UNIT -2

Getting Stating with Python

1. INTRODUCTION

- Computers are used for solving various day-to-day problems.
- It is pertinent to mention that computers themselves can not solve a problem.
- Precisestep-by-stepinstructionsshouldbegivenbyustosolvetheproblem.
- Thus, the success of a computer in solving a problem depends on how correctly and precisely we define the problem, design a solution (algorithm) and implement the solution (program) using a programming language.
- Thus, problem solving is the process of identifying a problem, developing an algorithm for the identified problem and finally implementing the algorithm to develop a computer program.

2. Steps for ProblemSolving

There are four steps in problem solving

- Analysing the problem, Developing an Algorithm, Coding, Testing and Debugging

2.1 Analysing the problem

- It is important to clearly understand a problem before we begin to find the solution for it.
- If we are not clear as to what is to be solved,we may end up developing a program which may not solve our purpose.
- By analysing a problem, we would be able to figure out what are
- the inputs that our program should accept and the outputs that it should produce.

2.2 Developing an Algorithm

- The solution for a problem is represented in step by step procedure called analgorithm.
- For a given problem, more than one algorithm is possible and we have to select the most suitable solution.

2.3 Coding

- After finalising the algorithm, we need to convert the algorithm into the format which can be understood by the computer to generate the desired solution.

2.4 Testing and Debugging

- The program created should be tested on various parameters.
- The program should meet the requirements of the user.
- In the presence of syntactical errors, no output will be obtained.
- In case the output generated is in correct, then the program should be checked for logical errors, if any.

3. Algorithm

Algorithm is the step by step procedure for solving the problem. Suppose following are the steps required for an activity 'riding a bicycle':

- Remove the bicycle from the stand,
- Sit on the seat of the bicycle,
- Start peddling,
- Use breaks when ever needed and
- Stop on reaching the destination.

Example:
Algorithm to find square of anumber.

```
Step1: Input a number and store it to num
Step 2: Compute num * num and store
it in square
Step 3:Print square
```

3.1 Why do we need an Algorithm

- Writing an algorithm is mostly considered as a first step to programming.
- Once we have an algorithm to solve a problem, we can write the computer program for giving instructions to the computer in high level language.
- If the algorithm is correct, computer will run the program correctly, everytime.
- So, the purpose of using an algorithm is to increase the reliability, accuracy and efficiency of obtaining solutions.

Characteristics of a good algorithm

• Precision—the steps are precisely stated or defined.
• Uniqueness—results of each step are uniquely defined and only

depend on the input and the result of the preceding steps.

• Finiteness—the algorithm always stops after a finite number of steps.

• Input—the algorithm receives some input.

• Output—the algorithm produces some output.

While writing an algorithm ,it is required to clearly identify the following:

• The input to be taken from the user

• Processing or computation to be performed to get the desired result

• The output desired by the user

4. Representation of Algorithms

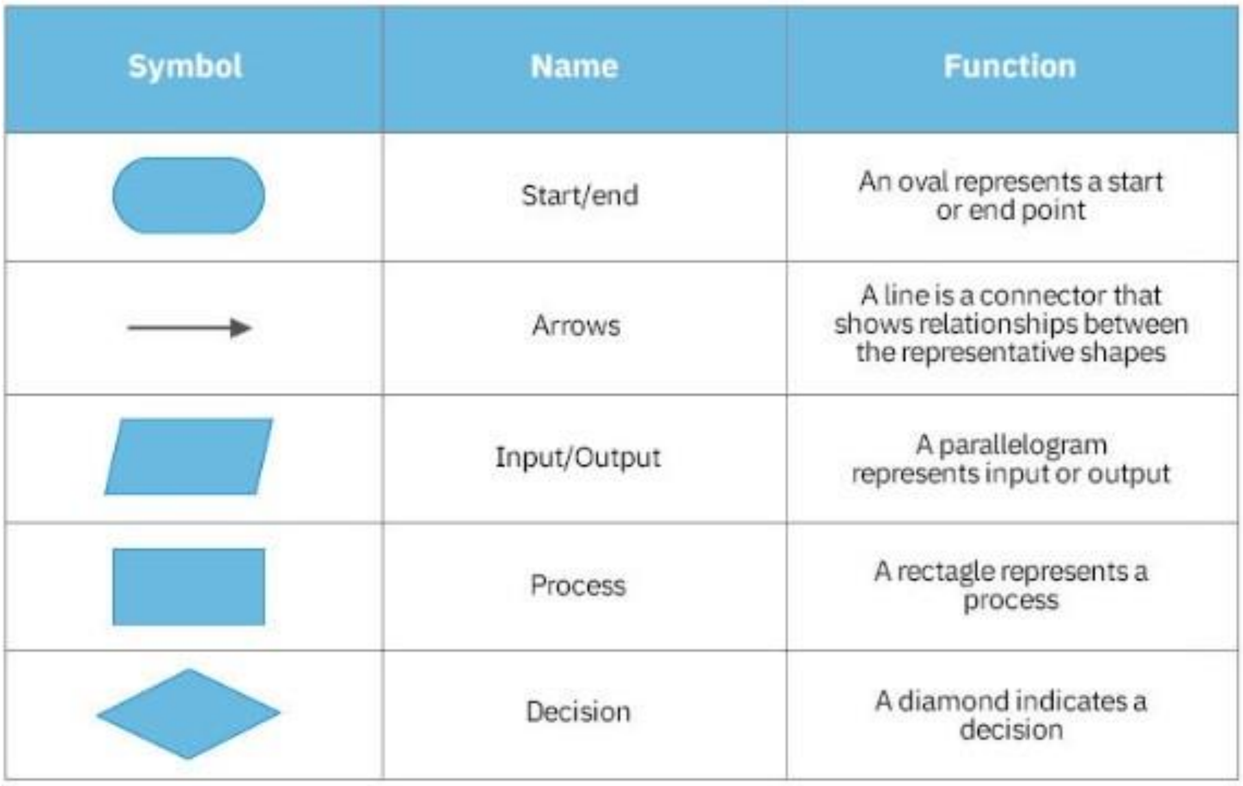
There are two common methods of representing an algorithm —flowchart and pseudocode. Either of the methods can be used to represent an algorithm while keeping in mind the following:

• it show cases the logic of the problem solution, excluding any implementational details

• it clearly reveals the flow of control during execution of the program

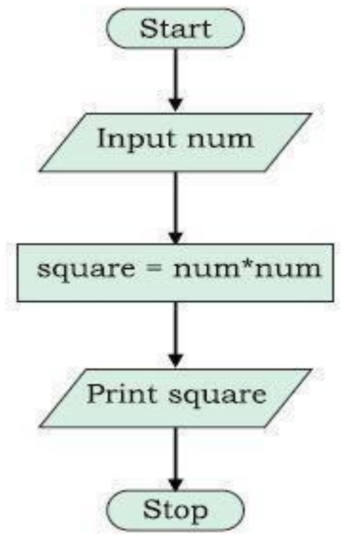
4.1 Flowchart —Visual Representation of Algorithms

- A flow chart is a visual representation of an algorithm.
- A flowchart is a diagram made up of boxes, diamonds and other shapes, connected by arrows.



Symbol	Name	Function
⬭	Start/end	An oval represents a start or end point
→	Arrows	A line is a connector that shows relationships between the representative shapes
▱	Input/Output	A parallelogram represents input or output
▭	Process	A rectagle represents a process
◆	Decision	A diamond indicates a decision

Flow chart to calculate square of a number



4.2 Pseudocode

A pseudocode (pronounced Soo-doh-kohd) is another way of representing an algorithm. It is considered as a non-formal language that helps programmers to write algorithm. The word“pseudo” means “not real,” so “pseudocode” means “not real code”. Following are some of the frequently used keywords while writing pseudocode:

- INPUT/•COMPUTE/•PRINT/•INCREMENT/•DECREMENT
- IF/ELSE,•WHILE,•TRUE/FALSE

Example:

Pseudo code for the sum of two numbers will be:

```
Input num1
input num2
COMPUTE Result = num1 + num2
PRINT Result
```

4.3 Coding

- Once analgorithm is finalised,it should be coded in ahigh-level programming language as selected by the programmer.
- The ordered set of instructions are written in that programming language by following its syntax.
- Syntax is the set of rules or grammar that governs the formulation of the statements in the language,such as spellings,order of words, punctuation,etc.

4.4 Decomposition

- The basic idea of solving a complex problem by decomposition is to 'decompose' or break down a complex problem into smaller sub problems.

Answer the Following Questions **(Very ShortAnnswers)**

1. Define Algorithm
2. What is decomposition?
3. Why do we need Algorithm?
4. What is meant by Debugging?

Answer the Following Questions **(Short Answers)**

1. Write an algorithm to find the greatest among two different numbers.
2. Write a pseudo code to calculate the factorial of a number.
3. Write an algorithm to find greater among three numbers

Answer the Following Questions **(Long Answers)**

1. Write pseudo code and draw flow chart to accept number still the user enters and then find their average.
2. Write a pseudocode and draw a flowchart where multiple conditions are checked to categorize a person as either child(<13), teenager(>=13but<20) or adult(>=20),based on age specified:
3. Write an algorithmth at accepts four numbers as input and find the largest and smallest of them.

PYTHON PROGRAMMING FUNDAMENTAL:

5.1.1 Features of Python

• Python is a high level language. It is a free and open source language.
• It is an interpreted language, as Python programs are executed by an interpreter.
• Python programs are easy to understand as they have a clearly defined syntax and relatively simple structure.
• Python is case-sensitive. For example, NUMBER and number are not same in Python.
• Python is portable and platform independent, means it can run on various operating systems and hardware platforms.
• Python has a rich library of predefined functions.
• Python uses indentation for blocks and nested blocks.

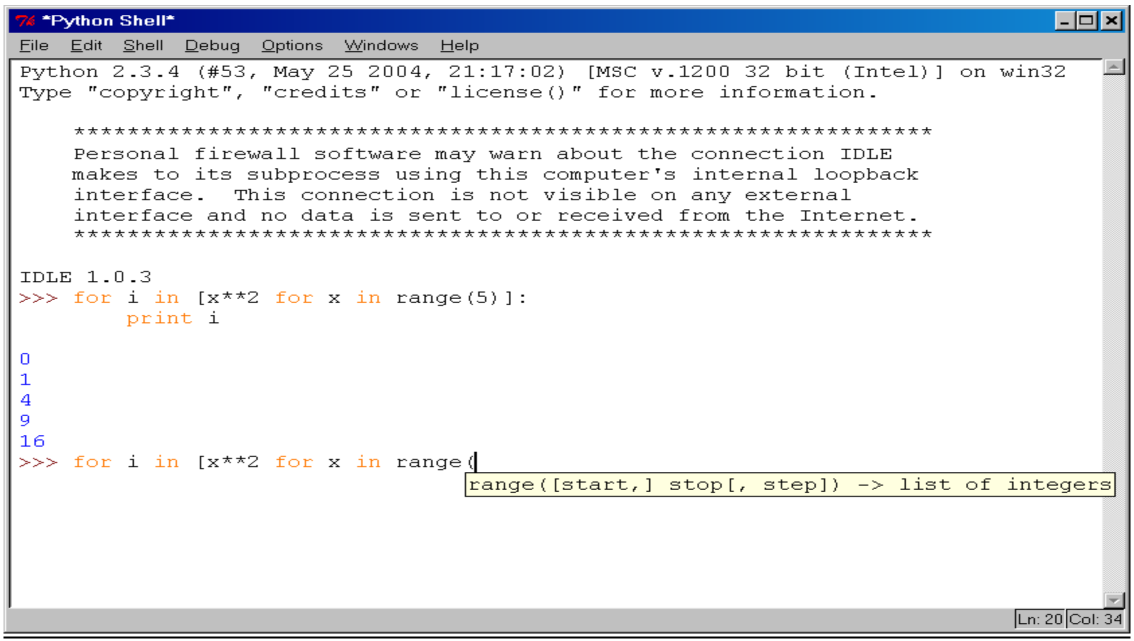
How to Install Python

- Python is pre-installed on most Unix systems, including Linux and MAC OS X
- The pre-installed version may not be the most recent one (2.6.2 and 3.1.1 as of Sept 09)
- Download from http://python.org/download/
- Python comes with a large library of standard modules
- There are several options for an IDE

IDLE – works well with Windows

IDLE (Integrated Development Environment)

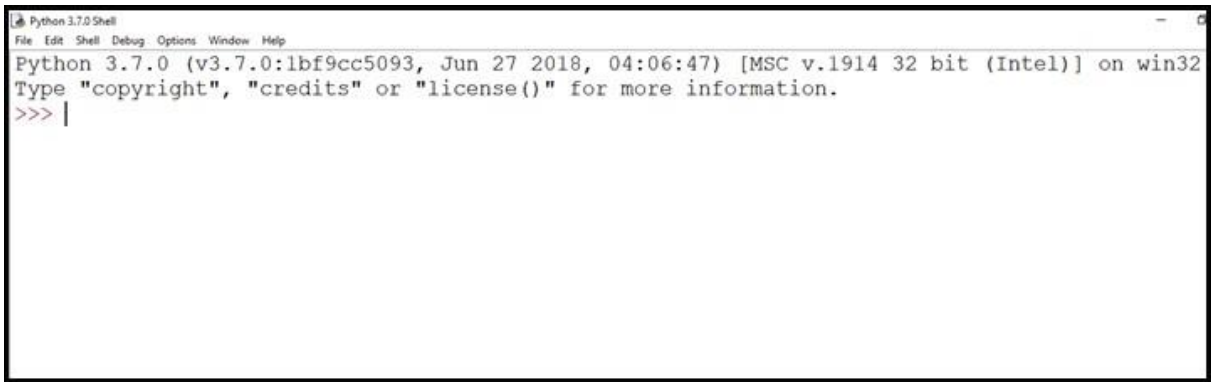
- IDLE is an Integrated DeveLopment Environment for Python, typically used on Windows
- Multi-window text editor with syntax highlighting, auto-completion, smart indent and other.
- Python shell with syntax highlighting.
- Integrated debugger with stepping, persistent breakpoints, and call stack visibility



```
Python 2.3.4 (#53, May 25 2004, 21:17:02) [MSC v.1200 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.

    ****************************************************************
    Personal firewall software may warn about the connection IDLE
    makes to its subprocess using this computer's internal loopback
    interface.  This connection is not visible on any external
    interface and no data is sent to or received from the Internet.
    ****************************************************************

IDLE 1.0.3
>>> for i in [x**2 for x in range(5)]:
        print i

0
1
4
9
16
>>> for i in [x**2 for x in range(
                                  range([start,] stop[, step]) -> list of integers
```

5.1.2 Working with Python

To write and run (execute) a Python program, we need to have a Python interpreter installed on our computer or we can use any online Python interpreter. The interpreter is also called Python shell. A sample screen of Python interpreter is shown in Figure:



```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> |
```

In the above screen, the symbol >>> is the Python prompt, which indicates that the interpreter is ready to take instructions. We can type commands or statements on this prompt to execute them using a Python interpreter.

