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.

Module Question Paper - I

1a. **break** statement is used to terminate the loop immediately when it is encountered.

Used inside **for loop** or **while loop**

Syntax

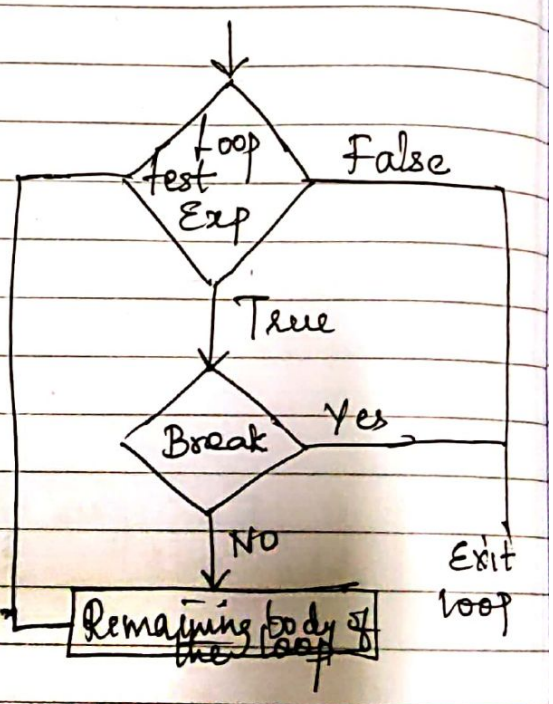
```
break.
```

Example program.

```
# using break to come out of while loop
x = 10
while x >= 1:
    print ('x = ', x)
    x -= 1
    if x == 5:
        break.
print ("out of loop")
```

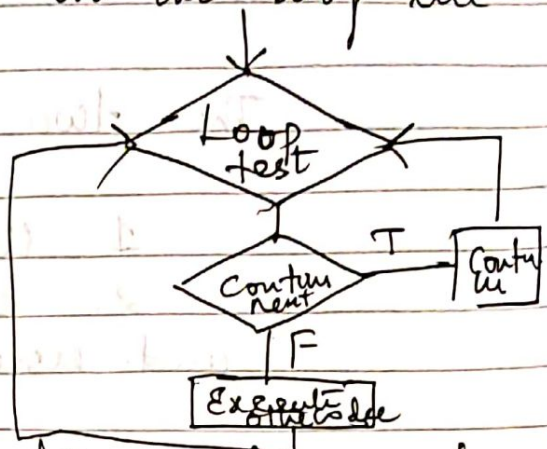
Output

```
x = 10
x = 9
x = 8
x = 7
x = 6
out of loop.
```



Continue : This statement is used to go back to the beginning of the loop.

When continue is executed, the next repetition will start and the subsequent statements in the loop are not executed



Syntax

continue

Example.

#using continue to execute next iteration of while loop

```
x=10
while x<10:
    x+=1
    if x>5:
        continue
    print('x=', x)
print("out of loop")
```

o/p

```
C:\> Python Demo.py
x=0
x=2
x=3
x=4
x=5
Out of loop
```

16. A module is a file with Python code. The code can be in the form of variables, functions or class defined.

File name can become module name.

The two way of importing module.

1. Using import keyword.
2. creating file in .py notation and using fctn.

Using import keyword

Example import os.

```
r1 = os.path("C:/document/python.txt")
open(r1)
```

Using .py codes and calling fctn.

Example

Step 1 test.py

Step 2

```
def display_msg():
    return "Welcome Python"
```

Step 3 : display.py

Step 4 : import test.py as import test

Step 5 : print (test.display_msg())

Step 6 : o/p execution

display.py

Welcome python.

1c. Python program to find factorial of a number.

working

$n! = n*(n-1)*(n-2)*\ldots 1$

```
num = int (input ("Enter a number : "))
factorial = 1
if num < 0:
    print ("Factorial does not exist for negative numbers")
elif num == 0:
    print ("The factorial of 0 is 1")
else
    for i in range (1, num+1):
        factorial = factorial * i
print ("The factorial of", num, "is" factorial)
```

Output : enter the number
4
The factorial of 4 is 24.

Binomial Coefficient

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

```
def binomialCoeff(n, k):
    if k > n:
        return 0
    if k == 0 or k == n:
        return 1
    return binomialCoeff(n-1, k-1) + binomialCoeff(n-1, k)

n = 5
k = 2
print("Value of C(%d, %d) is (%d)" % (n, k, binomialCoeff(n, k)))
```

O/P

Value of C(5,2) is 10

2a. ~~While to~~

2a. The loop control statement are as follows.

