CBCS SCHEME

USN | | | | | | | | | | | | BPLCK205B

Second Semester B.E./B.Tech. Degree Examination, Dec.2024/Jan.2025
Introduction to Python Programming

Time: 3 hrs. Max. Marks: 100

Note: 1. Answer any FIVE full questions, choosing ONE full question from each module.
2. M : Marks , L: Bloom's level , C: Course outcomes.

			M	L	C
		Module – 1			
Q.1	a.	Describe the following flow control statements in python programming: (i) if (ii) else (iii) while	8	L1	CO1
	b.	Write a function to calculate factorial of a number.	8	L3	CO1
	c.	Define def statements with parameters.	4	L1	CO1
		OR			
Q.2	a.	Explain the syntax and control flow diagrams for break and continue statement	8	L2	CO1
	b.	Illustrate the dissection of python program.	8	L2	CO1
	c.	Describe the return values and return statements.	4	L1	CO1
		Module – 2			
Q.3	a.	Read N numbers from the console and create a list. Develop a program to print, mean, variance and standard deviation with suitable messages.	8	L3	CO2
	b.	Summarize the sequence data types of python programming.	8	L2	CO2
	c.	Compare and contrast dictionaries Vs lists.	4	L2	CO2
		OR			
Q.4	a.	Develop a program to print 10 most frequently appearing words in a text file. [Hint : Use dictionary with distinct words and their frequency of occurrences. Sort the dictionary in reverse order of frequency and display dictionary slice of first 10 items].	8	L3	CO2
	b.	Explain the slots of tic – tac – toe board with its corresponding keys using data structures in python programming.	6	L2	CO2
	c.	Paraphrase the working with lists in python programming	6	L1	CO2
		Module – 3			
Q.5	a.	Describe the Python string handling methods with examples : Split ( ), endswith ( ), ljust ( ), center ( ), lstrip ( )	10	L2	CO3
	b.	Summarize the process of input validation in python programming.	10	L2	CO3

OR

Q.6	a.	Explain Python string handling methods with examples : join ( ), startswith ( ), rjust ( ), strip ( ), rstrip ( ).	10	L2	CO3
	b.	Demonstrate the process of copying and pasting strings with pyperclip module.	10	L2	CO3

Module – 4

Q.7	a.	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	10	L3	CO3
	b.	Describe the file reading or writing process in python programming.	10	L2	CO3

OR

Q.8	a.	Explain the process of compressing files with the zip file module.	10	L2	CO3
	b.	Summarize the organization of files using shutil module.	10	L3	CO3

Module – 5

Q.9	a.	Explain about class and objects.	10	L2	CO4
	b.	Explain purefunction and modifier.	10	L2	CO4

OR

Q.10	a.	Illustrate operator overloading and polymorphism in python with an example.	10	L2	CO4
	b.	Explain init and str method with examples.	10	L2	CO4

* * * * *

BPLCK205B

Introduction to Python Programming

Dec 2024/Jan. 2025

Time: 3hrs

Max. Marks - 100

Q1)

a. Describe the following flow Control Statements in Python Programming.

i) if ii) else iii) While.

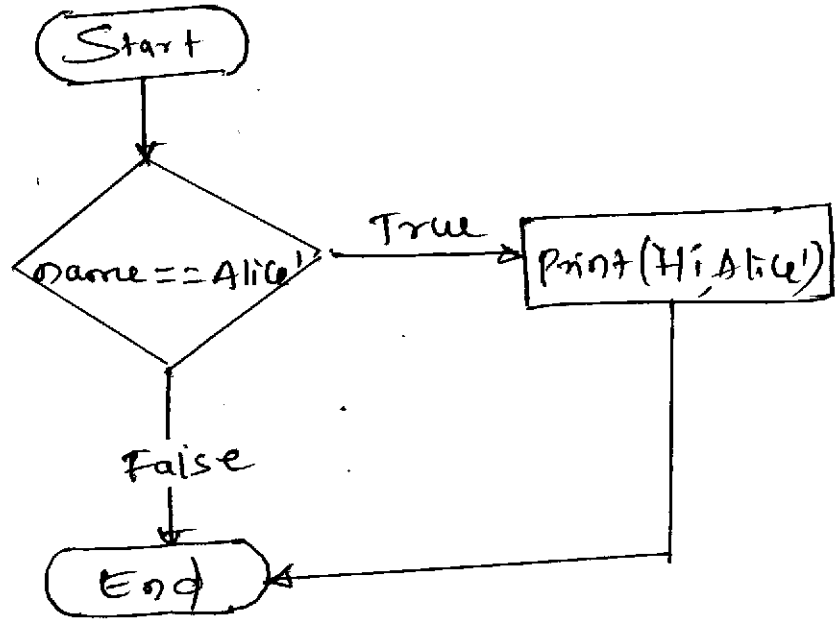
1) if.