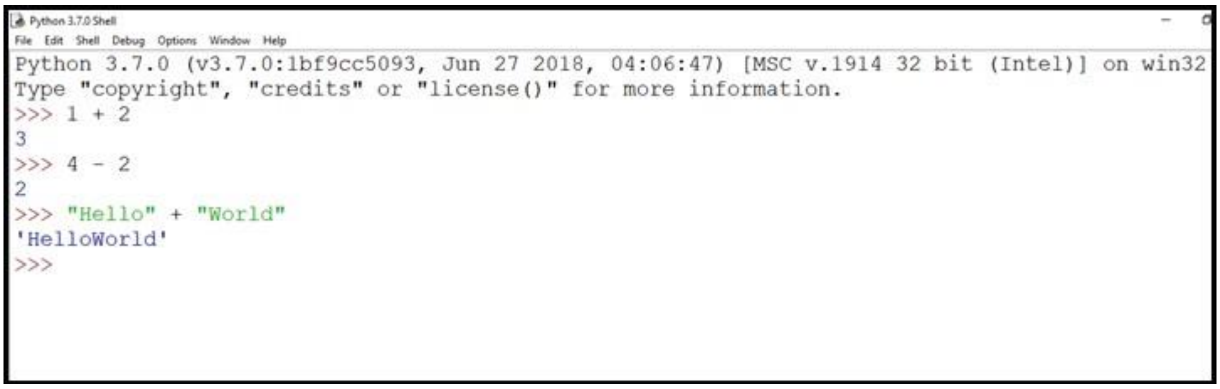
5.1.3 Execution Modes

There are two ways to use the Python interpreter:
a) Interactive mode
b) Script mode

Interactive mode allows execution of individual statement instantaneously. Whereas, Script mode allows us to write more than one instruction in a file called Python source code file that can be executed.

(A) Interactive Mode

To work in the interactive mode, we can simply type a Python statement on the >>> prompt directly. As soon as we press enter, the interpreter executes the statement and displays the result(s), as shown in Figure:



```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 1 + 2
3
>>> 4 - 2
2
>>> "Hello" + "World"
'HelloWorld'
>>>
```

Working in the interactive mode is convenient for testing a single line code for instant execution. But in the interactive mode, we cannot save the statements for future use and we have to retype the statements to run them again.

(B) Script Mode

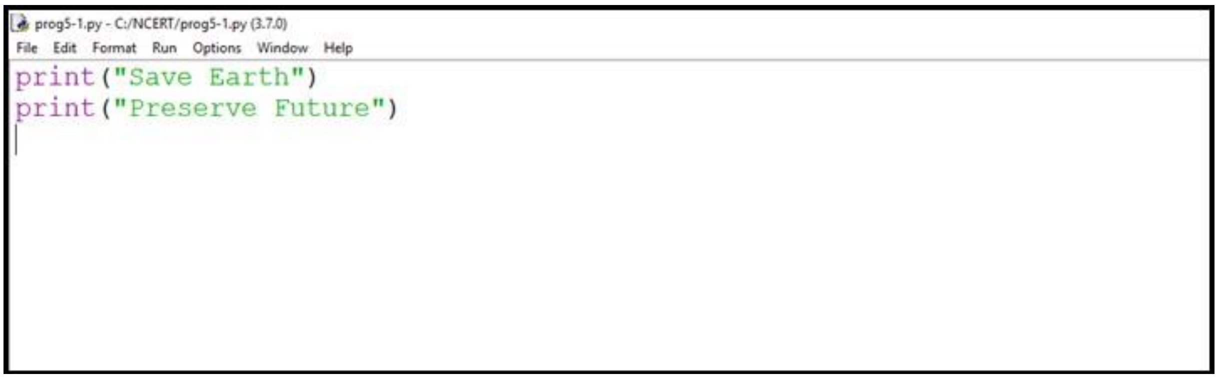
In the script mode, we can write a Python program in a file, save it and then use the interpreter to execute it. Python scripts are saved as files where file name has extension “.py”. By default, the Python scripts are saved in the Python installation folder. To execute a script, we can either:

a) Type the file name along with the path at the prompt. For example, if the name of the file is prog5-1.py, we type prog5-1.py. We can otherwise open the program directly from IDLE.

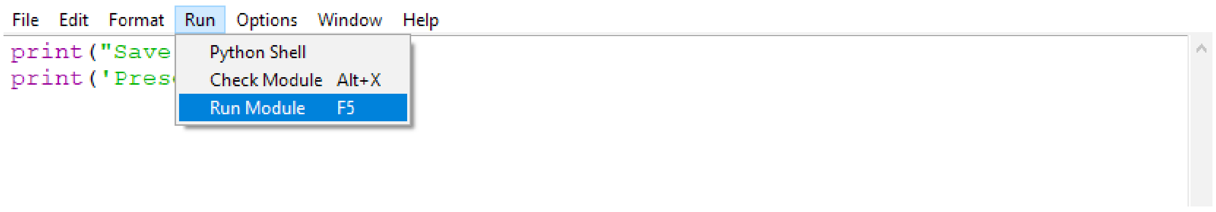
b) While working in the script mode, after saving the file, click [Run]->[Run Module] from the menu as shown in Figure below.

c) The output appears on shell as shown in figure below:

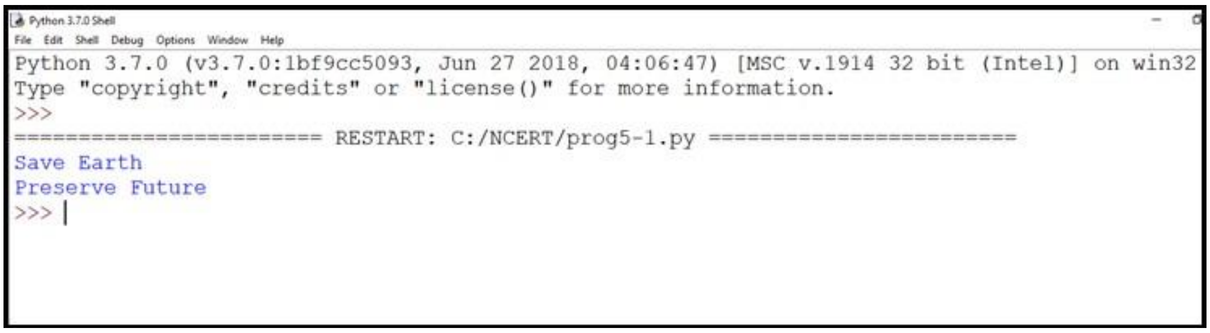
Program 5-1 Write a program to show print statement in script mode.


prog5-1.py - C:/NCERT/prog5-1.py (3.7.0)
File Edit Format Run Options Window Help

```
print("Save Earth")
print("Preserve Future")
```



```
print("Save
print('Pres
```



```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
======================== RESTART: C:/NCERT/prog5-1.py ========================
Save Earth
Preserve Future
>>> |
```

Basic DataTypes:

- **Integers (default for numbers)- Number without decimal point**
  z = 5 / 2   # Answer 2.5, integer division
- **Floats : Number with decimal Point**
  x = 3.456

- **Strings -**

Can use “” or ‘’ to specify with “abc” == ‘abc’

Use triple double-quotes for multi-line strings or strings than contain both ‘ and “ inside of them:
“““a‘b“c”””

Comments :

- **Comments are very important while writing a program. It describes what's going on inside a program so that a person looking at the source code does not have a hard time figuring it out.**
- **In Python, we use the hash (#) symbol to start writing a comment.**
- **If we have comments that extend multiple lines, one way of doing it is to use hash (#) in the beginning of each line.**
- **For example:**

```
#This is a long comment
#and it extends
#to multiple lines
```

- **Another way of doing this is to use triple quotes, either ''' or """.**
- **These triple quotes are generally used for multi-line strings. But they can be used as multi-line comment as well.**

```
"""This is also a
perfect example of
multi-line comments"""
```

Assignment :

- ***Binding a variable*** **in Python means setting a** ***name*** **to hold a** ***reference*** **to some** ***object***
  Assignment creates references, not copies
- **Names in Python do not have an intrinsic type, objects have types**
  Python determines the type of the reference automatically based on what data is assigned to it
- **You create a name the first time it appears on the left side of an assignment expression:** **x = 3**
- **A reference is deleted via garbage collection after any names bound to it have passed out of scope**
- **Python uses** ***reference semantics*** **(more later)**
- **You can assign to multiple names at the same time**

```
>>> x, y = 2, 3
>>> x
2
>>> y
3
```

- **This makes it easy to swap values**

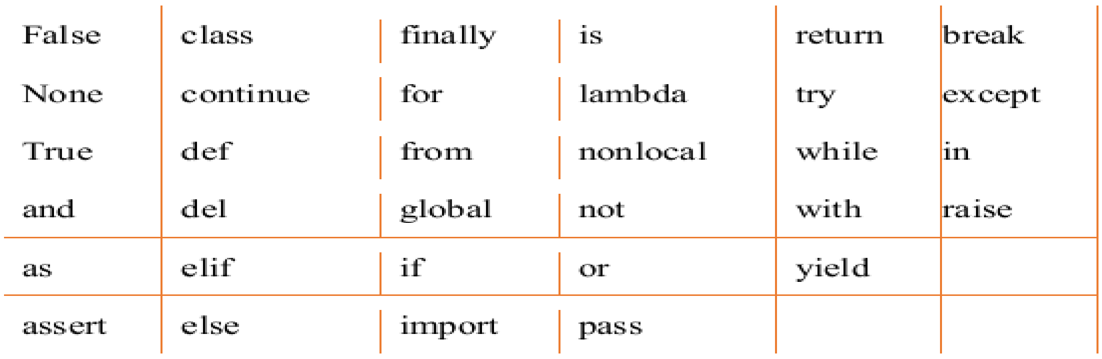
```
>>> x, y = y, x
```

- **Assignments can be chained**

```
>>> a = b = x = 2
```

Python keywords

Reserved words in the library of a language.There are 3 3keywords in python.



False	class	finally	is	return	break
None	continue	for	lambda	try	except
True	def	from	nonlocal	while	in
and	del	global	not	with	raise
as	elif	if	or	yield	
assert	else	import	pass		

All the keywords are in lowercase except 03 keywords(True, False, None)

Identifier

- **An identifier is a name given to entities like class, functions, variables, etc.**
- **It helps to differentiate one entity from another.**

Rules for naming Identifier :

- **Identifiers can be a combination of letters in lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore _.**
- **Names like myClass, var_1 and print_this_to_screen, all are valid example.**
- **An identifier cannot start with a digit. 1variable is invalid, but variable1 is perfectly fine.**
- **Keywords cannot be used as identifiers.**
- **We cannot use special symbols like !, @, #, $, % etc. in our identifier.**

bob Bob _bob _2_bob_ bob_2 BoB

- **There are some reserved words:**

and, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while

Program : Write a program to display values of variables in Python.

```
#To display value of variable
message = "hello"
print(message)
)

Output:
Hello
```

Program : Write a Python program to find the area of a rectangle given that its length is 10 units and breadth is 20 units.

```
#To find the area of a rectangle
length = 10
breadth = 20
area = length * breadth
print("Area of Rectangle=", area)

Output: 200
```

Do it yourself:

1. Write a Python program to find the sum and subtract of two numbers
2. Find the simple interest. Principle amount, rate of interest and time given by user.

Constants

- A constant is a type of variable whose value cannot be changed.
- It is helpful to think of constants as containers that hold information which cannot be changed later.

Literals

- Literals
- Literal is a raw data given in a variable or constant. In Python, there are various types of literals they are as follows:
    - Numeric Literals
    - Numeric Literals are immutable (unchangeable). Numeric literals can belong to 3 different numerical types Integer, Float and Complex.
    - a = 0b1010 #Binary Literals
    - b = 100 #Decimal Literal
    - c = 0o310 #Octal Literal
    - d = 0x12c #Hexadecimal Literal
- #Float Literal
    - float_1 = 10.5
    - float_2 = 1.5e2
- #Complex Literal
    - x = 3.14j
- print(a, b, c, d)
- print(float_1, float_2)
- print(x, x.imag, x.real)

OUTPUT

```
10 100 200 300
10.5 150.0
3.14j 3.14 0.0
```

Python Data Types

Built-in Data Types

In programming, data type is an important concept. Variables can store data of different types, and different types can do different things. Python has the following data types built-in by default, in these categories:

	Str
Numeric Types:	int, float, complex
Sequence Types:	list, tuple, range
Mapping Type:	dict
Set Types:	set, frozen set
Boolean Type:	bool

Binary Types: bytes, byte array, memory view

Let us now try to execute few statements in interactive mode to determine the data type of the variable using built-in function type().
Example :

```
>>> num1 = 10
>>> type(num1)
<class 'int'>
>>> num2 = -1210
>>> type(num2)
<class 'int'>
>>> var1 = True
>>> type(var1)
<class 'bool'>
```

Operators

An operator is used to perform specific mathematical or logical operation on values. The values that the operators work on are called operands. For example, in the expression 10 + num, the value 10, and the variable num are operands and the + (plus) sign is an
operator. Python supports several kinds of operators whose categorisation is briefly explained in this section

Python divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators

Python Arithmetic Operators

Arithmetic operators are used with numeric values to perform common mathematical operations:

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

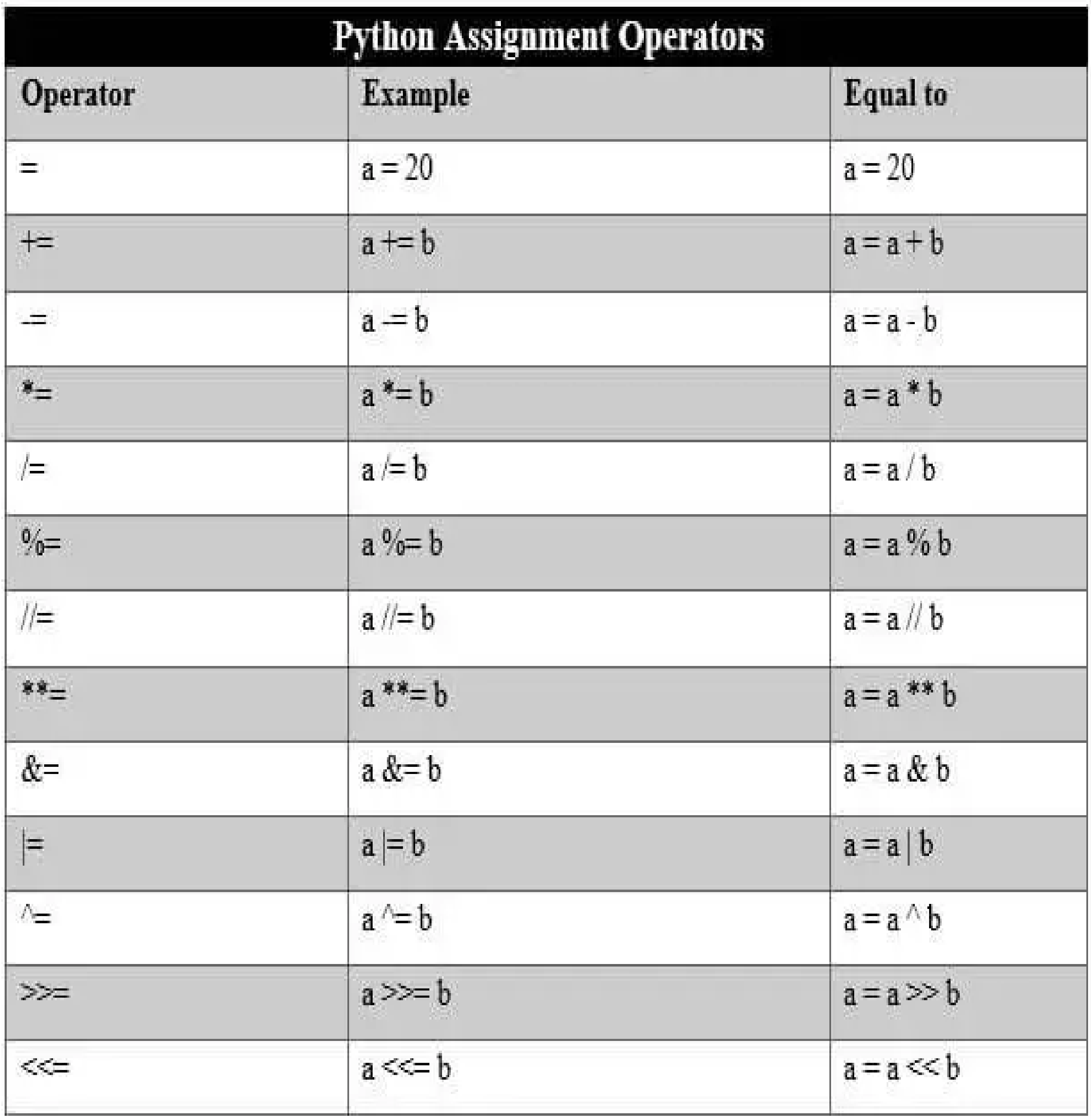
Concatenation

Concatenating means obtaining a new string that contains both of the original strings. In Python, there are a few ways to concatenate or combine strings. The new string that is created is referred to as a string object. In order to merge two strings into a single object, you may use the + operator.