1. If loop.
2. while loop
3. for loop.
4. break and continue

If loop

```
If conditions:
    Statements
```

Example

```
num=1
if num == 1:
    print("one")
```

If ... else

```
if cond:
    Statment 1
else:
    Statement 2
```

Example

```
x = 10
if x%2 == 0:
    print("Even")
else:
    print("odd")
```

If...elif...else statement

```
if condition 1
    statement 1
elif condition 2
    statement 2
elif condition 3
    statement 3
else
    statement 4
```

example

```
num = -5
if num == 0:
    print(num, "is Zero")
elif num > 0:
    print(num, "is positive")
else:
    print(num, "is negative")
```

output

```
python
C:\>"Demo.py
-5 is negative
```

While loop

This is need to run set of code to run only once from top to bottom

2b Lab program, refer Manual.

2c Lab program, refer Manual.

2d Variable scopes are as follows.

1. Variable should not be Keyword.
2. Variable should start with alphabet or underscore
   eg A, a, A_1, _a1
3. Vareable should not start with numbers but can end with numbers.
   1a = This is not possible
   a1 = This is possible
4. Apart from underscore (_) no other special characters are allowed
   #a = This is not allowed.
   a_1 = This is allowed.

3a. get(): In python Dictionary get() Method return the value for the given key, If present in the dictionary. If not then it will return None.

Syntax
Dict.get(key)

Return the value of the item with

the specified key or the default value

eg.

```
d = {'coding': 'Python', 'think': 'better'}
print(d.get('coding'))
```

op

Python

If the particular key is not found, then compiler throw an error called KeyError.

```
print(d.get('sub'))
```

KeyError: Key not found.

setdefault()

The setdefault() method returns the value of a key if the key present in the dictionary

Syntax: setdefault()

```
dict.setdefault(key, default value)
```

↓ This is optional

if not specified, value will be none.

If key is found, value will be the output.

```
d = {'country' : 'India', 'sports' : 'cricket'}
print(d.setdefault('country'))
```

o/p

India

```
d.setdefault('team', 'RCB')
print(d)
```

o/p

```
d={'country': 'India', 'team': 'RCB', 'sports': 'cricket'}
```

In the above code key and value both are mentioned.

If the value is not mentioned, then the value it be None

```
d.setdefault('player')
print(d)
```

o/p

```
d={'country': 'India', 'team': 'RCB', 'sports': 'cricket',
   'player': None}
```

In the above code, key and value are arranged randomly.

3b. copy.copy()

We can use copy module of python for shallow and deep copy operation.

```
import copy
copy.copy(x)
copy.deepcopy(x)
```

copy.copy(x) This is shallow copy, this creates a new object which stores the reference of the original elements.

This doesnot create a copy of nested objects. instead it just copies the reference of nested objects.

This means, a copy process does not recurse or create copies of nested list.

Example

```
import copy
old_list = [[1,2,3], [4,5,6], [7,8,9]]
new_list = copy.copy(old_list)

print("old list :", old_list)
print("new list :", new_list)
```

o/p

old_list : [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
New_list : [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

Suppose we need to add [4, 4, 4] to old_list, then

old_list.append([4, 4, 4])

when we run the program again we get

old_list [[1, 2, 3], [4, 5, 6], [7, 8, 9] [4, 4, 4]]
new_list [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

This above technique is called copy.copy() ie. shallow copy

old_list : [[2, 2, 2] [3, 3, 3], [5, 5, 5]]
new_list = copy.copy(old_list)

old_list[1][1] = 'AP'

print ("old list :", old_list)
print ("New list :", New_list).

o/p: old_list : [[2, 2, 2] [3, AP, 3] [5, 5, 5]]
new_list : [[2, 2, 2], [3, AP, 3], [5, 5, 5]]

In the above code, both old list and new list shares the same reference number as index [1] [1], so both the list values will be changed.

Deep Copy

A deep copy creates a new object and recursively adds the copies of nested objects present in the original elements.

The deep copy creates independent copy of original copy ~~object~~ list and all its nested list

```
import copy
old_list = [[1,1,1], [2,2,2], [3,3,3]]
new_list = copy.deepcopy(old_list)

print("old list:", old_list)
print("new list:", new_list)
```

o/p

```
old_list: [[1,1,1], [2,2,2], [3,3,3]]
New list: [[1,1,1], [2,2,2], [3,3,3]]
```

In the above code, deepcopy() function to create copy which look similar

If any changes done in oldlist it leads to no change in the new list

```
old_list[1][0] = "R.P"

print ("old list :", old_list)
print ("new list :", new_list)
```

Op

```
old_list : [[1, 1, 1], ['RP', 2, 2], [3, 3, 3]]
new list : [[1, 1, 1], [2, 2, 2], [3, 3, 3]]
```

3c. In python append() is a method used in list to add element, append() method adds the element at the end of the list. i.e, index will be incrimint by 1.

