year is leap year or not.

Ans. Refer Q. 2.8, Page 2–7V, Unit-2.

5. Attempt any **two** part of the following : **(4 × 2 = 8)**

a. There is a file named input. Txt. Enter some positive numbers into the file named Input.Txt. Read the contents of the file and if it is an odd number write it to ODD.TXT and if the number is even, write it to EVEN.TXT.

Ans. Refer Q. 4.15, Page 4–11V, Unit-4.

b. Discuss exception and assertions in python. Explain with a suitable example. Explain any two built-in exceptions.

Ans. This question is out of syllabus from session (2023-24).

c. What is the significance of module in Python ? What are the methods of importing a module ? Explain with suitable example.

Ans. Refer Q. 5.4, Page 5–4V, Unit-5.

6. Attempt any **two** part of the following : **(4 × 2 = 8)**

a. What do you mean by recursion ? Write a recursive function to compute the factorial of an input number N.

Ans. This question is out of syllabus from session (2023-24).

b. Implement selection sort using Python.

Ans. This question is out of syllabus from session (2023-24).

c. What are different types of inheritance supported by Python ? Explain.

Ans. This question is out of syllabus from session (2023-24).

☺☺☺

B. Tech.
(SEM. III) ODD SEMESTER THEORY EXAMINATION, 2022-23
PYTHON PROGRAMMING

Time : 3 Hours **Max. Marks : 100**

Note : 1. Attempt **all** sections. If require any missing data; then choose suitably.

Section-A

1. Attempt **all** questions in brief. **(2 × 10 = 20)**

a. Explain the programming cycle for Python in detail.

Ans. Refer Q. 5.18, Page SQ–30V, Unit-5, Two Marks Questions.

b. Describe the concept of List Slicing with a suitable example.

Ans. Refer Q. 3.33, Page SQ–18V, Unit-3, Two Marks Questions.

c. Show the way to import the module in Python.

Ans. Refer Q. 5.19, Page SQ–31V, Unit-5, Two Marks Questions.

d. Differentiate between Python arrays and lists ?

Ans. Refer Q. 3.28, Page SQ–16V, Unit-3, Two Marks Questions.

e. Define floor division with an example.

Ans. Refer Q. 1.16, Page SQ–3V, Unit-1, Two Marks Questions.

f. Explain the difference between 'append' and 'extend' in Python ?

Ans. Refer Q. 3.34, Page SQ–18V, Unit-3, Two Marks Questions.

g. What is a dictionary in Python ?

Ans. Refer Q. 3.38, Page SQ–20V, Unit-3, Two Marks Questions.

h. What is object oriented programming (OOP) in Python ? Give an example.

Ans. This question is out of syllabus from session (2023-24).

i. What will be the output of the following Python code

```
def count1 (s) :
    vowels = "AEIOUaeiou"
    count = 0
    for c in s :
        if c in vowels :
```

```
        count + = 1
    return count
print (count1 ('I love India'))
```

Ans. Refer Q. 3.35, Page SQ–19V, Unit-3, Two Marks Questions.

j. What will be the output of the following code ?

```
list1 = ['M', 'n', 'k', 'y']
print ("@".join(list1))
```

Ans. Refer Q. 3.36, Page SQ–19V, Unit-3, Two Marks Questions.

Section-B

2. Attempt any **three** of the following : (10 × 3 = 30)

a. Demonstrate five different built in functions used in the string. Write a program to check whether a string is a palindrome or not.

Ans. Refer Q. 3.5, Page 3–4V, Unit-3.

b. Explain the following loops with a flow diagram, syntax, and suitable examples.

i. For

ii. While

Ans. Refer Q. 2.27, Page 2–18V, Unit-2.

c. Explain the continue, break, and pass statements with a suitable example.

Ans. Refer Q. 2.33, Page 2–21V, Unit-2.

d. Develop a program to calculate the reverse of any entered number.

Ans. Refer Q. 2.38, Page 2–23V, Unit-2.

e. Explain the list comprehension with any suitable example.

Ans. Refer Q. 3.11, Page 3–9V, Unit-3.

Section-C

3. Attempt any **one** part of the following : (10 × 1 = 10)

a. Illustrate unpacking sequences, mutable sequences, and list comprehension with examples.

Ans. This question is out of syllabus from session (2023-24).

b. Explain the lambda function. How it is helpful in the higher order function. Explain map() function with a suitable example.

Ans. This question is out of syllabus from session (2023-24).

4. Attempt any **one** part of the following : (10 × 1 = 10)

a. Discuss the different types of argument-passing methods in Python. Explain the variable length argument with any suitable example.

Ans. Refer Q. 3.38, Page 3–28V, Unit-3.

b. Write short notes on the following with a suitable example

i. Encapsulation

ii. Inheritance

Ans. This question is out of syllabus from session (2023-24).

5. Attempt any **one** part of the following : (10 × 1 = 10)

a. Demonstrate the file handling procedure in detail. Write a Python code to create file with 'P.txt' name and write your name and father's name in this file and then read this file to print it.

Ans. Refer Q. 4.18, Page 4–14V, Unit-4.

b. Demonstrate the 'Sieve of Eratosthenes' theorem and write the python function to print prime numbers between 1 to 100.

Ans. This question is out of syllabus from session (2023-24).

6. Attempt any **one** part of the following : (10 × 1 = 10)

a. Develop and write the Python code of selection sort to sort 41, 65, 43, 91, 12, 14, 62 elements. Also, explain its complexity.

