WINTER – 2023 EXAMINATION
Model Answer – Only for the Use of RAC Assessors

Subject Name: Programming with Python **Subject Code:** 22616

Important Instructions to examiners:

1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills.
4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.
8) As per the policy decision of Maharashtra State Government, teaching in English/Marathi and Bilingual (English + Marathi) medium is introduced at first year of AICTE diploma Programme from academic year 2021-2022. Hence if the students in first year (first and second semesters) write answers in Marathi or bilingual language (English +Marathi), the Examiner shall consider the same and assess the answer based on matching of concepts with model answer.

Q. No.	Sub Q. N.	Answer	Marking Scheme
1		**Attempt any FIVE of the following:**	**10 M**
	a)	**Enlist applications for python programming.**	**2 M**
	Ans	• Google's App Engine web development framework uses Python as an application language. • Maya, a powerful integrated 3D modeling and animation system, provides a Python scripting API. • Linux Weekly News, published by using a web application written in Python. • Google makes extensive use of Python in its Web Search Systems. • The popular YouTube video sharing service is largely written in Python programming. • The NSA uses Python for cryptography and intelligence analysis. • iRobot uses Python programming to develop commercial and military robotic devices. • The Raspberry Pi single-board computer promotes Python programming as its educational language. • Nextflix and Yelp have both documented the role of Python in their software infrastructures. • Industrial Light and Magic, Pixar and others uses Python in the production of animated movies. • Desktop GUI Applications • Image Processing Applications • Scientific and Numeric Applications • Audio and Video Based Applications • 3D CAD Applications	Any two application, One application for 1 M each

		• Software Development • Web Applications	
	b)	**Write the use of elif keyword in python.**	**2 M**
	Ans	**elif'** stands for 'else if' and is used in Python programming to test multiple conditions. The if statements are executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the ladder is bypassed. If none of the conditions is true, then the final else statement will be executed.	Correct explanation 2 M
	c)	**Describe the Role of indentation in python.**	**2 M**
	Ans	• Indentation refers to the spaces at the beginning of a code line. • Generally, four whitespaces are used for indentation and is preferred over tabs. • Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important. • Indentation helps to convey a better structure of a program to the readers. It is used to clarify the link between control flow constructs such as conditions or loops, and code contained within and outside of them. • Python uses indentation to indicate a block of code. **Example:** if 5 > 2: print("Five is greater than two!")	Correct Role 2 M
	d)	**Define Data Hiding concept? Write two advantages of Data Hiding.**	**2 M**
	Ans	• Data hiding is a concept which underlines the hiding of data or information from the user. • Data hiding is a software development technique specifically used in Object-Oriented Programming (OOP) to hide internal object details (data members). • Data hiding includes a process of combining the data and functions into a single unit to conceal data within a class by restricting direct access to the data from outside the class. **Advantages of Data Hiding** • Data hiding ensures exclusive data access to class members and protects object integrity by preventing unintended or intended changes. • Data hiding is also known as information hiding. An object's attributes may or may not be visible outside the class definition. • Data hiding also minimizes system complexity for increase robustness by limiting interdependencies between software requirements. • The objects within the class are disconnected from irrelevant data. • It heightens the security against hackers that are unable to access confidential data. • It helps to prevent damage to volatile data by hiding it from the public. • A user outside from the organization cannot attain the access to the data. • Within the organization/ system only specific users get the access. This allows better operation.	Any relevant Definition 1 M, any two Advantages 1 M
	e)	**State use of namespace in python.**	**2 M**
	Ans	• Namespaces bring you three advantages: they group names into logical containers, they prevent clashes between duplicate names, and third, they provide context to names.	Correct/relevant use 2 M

		• Namespaces prevent conflicts between classes, methods and objects with the same name that might have been written by different people. • A namespace is a system to have a unique name for each and every object in Python. An object might be a variable or a method. Python itself maintains a namespace in the form of a Python dictionary. • A namespace in python is a collection of names. So, a namespace is essentially a mapping of names to corresponding objects.	
	f)	**State the use of read() and readline () functions in python file handling.**	**2 M**