Syntax list.append(item)

This method does not return any value.

Eg cricketPlayer = ['Sachin', 'Virat', 'Rahul']

```
NewPlayer = cricketPlayer.append('Dhoni')
print ("Newplayer")
print
```

print("old Player:", CricketPlayer)
print("New Player:", NewPlayer).

o/p

Old Player = ['sachin', 'Virat', 'Rahul']
New Player = ['Sachin', 'Virat', 'Rahul', 'Dhoni']

→ Index() method is used in list to return the index of the specified element in the list

Syntax index('element_name')

CricketPlayer = ['Sachin', 'Virat', 'Raina', 'Rahul']

index = cricket.index('Raina')
print(Index).

o/p = 2

The index is used for searching element, the list also

in list.index(strat,

Syntax:

list.index(element, start, end)

list index() parameters

dment: the element to be searched

start : Start searching from this index
end : Search the element up to this index

→ start and end are optional

→ If the indexed element is not found then it raise ValueError exception is raised

If same type of element found in the list then index() return the first index value.

Eg Animal = ['cat', 'rat', 'bat', 'cat']
index = Animal.index('cat')

O/P

0

alphabets = ['a', 'e', 'i', 'o', 'u', 'g', 'k', 'i', 'u']
index = alphabets.index('e') → 1

index = alphabets.index('i', 4) → 7

index = alphabets.index('i', 3, 7) → 7

O/P 1
7
7

Ans. In Python we can remove the element from the list using two ways.

1. remove()
2. del key word

1. using remove()

We can remove an item from the list by passing the value of the item to be deled as the parameter to remove() function

```
list = ['India', 'Romania', 'Rome', 'Spain']

list.remove('Rome')
print("After deleting the item :", list)
```

O/P

['India', 'Romania', 'Spain']

2. Using del

The python del statement is not function of list. items of the list can be deleted using the del statement by specifying the index of of item (element) to be deleted

```
fruit = ['Apple', 'Banana', 'Grapes', 'Watermelon']
print('Original list :', fruit)
del fruit[1]
print('After deleting the item :', fruit)
```

o/p

Original list :['Apple', 'Banana', 'Grapes', 'Watermelon']

After deleting :['Apple', 'Grapes', 'watermelon']

4b Lab Program, Ref Lab Manual

4c. A tuple is a sequence of value much like a list. The valued stored in a tuple can be any type, and they are indexed by integers. The important difference is that tuples are immutable.

Tuples are also comparable and hashable so we can lists of them and use tuples as key values in Python dictionaries

```
t = 'a', 'b', 'c', 'd', 'e'
```

Although it is not necessary, it is common to enclose Tuples in parantheses to help us.

quickly identify tuples when we look at python code:

```
t = ('a', 'b', 'c', 'd', 'e')
t[2] = f
print(t)
```

o/p

```
('a', 'b', 'f', 'd', 'e')
```

Module 3

5a. split()

```
'My country is India'.split()
['My', 'country', 'is', 'India']
```

This split the string into list each and every words will be converted into element of list.

```
'MyABCnameABCisABCFlowers'.split('ABC')
['My', 'name', 'is', 'Flowers']
```

endswith()

The endswith() methods return True if the string value they are called on begins or ends with the string passed to the method; otherwise, they return False.

```
'Hello world!'.endswith('world!')
True

'abc123'.endswith('12!')
False
```

ljust() → Left Justify

This method is used string to do left justify

Eg:

```
'Hello'.ljust(10)
                ↳ 10 space.
'Hello
'Hello'.ljust(5,'*') → 'Hello*****'
```

Center() → Center the string

This center() method is used in String

```
'Hello'.center(20)
        'Hello'
'Hello'.center(20,'=')
=======Hello========
```

lstrip() → This method will remove whitespace characters from the left ends.

```
Spam = '    HelloWorld  '
Spam.lstrip()
'HelloWorld  '
```

58. Using Shelve Module we can save variable in python program to binary shelf

Shelve module will add, save and open feature of the program.