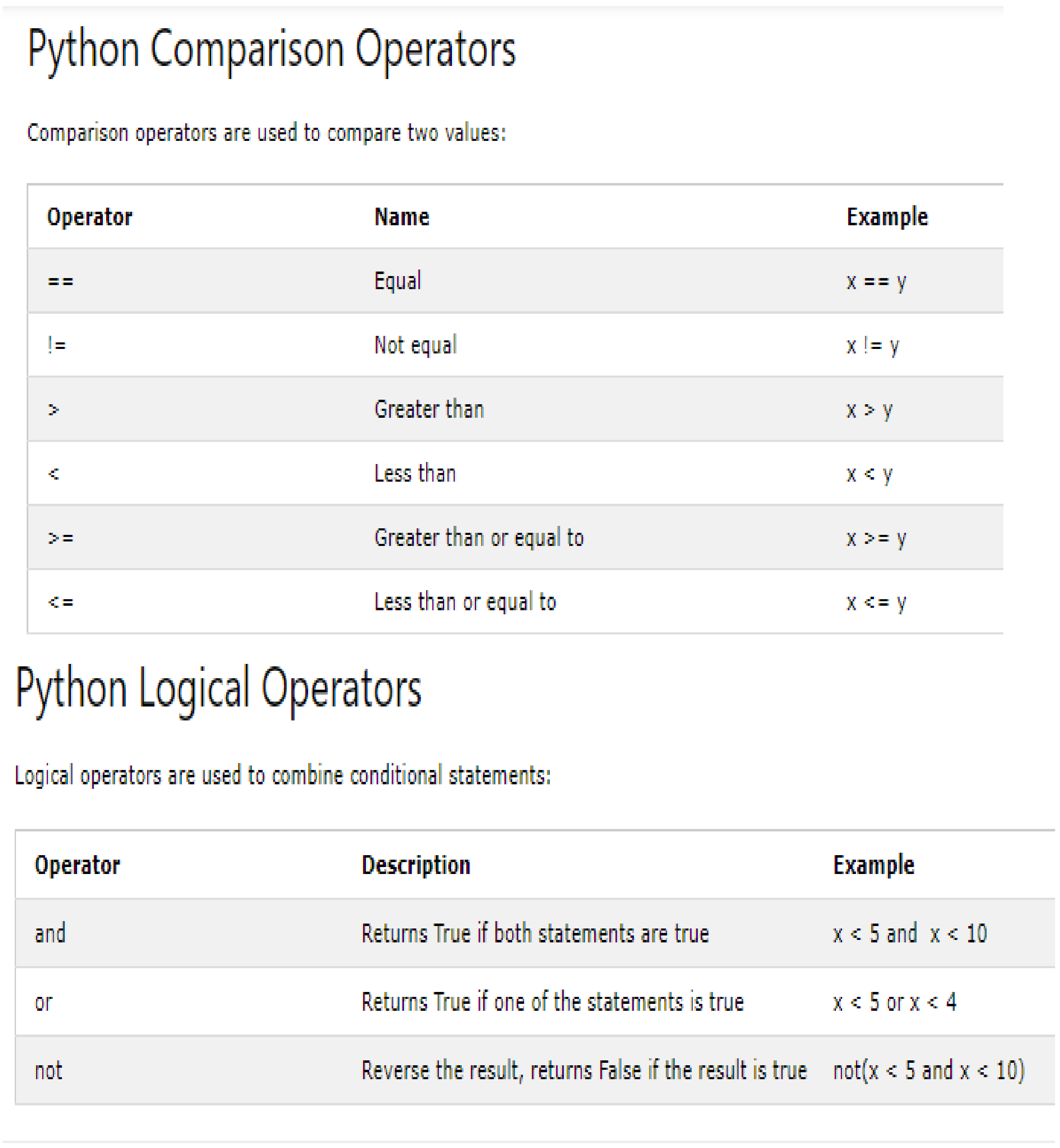
For Example
>>>"CBSE "+"India"

Output is CBSEIndia

Replication. : The multiplication operator acts as a replication operator when we have one string and one integer as operands. What is replication? First, understand the meaning of the word replication**.**
>>>"CBSE" 3
Output is CBSECBSECBSE



Python Assignment Operators		
Operator	Example	Equal to
=	a = 20	a = 20
+=	a += b	a = a + b
-=	a -= b	a = a - b
*=	a *= b	a = a * b
/=	a /= b	a = a / b
%=	a %= b	a = a % b
//=	a //= b	a = a // b
**=	a **= b	a = a ** b
&=	a &= b	a = a & b
|=	a |= b	a = a | b
^=	a ^= b	a = a ^ b
>>=	a >>= b	a = a >> b
<<=	a <<= b	a = a << b



Python Comparison Operators

Comparison operators are used to compare two values:

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Python Logical Operators

Logical operators are used to combine conditional statements:

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Python Identity Operators

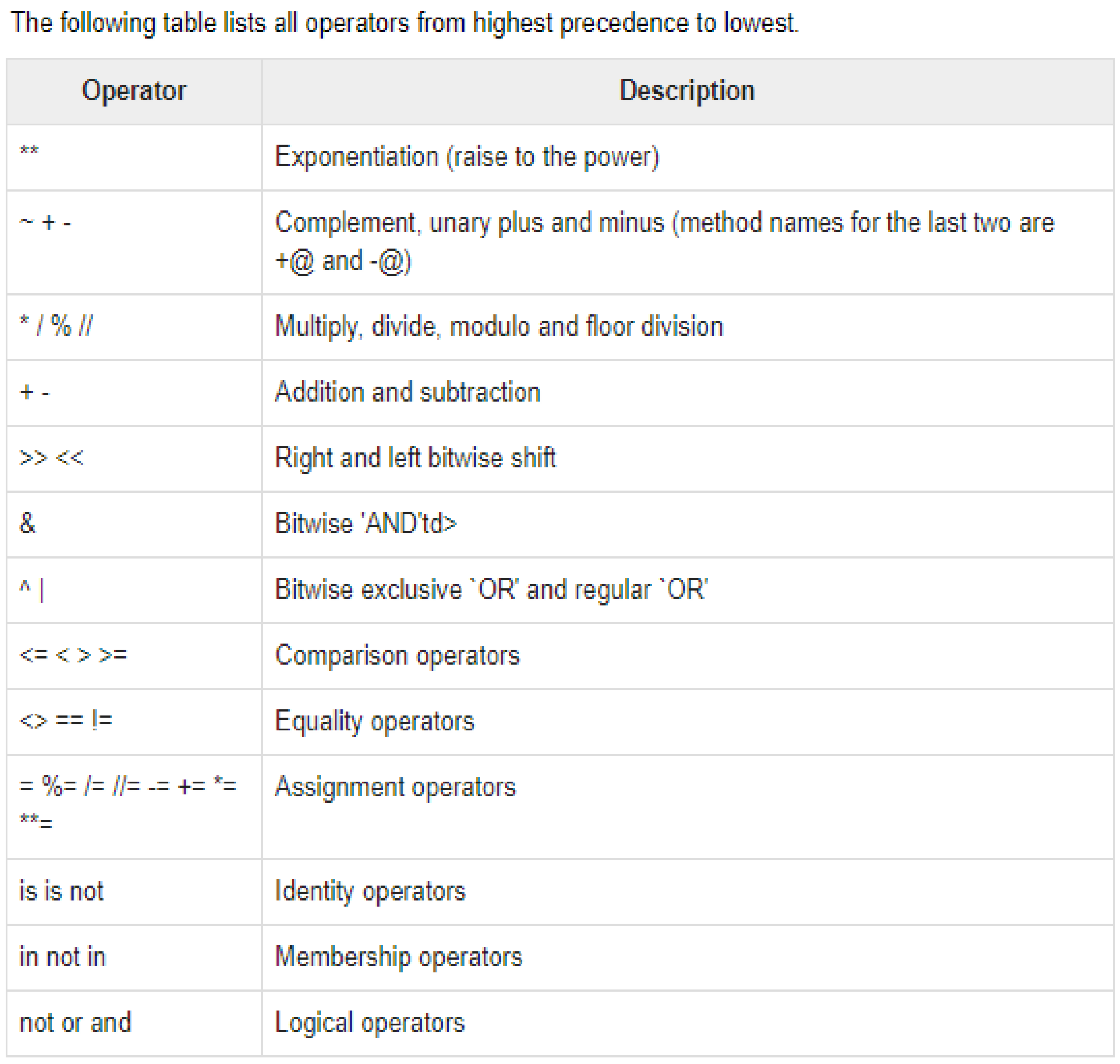
Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

Python Membership Operators

Membership operators are used to test if a sequence is presented in an object:

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y



The following table lists all operators from highest precedence to lowest.

Operator	Description
**	Exponentiation (raise to the power)
~ + -	Complement, unary plus and minus (method names for the last two are +@ and -@)
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'td>
^ |	Bitwise exclusive `OR' and regular `OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

Statement

In Python, a statement is a unit of code that the Python interpreter can execute.

Example

```
>>> x = 4        #assignment statement
>>> cube = x ** 3       #assignment statement
>>> print (x, cube)     #print statement

        4 64
```

- **Multi-line statement**
  In Python, end of a statement is marked by a newline character. But we can make a statement extend over multiple lines with the line continuation character (\).

For example:

```
a = 1 + 2 + 3 + \
4 + 5 + 6 + \
7 + 8 + 9
```

Input and Output

Sometimes, a program needs to interact with the user’s to get some input data or information from the end user and process it to give the desired output. In Python, we
have the input() function for taking the user input. The input() function prompts the user to enter data. It accepts all user input as string. The user may enter a number or a string but the input() function treats them as strings only. The syntax for input() is:

input ([Prompt])

Example

```
>>>fname = input("Enter your first name: ")
Enter your first name: Arnab

>>> age = input("Enter your age: ")
Enter your age: 19

>>> type(age)
<class 'str'>
```

Example

```
#function int() to convert string to integer
>>> age = int( input("Enter your age:"))

Enter your age: 19
>>> type(age)
<class 'int'>
```

Example

Statement	**Output**
print("Hello")	Hello
print(10*2.5)	25.0
print("I" + "love" + "my" + "country")	Ilovemycountry
print("I'm", 16, "years old")	I'm 16 years old

More Examples:

```
print(1,2,3,4)
# Output: 1 2 3 4

print(1,2,3,4,sep='*')
# Output: 1*2*3*4

print(1,2,3,4,sep='#',end='&')
# Output: 1#2#3#4&
```

Type Conversion

The process of converting a Python data type into another data type is known as type conversion. There are mainly two types of type conversion methods in Python: implicit type conversion and explicit type conversion

Explicit Conversion

Explicit conversion, also called type casting happens when data type conversion takes place because the programmer forced it in the program. The general form of an explicit data type conversion is:

(new_data_type) (expression)

With explicit type conversion, there is a risk of loss of information since we are forcing an expression to be of a specific type. For example, converting a floating value of x = 20.67 into an integer type, i.e., int(x) will discard the fractional part .67. Following are some of the functions in Python that are used for explicitly converting an expression or a variable to a different type.

Explicit type conversion functions in Python

Function	**Description**
int(x)	Converts x to an integer
float(x)	Converts x to a floating-point number
str(x)	Converts x to a string representation
chr(x)	Converts x to a character

Program of explicit type conversion from int to float.

```
#Explicit type conversion from int to float
num1 = 10
num2 = 20
num3 = num1 + num2
print(num3)
print(type(num3))
num4 = float(num1 + num2)
print(num4)
print(type(num4))
```

Output:

```
30
<class 'int'>
30.0
<class 'float'>
```

Implicit Conversion

Implicit conversion, also known as coercion, happens when data type conversion is done automatically by Python and is not instructed by the programmer.

Program to show implicit conversion from int to float.

```
#Implicit type conversion from int to float
num1 = 10 #num1 is an integer
num2 = 20.0 #num2 is a float
sum1 = num1 + num2 #sum1 is sum of a float and an integer
print(sum1)
print(type(sum1))
```

Output:

```
30.0
<class 'float'>
```

MCQ:

Q.1 Who developed the Python language?

a) Zim Den
b)Wick van Rossum
c) Guido van Rossum
d) NieneStom

Q.2.. Which of the following is an invalid variable?

a) my_string_1
b) 1st_string
c) foo
d) _amount

Q.3.. Which one of these is floor division?

a) /
b) //
c) %
d) None of the mentioned

Q.4.. Which character is used in Python to make a single line comment?

a) /
b) //
c) #
d) !

Q.5.. If x=3.123, then int(x) will give ?

a) 1

b) 0

c) 1

d) 3

Answers :

Q.1. C	Q.2. B	Q.3. B
Q.4. C	Q.5. D	

Practice Questions

1. Write a Python program to convert temperature in degree Celsius to degree Fahrenheit. If water boils at 100 degree C and freezes as 0 degree C, use the program to find out what is the boiling point and freezing point of water on the Fahrenheit scale.
   (Hint: T(°F) = T(°C) × 9/5 + 32)

2. Write a Python program to calculate the amount payable if money has been lent on simple interest. Principal or money lent = P, Rate of interest = R% per annum and Time = T years.P,R,T are given by user. Then Simple Interest (SI) = (P x R x T)/ 100.
3. Amount payable = Principal + SI

4. Write a program to enter two integers and perform all arithmetic operations on them.

5. Write a program to swap two numbers using a third variable.

6. Write a program to swap two numbers without using a third variable.

DEBUGGING

Errors occurring in programs can be categorized as:

i) Syntax error

ii) Logical error

iii) Run Time error

Syntax Errors

Python has its own rules that determine its syntax. The interpreter interprets the statements only if it is syntactically (as per the rules of Python) correct. For example, parentheses must be in pairs, so the expression (10 + 12) is syntactically correct, whereas (7 + 11 is not due to absence of right parenthesis. Such errors need to be removed before the execution of the program

Logical Errors

Logical errors are also called semantic errors as they occur when the meaning of the program (its semantics) is not correct.For example, if we wish to find the average of two numbers 10 and 12 and we write the code as 10 + 12/2, it would run successfully and produce the result 16. Surely, 16 is not the average of 10 and 12. The correct code to find the average should have been (10 + 12)/2 to give the correct output as 11.

Runtime Error

A runtime error causes abnormal termination of program while it is executing. Runtime error is when the statement is correct syntactically, but the interpreter cannot execute it. Runtime errors do not appear until after the program starts running or executing.

Flow of Control

Sequential flow : In sequential flow instructions of a program are executed one after the another
Conditional Flow : In Conditional flow , flow of control of instructions in a program are changed based on a condition.
Iterative Flow : In iterative flow , a set of instructions are executed repeatedly based on a certain condition
Flow control statements are used to control the flow of execution depending upon the specified condition/logic.

There are three types of control statements.