	Ans	**1. read([n]) Method:**
		The read method reads the entire contents of a file and returns it as a string, if number of bytes are not given in the argument. If we execute read(3), we will get back the first three characters of the file.
		Example: for read( ) method.
		f=open("sample.txt","r")
		print(f.read(5)) # read first 5 data
		print(f.read()) # read rest of the file
		
		2. readline([n]) Method:
		The readline() method just output the entire line whereas readline(n) outputs at most n bytes of a single line of a file. It does not read more than one line. Once, the end of file is reached, we get empty string on further reading.
		Example: For readline ( ) method.
		f=open("sample.txt","r")
		print(f.readline()) # read first line followed by\n
		print(f.readline(3))
		print(f.readline())
			read() 1 M and readline()
			1 M
			(example is not mandatory)

	g)	**Explain two ways to add objects / elements to list.**	**2 M**

	Ans	**1)append method:** The append() method adds an element to the end of a list. We can insert a single item in the list data time with the append().
		Example: For append() method.
		>>> list1=[10,20,30]
		>>> list1
		[10, 20, 30]
		>>> list1.append(40) # add element at the end of list
		>>> list1
		[10, 20, 30, 40]
		
		2. extend() Method: The extend() method extends a list by appending items. We can add several items using extend() method.
		Example: Program for extend() method.
		>>>list1=[10, 20, 30, 40]
		>>>list1
		[10, 20, 30, 40]
		>>> list1.extend([60,70]) #add elements at the end of list
		>>> list1
		[10, 20, 30, 40, 60, 70]
			1 Method for 1 M (any two methods)
			(example is not mandatory)

3. insert() Method: We can insert one single item at a desired location by using the method insert() or insert multiple items by squeezing it into an empty slice of a list.
Example: Program for insert() method.

```
>>> list1=[10, 20]
>>>list1
[10,20]
>>> list1.insert(1,30)
>>> list1
[10, 30, 20]
```

2. **Attempt any THREE of the following:** **12 M**

a) **Explain membership and identity operators in Python.** **4 M**

Ans

Membership Operators:
The membership operators in Python are used to find the existence of a particular element in the sequence, and used only with sequences like string, tuple, list, dictionary etc.
Membership operators are used to check an item or an element that is part of a string, a list or a tuple. A membership operator reduces the effort of searching an element in the list.
Python provides 'in' and 'not in' operators which are called membership operators and used to test whether a value or variable is in a sequence.

Operator	Description	Example

in	True if value is found in list or in sequence, and false it item is not in list or in sequence	>>> x="Hello World"
		>>> print('H' in x)
		True

not in	True if value is not found in list or in sequence, and false it item is in list or in sequence.	>>> x="Hello World"
		>>> print("Hello" not in x)
		False

Identity Operators: Sometimes, in Python programming a need to compare the memory address of two objects; this is made possible with the help of the identity operator. Identity operators are used to check whether both operands are same or not. Python provides 'is' and 'is not' operators which are called identity operators and both are used to check if two values are located on the same part of the memory. Two variables that are equal does not imply that they are identical.

Operator	Description	Example

is	Return true, if the variables on either side of the operator point to the same object and false otherwise.	>>> a=3
		>>> b=3
		>>> print(a is b)
		True

is not	Return false, if the variables on either side of the operator point to the same object and true otherwise.	>>> a=3
		>>> b=3
		>>> print(a is not b)
		False

Membership operator 2 M and Identity operator 2 M

b) **Write python program to display output like.** **4 M**

2
4 6 8
10 12 14 16 18

Ans

```
a=2
for i in range(1,5):
  for j in range(i):
     print(a,end="\t")
     a+=2
  print()
```

Correct or any relevant Logic/ any other suitable Program 4 M

c) Explain four built-in list functions. **4 M**

Ans

Sr. No.	Function	Description	Example

1	len(list)	It returns the length of the list.	>>> list1
			[1, 2, 3, 4, 5]
			>>> len(list1)
			5