Ans. This question is out of syllabus from session (2023-24).

b. Explain binary search with its Python code and complexity.

Ans. This question is out of syllabus from session (2023-24).

7. Attempt any **one** part of the following : (10 × 1 = 10)

a. Explain the importance of exception handling in any object oriented programming language. Explain try exceptions and finally block with any suitable example.

Ans. This question is out of syllabus from session (2023-24).

b. Summarize the 'Tower of Hanoi' puzzle and write its recursive function to implement it.

Ans. This question is out of syllabus from session (2023-24).

☺☺☺

B. Tech.
(SEM. IV) EVEN SEMESTER THEORY EXAMINATION, 2022-23
PYTHON PROGRAMMING

Time : 3 Hours **Max. Marks : 100**

Note : Attempt **all** Sections. If you require any missing data, then choose suitably.

SECTION-A

1. Attempt **all** questions in brief. **(2 × 10 = 20)**

a. Define the programming cycle for Python.

Ans. Refer Q. 5.18, Page SQ–30V, Unit-5, Two Marks Questions.

b. Describe the use of class '__init__()' method.

Ans. This question is out of syllabus from session (2023-24).

c. Explain features of any two python IDEs.

Ans. Refer Q. 5.20, Page SQ–31V, Unit-5, Two Marks Questions.

d. Discuss the purpose of the break and continue statement in loops.

Ans. Refer Q. 2.22, Page SQ–10V, Unit-2, Two Marks Questions.

e. Explain mutable sequences in Python.

Ans. This question is out of syllabus from session (2023-24).

f. Explain the output of following function.

```
defprintalpha (abc_list, num_list) :
    for char in abc_list :
        for num in num_list :
            print(char, num)
    return
printalpha ('a', 'b', 'c'], [1, 2, 3]
```

Ans. Refer Q. 3.32, Page SQ–17V, Unit-3, Two Marks Questions.

g. Discuss the special methods__le__and__ne__.

Ans. This question is out of syllabus from session (2023-24).

h. Discuss various file opening modes of Python.

Ans. Refer Q. 4.2, Page SQ–21V, Unit-4, Two Marks Questions.

i. Compare assert, try-except and raise statements.

Ans. This question is out of syllabus from session (2023-24).

j. Write Python function to perform linear search.

Ans. This question is out of syllabus from session (2023-24).

SECTION-B

2. Attempt any **three** of the following : **(10 × 3 = 30)**

a. Write Python code snippet to display n terms of Fibonacci series using recursion.

Ans. This question is out of syllabus from session (2023-24).

b. Write and explain an algorithm through Python code to generate prime numbers.

Ans. Refer Q. 2.21, Page 2–14V, Unit-2.

c. Compare list and tuple data structure with suitable examples. Explain the concept of list comprehension.

Ans. Refer Q. 3.23, Page 3–18V, Unit-3.

d. Explain the Python implementation of stack ADT.

Ans. This question is out of syllabus from session (2023-24).

e. Describe Python program to implement selection sort. Simulate your code on the following data 2, 52, 1, 5, 2, 54, 62, 64.

Ans. This question is out of syllabus from session (2023-24).

SECTION-C

3. Attempt any **one** part of the following : **(10 × 1 = 10)**

a. Discuss various categories of operators in Python. Find and explain stepwise solution of following expressions if a = 3, b = 5, c = 10 :

i. a & b << 2//52 + c^b**

ii. b >> a2 << 2 >> b**2^c**3**

Ans. Refer Q. 1.5, Page 1–5V, Unit-1.

b. Discuss why Python is interpreted language. Explain history and features of python while comparing Python version 2 and 3.

Ans. Refer Q. 5.25, Page 5–22V, Unit-5.

4. Attempt any **one** part of the following : **(10 × 1 = 10)**

a. Write a Python program to count the vowels present in given input string. Explain the output of program through example.

Ans. Refer Q. 3.31, Page 3–24V, Unit-3.

b. Write a program that accepts sequence of lines as input and prints the lines after making all characters in the sentence capitalized.
e.g., If Input :
Hello world
Practice makes perfect
Then, Output :
HELLO WORLD
PRACTICE MAKES PERFECT

Ans. Refer Q. 3.33, Page 3–24V, Unit-3.

5. Attempt any **one** part of the following : **(10 × 1 = 10)**

a. Discuss list data structure of Python. Explain various inbuilt methods of list with suitable example of each.

Ans. Refer Q. 3.10, Page 3–9V, Unit-3.

b. Compare list and dictionary data structure. Explain various dictionary methods with suitable examples of each.

Ans. Refer Q. 3.24, Page 3–18V, Unit-3.

6. Attempt any **one** part of the following : **(10 × 1 = 10)**

a. Discuss the concept of Iterators in Python. Explain, how we can create text file in Python ? Describe Python program to write the number of letters and digits in given input string into a File object.

Ans. Refer Q. 4.16, Page 4–12V, Unit-4.

b. Illustrate through examples, how we can implement different type of inheritance in Python ? Explain the static, instance and class method through suitable example.

Ans. This question is out of syllabus from session (2023-24).

7. Attempt any **one** part of the following : **(10 × 1 = 10)**

a. Explain the Tower of Hanoi problem and its recursive solution in Python.

Ans. This question is out of syllabus from session (2023-24).

b. Write Python program to implement merge sort algorithm. Discuss, how we can compare the time complexity of two algorithms ?

Ans. This question is out of syllabus from session (2023-24).

☺☺☺