Decision Making Statements /If control statement
Iteration Statements(Loop control statement, for, while)
Jump Statements(break,continue)

Decision Making Statements /If control statement

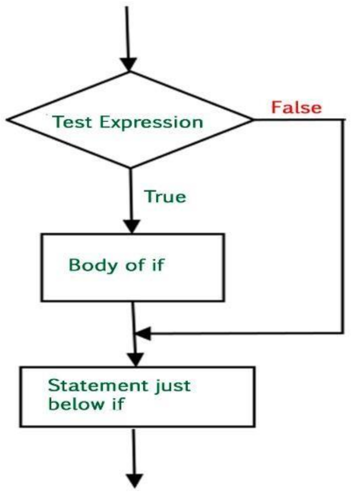
Decision making statement used to control the flow of execution of program depending upon condition.
Types of decision making statement.

1. If statements
2. if-else statements
3. if-else-elif ladder

if statements

An if statement is a programming conditional statement that, if proved true, performs a function or displays information.
Syntax

```
if <condition>:
        Statement(s)
```



Example:

```
a=2
b=3
if a < b :
    print( 'a is greater' )
```

Output : a is greater

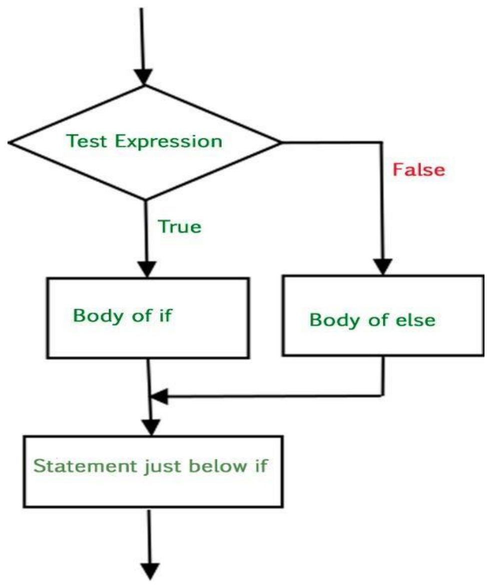
if-else statement

The statements inside the if block are executed only when condition is True, otherwise the statements in the else block are executed.

Syntax

```
if <condition>:
      Statement(s)
else:
      statements
```

Flow Chart



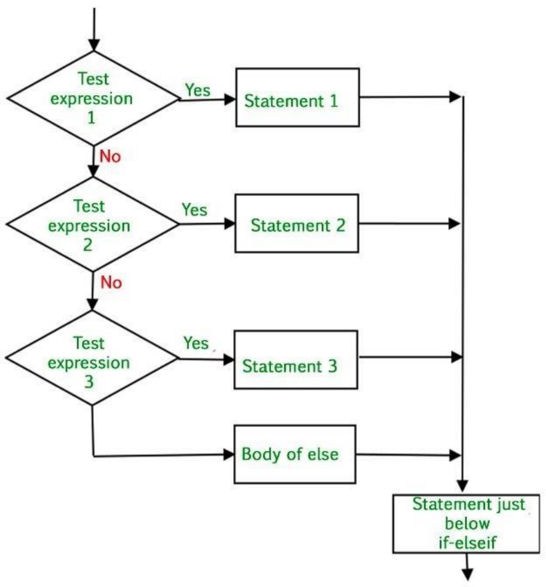
Example:

```
a=5
b=6
if a < b :
    print( 'b is greater' )
else:
    print( 'a is greater' )
```
Output : b is greater

if elif statement

The if...elif...else statement allows you to check for multiple test expressions and execute different codes for more than two conditions.

FlowChart



Example :

```
if 51<5:
    print("False, statement skipped")
elif 6<5:
    print("False, statement skipped")
elif 0<3:
    print("true, block will execute")
else:
    print("If all fails.")
```
output : true, block will execute

Iteration statements

Iteration statements(loop) are used to execute a block of statements as long as the condition is true.

Python Iteration (Loops) statements are of three type:-

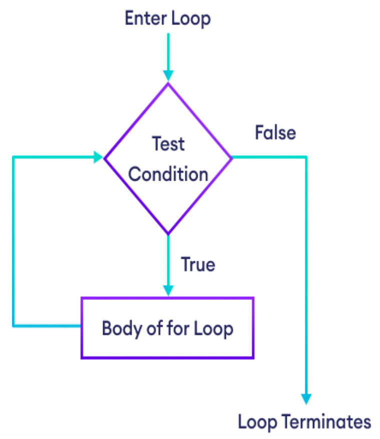
1. while Loop
2. for Loop

WHILE LOOP

It is used to execute a block of statement if a given condition is true. And when the condition become false, the control will come out of the loop. The condition is checked every time at the beginning of the loop.

Syntax

```
while(condition) :
      [statement]
```



Example :

```
count =0
while(count <2):
  count=count +1
      print("Hello")
```

Output :

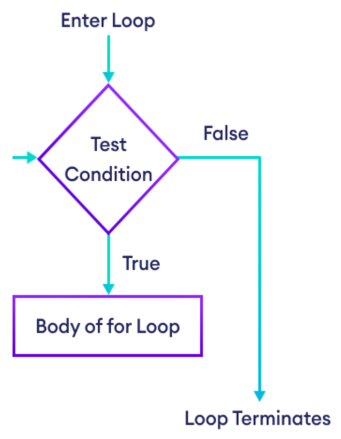
```
Hello
Hello
```

FOR LOOP

It is used to iterate over items of any sequence, such as a list or a string.

Syntax

```
for val in sequence: #here val will take value of each element in the sequence
        statements
```



Example:

```
for i in [1,2,3,4,5]:
      print(i*5)
```

OUTPUT:

```
5
10
15
20
25
```

range() Function

This function generates a sequence of numbers based on the parameters passed.

Parameters

start: Starting number of the sequence.

stop: Generate numbers up to, but not including this number.

step(Optional): Determines the increment between each numbers in the sequence.

Python use range() function in three ways:

a. **range**(stop) : By default, It starts from 0 and increments by 1 and ends upto stop, but not including **stop** value.

```
for i inrange(5):
   print(i, end=" ")
```

Output : 0 1 2 3 4

b.range(start,stop) : It starts from the **start** value and upto stop, but not including stop value.

```
for i inrange(5, 10):
   print(i, end=" ")
```

Output : 5 6 7 8 9

c. **range**(start,stop,step): Third parameter specifies to increment or decrement the value by adding or subtracting the value.

```
for i inrange(0, 10, 2):
    print(i, end=" ")
```

Output : 0 2 4 6 8

BREAK STATEMENT

It is used to terminate the loop.

Example

```
for val in "string":
     if val == "i":
          break
     print(val, end=" " )
print("The end")
```

Output: s t r The end

CONTINUE STATEMENT

- Used to skip the rest of the statements of the current loop block and to move to next iteration, of the loop.
- Continue will return back the control to the beginning of the loop.

Example

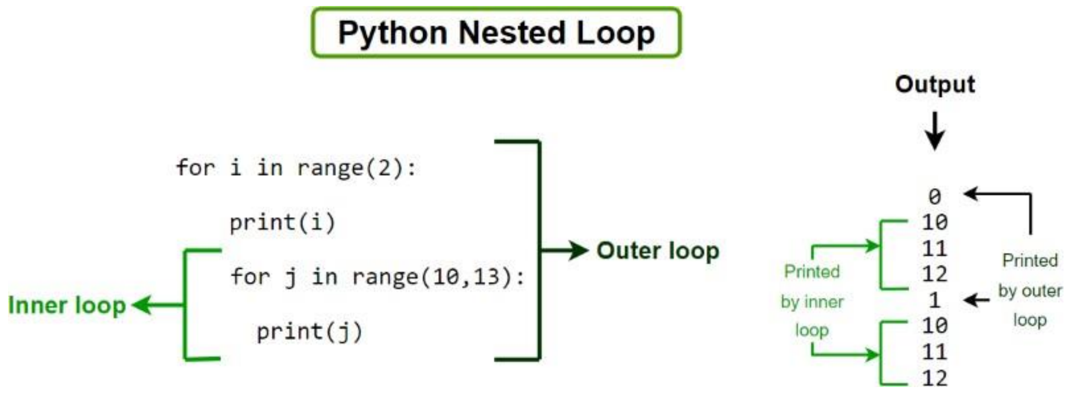
```
for letter in  'Python' :
     if letter ==  'h' :
          continue
print (letter)
```

Output

```
Pyton
```

Nested Loops

Nested loops mean loops inside a loop. For example, while loop inside the for loop, for loop inside the for loop, etc.



```
for i in range(2):
    print(i)
    for j in range(10,13):
      print(j)
```

```
0
10
11
12
1
10
11
12
```

SOLVED QUESTIONS

MCQ (1 mark)

Q1:Which keyword is used to add multiple conditions in a statement?

1. if
2. else
3. elif
4. while

Ans : 3. elif

Q2 : Choose correct output :

```
namesl = [‘Amir’, ‘Barry’,Chales’,‘Dao’]
  if ‘amir’ in namesl:
      print 1
  else:
      print 2
```

1. 1
2. 2
3. Error
4. 12

Ans : 2. 2

Q3 : Find correct output :

```
x=3
        if x>2 or x<5 and x==6:
              print("ok")
        else:
              print("no output")
```

1. Ok
2. no output
3. error
4. ok no output

Ans : 2. No output

Q4 : Which keyword is used to terminate the looping in Python when certain condition is met ?

1. Break
2. Continue
3. Raise
4. Pass

Ans 1. Break

Q5. What will be the output of the following Python code?

```
i = 1
while True:
      if i % 3 == 0:
             bre
         ak
      print(i)
       i + = 1
```

1. 1 2
2. 1 2 3
3. Error
4. None of the mentioned

Ans : 1. 1 2

Q 6 : Which of the following sequences would be generated in the given line of code?

range (5,0, -2)

1. 5 4 3 2 1 0 -1
2. 5 3 1 -1
3. 5 3 1
4. 5 3
5.

Ans : 3. 5 3 1

Very Short Answer type Questions (2 marks)

Q1 : Find output

```
x=True
        y=False
        z=False
        if x or y and z:
                print("YES")
        elif x and y or z:
                print("yes")
        else:
                 print("no")
```

Ans : no

Q2 . What is the output for the following code

```
i=1
         while(i<=7):
                i*=2
         print(i)
```

Ans : 128

Q3**.** What is the difference between break and continue statements?

Ans:

Break: breaks the iteration of loop when a certain condition is met .

Continue : skip the rest of the statements of the current loop block and to move to next iteration, of the loop.

Q4. Write a program to print the sum of series: s=1+x+x^2+x^3…..+x^n

Ans :

```
s=0
        x=int(input("enter x"))
        for i in range(1,n+1):
            s=s+x**i
        print(s)
```

Q5 . Write a Python program that accepts two integers from the user and prints a message saying if first number is divisible by second number or if it is not?

Ans:

```
x=int(input("enter first number"))
               y=int(input("enter second number"))
               if(X%Y ==0):
                    print(x, "is divisible by",y)
              else :
                    print(x,"is not divisible by " , y)
```

Syntax Error	**Logical Error**
Syntax Errors occur when we violate the rules of writing the statements of the programming language.	Logical Errors occur due to our mistakes in programming logic.
Program fails to compile and execute.	Program compiles and executes but doesn't give the desired output.
Syntax Errors are caught by the compiler.	Logical errors need to be found and corrected by people working on the program.
Example : print("hello) . closing " is missing	Example : average=a+b/2 gives incorrect value for average of a and b .

Q6. Differentiate between Syntax Error and Logical Error with Example.

<u>Short Answer Type Questions (3 marks)</u>

Q 1 Write a program to print grade of a student as per input percentage as per criteria given below:-

Percentage range	Grade
>=90	A
75<=89	B
60<=74	C
40<=59	D
>=40	E

Ans:

```
perc= int(input("enter percentage"))
    if(perc>=90):
    Print(" grade A")
    elif(perc>=75 and perc< 89 ) :
            Print ( " grade B ")
    elif( perc>= 60 and perc< 74 ) :
             Print("grade C")
    elif(perc>=40  and perc< 60):
              Print("grade D")
     else:
              Print("E")
```

Q2. Write a program to find maximum of 3 integers entered by user .

Ans :

```
a=int(input ("Enter First Number ?"))
b=int(input("Enter First Number ?"))
c=int(input("Enter First Number ?"))
if (a>b) and (a>c) :
   print ("Max number is ",a)
if (b>a) and (b>c) :
   print ("Max number is ",b)
 if (c>a) and (c>b) :
    print ("Max number is ",c)
```

<u>Long answer question (5marks)</u>

Q2. **Q1 Write python code to Generate following pattern using Nested loop**

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

Ans :

```
n = int(input("Enter the number of rows"))
# outer loop to handle number of rows
```

```
for i in range(0, n):
        for j in range(0, i + 1):
            print("* ", end="")
          # ending line after each row
        print()
```

Unsolved questions :

MCQs

Q1 : How do we differentiate the body of the loop from the rest of the code?

1. Writing loop above.
2. Writing loop below the whole code.
3. Using proper indentation.
4. All are correct.

Q2: Which of the following is False regarding loops in Python?

A. Loops are used to perform certain tasks repeatedly.
B. While loop is used when multiple statements are to executed repeatedly until the given condition becomes False
C. While loop is used when multiple statements are to executed repeatedly until the given condition becomes True.
D. for loop can be used to iterate through the elements of lists.

Q3 : The__________statement skips the rest of the loop statements and causes the next iteration of the loop to take place.

1. Break
2. Continue
3. Raise
4. Pass

VSA (2marks)

Q1. Rewrite the following code after removing syntax errors :

```
name = ("Enter Name :" )
age = int(input("Enter Age: ")
if age >=18
        print(name, “is eligible for driving license")
else
print(name, "isot eligible for driving license")
```

Q2 . Find Output

```
i=0
   while (i<10):
   i=i+1
         if i==5:
                break
 print(i,end=” “)
```

Q3 Find ouput of following code :

```
for x in range(0,10,2):
        print(x,"#", end=' ')
```

SA (3 marks)

Q1. Write a program to find sum of all even numbers from 1 to 100.
Q2.Write a program which accept Sales Amount from user then calculate and print discount amount per following criteria :

Sales Amount	Discount
Less than 5000	5% of Sales Amount
5001 to 10000	7% of sales Amount
More than 10000	10% of Sales Amount

(e.g. if Sales amount is 400 then Discount amount would be 400*5/100 i.e. 20)

LA (5 marks)

Q1 Write a python program to accept age of a person from user and check whether he has completed 18 years or more. If yes print 'Eligible for Voting' or otherwise print 'Minor'.

Q2 Write a program in python to print the table of a number given by the user using for loop

STRING MANIPULATION

Strings: Sequence of characters that is enclosed in single or double quotes referred as String. Strings are immutable. Various string operations are as follows:-

Operators	**Syntax**	**Description**
+ (Concatenation)	Str1+Str2	Adds or join two or more strings
*(Repetition)	Str*3	*operator creates a new string by repeating multiple copies of the same string.
in/not in (Membership)	'M' in 'Mumbai' 'M' not in 'mumbai'	Returns true if character exist in given the string and return false if the given character does not exist in the string
[:] (range(start, stop, step)	Str[1:8:2]	Extracts the characters from the given range or to extract a subset of values.
[ ] Slice[n:m]	Str[2:7]	Extracts the characters from the given index
Traversing a String	for i in range(len(str)):	iterate through the elements of a string,one character at a time.