* An if Statement's clause will execute if the Statement's Condition is True. The clause is skipped if the Condition is False.

* If Statement Consists of the following.
  - → if Keyword
  - → A Condition
  - → A Colon
  - → Next line start with indentation.

Ex:-

```
if name == 'Alice':
    Print('Hi, Alice')
```



ii) else

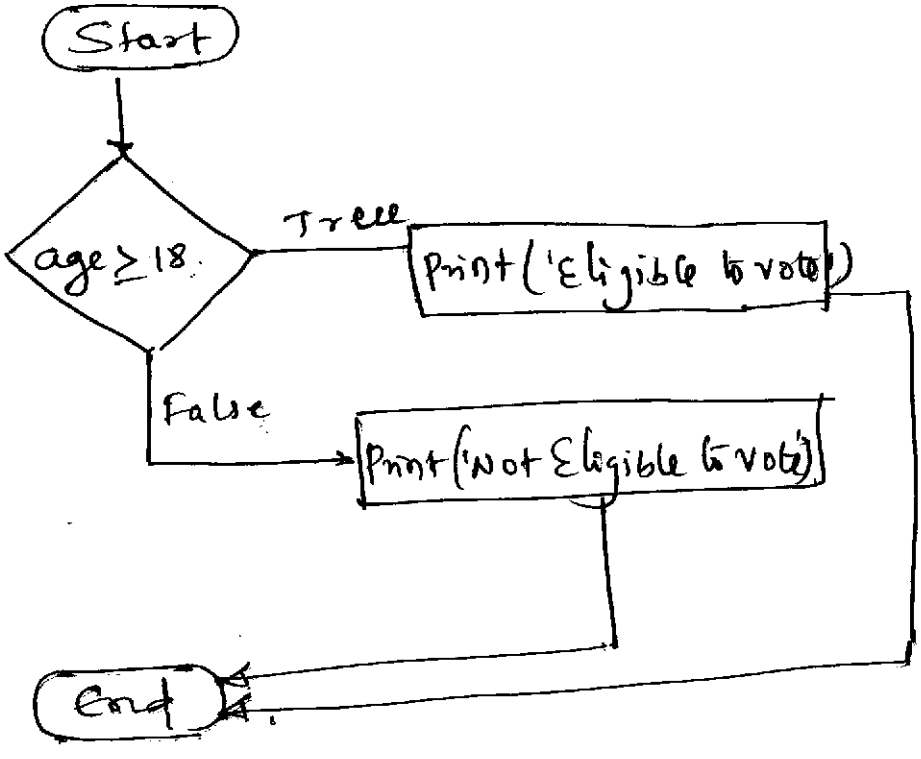
* An if clause optionally followed by an else statement. The else clause is executed only when the if statement's condition is false.

* else statement always consists of the following.
  - → The else keyword
  - → A colon
  - → Next line start with the indentation.

Ex:-

```
if age ≥ 18:
    Print ('Eligible to vote')
else:
    Print ('Not eligible to vote')
```

Flow chart



(ii) While

* We can make a block of code execute over and over again with a while statement.
* The code in a while clause will be executed as long as the while statements condition is true.
* while statement always consists of the following.

1. The while keyword
2. A condition
3. A colon.
4. Next line start with the indentation.

Ex:

```
Spam=0
while Spam<5:
    Print('Hello, World,')
    Spam=Spam+1
```

O/p

```
'Hello, World'
'Hello, World'
'Hello, World'
'Hello, World'
'Hello, World'
```

Q16) Write a function to calculate factorial of a number

```
import math
def factorial(n):
    if n==0:
        return 1
    else:
        return n * factorial(n-1)
Print('Enter a number to Compute the factorial')
n=int(input())
res=factorial(n)
Print('factorial is', res)
```

O/p

```
Enter a number to
Compute the factorial
4
factorial is 24
```

Q1) c) Define def statement with parameters

* When we call the print() or len() function we pass in values, called arguments.