2	max(list)	It returns the item that has the maximum value in a list.	>>> list1
			[1, 2, 3, 4, 5]
			>>> max(list1)
			5

3	sum(list)	Calculates sum of all the elements of list.	>>>list1 [1, 2, 3, 4, 5] >>>sum(list1) 15

4	min(list)	It returns the item that has the minimum value in a list.	>>> list1
			[1, 2, 3, 4, 5]
			>>> min(list1)
			1

5	list(seq)	It converts a tuple into a list.	>>> list1
			[1, 2, 3, 4, 5]
			>>> list(list1)
			[1, 2, 3, 4, 5]

6	list.append(item)	It adds the item to the end of the list.	>>> list1
			[1, 2, 3, 4, 5]
			>>> list1.append(6)
			>>> list1
			[1, 2, 3, 4, 5, 6]

7	list.count(item)	It returns number of times the item occurs in the list.	>>> list1
			[1, 2, 3, 4, 5, 6, 3]
			>>> list1.count(3)
			2

8	list.extend(seq)	It adds the elements of the sequence at the end of the list.	>>> list1
			[1, 2, 3, 4, 5]
			>>> list2
			['A', 'B', 'C']
			>>> list1.extend(list2)
			>>> list1
			[1, 2, 3, 4, 5, 'A', 'B', 'C']

9	list.pop(item=list[-1])	It deletes and returns the last element of the list.	>>> list1
			[1, 2, 7, 3, 4, 5, 3]
			>>> list1.pop()
			3

Any 4 functions 4 M, 1 M each

					>>> list1.pop(2) 7

		10	list.remove(item)	It deletes the given item from the list.	>>> list1
					[1, 2, 3, 4, 5]
					>>> list1.remove(3)
					>>> list1
					[1, 2, 4, 5]

		11	list.reverse()	It reverses the position (index number) of the items in the list.	>>> list1
					[1, 2, 3, 4, 5]
					>>> list1.reverse()
					>>> list1
					[5, 4, 3, 2, 1]

		12	list.sort([func])	It sorts the elements inside the list and uses compare function if provided.	>>> list1
					[1, 3, 2, 5, 4]
					>>> list1.sort()
					>>> list1
					[1, 2, 3, 4, 5]

	d)	**Write python program using module, show how to write and use module by importing it.**			**4 M**

	Ans	**For creating a module write following code and save it as p1.py**
		#p1.py
		def add(a, b):
		"This function adds two numbers and return the result"
		result = a + b
		return result
		
		def sub(a, b):
		"This function subtract two numbers and return the result"
		result = a – b
		return result
		
		Import the definitions inside a module:
		
		import p1
		print(p1.add(10,20))
		print(p1.sub(20,10))
		
		Output:
		30
		10
						Write module 2 M and Import 2 M

3.		**Attempt any <u>THREE</u> of the following:**			**12 M**
	a)	**Describe various modes of file object? Explain any two in detail.**			**4 M**

Ans

Like, C, C++, and Java, a file in Python programming can be opened in various modes depending upon the purpose. For that, the programmer needs to specify the mode whether read 'r', write 'w', or append 'a' mode. Apart from this, two other modes exist, which specify to open the file in text mode or binary mode.

1. The **text mode** returns strings while reading from the file. The default is reading in text mode.

2. The **binary mode** returns bytes and this is the mode to be used when dealing with non-text files like image or executable files.

The text and binary modes are used in conjunction with the r, w, and a modes. The list of all the modes used in Python are given in following table:

Sr. No.	Mode	Description
1	r	Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.
2	rb	Opens a file for reading only in binary format. The file pointer is placed at the beginning of the file. This is the default mode.
3	r+	Opens a file for both reading and writing. The file pointer placed at the beginning of the file.
4	rb+	Opens a file for both reading and writing in binary format. The file pointer placed at the beginning of the file.
5	w	Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
6	wb	Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
7	w+	Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
8	wb+	Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
9	a	Opens a file for appending. The file pointer is at the end of the file if the file exists.