String Methods and Built-in functions:

Function/Method	**syntax**	**Description**
len()	len (str)	Returns the length of the string
count()	str.count(sub, start, end)	It returns number of times substring str occurs in the given string.
split()	str.split(",")	Breaks up a string at the specified separator and returns a list of substrings
capitalize()	str.capitalize ()	Converts the first letter of the string in uppercase
title()	str.title()	It returns the string with first letter of every word in the string in uppercase and rest in lowercase.

find()	str.find (sub, start, end)	It is used to search the first occurrence of the substring in the given string.
replace()	str.replace (old, new)	It replaces all the occurrences of the old string with the new string.
index()	index(substr, start, end)	It also searches the first occurrence and returns the lowest index of the substring.
lower()	str,lower ()	It converts the string into lowercase
islower()	str.islower ()	It returns True if all the letters in the string are in lowercase otherwise false
upper()	str.upper()	It converts the string into uppercase
isupper()	str.isupper ()	It returns True if all the letters in the string are in uppercase otherwise false.
isalpha()	str.isaplha ()	It checks for alphabets in an inputted string and returns True in string contains only letters else false.
isalnum()	str.isanum ()	It returns True if all the characters are alphanumeric else false.
isdigit()	str.sidigit ()	It returns True if the string contains only digits, otherwise false.

lstrip()	str.istrip(chars)
	str.istrip()
		It returns the string after removing the space from the left of the string

rstrip()	str.restrip(chars)
	str.rstrip()
		It returns the string after removing the space from the right of the string

strip()	str.strip()	It returns the string after removing the space from the both side of the string
index()	str.index(substring)	It returnsthe index position of an element or an item in a string of characters or a list of items.
startswith()	string.startswith(substring, start, end)	It returns True if the string starts with the given substring, and False otherwise.
partition()	str.partition(separator)	It splits a given string into three parts based on the first occurrence of a specified separator. It returns a tuple containing the three parts: the portion before the separator, the separator itself, and the portion after the separator.
join()	separator.join(iterable)	It is used to concatenate elements from an iterable (like a list or tuple) into a single string, with a specified separator between each element.
endswith()	string.endswith(substring, start, end)	It returns True if the string ends with the given substring, and False otherwise.

MULTIPLE CHOICE QUESTIONS (MCQ)-1 Mark

1	Which of the following operations on a string will generate an error? (a) "PYTHON"*3 (b) "PYTHON" + "20" (c)"PYTHON" + 10 (d) "PYTHON" + "LANGUAGE"
2	If the following code is executed, what will be the output of the following code? name="Computer_Science_with_Python" print (name [-25:10]) (a) puter_S (b) hon (c) puter_Science (d) with python
3	Which of the following functions will return the total number of characters in a string? count () b) index() c) len() d) all of these
4	Which of the following functions will return the last three characters of a string s ? s[3:] b) s[: 3] c) s[-3:] d) s[: -3]
5	Which of the following functions will raise an error if the given substring is not found in the string ? a) find() b) index() c) replace() d) all of these
6	Which of the following functions removes all leading and trailing spaces from a string ? a) lstrip() b) rstrip() c) strip() d) all of these
7	Find the operator which cannot be used with a string in Python from the following: (a) + (b) not in (c) * (d) //

8	What will be the output of above Python code?
	str1= “6/4”
	print(“str1”)
	a) 1 b) 6/4 c) str1 d (1.5)

9	Which of the following will result in an error? str1="python" a) print(str1[2]) b) str1[1]="x" b) c) print(str1[0:9]) d) Both (b) and (c)
10	Which of the following is False? a) String is immutable. b) capitalize() function in string is used to return a string by converting the whole given string into uppercase. c) lower() function in string is used to return a string by converting the whole given string into lowercase. d) None of these.

11	What will be the output of below Python code?
	str1="Information"
	print(str1[2:8])

	a) formatb) formation c) orma d) ormat

12	What will be the output of below Python code?
	str1="Aplication"
	str2=str1.replace('a','A')
	print(str2)
	a) application b) Application c) ApplicAtion d) applicAtion

13	What will be the output of below Python code?
	str1=”power”
	str1.upper( )
	print(str1)
	a) POWER b) Power c) power d) power
	b)

14	Which of the following will give "Simon" as output? If str1="John,Simon,Aryan" a) print(str1[-7:-12]) b) print(str1[-11:-7]) c) print(str1[-11:-6]) d) print(str1[-7:-11])
	In the following questions(15 to 20) , a statement of assertion (A) is followed by a statement of reason(R) . Make the correct choice as : (a) Both A and R are true and R is the correct explanation for A (b) Both A and R are true and R is not the correct explanation for A (c) A is True but R is False (or partially True) (d) A is false(or partially True) but R is True
15	Assertion(A): The position and index of string characters are different. Reason(R) : The positions for string’s characters vary from 1..n , where n is size of the string . The indices for string’s characters vary from 0 to n-1.
16	Assertion(A): Operators + and * can work with numbers as well as strings. Reason(R) : Unlike numbers , for strings , + means concatenation and * means replication.
17	Assertion(A):String slices and substrings ,both are extracted subparts of a string. Reason(R) : String slices and substrings mean the same.
18	Assertion(A):Like numbers, ==,>,< can also compare two strings. Reason(R) :comparison of python strings takes place in dictionary order by applying character-by-character comparison rules for ASCII/Unicode.
19	Assertion(A):String slices and substrings, despite being subparts of a string , are not the same. Reason(R) :While the substring contains a continuous subparts of the string , the slices may or may not contain continuous subparts of a string.
20	Assertion(A): The individual characters of a string are randomly stored in memory. Reason(R) :Python strings are stored in memory by storing individual characters in contiguous memory locations.

ANSWER (1 MARK- MULTIPLE CHOICE QUESTIONS (MCQ)

1	c	11	a
2	a	12	c
3	c	13	a
4	c	14	c
5	b	15	a
6	c	16	a
7	d	17	c
8	c	18	b
9	b	19	a
10	b	20	d

VeryShortanswerTypeQuestions

Q1.Which of the following is not a Python legal string operation?

a) "abc"+"abc" (b)„abc"*3

b) (c)"abc"+3 (d)"abc".lower()Ans:

Ans. (c)‗abc'+3
Q2. Out of the following operators ,which one scan be used with strings?
=,-,*,/,//,%,>,<>,in, not in,<=
Ans: /,// and %
Q3. From the string S="CARPEDIEM".Which ranges return"DIE"and"CAR"?
Ans. S[6:9]forDIEandS[0:3]for CAR
Q4. Given a string S = "CARPE DIEM". If n is length/2 then what would following return?
(a)S[:n] (b)S[n:] (c)S[n:n] (d)S[1:n] (e)S[n:length-1]
Ans: (a)"CARPE(b)"DIEM" (c)" " (d)"ARPE" (e)"DIE"

Q5.Whatwouldfollowingexpressionreturn?
(a) "HelloWorld".upper().lower() (b)"HelloWorld".lower().upper()
(c)"HelloWorld".find("Wor",1,6) (d)"HelloWorld".find("Wor")
(e)"HelloWorld".find("wor") (f)"HelloWorld".isalpha()
(g) "HelloWorld".isalnum() (h)"HelloWorld".isdigit()
(i)"123FGH".isdigit()

Ans: (a) 'helloworld' (b)'HELLOWORLD'
(c)-1 (d)6
(e)-1 (f)False
(g) False (h)False
(i)False

Q-6. Find the output:

```
 str="PYTHON@LANGUAGE"
print(str[2:12:2])
```

Ans. TO@AG

Q-7. Find and write the output of the following python code:

```
x = "abcdef"
i =  "a"
while i in x:
        print(i, end = " ")
```

Ans: aaaaaa ---- OR infinite loop

ShortAnswerTypeQuestions

Q1.What is a string slice?How is it useful?

Ans: String Slice is a part of a string containing some contiguous characters from the string.

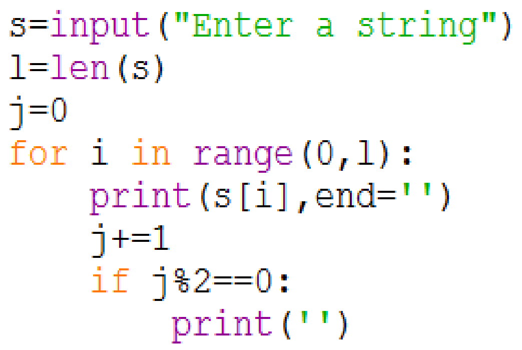
It is accessed from the string by providing a rangein“[]”bracketsi.e.S[n:m].Python returns all the characters at indices n,n+1, n+2. .n-1e.g.

S=”Barabanki”

S[4:7]will return ban

Q2.Write a python script that traverses through an input string and prints its characters in different lines–two characters per line.

Ans:



```
s=input("Enter a string")
l=len(s)
j=0
for i in range(0,l):
    print(s[i],end='')
    j+=1
    if j%2==0:
        print('')
```

Q3.Which functions would you chose to use to remove leading and trailing white spaces from a given string?

Ans: Python String strip() function will remove leading and trailing white spaces.If you want to remove only leading or trailing spaces,use lstrip()or rstrip()function instead.

Q4.Suggest appropriate functions for the following tasks –

(a) To check whether the string contains digits.
(b) To find the occurrence a string within another string.
(c) Toconvertthefirstletterofastringtouppercase.
(d) To convert all the letters of a string toupper case.
(e) To check whether all the letters of the string are in capital letters.
(f) to remove all the white spaces from the beginning of a string.

Ans:(a) isalnum()
(b) find()
(c) capitalize()
(d) upper()
(f) isupper()
(g) lstrip()

Q5. Find the errors-

s=”PURAVIDA”
print(s[9]+s[9:15])

Ans: Here the error is:String index out of range.

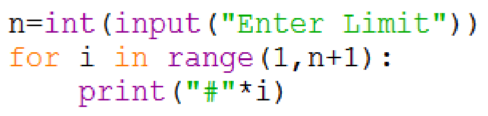
Q6- How many types of strings are supported in Python?
Ans: Python allows two string types:
Single Line strings –Strings that are terminated in a single line enclosed within single and double quotes.
Multiline strings- String storing multiple lines of text enclosed within three single or double quotes.

HOTs Questions

Q.1 WAP to print following pattern without using any nested loop.

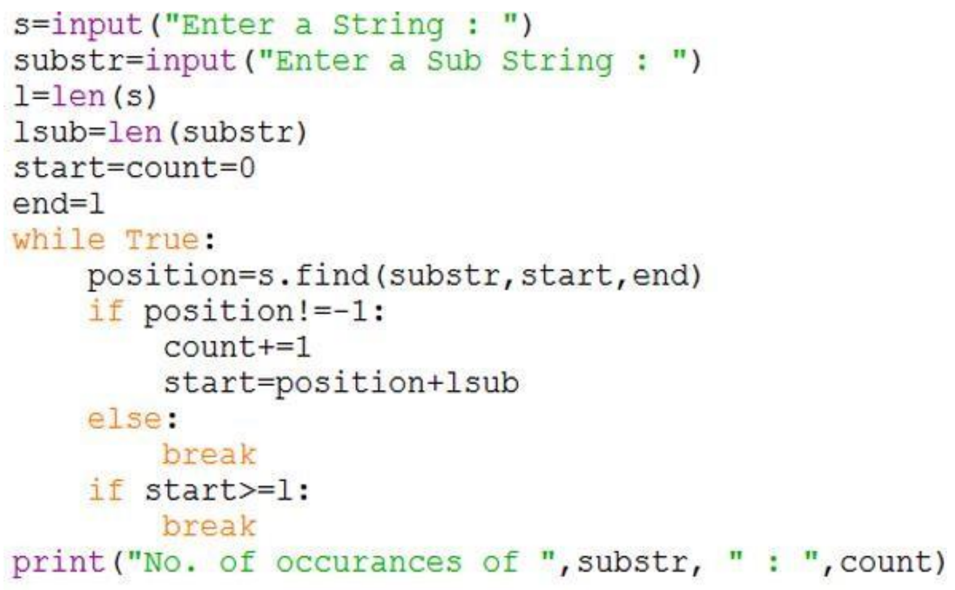
```
#
# #
# # #
# # #  #
# # # # #
```

Ans:



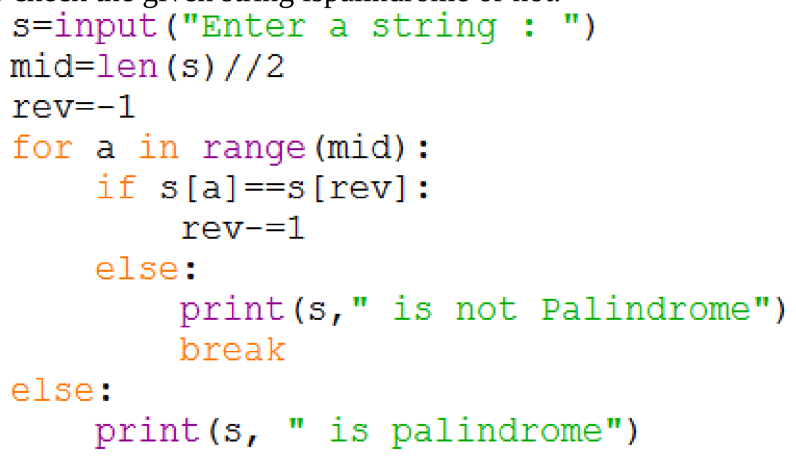
```
n=int(input("Enter Limit"))
for i in range(1,n+1):
    print("#"*i)
```

Q2. WAP to print the number of occurrences of a substring into a line.
Ans.



```
s=input("Enter a String : ")
substr=input("Enter a Sub String : ")
l=len(s)
lsub=len(substr)
start=count=0
end=l
while True:
    position=s.find(substr,start,end)
    if position!=-1:
        count+=1
        start=position+lsub
    else:
        break
    if start>=l:
        break
print("No. of occurances of ",substr, " : ",count)
```

Q3. WAP to check the given string ispalindrome or not.
Ans.



```
s=input("Enter a string : ")
mid=len(s)//2
rev=-1
for a in range(mid):
    if s[a]==s[rev]:
        rev-=1
    else:
        print(s," is not Palindrome")
        break
else:
    print(s, " is palindrome")
```

Unsolved Questions:

Q1.Find output of the following “abcd”.startswith(“cd”).
Q2. Find output of the following “COMPUTER SCIENCE”.title().
Q3. Differentiate between partition ( ) and split( ) functions.
Q4. Find the length of string: name= “Computer Science”).
Q5. Find the output:

```
str = "Python Output based Questions"
word=str.split()
for i in word:
   print(i)
```

Q6. . Find and write the output of the following python code:

```
Text1="DIGITAL INDIA2023"
Text2=""
I=0
while I<len(Text1):
       if Text1[I]>="0" and Text1[I]<="9":
              Val = int(Text1[I])
              Val = Val + 1
              Text2=Text2 + str(Val)
       elif  Text1[I]>="A" and Text1[I] <="Z":
              Text2=Text2 + (Text1[I+1])
       else:
              Text2=Text2 + "*"
       I=I+1
print (Text2)
```

Q7. Find and write the output of the following python code:

```
s="welcome2cs"
n = len(s)
m=""
for i in range(0, n):
    if (s[i] >= 'a' and s[i] <= 'm'):
         m = m +s[i].upper()
    elif (s[i] >= 'n' and s[i] <= 'z'):
         m = m +s[i-1]
    elif (s[i].isupper()):
          m = m + s[i].lower()
    else:
          m = m +'&'
print(m)
```

Answers:
Q1. False
Q2. Computer Sceince
Q3. The split() method is used to split a string into a list of substrings based on a delimiter. The delimiter can be any character or sequence of characters that you specify. By default, if no delimiter is provided, it will split the string at spaces.
The partition() method is used to split a string into three parts based on the first occurrence of a specified delimiter. It returns a tuple containing the part before the delimiter, the delimiter itself, and the part after the delimiter.
Q4. 16
Q5. Python

```
    Output
     based
     Questions
```

Q6. IGITAL *NDIA23134
Q7. sELCcME&Cc

LIST MANIPULATION

List: A list is a collection of comma-separated values (items) of same or different type within square ( ) brackets. List is a mutable data type.

