Model Question Paper-I with effect from 2022-23 (CBCS Scheme)

USN | | | | | | | | | | |

First/Second Semester B.E. Degree Examination

Introductionto Python Programming

TIME: 03 Hours **Max. Marks: 100**

Note: 01. Answer any **FIVE** full questions, choosing at least **ONE** question from each **MODULE**.

		Module -1	*Bloom's Taxonomy Level	Marks
Q.01	a	With Python programming examples to each, explain the syntax and control flow diagrams of break and continue statements.	L2	08
	b	Explain TWO ways of importing modules into application in Python with syntax and suitable programming examples.	L2	06
	c	Write a function to calculate factorial of a number. Develop a program to compute binomialcoefficient (Given N and R).	L3	06
		OR		
Q.02	a	Explain looping control statements in Python with a syntax and example to each.	L2	06
	b	Develop a Python program to generate Fibonacci sequence of length (N). Read N from the console.	L3	04
	c	Write a function named DivExp which takes TWO parameters a, b and returns a value c (c=a/b). Write suitable assertion for a>0 in function DivExp and raise an exception for when b=0. Develop a Python program which reads two values from the console and calls a function DivExp.	L3	06
	d	Explain FOUR scope rules of variables in Python.	L2	04
		Module-2		
Q. 03	a	Explain with a programming example to each: (ii) get() (iii) setdefault()	L2	06
	b	Develop suitable Python programs with nested lists to explain copy.copy( ) and copy.deepcopy( ) methods.	L3	08
	c	Explain append() and index() functions with respect to lists in Python.	L2	06
		OR		
Q.04	a	Explain different ways to delete an element from a list with suitable Python syntax and programming examples.	L2	10
	b	Read a multi-digit number (as chars) from the console. Develop a program to print the frequency of each digit with suitable message.	L3	06
	c	Tuples are immutable. Explain with Python programming example.	L2	04
		Module-3		
Q. 05	a	Explain Python string handling methods with examples: split(),endswith(), ljust(), center(), lstrip()	L2	10
	b	Explain reading and saving python program variables using shelve module with suitable Python program.	L2	06
	c	Develop a Python program to read and print the contents of a text file.	L3	04
		OR		
Q. 06	a	Explain Python string handling methods with examples: join(), startswith(),rjust(),strip(),rstrip()	L2	10
	b	Explain with suitable Python program segments: (i) os.path.basename() (ii) os.path.join().	L2	05
	c	Develop a Python program find the total size of all the files in the given	L3	05

		directory.		
		Module-4		
Q. 07	a	Explain permanent delete and safe delete with a suitable Python programming example to each.	L2	08
	b	Develop a program to backing Up a given Folder (Folder in a current working directory) into a ZIP File by using relevant modules and suitable methods.	L3	06
	c	Explain the role of Assertions in Python with a suitable program.	L2	06
		OR		
Q. 08	a	Explain the functions with examples: (i) shutil.copytree() (ii) shutil.move() (iii) shutil.rmtree().	L3	06
	b	Develop a Python program to traverse the current directory by listing sub-folders and files.	L2	06
	c	Explain the support for Logging with logging module in Python.	L2	08
		Module-5		
Q. 09	a	Explain the methods __init__ and __str__ with suitable code example to each.	L2	06
	b	Explain the program development concept 'prototype and patch' with suitable example.	L2	06
	c	Define a function which takes TWO objects representing complex numbers and returns new complex number with a addition of two complex numbers. Define a suitable class 'Complex' to represent the complex number. Develop a program to read N (N >=2) complex numbers and to compute the addition of N complex numbers.	L3	08
		OR		
Q. 10	a	Explain the following with syntax and suitable code snippet: i) Class definition ii) instantiation iii) passing an instance (or objects) as an argument iv) instances as return values.	L2	10
	b	Define pure function and modifier. Explain the role of pure functions and modifiers in application development with suitable python programs.	L2	10

*Bloom's Taxonomy Level: Indicate as L1, L2, L3, L4, etc. It is also desirable to indicate the COs and POs to be attained by every bit of questions.