Ex:-

```
def hello(name):
    Print('Hello', +name)
hello('Alice')
hello('Bob')
```

o/p

Hello Alice

Hello Bob.

* The definition of the hello() function in this program has a parameter called name.
* A parameter is a variable that an argument is stored in when a function is called.
* One special thing to note about parameters is that the value stored in a parameter is forgotten when the function returns.

Q2)

a) Explain the Syntax and Control flow diagrams for break and continue statement.

i) break Statement

* There is a shortcut to getting the program execution to break out of a while loop's clause early.
* If the execution reaches a break statement it immediately exist the while loop's clause.
* In code, a break statement simply contains the break keyword.

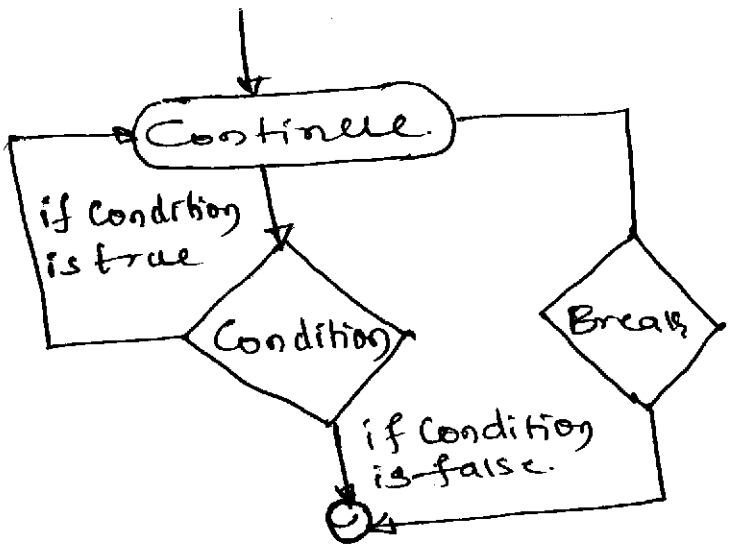
Syntax of break Statement

```
while (test_expression1):
    ---
    Statement(s)
    ---
    if (test_expression2):
        b.break.
```

true

out of the loop.

Flow chart



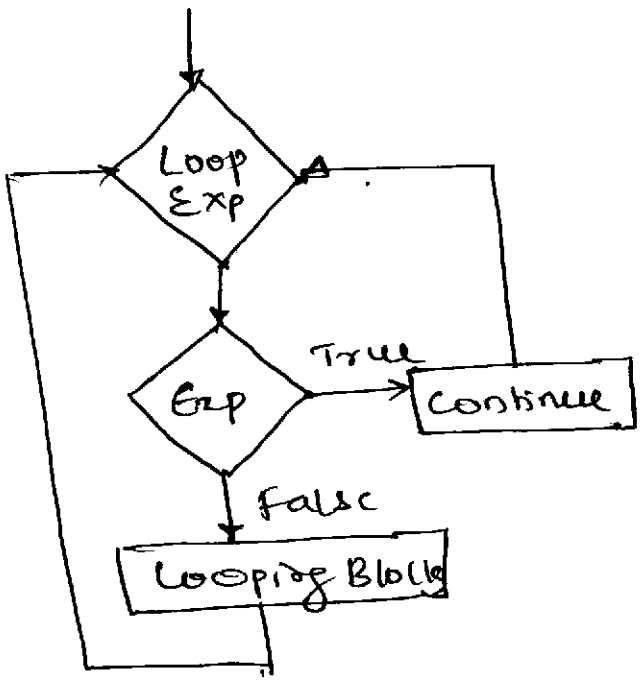
Continue Statement

* When the Program execution reaches a Continue statement, the program execution immediately jumps back to the start of the loop & recvaluates the loop's condition.

Syntax

```
while Condition:
    # code
    if Condition:
        Continue
    # Code.
```

flow chart



Q.2) b) Illustrate the dissection of python program.

Comments

* The following line is called a Comment.

This program says hello and ask for my name.

* Python ignores comments, and we can use them to write notes or remind ourselves what the code is trying to do.

The print() function

* The print() function displays the string value inside the parentheses on the screen.

Ex: print('Hello world!')
print('What is your name?') # ask for their name.