Slicing: Slicing is an operation in which we can slice a particular range from a sequence.

List slices are the sub parts extracted from a list.

Nested List: When a list appears as elements of another list, it is called a nested list.

Built-in Function (Manipulating Lists)

Function	**Syntax**	**Description**
list()	list(sequence)	It returns a list created from the passed arguments, which should be a sequence type(string, list, tuple etc.). if no argument is passed, it will create an empty list.
append()	list.append (items)	It adds a single item to the end of the list.
extend()	list1.extend (list2)	It adds one list at the end of another list.
insert()	list.insert (index_no, value)	It adds an element at a specified index
reverse()	list.reverse ()	It reverses the order of the elements in a list.
index()	list.index (item)	It returns the index of first matched item from the list.
len()	len (list)	Returns the length of the list i.e. number of elements in a list
sort()	list.sort ()	This function sorts the items of the list.
clear()	list.clear ()	It removes all the elements from the list.
count()	list.count (element)	It counts how many times an element has occurred in a list and returns it.
sorted()	sorted(sequence,reverse=False)	It returns a newly created sorted list; it does not

		change the passed sequence.
pop()	list.pop (index)	It removes the element from the specified index and also returns the element which was removed.
remove()	list.remove (value)	It is used when we know the element to be deleted, not the index of the element.
max()	max(list)	Returns the element with the maximum value from the list.
min()	min(list)	Returns the element with the minimum value from the list
sum()	sum(list)	It returns sum of elements of the list.

MCQs

1	**1. The data type list is an ordered sequence which is_________and made up of one or more elements.** a. Mutable b. Immutable c. Both a) and b) d. None of the above
2	**Which statement from the list below will be create a new list?** a. new_l = [1, 2, 3, 4] b. new_l = list() c. Both a) and b) d. None of the above

3	**What will be the output of the following python code**
	new_list = [‘P’,’y’,’t’,’h’,’o’,’n’]
	print(len(new_list))
	a. 6 b. 7c. 8d. 9

4	**Python allows us to replicate a list using repetition operator depicted by symbol_________.** a. – b. + c. / d. *
5	**We can access each element of the list or traverse a list using a_______.** a. for loop b. while loop c. Both a) and b) d. None of the above
6	**________a single element passed as an argument at the end of the list.** a. append() b. extend() c. insert() d. None of the above
7	**_________returns index of the first occurrence of the element in the list. If the element is not present, ValueError is generated.** a. insert() b. index() c. count() d. None of the above
8	**_______function returns the element whose index is passed as parameter to this function and also removes it from the list.** a. push() b. remove() c. pop() d. None of the above

9	**What will be the output of the following python code.** **new_list = “1234”** **print(list(new_list))** a. [‘1’, ‘2’, ‘3’, ‘4’] b. (‘1’, ‘2’, ‘3’, ‘4’) c. {‘1’, ‘2’, ‘3’, ‘4’} d. None of the above
10	**Suppose list1 is [4, 2, 2, 4, 5, 2, 1, 0], which of the following is correct syntax for slicing operation?** **a) print(list1[0]) b) print(list1[:2])** **c) print(list1[:-2]) d) all of the mentioned**
11	**Suppose list1 is [2, 33, 222, 14, 25], What is list1[-1] ?** **a) Error b) None c) 25 d) 2**
12	**Suppose list1 is [1, 3, 2], What is list1 * 2 ?** **a) [2, 6, 4] b) [1, 3, 2, 1, 3]** **c) [1, 3, 2, 1, 3, 2] d) [1, 3, 2, 3, 2, 1].**
13	**What is the output when following code is executed ?** **>>>list1 = [11, 2, 23]** **>>>list2 = [11, 2, 2]** **>>>list1 < list2 is** **a) True b) False c) Error d) None**
14	**To insert 5 to the third position in list1, we use which command ?** **a) list1.insert(3, 5) b) list1.insert(2, 5)** **c) list1.add(3, 5) d) list1.append(3, 5)**
	In the following questions(15 to 20) , a statement of assertion (A) is followed by a statement of reason(R) . Make the correct choice as : (a) Both A and R are true and R is the correct explanation for A (b) Both A and R are true and R is not the correct explanation for A (c) A is True but R is False (or partially True) (d) A is false(or partially True) but R is True
15	Assertion: In python, unlike other types, you can change elements of a list in place. Reason: Lists are mutable sequences.
16	Assertion: Any comma-separated group of values creates a list. Reason: Only a group of comma-separated values or expressions enclosed in [ ] , creates a list.
17	Assertion: Lists and strings have similar types of indexing. Reason: Both lists and strings have two-way indexing , forward indexing and backward indexing.
18	Assertion :Lists are similar to strings in a number of ways like indexing , slicing and accessing individual elements. Reason : Lists, unlike strings , are mutable.
19	Assertion : The membership operators in and not in work in the same way on lists as they do , with strings. Reason :Some operators work differently on strings and lists , such as + and * .
20	Assertion: A list slice is an extracted part of a list. Reason: A list slice is a list in itself.

1	a	11	c
2	c	12	c
3	a	13	b
4	d	14	b
5	c	15	a
6	c	16	d
7	b	17	a
8	c	18	b
9	a	19	c
10	d	20	b

Very Short Answer Type Questions

Q1. What do you understand by mutability?

Ans: Mutable means changeable. In Python, mutabletypes are those whose values can be changed inplace.Onlythreetypesaremutablein python–Lists,Dictionariesand Sets.

Q2. Startwiththelist[8,9,10].Dothefollowingusinglistfunctions:

(a) Setthesecondentry(index1)to17
(b) Add4,5and6tothe endofthelist.
(c) Removethefirstentryfromthelist.
(d) Sortthelist
(e) Doublethelist.
(f) Insert 25 at index 3

Answers:

(a) list[1]=17

(b) list.append(4)

list.append(5)

list.append(6)

(c) list.pop(0)

(d) list.sort()

(e) list=list*2

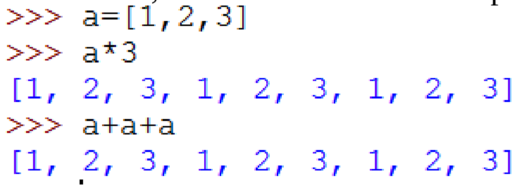
(f) list.insert(3,25)

Q3. Ifais[1,2,3],what is the difference(ifany) between a*3 and [a,a,a]?

Ans: a*3 will produce [1,2,3,1,2,3,1,2,3],means a list of integers and [a,a,a] will produce[[1,2,3],[1,2,3],[1,2,3]], means list of lists

Q4. If a is [1,2,3],is a*3 equivalent to a+ a+a?

Ans: Yes, Both a*3and a+a+a will produce same result.



```
>>> a=[1,2,3]
>>> a*3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
>>> a+a+a
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

Q5. If ai s[1,2,3],what is the meaning of a[1:1]=9?
Ans: This will generate an error–Type Error: can only assign an iterable.
Q6. If a is[1,2, 3],what is the meaning of a[1:2]=4 and a[1:1]=4?
Ans: These will generate an error–Type Error:can only assign an iterable.
Q7. What are list slices?
Ans: List slices are the sub-part of a list extracted out. You can use indexes of the list elements to create *list slices* as per following format. Syntax is as follows:- *Seq=ListName [start:stop]*
Q8.Does a slice operatoral ways produce a newlist?
Ans: Yes, this will create a new list.

ShortAnswerTypeQuestions

Q1. How are lists different from strings when both are sequences?
Ans: Lists are similar to strings in many ways like indexing, slicing, and accessing individual elements but they are different in the sense that *Lists are mutable while strings are immutable.*
Inconsecutive locations, strings store the individual characters while list stores the references of its elements.
Strings store single type of elements-all characters while lists can store elements belonging to different types.

Q2. What are nested Lists?
Ans: A list can have an element in it,which itself is a list.Such a list is called nested list.e.g.
L=[1,2,3,4,[5,6,7],8]
Q3.Discuss the utility and significance of Lists.
Ans: The list is a most versatile datatype available in Python which can be written as a list of comma-separated values(items) between square brackets. Important thing about a list is that items in a list need not be of the same type. List is majorly used with dictionaries when there is large number of data.
Q4. What is the purpose of the del operator and pop method? Try deleting a slice.
Ans: *del operator* is used to remove an individual item,or to remove all items identified by a slice.
It is to be used as per syntax given below–

```
>>>del List[index]
>>>del List[start:stop]
```

Pop method is used to remove single element, not list slices.The pop() method removes an individual

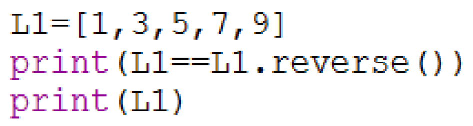
item and returns it.I ts syntax is–

```
>>>a=List.pop()     #thiswill remove last item and deleted item will be assigned to a.
>>>a=List[10]       #thiswill remove the item at index10 and deleted item will be assigned to a.
```

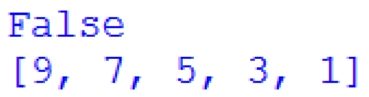
Q5. What are list slices?
Ans: List slices,like string slices are the sub part of a list extracted out. Indexes can be used to create list slices as per following format:
seq=L[start:stop]
Q6. What do you understand by true copy of a list? How is it different from shallow copy?
Ans:A shallow copy means constructing a new collection object and then populating it with references to the child objects found in the original. In essence, a shallow copy is only *one leveldeep*. The copying process does not recurse and therefore won‘t create copies of the child objects themselves.

True Copy means you can create a copy of a list using *New_list=My_list*. The assignment just copies the reference to the list, not the actual list, so both new_list and my_list refer to thesamelistaftertheassignment.

Q7. Predict the output – Ans:

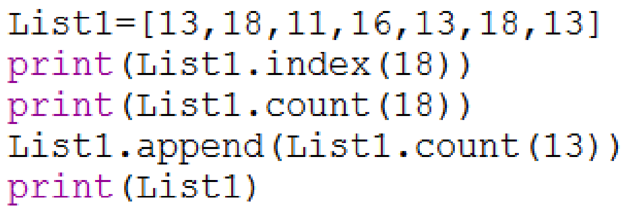


```
L1=[1,3,5,7,9]
print(L1==L1.reverse())
print(L1)
```

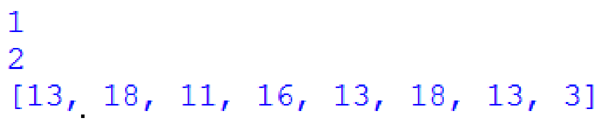


```
False
[9, 7, 5, 3, 1]
```

Q8. Predict the output – Ans:



```
List1=[13,18,11,16,13,18,13]
print(List1.index(18))
print(List1.count(18))
List1.append(List1.count(13))
print(List1)
```



```
1
2
[13, 18, 11, 16, 13, 18, 13, 3]
```

UNSOLVED QUESTIONS

Q1. WAP to find minimum element from a list of elements along with its index in the list.
Q2.WAP to Calculate mean of the given list of numbers.
Q3. WAP to search for an element in a given list of numbers.
Q4. WAP to count the frequency of a given element in the list of numbers.
Q5. Differentiate between append( ) and extend ( ) functions in Python.

UNSOLVED QUESTIONS(ANSWERS)

Q1. WAP to find minimum element from a list of elements along with its index in the list.

```
Ans. lst = eval(input("enter list :"))
min_element = lst[0]
min_index = 0
for i in range(1, len(lst)):
 if lst[i] <min_element:
min_element = lst[i]
min_index = i
print("Given list is : ",lst)
print("The minimum element of the given list is :")
print(min_element,"at index",min_index)
```

Q2.WAP to Calculate mean of the given list of numbers.

```
Ans.numbers = [23, 45, 12, 67, 9, 31]
# Calculate the sum of the numbers
total = 0
for num in numbers:
 total += num
# Calculate the mean
mean = total / len(numbers)
print("The mean of the numbers is :",mean)
```

Q3. WAP to search for an element in a given list of numbers.
Ans.lst=eval(input(“Enter list:”))

```
        length=len(lst)
        element=int(input(“Enter element to be searched for:”))
        for i in range(0,length):
                if element==lst[i]:
                        print(element,”found at index”,i)
                break
        else:
                print(element,”not found in given list”)
```

Q4. WAP to count the frequency of a given element in the list of numbers.
Ans .

```
        L=[2,58,95,999,65,32,15,1,7,45]
        n=int(input("Enter the number : "))
        print("The frequency of number ",n," is ",L.count(n))
```

Q5. Differentiate between append( ) and extend( ) functions.
Ans. The append() function/method adds a single item to the end of the existing list. It doesn’t return a new list. Rather, it modifies the original list. The extend() adds all multiple items in the form of a list at the end of another list.

TUPLES

A tuple is a collection which is ordered and immutable (We cannot change elements of a tuple in place). Tuples are written with round brackets. Tuples are used to store multiple items in a single variable. Tuples may have items with same value.

```
Exp.    T = ()                          # Empty Tuple
        T = (1, 2, 3)                   # Tuple of integers
        T = (1,3.4,7)                   # Tuple of numbers
        T = (‘a’, ‘b’, ‘c’)             # Tuple of characters
        T = (‘A’,4.5,’Ram’,45)          # Tuple of mixed values
        T = (‘Amit’, ‘Ram’, ‘Shyam’)    # Tuple of strings
```

Creating Tuples

A tuple is created by placing all the items (elements) inside parentheses (), separated by commas.
Exp. T = (10,20, ‘Computer’,30.5)

If there is only a single element in a tuple then the element should be followed by a comma, otherwise it will be treated as integer instead of tuple. For example

```
        T = (10)        # Here (10) is treated as integer value, not a tuple
        T1 = (10,)      # It will create a tuple
```

Difference between List and Tuple

List	Tuple
Elements are enclosed in square brackets i.e. [ ]	Elements are enclosed in parenthesis i.e. ( )
It is mutable data type	It is immutable data type
Iterating through a list is slower as compared to tuple	Iterating through a tuple is faster as compared to list

Accessing Elements of a tuple (Indexing)

Elements of a tuple can be accessed in the same way as a list or string using indexing and slicing, for example
If str = (‘C’, ‘O’, ‘M’, ‘P’, ‘U’, ‘T’, ‘E’, ‘R’)

ELEMENTS	C	O	M	P	U	T	E	R
POSITIVE INDEX VALUE	0	1	2	3	4	5	6	7
NEGATIVE INDEX VALUE	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> str[2] = ‘M’
>>> str[-3] = ‘T’
```

Traversing a Tuple

Traversing a tuple means accessing and processing each element of it. The for loop makes it easy to traverse or loop over the items in a tuple. For example:

```
str = (‘C’, ‘O’, ‘M’, ‘P’, ‘U’, ‘T’, ‘E’, ‘R’)
for x in str:
   print(str[x])
```

The above loop will produce result as :

```
C
O
M
P
U
T
E
R
```

Tuple Operations

Concatenation (Joining Tuples)

The + operator is used to join (Concatenate) two tuples

Exp.

```
>>> T1 = (10,20,30)
>>> T2 = (11,22,33)
>>> T1 + T2
```

+ Operator concatenates Tuple T1 and Tuple T2 and creates a new Tuple

(10, 20, 30, 11, 22, 33)

Repetition

* Operator is used to replicate a tuple specified number of times, e.g.

```
>>> T = (10, 20, 30)
>>> T * 2
(10, 20, 30, 10, 20, 30)
```

Membership:

The **'in'** operator checks the presence of element in tuple. If the element present it returns True, else it returns False.