1a. **Break - statement** :- It is used to terminate the loop immediately when it is encountered.

eg:-

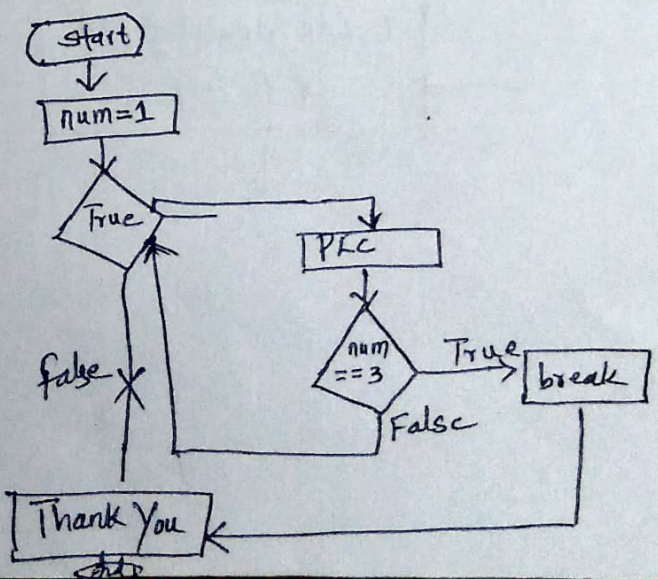
```
num=1
while True:
    print('PLC',num)
    if num==3:
        break
    num=num+1
print('Thank you')
```

output:

```
PLC 1
PLC 2
PLC 3.
Thank You.
```

1) The first line is initialization of variable num.
2) Second one creates an infinite loop whose condition is always true.
3) If we use break statement inside the loop. It will terminate the loop and exit from the loop.
4) In this example 'PLC' will print upto 3 times.
5) After printing 3rd time num ==3, condition becomes true. then break statement will terminate the loop.

* **Flow control**



* Continue Statement :-

→ It is used inside the loops.

→ It is used to skip the current iteration of the loop. and control flow of the program goes to the next iteration.

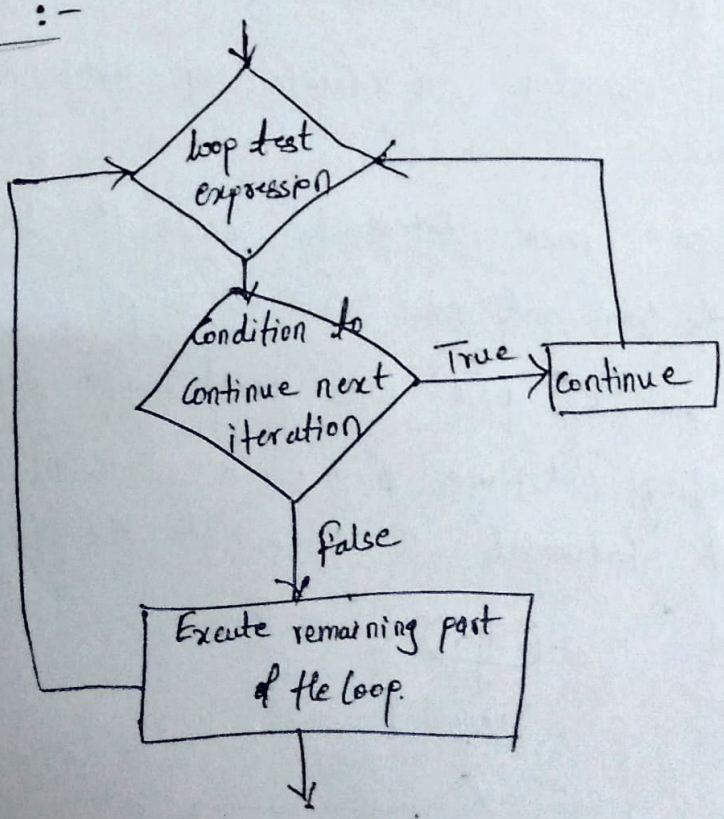
* example :-

```
num = 0
while num ≤ 5:
    num = num + 1
    if num == 3
        continue
    print('PLC', num)
```

output :

```
PLC 1
PLC 2
PLC 4
PLC 5
PLC 6
```

* Flow-Control :-



1b. Two ways of importing Modules :-

An import statement consists of the following.

-) The import keyword.
-) The name of the module.
-) Optionally more module names can be used separated by commas.

eg:- ①
```
import random
for i in range(5):
    pt = random.randint(1, 5)
print(pt)
```

o/p:-
```
4
1
3
1
2
```

eg:2:- import random, sys, os, math.

Here we are importing more modules seperated by commas.