Listing of modes- 1 M and explain ant 2- 3 M

			That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.	
		10	ab	Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
		11	a+	Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.
		12	ab+	Opens a file for both appending and reading in binary format. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.
		13	t	Opens in text mode (default).
		14	b	Opens in binary mode
		15	+	Opens a file for updating (reading and writing).
	b)	**Explain use of Pass and Else keyword with for loops in python.**		**4 M**

	Ans	**Pass Statement:** It is used when a statement is required syntactically but we do not want any command or code to execute. A pass statement in Python also refers to as a will statement. The pass statement is a null operation; nothing happens when it executes. The pass is also useful in places where your code will eventually go, but has not been written yet.
		Syntax: pass
		Example: For pass statement.
		for i in range(1,11):
		if i%2==0: # check if the number is even
		pass # (No operation)
		else:
		print("Odd Numbers: ",i)
		Output:
		Odd Numbers: 1
		Odd Numbers: 3
		Odd Numbers: 5
		Odd Numbers: 9
		Odd Numbers: 7
				Pass for 2 M and Else for 2 M

Else Statement: The else block just after for/while is executed only when the loop is NOT terminated by a break statement. The else keyword in a for loop specifies a block of code to be executed when the loop is finished:

Example:

```
for i in range(1, 4):
   print(i)
else:                  # Executed because no break in for
   print("Finally Exit")
```

Output:

1
2
3
Finally Exit

c) **T = ('spam, Spam', SPAM!', 'SaPm')**
print (T [2])
print (T[-2])
print (T[2:])
print (List (T)) — **4 M**

Ans

Input
T = ('spam', 'Spam', 'SPAM!', 'SaPm')

Python statement	Output
print (T [2])	**SPAM!**
print (T[-2])	**SPAM!**
print (T[2:])	**['SPAM!', 'SaPm']**
print (list (T))	**['spam', 'Spam', 'SPAM!', 'SaPm']**

Each Print Statement/ output for 1 M

d) **Explain method overloading and overriding in python.** — **4 M**

Ans

Method Overloading: Method overloading is the ability to define the method with the same name but with a different number of arguments and data types. With this ability one method can perform different tasks, depending on the number of arguments or the types of the arguments given. Method overloading is a concept in which a method in a class performs operations according to the parameters passed to it. Python does **not** support method overloading, that is, it is not possible to define more than one method with the same name in a class in Python. This is because method arguments in python do not have a type. A method accepting one argument can be called with an integer value, a string or a double as shown in next example.

```
class Demo:
def method(self, a):
    print(a)
obj= Demo( )
obj.method(50)
obj.method('Meenakshi')
obj.method(100.2)
```

Output:

50
Meenakshi
100.2

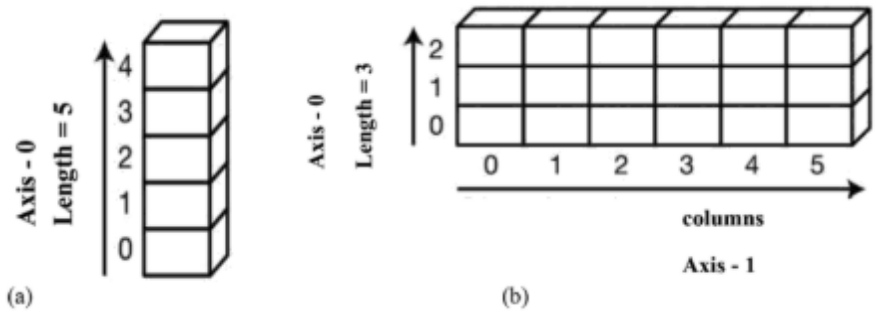
Method Overloading 2 M and Overriding 2 M