For Exp.
```
str = ('C', 'O', 'M', 'P', 'U', 'T', 'E', 'R')
'M' in str          => Returns True
'S' in str          => Returns False
```

The **not in** operator returns True if the element is not present in the tuple, else it returns False.
```
'M' not in str        => Returns False
'S' not in str        => Returns True
```

Slicing

It is used to extract one or more elements from the tuple. Slicing can be used with tuples as it is used in Strings and List. Following format is used for slicing:

S1 = T[start : stop : step]

The above statement will create a tuple slice namely S1 having elements of Tuple T on indexes start, start+step, start+step+step, …, stop-1.

By default value of start is 0, value of stop is length of the tuple and step is 1

For Example:
```
>>> T = (10,20,30,40,50,60,70,80,90)
>>> T[2:7:3]
(30, 60)

>>> T[2:5]
(30, 40, 50)

>>> T[::3]
(10, 40, 70)

T[5::]
(60, 70, 80, 90)
```

Built-in functions/methods:

The len( ) function

This method returns length of the tuple or the number of elements in the tuple, i.e.,
```
>>> T = (10,20,30,40,50,60,70,80,90)
>>> len(T)
9
```

The max( ) function

This method returns element having maximum value, i.e.,

```
>>> T = (10,20,30,40,50,600,70,80,90)
>>>max(T)
600

>>> T = ('pankaj','pinki','parul')
>>> max(T)
'pinki'
```

The min( ) function

This method returns element having minimum value, i.e.,

```
>>> T = (10,20,30,40,50,600,70,80,90)
>>>min(T)
10

>>> T = ('pankaj','pinki','parul')
>>> min(T)
'pankaj'
```

The sum( ) function

This method is used to find the sum of elements of the tuple, i.e.,

```
>>> T = (10,20,30,40,50,600,70,80,90)
>>> sum(T)
990
```

The index( ) method

It returns the index of an existing element of a tuple., i.e.,

```
>>> T = (10,20,30,40,50,600,70,80,90)
>>> T.index(50)
4
```

But if the given item does not exist in tuple, it raises **ValueError** exception.

```
>>> T = (10,20,30,40,50,600,70,80,90)
>>> T.index(55)
ValueError: tuple.index(x): x not in tuple
```

The count( ) method

This method returns the count of a member / element (Number of occurrences) in a given tuple., i.e.,

```
>>> T = (10,20,30,20,20,0,70,30,20)
>>> T.count(20)
4

>>> T = (10,20,30,20,20,0,70,30,20)
>>> T.count(30)
2
```

The tuple( ) method

This function creates an empty tuple or creates a tuple if a sequence is passed as argument.

```
>>> L = [10, 20, 30]      # List
>>> T = tuple(L)          # Creates a tuple from the list
>>> print(T)
(10, 20, 30)

>>> str = "Computer"              # String
>>> T = tuple(str)                # Creates a tuple from the string
>>> print(T)
('C', 'o', 'm', 'p', 'u', 't', 'e', 'r')
```

The sorted( ) method

This function takes the name of the tuple as an argument and returns a new sorted list with sorted elements in it.

```
>>> T = (10,40,30,78,65,98,23)
>>> X = sorted(T)                       # Make a list of values arranged in ascending order
>>> print(X)
[10, 23, 30, 40, 65, 78, 98]

>>> Y = sorted(T, reverse = False)      # Make a list of values arranged in ascending order
>>> print(Y)
[10, 23, 30, 40, 65, 78, 98]

>>> Z = sorted(T, reverse = True)       # Make a list of values arranged in descending order
>>> print(Z)
[98, 78, 65, 40, 30, 23, 10]
```

Tuple Assignment (Unpacking Tuple)

It allows a tuple of variables on the left side of the assignment operator to be assigned respective values from a tuple on the right side. The number of variables on the left should be same as the number of elements in the tuple.

Exp.

```
>>> T = ('A', 100, 20.5)
>>> x,y,z = T
>>> print(x,y,z)
A 100 20.5
```

Nested Tuple

A tuple containing another tuple in it as a member is called a nested tuple, e.g., the tuple shown below is a nested tuple:

```
>>> students = (101,'Punit', (82,67,75,89,90))              # nested tuple
>>> len(students)
3
```

```
>>> print(students[1])                        # 2nd element of tuple
Punit
>>> print(students[2])                        # 3rd element of tuple
(82, 67, 75, 89, 90)
>>> print(students[2][3])                     # Accessing 4th element of inner tuple
89
```

# Program to find sum of all the elements of a tuple

```
T = (10, 2, 30, 4, 8, 5, 45)
print(T)
s = sum(T)
print("Sum of elements : ", s)
```

Output

```
(10, 2, 30, 4, 8, 5, 45)
Sum of elements :  104
```

# Program to find minimum and maximum values in a tuple

```
T = (10,2,30,4,8,5,45)
print(T)
minimum = min(T)
maximum = max(T)
print("Minimum Value : ", minimum)
print("Minimum Value : ", maximum)
```

Output

```
 (10, 2, 30, 4, 8, 5, 45)
Minimum Value : 2
Minimum Value :  45
```

# Program to find mean of values stored in a tuple

```
T = (10,2,30,4,8,5,46)
print("Tuple :", T)
s = sum(T)
NumberOfElements = len(T)
average = s/NumberOf Elements
print("Mean of elements : ", average)
```

Output

```
Tuple : (10, 2, 30, 4, 8, 5, 46)
Mean of elements :  15.0
```

# Write a program to find the given value in a tuple

```
T = (10,2,34,65,23,45,87,54)
print("Tuple :", T)
x = int(input("Enter value to search:"))
for a in T:
   if a == x :
      print("Value found")
      break
else:
   print("Value not found")
```

Output

```
Tuple :  (10, 2, 34, 65, 23, 45, 87, 54)
Enter value to search:45
Value found
```

Q) Write a program to find sum of elements of tuple without using sum() function?

```
T = (10,20,30)
sum = 0
for x in T:
   sum = sum + x
print(T)
print(sum)
```

Q) Write a program to find sum of even and odd elements of tuple

```
T = (10,23,30,65,70)
sumE = 0
sumO = 0
for x in T:
if (x%2 == 0):
      sumE = sumE + x
   else:
      sumO = sumO + x
print(T)
print("sum of Even numbers :",sumE)
print("Sum of Odd Numbers : ", sumO)
```

MCQs

1.	Consider a tuple in python named Months = ('Jul', 'Aug', 'Sep'). Identify the invalid statement(s) from the given below statements:- a) S = Months[0] b) print(Months[2]) c) Months[1] = 'Oct' d) LIST1 =list(Months)	C
2.	Suppose tuple1 = (2, 33, 222, 14, 25), What is tuple1[::-1]? a) [2, 33, 222, 14] b) Error c) 25 d) [25, 14, 222, 33, 2]	D
3.	Consider the following declaration of Record, what will be the data type of Record? Record=(1342, "Pooja" , 45000, "Sales") a) List b) Tuple c) String d) Dictionary	B
4.	Which operator is used for replication? a) + b) % c) * d) //	C

5.	What will be the output of following Python Code? Tp = (5)
	Tp1 = tp * 2
	Print(len(tp1))
	a) 0 b) 2 c) 1 d) Error
		D

Q1. Define Tuple?
Ans. A tuple is an ordered sequence of elements of different data types, such as integer, float, string, list or even a tuple. Elements of a tuple are enclosed in parenthesis (round brackets) and are separated by commas.

For example T = (1,2, 'a', 5) # T is a tuple

Q2) Write a statement to create an empty tuple named T1?
Ans: T1 = ( ) or T1 = tuple( )

Q3) Write the code to convert given list L into a tuple?
Ans: T = tuple(L)

Q4) What is the difference between a List and a Tuple?
Ans:

List	Tuple
Elements are enclosed in square brackets i.e. [ ]	Elements are enclosed in parenthesis i.e. ( )
It is mutable data type	It is immutable data type
Iterating through a list is slower as compared to tuple	Iterating through a tuple is faster as compared to list

Q5) What is unpacking Tuple?
Ans: It allows a tuple of variables on the left side of the assignment operator to be assigned respective values from a tuple on the right side. The number of variables on the left should be same as the number of elements in the tuple.

Q6) Which error is returned by the following code:

```
T = (10,20,30,40,50,60,70)
Print(T[20])
```

Ans: IndexError : tuple index out of range

Q6) What are nested tuples?

Ans: A tuple containing another tuple in it as a member is called a nested tuple, e.g., the tuple shown below is a nested tuple:

```
>>> students = (101,'Punit', (82,67,75,89,90))                    # nested tuple
```

Q7) Write a program to create a tuple and find sum of its alternate elements?

```
Ans:T = (10,23,30,65,70)
        sum = 0
        for a in T[0:5:2]:
           sum = sum + a
        print(sum)
```

Q8) Write a program to count vowels in a tuple?

```
Ans: T = tuple(input("Enter Name :"))
        print(T)
        v = 0
        for ch in T:
           if ch in ['a','e','i','o','u']:
              v = v + 1
        print("No. of vowels : ",v)
```

Dictionaries

A dictionary is an unordered sequence of key-value pairs.

- Key and value in a key-value pair in a dictionary are separated by a colon. Further, the key : value
- pairs in a dictionary are separated by commas and are enclosed between curly parentheses.
- The keys of the dictionaries are immutable types such as Integers or Strings etc.
- Indices in a dictionary can be of any immutable type and are called keys.
- Dictionaries are mutable.

Creating Dictionaries

A Dictionary can be created in three different ways:

Empty Dictionary

```
>>>D = { }                                    # Empty Dictionary
```

Dictionary using literal notation

```
>>>D = {“Name” : “Mohan”, “Class” : “XI”, “City” : “Gurdaspur”}
>>>print(D)
{“Name” : “Mohan”, “Class” : “XI”, “City” : “Gurdaspur”}
>>>print(D[“City”])
“Gurdaspur”
```

Dictionary using dict() function

Dict() function is used to create a new dictionary with no items. For example,

```
>>> Months = dict()    # Creates an empty dictionary
>>>print(Month)        # Prints an empty dictionary
```

We can use Square Brackets([ ] with keys for accessing and initializing dictionary values. For Example:

```
>>> Months[0] = ‘January’
>>> Months[1] = ‘February’
>>>Months[2] = ‘March’
>>> print(Months)
{0 : ’January’, 1: ’February’ , 2 : ’March’}
```

```
>>> Months = dict(Jan = 31, Feb = 28, March = 31) # Creating dictionary by giving values in dict()
function
>>> print(Months)
{‘Jan’ : 31, ‘Feb’ : 28, ‘March’ : 31}
```

Accessing Elements of a Dictionary

Elements of Dictionary may be accessed by writing the Dictionary name and key within square brackets ([ ]) as given below:

```
>>> D = {0 : “Sunday”, 1 : “Monday”, 2: “Tuesday”}
>>> print(D[1])
Monday
```

Attempting to access a key that does not exist, causes an error. Consider the following statement that is trying to access a non – existent key (7) from dictionary D. it raises KeyError.

```
>>> D[7]
KeyError : 7
```

Traversing a Dictionary

Dictionary items can be accessed using a for loop.

```
for x in D:
   print(D [x])
```

Output

```
Sunday
Monday
Tuesday
```

Mutability of Dictionary

Dictionary like lists are mutable, it means dictionary can be changed, new items can be added and existing items can be updated.

Adding an Element in Dictionary

We can add new element (key : value pair) to a dictionary using assignment, but the key being added must not exist in dictionary and must be unique.

```
>>> D[4] = “Wednesday”  # if a new key is given, new item is added
>>> print(D)
{0 : “Sunday”, 1 : “Monday”, 2: “Tuesday”, 3: “Wednesday”}
```

Updating / Modifying an element in Dictionary

We can change the individual element of dictionary as given below:

```
>>> D[1] = “Mon”       # Value of Key (1) is changed
>>> print(D)
{0 : “Sunday”, 1 : “Mon”, 2: “Tuesday”, 3: “Wednesday”}
```

Dictionary Functions and Methods

Built –in functions and methods are provided by Python to manipulate Python dictionaries.

len() function:

It is used to find the length of the dictionary, i.e., the count of the key : value pair.

Exp.

```
D = {'Ram' : 20, 'Mohan' : 30, 'Rama' : 22, 'Rashi' : 32}
length = len(D)
print("Length of Dictionary : ", length)
```

Output

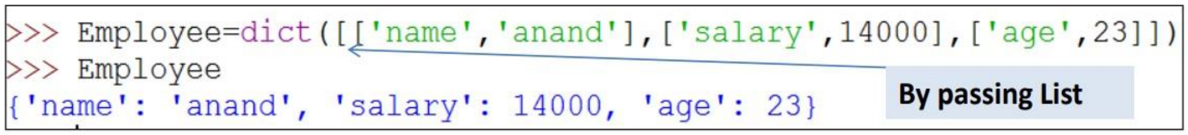
```
Length of Dictionary :  4
```

dict() function:

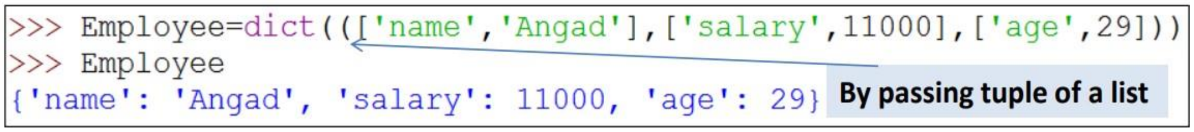
The dict() function creates a dictionary.

Exp.