* From import statements :-

It is composed of the from keyword, followed by module name, import keyword, and a star.

eg2:- from random import *.

1c Factorial and Binamial Coefficient program :-

```
import math.
def fact(n):
    if n==0:
        return 1
    else:
        return n * fact(n-1)
print('enter n value')
n= int(input())
print('enter r value')
r=(int(input())
res=fact(n)
res2= math.comb(n,r)
print('factorial is', res)
print('Binomial Coeff', res2)
```

2a) **For loop :-** It is used to run a block of code for certain number of times. It can be used to iterate over any sequences such as list, tuple, string.

eg:-
```
var = 'hi plc'
for pt in var:
    print(pt)
```

o/p:-
```
h
i
p
l
c
```

Here for loop contains :-

1) for is a keyword
2) pt is a variable name
3) in is a keyword
4) var is a stored variablename
5) A colon.

* **While loop :-** It repeats a statement or group of statements while a given condition is True.

- It tests the condition before executing the loop body.
- It contains
  1) While keyword
  2) A codition
  3) A colon
  4) Block of code

eg:-
```
num = 1
while num <= 3:
    print('hi PLC')
    num = num + 1
```

o/p:-
```
hi plc
hi plc
hi plc
```

2b. Fibonacci sequence of length(n).

```
print('Enter the value of n')
n= int(input())
f1= 0
f2= 1
print('fib series are')
print(f1)
print(f2)
for x in range(2, n):
    f3= f1+f2
    print(f3)
    f1= f2
    f2= f3.
```

output :-

```
Enter the value of n
5
fib series are
0
1
1
2
3.
```

2c.

```
def divexp(a, b):
    try:
        c=a/b
        return c
    except ZeroDivisionError:
        print('Invalid argument s')

print('enter 1st number')
a= int(input())
print('enter 2nd number')
b= int(input())
res= divexp(a, b)
print('Division of two number is', res)
```

2d. Four Rules of Variable

1) It can be only one word.
2) It can be only letters, numbers, & under score
3) It can't begin with number. 4) Variables are case-sensitive

3a. **get() Method** :- It returns the value of the item with specified key.

→ It takes two arguments

1) one is key.
2) fall back value to return. If that key does not exist.

eg:- marks={'phy': 77, 'maths': 78}

marks.get('phy')

77

* **setdefault()** :- It takes two arguments

1) 1st Argument passed to the method is key. to check for
2) the value to set at that key.

eg:- >>> marks = {'phy': 77, 'maths': 78}

>>> marks.setdefault('Eng': 95)

>>> marks

{'Eng': 95, 'phy': 77, 'maths': 98}.

3b) The copy.copy() returns a shallow copy of the list. and deepcopy() returns a deep copy of the list. Both have same value but have different IDs.

```
import copy.
list1 = [1, 2, [3, 5], 4]
list2 = copy.copy(list1)
print ('list2 ID' id(list2), .list2)

list3 = copy.deepcopy(list1)
print ('list3 ID', id(list3), list3)
```

o/p:- list2 ID 123455 [1, 2, [3, 5], 4]
list3 ID 1354322 [1, 2, [3,5], 4].

3c) append():- It appends an element to the end of the list. It can be called on list values not other values.

eg:-
```
>>> fruits = ['apple', 'banana', 'cherry']
>>> fruits.append('orange')
>>> fruits
['apple 'banana', 'cherry', 'orange']
```

index():- It returns the position at the first occurrence of the specified value. If the value is not in the list. then it produces value error. error.

```
>>> fruits = ['apple', 'mango', 'cherry']
>>> fruits.index('mango')
1
```

4a) Different ways to delete an element from a list:-

1) remove(): It removes the first occurrence of the element with specified value.

eg:- >>> animal=['cat', 'bat', 'rat', 'elephant']
>>> animal.remove('bat')
>>> animal
['cat', 'rat', 'elephant']

→ Attempting to delete a value that doesnot exist in the list it will result an error.

→ If the value appears multiple times in the list, only 1st instance of the value will be removed.

2) Using del statement :- The del statement will delete values at an index in a list.