To read data in the file we use Shelve module and use Shelve.open() method and pass it a filename

```
import shelve
shelfFish = shelve.open('mydata')
cats = ['Zophie', 'Pooka', 'Simon']
shelfFish['cats'] = cats

shelfFish.close()
```

Once this code is runned we get three current working directory mydata.bak, mydata.dat and mydata.dir

All this binary files store data in Shelf.

To save program we use pprint.pformat() function. This is by import module.
pprint

import pprint, this enable fun^n.

pprint.pprint() and pprint.pformat().

```
import pprint
cats = [{'name':'zo', 'dse':'chub'}, {'name':'poka',
                                      'desc':'hff'}]
pprint.pformat(cats)
"[{'desc':'chubb', 'name':'zo'}, {'desc':'hff',
                                  'name':'Poka'}]
```

5c. Ref Lab Manual.

6a join() — This is a string Method which inserted between each string of the list argument

Eg >>> ', '.join(['cats', 'rats', 'bats'])

'Cats', 'rats', 'bats'

```
>>> ' '.join(['My', 'name', 'is', 'Dravid'])
'My name is Dravid
>>> 'RM'.join(['My', 'name', 'is', 'Dravid'])
MyRMnameRMisRMDravid
```

This method gives output in the form of string

< Ref Notes >

Startswith(), rjust(), strip(), rstrip()

6b C:\Window\System32\Calc.exe

Dir Name | Base name

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

The basename follows the last slash in a path and is the same as the filename.

path = 'c:\\Windows\\System32\\calc.exe'
os.path.basename(path).

o/p

calc.exe

os.path.join() method in python join one or more path components intelligently.

This concatenates various path components with exactly one directory seperator ('/') following each non empty part except the last path component.

If the last path component to be joined is empty then directory seperator ('/') is put at the end.

```
import os
path = "/home"

print(os.path.join(path, "User/Desktop", "file.txt"))
```

o/p

"home/user/Desktop/file.txt"

6c. os.path Module provides fun^n for finding the size of the file

The size of the file will be in the form of bytes, this includes the files and foldres and folder inside the folder

* Don't Miss any points
* Notes given

Syntax

os.path.getsize()

os.path.getsize('C:\\Window\\System32\\Cale.exe')

O/P → 776192 → This is the byte size strats from C to cale.exe

program

```
totalSize = 0
for filename in os.listdir('C:\\Window\\System32'):
    totalSize = totalsize + os.path.getsize(os.path.join('C:\\window\\System32', fil
print(totalsize)
```

O/P ⇒ 1117846456

Module 4

7a. We can delete a single file or a single empty folder with functions in the os module, whereas to delete a folder and all of its contents, shutil module should be use

1) calling os.unlink(path) will delete the file at path

2) Calling os.rmdir(path) will delete the folder at path. This folder must be empty of any files or folders

3) Calling shutil.rmtree(path) will remove the folder at path, and all files and folders it contains will also be deleted

Program

```
import os
for filename in os.listdir():
if filename.endswith('.txt'):
    os.unlink(filename)
```

Safe delete - This is a type of deletion that allows data to be removed from a storage device

```
import os

file_name = "example.txt"

with open(file_name, "w") as file:
    file.write("This file has been deleted")

os.remove(file_name)
```

Eg:

```
import os
from zipfile import ZipFile
from os import path
from shutil import make_archive

if os.path.exists("a.txt"):

src = path.realpath("a.txt")

root_dir, tail = path.split(src)
shutil.make_archive("a_archive", "zip", root_dir)

with ZipFile("a_archive.zip", "w") as newzip:

    newzip.write("a.txt")
    newzip.write("a.txtbak")
```

7c. assert is a keyword, This is used statement takes as input a boolean condition.

This return the true and doesn't do anything and continues the normal flow of execution, but if it computed to be false, then it raises an. Assertion Error

Syntax : assert condition, error_message (optional)

where :

: assert → keyword
condition → boolean
Error_message : The optional argument to be printed in Console in case of AssertionError

Example : Python assert keyword with Error message.

```
a = 4
b = 0
print ("The value of a/b is :")
assert b != 0, "Zero Division Error"
print (a/b)
```

o/p

AssertionError : Zero Division Error.