		It is clear that method overloading is not supported in python but that does not mean that we cannot call a method with different number of arguments. There are a couple of alternatives available in python that make it possible to call the same method but with different number of arguments. **Method Overriding:** Overriding is the ability of a class to change the implementation of a method provided by one of its base class. Method overriding is thus a strict part of the inheritance mechanism. To override a method in the base class, we must define a new method with same name and same parameters in the derived class. Overriding is a very important part of OOP since it is the feature that makes inheritance exploit its full power. Through method overriding a class may "copy" another class, avoiding duplicated code, and at the same time enhance or customize part of it. **Example:** For method overriding. class A: # parent class "Parent Class" def display(self): print ('This is base class.') class B(A): #derived class "Child/Derived class" def display(self): print ('This is derived class.') obj = B() # instance of child obj.display() # child calls overridden method **Output:** This is derived class.	
4.		**Attempt any <u>THREE</u> of the following:**	**12 M**
	a)	**Explain different functions or ways to remove key : value pair from Dictionary.**	**4 M**

	Ans	**pop():** We can remove a particular item in a dictionary by using the method **pop()**. This method removes as item with the provided key and returns the value.
		Example:
		>>> squares
		{1: 1, 2: 4, 3: 9, 4: 16}
		>>> squares.pop(2) # remove a particular item
		4
		>>> squares
		{1: 1, 3: 9, 4: 16}
		
		Popitem(): The method, **popitem()** can be used to remove and return an arbitrary item (key, value) form the dictionary.
		Example:
		>>> squares={1:1,2:4,3:9,4:16,5:25}
		>>> squares
		{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
			Any two functions, 2m each

		>>> print(squares.popitem()) # remove an arbitrary item
		(5, 25)
		>>> squares
		{1: 1, 2: 4, 3: 9, 4: 16}
		
		Clear(): All the items can be removed at once using the **clear()** method.
		Example:
		>>> squares
		{1: 1, 4: 16}
		>>> squares.clear() # removes all items
		>>> squares
		{}
		
		Del(): We can also use the **del** keyword to remove individual items or the entire dictionary itself.
		Example:
		>>> squares
		{1: 1, 3: 9, 4: 16}
		>>> del squares[3] # delete a particular item
		>>> squares
		{1: 1, 4: 16}	

	b)	**Explain Numpy package in detail.**	**4 M**
	Ans	• NumPy is the fundamental package for scientific computing with Python. NumPy stands for "Numerical Python". It provides a high-performance multidimensional array object, and tools for working with these arrays. • An array is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers and represented by a single variable. NumPy's array class is called **ndarray**. It is also known by the **alias array**. • In NumPy arrays, the individual data items are called elements. All elements of an array should be of the same type. Arrays can be made up of any number of dimensions. • In NumPy, dimensions are called axes. Each dimension of an array has a length which is the total number of elements in that direction. • The size of an array is the total number of elements contained in an array in all the dimension. The size of NumPy arrays are fixed; once created it cannot be changed again. • Numpy arrays are great alternatives to Python Lists. Some of the key advantages of Numpy arrays are that they are fast, easy to work with, and give users the opportunity to perform calculations across entire arrays. • Fig. shows the axes (or dimensions) and lengths of two example arrays; (a) is a one-dimensional array and (b) is a two-dimensional array.	Suitable explanation 4 M



- A one dimensional array has one axis indicated by Axis-0. That axis has five elements in it, so we say it has length of five.
- A two dimensional array is made up of rows and columns. All rows are indicated by Axis-0 and all columns are indicated by Axis-1. If Axis-0 in two dimensional array has three elements, so its length it three and Axis-1 has six elements, so its length is six.

Execute Following command to install numpy in window, Linux and MAC OS:

python -m pip install numpy
To use NumPy you need to import Numpy:
import numpy as np # alias np
Using NumPy, a developer can perform the following operations:
1. Mathematical and logical operations on arrays.
2. Fourier transforms and routines for shape manipulation.
3. Operations related to linear algebra.
4. NumPy has in-built functions for linear algebra and random number generation.

	c)	**Explain seek ( ) and tell ( ) function for file pointer manipulation in python with example.**	**4 M**