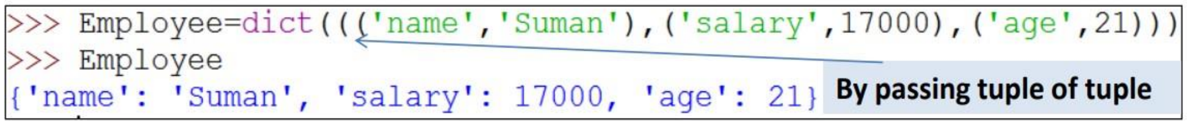
By giving Key:value pair in the form of separate sequence:



```
>>> Employee=dict([['name','anand'],['salary',14000],['age',23]])
>>> Employee
{'name': 'anand', 'salary': 14000, 'age': 23}
```



```
>>> Employee=dict((['name','Angad'],['salary',11000],['age',29]))
>>> Employee
{'name': 'Angad', 'salary': 11000, 'age': 29}
```



```
>>> Employee=dict((('name','Suman'),('salary',17000),('age',21)))
>>> Employee
{'name': 'Suman', 'salary': 17000, 'age': 21}
```

Accessing Items, Keys and Values – get(), items(), keys(), values () methods
get( ) Method : it returns value of the given key

```
>>> D = {'Ram' : 20, 'Mohan' : 30, 'Rama' : 22, 'Rashi' : 32}
>>> value = D.get('Rama', 'Key not Found')
>>> print("Age of Rama is : ", value)

>>> value = D.get('Mohit', 'Key not Found')       # if key is not in the dictionary, it shows key not found
>>> print("Age of Mohit is : ", value)
```

Output

```
Age of Rama is :  22
Age of Mohit is :  Key not Found
```

items( ) Method :
It returns all items of a dictionary in the form of list of tuple of (key:value)

```
>>> D = {'Ram' : 20, 'Mohan' : 30, 'Rama' : 22, 'Rashi' : 32}
>>> print(D.items())
```

Output

```
dict_items([('Ram', 20), ('Mohan', 30), ('Rama', 22), ('Rashi', 32)])
```

keys( ) Method :
It returns list of all the keys of the dictionary.

```
>>> D = {'Ram' : 20, 'Mohan' : 30, 'Rama' : 22, 'Rashi' : 32}
>>> print(D.keys())
```

Output

```
dict_keys(['Ram', 'Mohan', 'Rama', 'Rashi'])
```

values( ) Method :
It returns list of all the values of the dictionary.

```
>>> D = {'Ram' : 20, 'Mohan' : 30, 'Rama' : 22, 'Rashi' : 32}
>>> print(D.values())
```

Output

```
dict_values([20, 30, 22, 32])
```

update( ) Method :

This function merges key : value pairs from the new dictionary into the original dictionary, adding or replacing as needed. The items in the new dictionary are added to the old one and override any items already there with the same keys.

```
>>> D = {'Ram' : 20, 'Mohan' : 30, 'Rama' : 22, 'Rashi' : 32}
>>> D2 = {'A' : 20, 'Mohan' : 60, 'Rama' : 62, 'B' : 32}
>>> D.update(D2)
>>> print("D  => ",D)
>>> print("D2 => ", D2)
```

Output

```
D  =>  {'Ram': 20, 'Mohan': 60, 'Rama': 62, 'Rashi': 32, 'A': 20, 'B': 32}
D2 =>  {'A': 20, 'Mohan': 60, 'Rama': 62, 'B': 32}
```

Deleting elements from dictionary:

del statement

del statement is used to delete a dictionary element or dictionary entry, i.e., a key:value pair.

```
>>> D = {'Ram' : 20, 'Mohan' : 30, 'Rama' : 22, 'Rashi' : 32}
>>> print(D)
>>> del D["Mohan"]    # to delete a key from the dictionary i.e. "Mohan"
>>> print(D)
>>> del D                    # To delete the whole dictionary
```

Output

```
{'Ram': 20, 'Mohan': 30, 'Rama': 22, 'Rashi': 32}        # Complete Dictionary
{'Ram': 20, 'Rama': 22, 'Rashi': 32}                     # After deleting "Mohan"
```

clear ( ) method

It empties the dictionary.

```
>>> D = {'Ram' : 20, 'Mohan' : 30, 'Rama' : 22, 'Rashi' : 32}
>>> print(D)
>>> D.clear()
>>> print("Dictionary after Clear : ", D)
```

Output

```
{'Ram': 20, 'Mohan': 30, 'Rama': 22, 'Rashi': 32}
Dictionary after Clear : {}
```

pop ( ) method

It removes and returns the dictionary element associated to passed key.

```
>>> D = {'Ram' : 20, 'Mohan' : 30, 'Rama' : 22, 'Rashi' : 32}
>>> print("Removed Item : ", D.pop("Rama"))
```

Output

```
Removed Item :  22
```

popitem ( ) method

It removes and returns the last dictionary element.

```
>>> D = {'Ram' : 20, 'Mohan' : 30, 'Rama' : 22, 'Rashi' : 32}
>>> print("Removed Item : ", D.popitem())
```

Output

```
Removed Item :  ('Rashi', 32)
```

sorted ( ) method

It returns a sorted list of the dictionary keys. It is used as below:

```
>>> D = {'Ram' : 20, 'Mohan' : 30, 'Rama' : 22, 'Rashi' : 32}
>>> print(sorted(D))    # sort the keys in ascending order
>>> print(sorted(D), reverse = False) # sort the keys in ascending order
>>> print(sorted(D), reverse = True) # sort the keys in descending order
```

Output

```
['Mohan', 'Ram', 'Rama', 'Rashi']
['Mohan', 'Ram', 'Rama', 'Rashi']
['Rashi', 'Rama', 'Ram', 'Mohan']
```

max ( ) , min() and sum() Functions

These functions work with the keys of a dictionary. Dictionaries must be homogeneous to use the functions.

- max() function gives the maximum value
- min() function gives the minimum value
- sum() function gives the sum of keys

```
>>> D = {1: "Monday", 2: "Tuesday", 3:"Wednesday"}
>>> print("Max = ",max(D))
>>> print("min = ",min(D))
>>> print("Sum = ", sum(D))
```

Output

```
Max =  3
min =  1
Sum =  6
```

fromkeys ( ) method

This method is used to create a new dictionary from a sequence containing all the keys and common value, which will be assigned to all the keys. Keys argument must be an interable sequence (First argument). When value not given, it will take None as the values for the keys (Second Argument)

Exp.

```
>>> D = dict.fromkeys([2,4,5,8],200)            # When Value given
>>> print(D)
>>> D = dict.fromkeys([2,4,5,8])        # When value not given, it will take None as the values for the keys
print(D)
```

Output

```
        {2: 200, 4: 200, 5: 200, 8: 200}
        {2: None, 4: None, 5: None, 8: None}
```

setdefault ( ) method

This method inserts a new key: value pair only if the key does not exist. If the key already exists, it returns the current value of the key.

Exp.

```
>>> D = {1: "Monday", 2: "Tuesday", 3:"Wednesday"}
>>> D.setdefault(2,"Tuesday")          # Key already exists, it will not insert
>>> print(D)
>>> D.setdefault(4,"Thursday")         # New Key, it will be inserted
>>> print(D)
```

Output

```
        {1: 'Monday', 2: 'Tuesday', 3: 'Wednesday'}
        {1: 'Monday', 2: 'Tuesday', 3: 'Wednesday', 4: 'Thursday'}
```

Q) Write a program to count number of words in a string?

```
        str = "This is a book , book is very good"
        D = {}
        w = str.split()
        for x in w:
           key = x
           if key not in D:
              count = w.count(key)
              D[key] = count
        print("Count Frequencies...")
        print(D)
```

Output:

```
Count Frequencies...
{'This': 1, 'is': 2, 'a': 1, 'book': 2, ',': 1, 'very': 1, 'good': 1}
```

Q) Write a program to count number of characters in a string?

```
        str = "This is a book , book is very good"
        D = {}
        for x in str:
           if x not in D:
              count = str.count(x)
              D[x] = count
        print("Count Frequencies...")
        print(D)
```
Output:

```
Count Frequencies...
{'T': 1, 'h': 1, 'i': 3, 's': 3, ' ': 8, 'a': 1, 'b': 2, 'o': 6, 'k': 2, ',': 1, 'v': 1, 'e': 1, 'r': 1, 'y': 1, 'g': 1, 'd': 1}
```

MCQs

1	Dictionaries are..................... set of elements. (a) sorted (b) Ordered (c) unordered (d) random	C
2	Which of the following will create a dictionary with given keys and a common value? (a) fromkeys() (b) update () (c) setdefault() (d) all of the above	A

3	What will be printed by the following statements?
	D1 = {“cat”:12,”dog”:6,”elephant”:23,”bear”:20}
	Print(25 in D1)
	(a) True (b) False (c) Error (d) None
		B

4	What would the following code print?
	D = {‘spring’ : ‘autumn’, ‘autumn’: ’fall’, ‘fall’ : ‘spring’}
	Print(d[‘autumn’])
	(a) autumn (b) fall (c) spring (d) Error
		B

5	What will be the output of following Python code?
	d1 = {“a” : 10, “b” : 2, “c”:3}
	str1 = “ ”
	for i in d1:
	  str1 = str1 + str(d1[i]) + “ “
	  str2 = str1[:-1]
	print(str2[::-1])
	(a) 3, 2 (b) 3,2,10 (c) 3,2,01 (d) Error
		C

Question – Answers

Q1) What is a dictionary?

Ans: A dictionary is an unordered sequence of key-value pairs.

- ➢ Key and value in a key-value pair in a dictionary are separated by a colon. Further, the key : value
- ➢ pairs in a dictionary are separated by commas and are enclosed between curly parentheses.
- ➢ The keys of the dictionaries are immutable types such as Integers or Strings etc.
- ➢ Indices in a dictionary can be of any immutable type and are called keys.
- ➢ Dictionaries are mutable.

Q2) What is the use of from keys ( ) method?

Ans: This method is used to create a new dictionary from a sequence containing all the keys and common value, which will be assigned to all the keys. Keys argument must be an interable sequence (First argument). When value not given, it will take None as the values for the keys (Second Argument)

Q3) What do you understand by Mutability of Dictionary?

Ans: Dictionary is mutable, it means dictionary can be changed, new items can be added and existing items can be updated.

Q4) Why a list cannot be used as keys of dictionaries?
Ans: Lists cannot be used as keys in a dictionary because they are mutable and a Python dictionary can have only keys of immutable types.

Q5) If the addition of new key : value pair causes the size of the dictionary to grow beyond its original size, an error occurs, True or False?
Ans: False, There cannot occur an error because dictionaries being the mutable types, they can grow or shrink
on and as needed basis.

Q6) Write the output of following code:

```
x = {1:10, 2:20, 3:30}
x[4] = 20
print(x)
```

Ans:

```
{1: 10, 2: 20, 3: 30, 4: 20}
```

Q7) Write the output of following code:

```
d = {'x': 1, 'y': 2, 'z': 3}
for k in d:
   print (k, '=', d[k])
```

Ans:

```
x = 1
y = 2
z = 3
```

Q8) Write the output of following code:

```
x = {1:10}
d = {2:20, 3:30, 4:40}
x.update(d)
print(x)
```

Ans:

```
{1: 10, 2: 20, 3: 30, 4: 40}
```

Python Modules

Python Modules:

Modules are the files in python used for grouping similar codes, to get an easy access to those codes.

Python Modules facilitates reusability and easy categorization of codes.

Python Built In Modules:

Python facilitates its users a variety of modules already defined in Python library that makes Python a easy to use programming language. Some of these modules are:

Importing a Python Module:
Python modules can be imported in three ways, using:

Python import statement:
To use the functionality present in any module, you have to import it into your current program. You need to use the import keyword along with the desired module name.
Python "import" statement is used to import a Python module. For Example

```
import math
```

Python from.. import statement:
Python also facilitates to import a specific attribute from a Python module. This can be done using Python "from..import" statement. For example

```
from math import sqrt          # It will import only sqrt method from math module
```

Python from.. import* statement:
Python "from..import*" statement is used to import a complete Python module. For Example

```
from math import *             # it imports entire module
```

Python Math Module:

Python math module contains different built in mathematical functions and mathematical constants.

Math Functions:

FUNCTION	USES	Example
ceil ()	The ceil() function returns the smallest integer not less than num	math.ceil(1.03) gives 2.0 math.ceil(-1.03) gives -1.0
floor()	The floor() function returns the largest integer not greater than num	math.floor(1.03) gives 1.0 math.floor(-1.03) gives -2.0
sqrt()	The sqrt() function returns the square root of num. If num < 0 , domain error occurs.	math.sqrt(81.0) gives 9.0
pow()	The pow() function returns the base raised to exp power	math.pow(5.0,0) gives 1 math.pow(3.0,4) gives 81
fabs()	The fabs() returns the absolute value of num	math.fabs(2.0) gives 2.0 math.fabs(-2.0) gives 2.0

sin()	The sin() function returns the sine of arg. The value of arg must be in radians.	math.sin(val) (val is a number)
cos()	The cos() function returns the cosine of arg. The value of arg must be in radians.	math.cos(val) (val is a number)
tan()	The tan() function returns the tangent of arg. The value of arg must be in radians.	math.tan(val) (val is a number)

The math module of Python also makes available two useful constants namely pi and e. Which we can use as
math.pi gives the mathematical constant π = 3.141592…, to available precision.
math.e gives the mathematical constant e = 2.718281…, to available precision.

Random Module

Function	**Uses**	**Example**
random()	Used to generate a random floating – point number between 0.0 to 1.0 that is, including zero but excluding one.	**Import random()** **print(random.random())** **0.022353**
randint()	**randint(start, stop)** is used to generate a random number between **start** and **stop** where both the numbers are inclusive.	**random.randint(10,15)** it may generate any one of the values given below 10,11,12,13,14,15
randrange()	• **random.randrange(<stopvalue**>) is used to generate a random number in the range 0 to <stopvalue> • **random.randrange(start,stop)** is used to generate a random number in the range start to stop • **random.randrange(start,stop,step)** is used to generate a random number in the range start to stop, but here, the difference between two such generated random numbers will be a multiple of step value.	• **random.randrange(35)** It will generate a random number from 0 to 35 • **random.randrange(11,45)** It will generate a random number in the range 11 .. 45 • **random.randrange(11,45, 4)** it may generate any one of the values given below11,15,19,23,27,31,35,39,43

Statistics Module

mean()	This method calculates the mean (average) of the givendata set. It add up all the given values, then divide by number of values in the set.	import statistics
		seq = [5,6,7,5,6,5,5,9,11,12,23,5]
		statistics.mean(seq)
		it gives 8.25

median()	This method calculates the median (middle value) of the given data set.	import statistics
		seq = [5,6,7,5,6,5,5,9,11,12,23,5]
		statistics.median(seq)
		it gives 6.0

mode()	This method determines the central tendency of numerical or nominal data. It is used to find the most frequent number in a sequence.	import statistics
		seq = [5,6,7,5,6,5,5,9,11,12,23,5]
		statistics.mode(seq)
		it gives 5

MCQs

1. To include the use of functions which are present in the random library, we must use the option:
a) import random b) random.h c) import.random d) random.random
2. The output of the following Python code is either 1 or 2.
```
import random
        random.randint (1 , 2)
```
a) True b) False

3. What will be the output of the following Python function if the random module has already been imported?
```
        random.randint(3.5 ,7)
```
a) b) Any integer between 3.5 and 7, including 7
b) Any integer between 3.5 and 7, excluding 7
c) The integer closest to the mean of 3.5 and 7
d) Error

4. What will be the output of the following Python code?
```
        random.randrange(0,91,5)
```
a) 18 b) 10 c) 79 d) 95

5. Which of the following is not a built in module in Python?
a) math b) os c) pi d) random

6. What will the following code result as?
```
        import math
        x = 100
```

```
print(x > 0 and math.sqrt(x))
```

a) True b) 1 c) 10 d) 10.0

7. What will the following code result as?

```
import statistics
seq = [10,20,20,30,12,10]
print(statistics.mean())
```

a) 30 b) 10 c) 17 d) 1

8. What may be the possible values printed by following statements?

```
import random
for I in range(3):
        print(random.randint(10,18))
```

a) 13 12 10 b) 11 18 19 c) 10 9 18 d) 10 9 8

Answers:

1	2	3	4	5	6	7	8
A	A	d	b	c	d	c	a

Q1) What is random module in Python?
Ans: Python Random module is an in-built module of Python that is used to generate random numbers in Python.
Q2) Write a code fragment to generate a random floating number between 45.0 and 95.0. Print this number along with its nearest integer greater than it.
Ans:

```
import random
import math
fnum = random.random() * (95 - 45) + 45
inum = math.ceil(fnum)
print("Random Numbers between 45...95: ")
print(fnum)
print("Nearest higher integer : ", inum)
```

Output

```
Random Numbers between 45...95:
56.31601558476624
Nearest higher integer :  57
```

Q2) Write a program to generate two random integers between 500 and 760. Print the average of these number along with these numbers.
Ans:

```
import random
num1 = random.randint(500,760) - 500
num2 = random.randint(500,760) - 500
avg = (num1 + num2)/2
print("Number 1 : ", num1)
print("Number 2 : ", num2)
print("Average  : ", avg)
```

Output

```
Number 1 :  147
Number 2 :  153
Average :  150.0
```

Q2) Given a list containing these values [22,13,28,13,22,25,7,13,25] . Write code to calculate mean, median and mode of this list?

Ans:

```
import statistics as s
L = [22,13,28,13,22,25,7,13,25]
LMean = s.mean(L)
LMedian = s.median(L)
LMode = s.mode(L)
print("List....... ")
print(L)
print("Mean : ", LMean)
print("Median : ", LMedian)
print("Mode  : ", LMode)
```

Output

```
List........
[22, 13, 28, 13, 22, 25, 7, 13, 25]
Mean  :  18.666666666666668
Median :  22
Mode  :  13
```