8a) Shutil - This is python module elaborated as Shell utility that provides a file operations like copying, moving and deleting files and directions

1. Shutil.copytree() is used to copy the entire directory tree from the source directory to the destination directory

```
import shutil
shut.move('/path/to/source/file_or_directory',
'/path/to/destination/file_or_directory')
```

2. Shutil.move(src, dst, copy_fut = copy2): This function moves a file or directory from the source path to the destination path.

```
import shutil
shutil.move('/path/to/source/file_or_directory'
'/path/to/destination/file_or_directory')
```

3. Shutil.rmtree() is used to recursively delete the directory tree located at the specified path

```
import shutil
shutil.rmtree('/path/to/directory')
```

8b. import os

```
for foldername, subfolder, files in os.walk
                ('/home/secabiet/test'):
print ('Current folder is '+foldername)
for subf in subfolder:
    print ('Subfolder of :'+foldername+':'+subf)
for fname in files:
    print ('file inside of :'+foldername+':'+fname)
```

<u>o/p</u>

```
current folder is /home/secabiet/test
Subfolder of :/home/secabiet/test :test1
file inside of : /home/secabiet/test:a.py
current folder is /home/secabiet/test/test1
```

8c. Logging is a technique used in programming to record and report events that occur during the execution of a program.

This is used to track the behaviour of a program and helps to identifying issues and debugging problems

The logging module in Python provides a flexible and efficient way to handle logging in Python programs.

```
import logging
logging.debug('The debug message is displaying')
logging.info('The info message is displaying')
logging.warning('The warning message is displaying')
logging.error('The error message is displaying')
logging.critical('The critical message is displaying')
```

Output:

```
WARNING:root:The warning message is displaying
ERROR:root:The error message is displaying
CRITICAL:root:The critical message is displaying
```

9a. The init() method is used to initialize the object's attribute

```
class Person:
    def __init__(self, name, age)
    self.name = name
    self.age = age
```

The --str--() method is used to provide the data in string representation of the object

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def __str__(self):
        return f"Name: {self.name}, Age: {self.age}"

p = Person("Ashok", 37)
print(p)
```

O/P

Name: Ashok, Age: 37

9b. Ref Notes

9c. Lab Program Ref Manual

10a.

Class Definition : A class is a blueprint or a template for creating objects that encapsulate data and behaviours

In python, a class definition begins wi a class keyword followed by the nam of the class and a colon.

Syntax

```
class ClassName:
```

Instantiation : Instantiation is the proce of creating an object from a class.

To create an object, we use th name followed by parantheses.

Example

```
Ap = Apple()
```

Passing an instance (or objects) as an argument: We can pass instances of class (or objects) as arguments to function or methods.

iv) Instance as return values: We can also return instances of a class from fun^{n} or methods.

This allows the programmer to create and return objects based on some condition or data.

10b. A pure fun^{n} is a fution whose o/p value follows solely from its i/p values without any observable side effects

```
class Time :
    hr = 0
    min = 0
    sec = 0

def print_time(t):
print ('%.2d, %.2d, %.2d', %.(t.hr, t.min, t.sec))

def add_time (t1, t2)
    sum = Time()  → t1 = Time()
    t1.hr = 9
    t2.min = 45
    t3.sec = 20

t2 = Time()
    t2.hr = 1
    t2.min = 35
```

t2.sec = 0

t3 = Time()

t3 = Time_add_time(t1, t2)
Time.print_time(t3)

o/p
10:80:00 ⟶ o/p should be in form hr:min:sec

In the above code, the o/p is taken as mutable object arguments and this have to change during the execution of the program. This phenomenon is called Modifier

```
class Time:
    hr=0
    min=0
    sec=0
    def print_time(self):
    print('%.2d:%.2d:%.2d' %.(self.hr, self.
        min, self.sec))

    def increment(self):
        self.sec += self.sec
```

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

        if self.min >= 60:
            self.min -= 60
            self.hr += 1

time = Time()
time.hr = 9
time.min = 62
time.sec = 12
time.increment()
time.print_time()
```

O/P

10:02:24

2d: Global Scope
Local Scope
Enclosing Scope
Built in Scope.

Global Scope: Variables defined outside any function or class have a global scope.

These variable can be accessed or modified from anywhere in the program.

```
x = 300
def myfun():
    print(x)
    myfun()
    print(x)
```

o/p

300

300

local Scope: Variable defined inside a fun^{n} have a local scope.

Thire variable can only be accessed or modified within the fun^{n} in which they are defined. Local Variable take precedence over global variable with the same name.

o/p

```
x = 300
print(x)
myfun()
```

o/p

300

Enclosing Scope: Variable defined in an enclosing fun" hand an. enclosing scope.

These variables can be accessed or modified by nested fun", but not by fun" outside the enclosing fun".

Built in scope: this are the built in fun" auiable for the user.

Variable which are used in Built in fun" scope can be modified anywhere in the program.