	Ans	**seek():** In python programming, within file handling concept seek() function is used to shift/change the position of file object to required position. By file object we mean a cursor. And it's cursor, who decides from where data has to be read or write in a file.
		Syntax:
		f.seek(offset, fromwhere)
		where offset represents how many bytes to move fromwhere, represents the position from where the bytes are moving.
		Example:
		f = open("demofile.txt", "r")
		f.seek(4) #sets Reference point to fourth index position from the beginning
		print(f.readline())
		
		tell(): tell() returns the current position of the file pointer from the beginning of the file.
		Syntax: file.tell()
		Example:
		f = open("demofile.txt", "r")
		# points at the start
		print(f.tell())
			For seek() method: 2 M and for Tell() method 2 M

	d)	**WAP to read contents of first.txt file and write same content in second.txt file.**	**4 M**

	Ans	with open('first.txt', 'r') as f: # Open the first file for reading
		contents = f.read() # Read the contents of the file
		
		with open('second.txt', 'w') as f: # Open the second file for writing
		f.write(contents) # Write the contents of the first file to the second file
			Correct logic program/any suitable program 4 M

5.		**Attempt any <u>TWO</u> of the following:**	**12 M**
	a)	**Explain any four set operations with example.**	**6 M**

	Ans	Set Operations:
		1. **Set Union:** The union of two sets is the set of all the elements of both the sets without duplicates. You can use the '|' operator to find the union of a Python set.
		>>> first_set = {1, 2, 3}
		>>> second_set = {3, 4, 5}
		>>> first_set.union(second_set)
		{1, 2, 3, 4, 5}
		>>> first_set | second_set # using the '|' operator
		{1, 2, 3, 4, 5}
		
		2**. Set Intersection:** The intersection of two sets is the set of all the common elements of both the sets. You can use the '&' operator to find the intersection of a Python set.
		>>> first_set = {1, 2, 3, 4, 5, 6}
		>>> second_set = {4, 5, 6, 7, 8, 9}
		>>> first_set.intersection(second_set)
		{4, 5, 6}
		>>> first_set & second_set # using the '&' operator
		{4, 5, 6}
		
		3. **Set Difference**
		The difference between two sets is the set of all the elements in first set that are not present in the second set. You would use the '–' operator to achieve this in Python.
		>>> first_set = {1, 2, 3, 4, 5, 6}
		>>> second_set = {4, 5, 6, 7, 8, 9}
		>>> first_set.difference(second_set)
		{1, 2, 3}
		>>> first_set - second_set # using the '-' operator
		{1, 2, 3}
		>>> second_set - first_set
		{8, 9, 7}
		
		4. **Set Symmetric Difference:** The symmetric difference between two sets is the set of all the elements that are either in the first set or the second set but not in both. You have the choice of using either the symmetric_difference() method or the ^ operator to do this in Python.
		>>> first_set = {1, 2, 3, 4, 5, 6}
		>>> second_set = {4, 5, 6, 7, 8, 9}
		>>> first_set.symmetric_difference(second_set)
		{1, 2, 3, 7, 8, 9}
			Each operation -1 ½ M

```
>>> first_set ^ second_set          # using the `^` operator
{1, 2, 3, 7, 8, 9}
```

b) Explain building blocks of python. **6 M**

Ans

Any 6 building blocks=6M

1) Python Identifiers: Variable name is known as identifier.
To name an identifier following are the rules:

- The first character of the variable must be an alphabet or underscore (_).
- All the characters except the first character may be an alphabet of lower-case(a-z), upper-case (A-Z), underscore or digit (0-9).
- Identifier name must not contain any white-space, or special character (!, @, #, %, ^, &, *).
- Identifier name must not be similar to any keyword defined in the language. o Identifier names are case sensitive for example my name, and MyName is not the same.

Eg: a,b,c=5,10,15

2) Reserved Words
The following list shows the Python keywords. These are reserved words and cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

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

3) Indentation: Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is compulsory.
The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

```
if True:
   print "True"
else:
   print "False"
```

Thus, in Python all the continuous lines indented with same number of spaces would form a block.

4) Python Types: The basic types in Python are String (str), Integer (int), Float (float), and Boolean (bool). There are also built in data structures to know when you learn Python. These data structures are made up of the basic types, you can think of them like Legos, the data structures are made out of these basic types. The core data

structures to learn in Python are List (list), Dictionary (dict), Tuple (tuple), and Set (set).