→ All of the values in the list after deleted will be moved up one index.

eg:- >>> animal=['cat', 'rat', 'bat']
>>> del animal[1]
>>> animal
['cat', 'bat']

4b

```
import pprint
print('enter multidigit number')
msg = input()
dict = {}
for char in msg:
    dict.setdefault(char, 0)
    dict[char] = dict[char] + 1
pprint.pprint(dict)
```

output:-

```
enter a number
122344
{'1': 1,
 '2': 2,
 '3': 1,
 '4': 2}
```

4c) Tuples are immutable :-

. Tuples are immotable, These are cannot be changed.

eg:- creating a tuple using tuple name and round brackets ()

```
>>> sem1=('plc', 'cpgm', 'maths')
>>> sem1
('plc', 'cpgm', 'maths')
>>> sem1[0] = 'phy'
TypeError:- tuple object doesnot support item assignment
```

→ Tuples cannot have their values modified, appended, or removed.

5a) split() :- It splits a string at the specified separator and returns a list of substrings.

eg:- 'first hi second hi third'.split('hi')

['first', 'second', 'third']

endswith() :- It returns true if the string value they are called on ends with the string passed to the method. otherwise they return false.

eg:-
```
>>> 'hiplc'.endswith('plc')
True
>>> hiplc'.endswith('hi')
False
```

ljust() :- This method will left align the string using specified character as the fill character.

→ 1st argument is integer length for justified string.
→ 2nd argument is fill character other than space.

eg:- >>> 'hi'.ljust(10, '*')

'Hi********'

center() :- It centers the text rather than justifying it to the left or right.

>>> 'hi'.center(10, '*')

'****hi****'

lstrip() :- It removes the left side white spaces in the given string.

eg:- >>> ' hi '.lstrip()

'hi '

5b) We can save variables in our python program to binary shelf files using shelve module.

3 new files in current working directory

1) filename.bak
2) filename.dir
3) filename.dat

To read and write data using shelve module.

1) first import the shelve module
2) call shelve.open() and pass filename
3) Create variable name
4) store variable name into file object.
5) Then close() the file

eg:- import shelve
file = shelve.open('anyfile')
animal = ['cat', 'bat', 'rat']
file['view'] = animal
file.close()

* Reopen and retrieve the data from shelf files.

file = shelve.open('anyfile')
file['view']
['cat', 'bat', 'rat']

5c) File reading and writing Process

```
file = open('enc.txt','w')
file.write('hi pdc')
13
file.close()
file = open('enc.txt')
con = file.read()
file.close()
```

filewriting

file reading process

```
print(con)
hi pdc.
```

6a) **join()** :- It is used to join a list of strings together into a single string value.

eg:- >>> 'hi'.join(['first','second','third'])

first hi second hi third.

startswith() :- It returns true if the string value they are called on begins with specified string to the method.

eg:- >>> 'hello world'.startswith('hello')

True.

>>> 'hello world'.startswith('hi')

False.

rjust() :- It will right align the string, using a specified character as the fill characters.

eg:- >>> 'hi'.rjust(10,'*')

'********hi'

It will accepts two arguments

1) Integer length
2) Character fill.

* strip() :- It removes the both left and right side white spaces. in the given string

eg:- >>> ' hi '.strip()

'hi'

* rstrip() :- It removes the right side white space in the given string.

eg:- >>> ' hi '.rstrip()

' hi'

6b) os.path.basename(path) :- It will return a string of everything that comes after the last slash in the path argument.

eg
```
>>> path = 'C:\\encl\divAll\plc.exe'
>>> os.path.basename(path)
'plc.exe'
```

os.path.join() :- It will return a string with a file path using the correct path separators.

```
>>> import os
>>> os.path.join('enc', 'boys', 'girls')
'enc\\boys\\girls'
```

6c Finding file sizes and Folder Contents :-

→ **os.path.getsize(path)** :- It will return the size in bytes of the file in path argument.

→ **os.listdir(path)** :- It will return a list of filename strings for each file in the path argument.

To find the total size of all files in directory use both the methods.

```
>>> import os
>>> total=0
>>> for filename in os.listdir('c:\\encl\\diva'):
        total = total + os.path.getsize(os.path.join('c:\\encl\\diva', filename))
>>> print(total)
1117846456.
```

7a * Permanently Deleting files & folder :-

1) os.unlink(path) :- It will delete a file at the given path.

eg:-
```
import os
os.unlink('xyz.txt')
```

2) os.rmdir(path) :- It will delete the empty folder only.

eg:-
```
import os
os.rmdir('ENC')
```

3) shutil.rmtree(path) :- It will delete the folder, subfolder files in the given directory

eg:-
```
import shutil
shutil.rmtree('ENC')
```

7a) * Safe delete :- Safe deletes with send2trash Module

→ It is a third party module
→ We can install this by running pip install send2trash from command prompt window.