The Input() function

* The input() function waits for the user to type some text on the keyboard and press Enter.

myName = input()

The len() function

* We can pass the len() function a string value and the function evaluates to the integer value of the number of characters in that string.

Ex:- >> len('hello')
o/p → 5

The Str() function

The Str() function can be passed an integer value and will evaluate to a string value version of it.

Ex:- Str(29)
o/p → '29'

(Q2) c) Describe the return values and return statement

* The value that a function call evaluates to is called the return value of the function.

Ex:- len('Hello') → 5
Return value is 5

* When creating a function using def statement, we can specify what the return value should be with return statement

* A return statement consists of the following
  1) The return keyword
  2) value or expression that the function should return.

Ex:-
```
def Square(num)
    result = num * num
    return result
value = Square(4)
print("The Square is:", value)
```

o/p → The Square is: 16.

9) a) Read N number from the Console and create a list. Develop a program to print mean, variance and Standard deviation with suitable messages.

```
import Statistics
input_string = input('Enter elements of a list
                separated by Space')
user_list = input_string.Split()
Print ('list:', user_list)
for i in range (len(user_list)):
    user_list[i] = int(user_list[i])
Print("Sum =", Sum(user_list))
Print("mean:", (Statistics.mean(user_list)))
Print("variance:", (Statistics.variance(user_list)))
Print("Standard Deviation:", (statistics.stdev(user_list)))
```

o/p

Enter elements of a list Separated by Space
10 20 40 50
list: ['10', '20', '40', '50']

Sum = 120

mean: 30

Variance: 333.33

Standard Deviation: 18.257

(Q3) b) Summarize the Sequence data types of Python programming.

1) List

* List are mutable.
* List can store different data types.
* List are defined using Square brackets.
* We can perform common operations like. append, insert, remove, sort, slicing.

Ex:-

```
my_list = [1, "apple", "Hello", 4]
```

2) Tuple

* Tuples are immutable.
* These are used for fixed collections.
* Defined using Parentheses.
* Supports indexing and slicing but cannot modified.

Ex:-

```
my_tuple = (1, "banana", 3.2)
```

3) String

* String is a sequence of unicode characters.
* Strings are immutable.
* Defined using quotes.
* Supports indexing, slicing, and string methods like upper(), lower() .. etc.

Ex:-

```
my_string = "Hello"
```

Q3) c) Compare and Contrast dictionaries vs lists.

Lists	Dictionaries
1) Lists are ordered. Position of items	1) Dictionaries are unordered.
2) Indexed by it's position.	2) Indexed by it's key.
3) Best for collections of items.	3) Best for structured data with named fields.
4) Ex:- [90, 85, 70] (student marks)	4) {"name": "Alice", "age": "25}.

Q4) a) Develop a program to print 10 most frequently appearing words in a text file [Hint: Use dictionary with distinct words and their frequency of occurance Sort the the dictionary in reverse order of frequency & display dictionary slice of first 10 items]

```
file = open("gfg.txt", "r")
frequent_word = " "
frequenly = 0.
words = [ ]
for line in file:
    line_word = line.lower().replace(',', ' ').
        replace('.', ' ').split(" ")
    for w in line_word:
        words.append(w);
for i in range(0, len(words)):
    count = 1;
    for j in range(i+1, len(words)):
        if(words[i] == words[j]):
            count = count + 1;
```

~~frequent~~

```
        if (count > frequency):
            frequency = count;
            frequent_word = words[i];
    Print("Most repeated word: " + frequent_word)
    Print("frequency:" + str(frequency))
    file.close()
```

(Q4) b) Explain the slots of tic-tac-toe board with it's corresponding keys using data structure in python programming.

ATic-Tac-Toe Board

'top-L'	'top-M'	'top-R'
'mid-L'	'mid-M'	'mid-R'
'low-L'	'low-M'	'low-R'

A tic-tac-toe board looks like a large hash symbol (#) with a nine slots that can each contain an X, an O, or a blank. To represent the board with a dictionary, we can assign each slot a string value key as shown in fig.

```
theBoard = {'top-L': ' ', 'top-M': ' ', 'top-R': ' ',
            'mid-L': ' ', 'mid-M': ' ', 'mid-R': ' ',
            'low-L': ' ', 'low-M': ' ', 'low-R': ' '}
```

Program

```
def printBoard(board):
    print(board['top-L'] + '|' + board['top-M'] + '|' + board['top-L'])
    print('-+-+-')
    print(board['mid-L'] + '|' + board['mid-M'] + '|' + board['mid-L'])
    print('-+-+-')
    print(board['low-L'] + '|' + board['low-M'] + '|' + board['low-L'])
printBoard(theBoard)
```

Q4) c) Paraphrase the working with lists in Python Programming.