Strings

Strings in Python are assigned with single or double quotations. As in many other programming languages, characters in strings may be accessed as if accessing an array. In the example below we'll assign a string to a variable, access the first element, check for a
substring, and check the length of the string.

```
x = 'abcd'
```

Numbers: Integers and Floats in Python are both Number types. They can interact with each other, they can be used in all four operations. In the example code we'll explore how these numbers can interact.

```
x = 2.5
y = 2
```

Boolean: Boolean variables in Python are either True or False. They will also return True for 1 and False for 0. The example shows how to assign either True or False to a variable in Python

```
x = True
y = False
```

Lists: Lists in Python are represented with brackets. Like characters in a string, the elements in a list can be accessed with brackets. Lists can also be enumerate'd on to return both the index and the element. We'll go over enumerate when we cover for loops in Python. The example code shows how to declare lists, print elements in them, add to them, and remove from them.

```
x = [10, 25, 63, 104]
y = ['a', 'q', 'blah']
```

Dictionaries: Dictionaries in Python are a group of key-value pairs. Dictionaries are declared with curly braces and their entries can be accessed in two ways, a) with brackets, and b) with .get. The example code shows how we can access items in a dictionary.

```
_dict = {
'a': 'Sally sells sea shells',
'b': 'down by the seashore'
}
```

Tuples: Tuples is an immutable sequence in Python. Unlike lists, you can't move objects out of order in a Tuple. Tuples are declared with parenthesis and must contain a comma (even if it is a tuple of 1). The example below shows how to add tuples, get a tuple from a list, and return information about it.

```
x = (a, b)
```

Sets: Sets in Python are the non-duplicative data structure. That means they can only store one of an element. Sets are declared with curly braces like dictionaries, but do not contain ':' in them. The example code shows how to turn a list into a set, access set elements by index, add to a set, and remove from a set.

```
# we can turn a list into a set
x = ['a', 'a', 'b', 'c', 'c']
x = set(x)
```

5) Control structures: Control structures are used to determine the flow of execution of a Python program. Examples of control structures in Python include if-else statements, for and while loops, and try-except blocks.

6) Functions: Functions are reusable blocks of code that perform specific tasks. In Python, functions are defined using the def keyword.

7) Modules: Python modules are files that contain Python code and can be imported into other Python programs to reuse code and simplify development.

8) Packages: Packages are collections of related Python modules that can be installed and imported together. Packages are commonly used in Python for organizing and distributing libraries and tools.

	c)	**Write a program illustrating use of user defined package in python.**	**6 M**

	Ans	**# student.py**
		class Student:
		  def __init__(self, student):
		  self.name = student['name']
		  self.gender = student['gender']
		  self.year = student['year']
		
		  def get_student_details(self):
		  return f"Name: {self.name}\nGender: {self.gender}\nYear: {self.year}"
		
		# faculty.py
		class Faculty:
		  def __init__(self, faculty):
		  self.name = faculty['name']
		  self.subject = faculty['subject']
		
		  def get_faculty_details(self):
		  return f"Name: {self.name}\nSubject: {self.subject}"
		
		# testing.py
		# importing the Student and Faculty classes from respective files
		from student import Student
		from faculty import Faculty
			Create package: 2m
			
			Importing packages: 2m
			
			Logic: 2m
			
			(Any Similar/Suitable other logic/program can consider)

		# creating dicts for student and faculty
		student_dict = {'name' : 'ABC', 'gender': 'Male', 'year': '3'}
		faculty_dict = {'name': 'XYZ', 'subject': 'Programming'}
		
		# creating instances of the Student and Faculty classes
		student = Student(student_dict)
		faculty = Faculty(faculty_dict)
		
		# getting and printing the student and faculty details
		print(student.get_student_details())
		print()
		print(faculty.get_faculty_details())
		
		Output :
		Name: ABC
		Gender: Male
		Year: 3
		
		Name: XYZ
		Subject: Programming	

6.		**Attempt any <u>TWO</u> of the following:**	**12 M**
	a)	**Write a program to create class student with Roll no. and Name and display its contents**	**6 M**