eg:-
```
import send2trash
send2trash.send2trash('abc.txt')
```

7b) step1:- Figure out the zip file's name
step2:- Create the new zip file
step3:- Walk the directory Tree & add to the zip file.

```
import os, zipfile
file = zipfile.ZipFile('myzipfile', 'w')
print ('Creating the zip file')
for foldername, subdir, files in os.walk('enc') :
    print ('Adding files in', foldername)
    file.write(foldername)
    for filename in files:
        file.write(os.path.join(foldername, filename))

file.close()
print('Backup folder is created')
```

o/p:-

```
Creating the zip file
Adding files in enc
adding files in enc\diva.
Backup folder is created.
```

7c) Assertions :-

→ It is a debugging tool, and its primary task is to check the condition.

→ If it finds that condition is true, it moves to the next line of code, and

→ If not, then stops all its operations and throws an error.

* An assert statement consists of the following.

-) The assert keyword
-) A condition (expression that evaluates to True or false)
-) Comma
-) A string to display when condition is false.

eg:-

```
x = 'goodboy'
assert x=='badboy' , 'x should be goodboy only'
```

Keyword — Condition — Comma — Error Message.

* Disabling Assertions :- It can be disabled by passing the -O option when running Python.

8a) 1) shutil.copytree() :- It will copy an entire folder and subfolder, files contained in it.

→ The source and destination parameters are both strings

eg:-

```
>>> import shutil
>>> shutil.copytree('ENC', 'EEE')
'EEE
```

8a). ii) shutil.move() :- It is used to move and Rename files and folders.

⇒ It will move the file or folder from source to the destination path.

eg:- >>> import shutil
>>> shutil.move('a.txt', 'Enc')
'Enc\\a.txt'

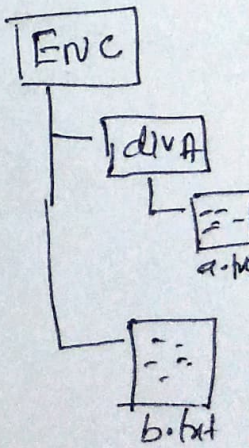
here the file a.txt is moved to the Enc folder. and returns the path.

eg2 :- >>> import shutil
>>> shutil.move('a.txt', 'b.txt')
'b.txt'

here a.txt file is moved and renamed to b.txt file.

iii) shutil.rmtree() :- It deletes all the files, subfolders in the given folder.

eg:- >>> import shutil
>>> shutil.rmtree('Enc')



Here it deletes all the files and sub folders in the given Enc folder.

8b) Traverse the current directory by listing sub-folders and files using os.walk() method.

→ It returns 3 values

1) A string of current folder name
2) A string of sub folder names
3) List of files in the current folder.

* program :-

```
import os
for foldername, subfolder, filenames in os.walk('xy2'):
    print('current folder is', foldername)
    for subfolders in subfolder
        print('subfolder is', subfolders)
    for filenam in filenames:
        print('filename is', filenam)
```

o/p:-

```
The current folder is xy2
subfolder is  xy2\pqr
filename is  xy2\pqr\a.txt.
```

30) **Logging:-** It is the process of writing information into log files.

→ Log files contain information about the events happened in operating system, software.

→ Logging is done for the following purposes

1) information gathering
2) Troubleshooting
3) Generating statistics

Using logging Module:- to enable the logging module to display log messages on our screen.

```
import logging
logging.basicConfig(level=loging.DEBUG, format='%(asctime)s -
                    %(levelname)s - %(message)s')
logging.debug('start of program')
def fact(n):
    total=1
    for i in range(1, n+1):
        total = total * i
        logging.debug('i is ' + str(i) + ' total is' + str(total))
    logging.debug('End of factorial')
print(fact(5))
logging.debug('End of program')
```

Output:

```
2023-03-13 17:13:40,650 - DEBUG - start of program
2023-03-13 17:13:40,651 - DEBUG - i is 1, total is 1
2023-03-13 17:13:40,651 - DEBUG - i is 2 total is 2
2023-03-13 17:13:40,652 - DEBUG - i is 3 total is 6
2023-03-13 17:13:40,653 - DEBUG - i is 4 total is 24
2023-03-13 17:13:40,654 - DEBUG - i is 5 total is 120
120.
```