Changing values in a list with indexes.

We can use an index of a list to change the value at that index.

Ex:- >>> Spam = ['cat', 'bat', 'rat', 'elephant']

```
>>> Spam[1] = lion.
>>> Spam
['cat', 'lion', 'rat', 'elephant']
```

Removing values from lists with del Statements

The del statement will delete values at an index in a list

Ex:- >>> Spam = ['cat', 'bat', 'rat', 'elephant']

```
>>> del Spam[2]
>>> Spam
['Cat', 'bat', 'elephant']
```

Getting a list's length with len()

The len() function will return the number of values that are in a list value.

Ex:- >>> Spam = ['Cat', 'dog', 'rat']

```
>>> len(Spam)
3
```

Getting Individual values in a List with Indexes

Ex:- >>> Spam = ['cat', 'rat', 'bat', 'elephant']

```
>>> Spam[0]
'Cat'
```

Negative Index

```
>>> Spam[-1]
elephant.
```

Q5) a) Describe the python string handling methods with examples Split(), endswith(), ljust(), center(), lstrip()

Sol:) 1) Split() — The Split() method is called on string value and returns a list of strings.

Ex:- >>> 'My name is simon'. Split()

['My', 'name', 'is', 'simon']

2) endswith() — endswith() method return True if the string value they are called on ends with the string passed to the method otherwise they return false.

Ex:- 'abc123'. endswith('12')

False

'abc123'. endswith('123')

True.

3) ljust() — ljust() string method return a padded version of the string they are called on, with spaces inserted to justify the text.

Ex:- 'Hello'. ljust(10)

'Hello '

4) Center() — The Center() String method works like ljust() & rjust() but centers the text rather than justifying it to the left or right.

Ex:- 'Hello'. Center(10)

' Hello '

5) lstrip() → lstrip() methods will remove whitespace characters from the left ends.

Ex:- Spam = ' Hello World '

Spam.lstrip()

'HelloWorld'

Q5) b) Summarize the process of input validation in python programming.

Sol's Input validation is the process of ensuring that the user provides valid and expected data before it is processed. It helps prevent Errors, Crashes, and security issues.

Some of the input validations are listed below.

1) inputPassword() - Is like the built-in input(), but displays characters as the user types so that passwords or other sensitive information are not displayed on the screen.

2) inputFilepath() - Ensures the user enters a valid file path and filename, and can optionally check that a file with that name exists.

3) inputNum() - Ensures the user enters a number and returns an int or float, depending on if the number has a decimal point in it.

~~4) inputBoolY()~~

4) inputDatetime() - Ensures the user enters a date and time.

5) inputStr() - Is like the built-in input() function but has the general pyInput-plus features. You can also pass a custom validation function to it.

6) inputEmail() - Ensures the user enters a valid email address.

7) inputYesNO() - Ensures the user enters a "yes" or "no" response.

Q.6) a) Explain python string handling methods with examples
join(), startswith(), rjust(), strip(), rstrip()

Sol'n 1) join() — join() method is useful when we have a list of strings that need to be joined together into a single string value.

Ex:- ', '.join(['cats', 'rats', 'bats'])

'cats, rats, bats'

2) startswith() — The startswith() methods return True if the string value they are called on begins with the string passed to the method, otherwise they return False.

Ex:- 'Hello world!'.startswith('Hello')

True.

3) rjust() — rjust() method return a padded version of the string they are called on with spaces inserted to justify the text.

Ex:- 'Hello'.rjust(10)

' Hello'

4) strip() — strip() string method will return a new string without any whitespace characters at the begining or end.

Ex:- Spam = ' Hello world! '

Spam.strip()

o/p → 'Hello world!'

5) rstrip() — rstrip() methode will remove whitespace characters from the right ends.

Ex:- Spam = ' Hello world '

Spam.rstrip()

o/p → ' Hello world'

Q6) Demonstrate the process of copying and pasting
b) strings with pyperclip module.