	Ans	class Student:
		  def __init__(self, name, rollno):
		  self.name = name
		  self.rollno = rollno
		
		  def __str__(self):
		  return f"{self.name}({self.rollno})"
		
		s1 = Student("ABC", 32)
		print(s1)
		
		Output:
		ABC 32
			Create Class: 3m
			
			Display Method: 3m

	b)	**Write program to implement concept of inheritance in python.**	**6 M**

	Ans	class Animal: #super class
		
		# attribute and method of the parent class
		name = ""
		def eat(self):
		print("I can eat")
		
		# inherit from Animal
		class Dog(Animal): sub(2 M class
		
		# new method in subclass
		def display(self):
		# access name attribute of superclass using self
		print("My name is ", self.name)
		
		# create an object of the subclass
		labrador = Dog()
		
		# access superclass attribute and method
		labrador.name = "Rohu"
		labrador.eat()
		# call subclass method
		
		labrador.display()
		
		Output:
		I can eat
		My name is Rohu
			Any other suitable program can consider
			
			Class creation: 2m
			
			Inherit one class to another: 2m
			
			Logic: 2m

	c)	**List and explain any four built-in functions on set.**	**6 M**
	Ans	**Built-in Functions with Set** add() discard() copy() remove() clear() union() difference() intersection() discard() issubset() issuperset() pop() update() symmetric_difference()	Listing: 2m Explanation of any two function: 4m (2 M each)

add():
Adds an element to the set. If an element is already exist in the set, then it does not add that element.
Example:

```
s = {'g', 'e', 'k', 's'}
# adding f into set s
s.add('f')
print('Set after updating:', s)
```

output:
Set after updating: {'s', 'f', 'e', 'g', 'k'}

discard():
Removes the element from the set
Example:

```
s = {'g', 'e', 'k', 's'}
print('Set before discard:', s)
s.discard('g')
print('\nSet after discard g:', s)
```

Output:
Set before discard: {'s', 'e', 'k', 'g'}
Set after discard g: {'s', 'e', 'k'}

remove():
Removes the specified element from the set. If the specified element not found, raise an error.
Example:

```
s = {'g', 'e', 'k', 's'}
print('Set before remove:', s)
s.remove('e')
print('\nSet after remove e:', s)
```

Output:
Set before remove: {'s', 'k', 'e', 'g'}
Set after remove e: {'s', 'k', 'g'}

clear():
Removes all elements from the set
Example:

```
s = {'g', 'e', 'k', 's'}
print('Set before clear:', s)
s.clear()
print('\nSet after clear:', s)
```

Output:
Set before clear: {'g', 'k', 's', 'e'}
Set after clear: set()

copy(): Returns a shallow copy of the set

Example:

```
s = {'g', 'e', 'k', 's'}
p=s.copy()
print("original set:",s)
print("Copied set:",p)
```

Output:

original set: {'k', 's', 'g', 'e'}
Copied set: {'k', 's', 'g', 'e'}

Union(): The set.union() method returns a new set with distinct elements from all the given sets.

Example:

```
nums1 = {1, 2, 2, 3, 4, 5}
nums2 = {4, 5, 6, 7, 7, 8}
distinct_nums = nums1.union(nums2)
print("The union of two sets is: ", distinct_nums)
```

Output:

The union of two sets is: {1, 2, 3, 4, 5, 6, 7, 8}

Difference(): The set.difference() method returns the new set with the unique elements that are not in the other set passed as a parameter.

Example:

```
nums1 = {1, 2, 2, 3, 4, 5}
nums2 = {4, 5, 6, 7, 8, 8}
nums3 = nums1.difference(nums2)
nums4 = nums2.difference(nums1)
print("nums1 - nums2: ", nums3)
print("nums2 - nums1: ", nums4)
```

Output:

nums1 - nums2: {1, 2, 3}
nums2 - nums1: {8, 6, 7}

Intersection(): The set.intersection() method returns a new set with the elements that are common in the given sets.

Example:

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
z = x.intersection(y)
print(z)
```

Output:

apple