9a) __init__ Method :-

- It is a special method that gets invoked when an object is instantiated.
- → Its full name is __init__ (two underscore character followed by init. and then two underscores)
- → Parameters have same name as the attributes.
- → Parameters are optional, so if we call method with no arguments we get default values.
- → If we provide one argument it overrides one argument and so on.

eg

```
class Time:
    """ Represents time in hr, min, sec"""
    def __init__(self, hr=0, min=0, sec=0):
        self.hr = hr
        self.min = min
        self.sec = sec
    def print_time(time)
        print(time.hr, time.min, time.sec)
```

```
time = Time()
time.print_time()
00:00:00
```

```
time = Time(9)
time.print_time()
09:00:00
```

9a). __str__ method

:→ It returns the string represention of the object.

→ It is called when print() or str() is invoked on an object.

eg:-

```
# inside class Time:
    def __str__(self):
        return '%.2d:%2d:.%2d' %(self.hr, self.min, self.sec)
```

→ When we print an object, python invokes the str method

```
>>> time = Time(9, 45)
>>> print (time)
09:45:00
```

9b) Prototype and Patch :-

The use of functions demonstrate an application development plan called "prototype and patch".

→ To illustrate we will define a class called Time that records the time of day.

```
class Time:
```

→ An alternative is designed development, in which high-level insight into the problem can make the programming much easier.

→ When we wrote add-time and increment, we were effectively doing addition in base 60, which is why we had to carry from one column to the next.

* The following function that converts Time to integers

```
def time_to_int(time):
    min = time.hour * 60 + time.min
    sec = min * 60 + time.sec
    return sec
```

* Another function that converts integer to Time.

```
def int_to_time(sec):
    time = Time()
    min, time.sec = divmod(sec, 60)
    time.hour, time.min = divmod(min, 60)
    return time
```

* Once the function is found correct, we can use them to rewrite add-time:

```
def add_time(t1, t2):
    sec = time_to_int(t1) + time_to_int(t2)
    return int_to_time(sec)
```

9c) Addition of two complex numbers :—

```
Class complexx:
    def add(self, a, b):
        return a+b
print('Enter 1st complex number')
num1 = complex(input())
print('Enter 2nd complex number')
num2 = complex(input())
obj = complexx()
res = obj.add(num1, num2)
print('Result is', res)
```

output :—

```
Enter 1st complex number
2+3j
Enter 2nd complex number
2+3j
Result is (4+6j)
```

10a). i) **class definition :-**

A programmer-defined type is called a class.

→ The class is a keyword.
→ class name
→ colon.

eg:- class enc :
"" Represents enc branch""

ii) **Instantiation :-** Creating a new object is called instantiation, and the object is an instance of the class.

eg:- class plc:

p1 = plc()

here p1 is an object of plc class.

iii) **Passing an Instance :-** We can pass an instance as an argument in the usual way.

eg:- def print-point(p):
print(p.x, p.y)

Here print-point takes a point as an argument and displays it in mathematical notation.
To invoke it, we can pass blank as an argument.

print-point(blank)
(3.0, 4.0)

Inside the function, p is an alias for blank, so if the function modifies p, blank changes.

The function creates a new Time object, initializes its attributes, and returns a reference to the new object.

```
>>> start = Time()
    start.hour = 9
    start.min = 45
    start.sec = 0

    duration = Time()
    duration.hour = 1
    duration.min = 35
    duration.sec = 0
    done = add_time(start, duration)
    print_time(done)
    10:80:00
```

Here the problem is that this function does not deal with cases where the number of seconds or minutes adds up to more than sixty.

Improved version.

```
if sum.sec >= 60:
    sum.sec -= 60
    sum.min += 1

if sum.min >= 60:
    sum.min -= 60
    sum.hour += 1

return sum.
```

Modifiers :- A function to modify the objects it gets as parameters. In that case, the changes are visible to the caller, These functions are called modifiers.

Increment() function adds a given number of seconds to a Time object which is visible to the called function.

eg
```
def increment(time, sec):
    time.sec t= seconds
    if time.seconds >= 60:
        time.second -= 60
        time.min += 1
    if time.min >= 60:
        time.min -= 60
        time.hr += 1
```

HOD
Computer Science & Engineering
KLS Vishwanathrao Deshpande
Institute of Technology, Haliyal

Dean, Academics
KLS VDIT, HALIYAL

Girish.N.S