Sol'n The pyperclip module has copy() and paste()
functions that can send text to and receive
text from your computer's clipboard.

```
>>> import pyperclip
>>> pyperclip.copy('Hello world!')
>>> pyperclip.paste()
'Hello world!'
```

If something outside of your program changes
the clipboard contents, the paste() function
will return it.

```
>>> pyperclip.paste()
'For Example, if I copied this sentence to the
clipboard and then called paste(),
it would look like this:'
```

Q7) a) 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.

```
from zipfile import ZipFile.
import os, zipfile
extension = input('Input file extension: ')
zippy = ZipFile ("Backup.zip", 'w')
for folder, subfolders, file in.
os.walk ("c:\\Users\\PyTest"):
    for subfolder in subfolders:
        Path = folder + subfolder
    for x in file:
        if x.endswith (extension):
            filepath = folder + "\\" + x
            Print (filepath)
            zippy.write (filepath, Compress_type
              = zipfile.ZIP_DEFLATED)
zippy.close()
```

O/p

Backup.zip file is created.

Q7)

b) Describe the file reading or writing process in Python programming.

1) Reading the Files

* If you want to read the entire contents of a file as a string value, use the file object's read() method

```
>>> helloContent = helloFile.read()
>>> helloContent
'Hello world!'
```

* Alternatively, we can use the readlines() method to get a list of string values from the file one string for each line of text

* Create a file named hello.txt in the same directory and write following lines in it.

```
Hello how are you,
I am studying 1st year BE.
Learning Python programming subject.
It is very easy to understand.
```

```
>>> helloFile = open('hello.txt')
    helloFile.readlines()
```

['Hello how are you,\n', 'I am studying 1st year BE,\n', 'Learning Python programming subject,\n', 'It is very easy to understand']

Writing file

→ Python allows you to write content to a file in a way similar to how the print() function 'writes' string to the screen.

→ Write mode will overwrite the existing file and start from scratch, just like when you overwrite a variables value with a new value.

→ Pass 'w' as the second argument to open() to open the file in write mode.

Ex:- baconFile = open('bacon.txt', 'w')

Q.8)

a) Explain the process of compressing files with the zipfile module.

Sol:) Step-by-Step process to compress files with zipfile.

1) Import the module

Import the zipfile module

import zipfile.

2) Open/Create ZIP File

Use zipfile.ZipFile() in write ('w') mode to create a new zipfile.

with zipfile.ZipFile ('Compressed.zip', 'w') as zipf:

3) Add files to the zip

Use .write() method to add files to the zip archive

zipf.write('file1.txt')

You can also specify the name inside the ZIP.

Zip.write('file1.txt', arcname = 'renamed_file.txt')

4) Close the ZIP file

If using with the file automatically closes otherwise, call zipf.close()

(Q 8) b) Summarize the organization of files using Shutil module.

1) Copying Files and Folders

The shutil module provides functions for copying files as well as entire folders.

Calling shutil.copy(source destination) will copy the file at the path source to the folder at the path destination

Ex:-

```
>>> import shutil, os
>>> os.chdir('c:\\')
>>> shutil.copy('c:\\spam.txt', 'c:\\delicious')
'c:\\delicious\\spam.txt'
```

2) Moving and Renaming Files and Folders

Calling shutil.move(source destination) will move the file or folder at the path source to the path destination and will return a string of the absolute path of the new location.

Ex:- >>> import shutil, os
>>> shutil.move('c:\\bacon.txt', 'c:\\eggs')
'c:\\eggs\\bacon.txt'

Permanently Deleting Files and Folders

→ You can delet a single file or single empty folder with functions fo the os module whereas to delet a folder and all its contents you use the shutil module

* Calling os.unlink(path)
  will delet the file at path.
* Calling os.rmdir(path)
  will delet the folder at path. This folder must be empty of any files or folders.
* Calling os.rmtree(path)
  ~~will delet the folder at path. This folder must be empty of any files or~~
  will remove the folder at path. and all files and folders it contains will also be deleted.

Qa)

a) Explain about class and objects.

Class → * It is a user defined data type which binds data and functions together in to single entity.

* class is just a prototype or logical entity or blueprint.
* which will not consume any memory.

Syntax

```
class ClassName:
    def __init__(self, attribute1, attribute2):
        self.attribute1 = attribute1
        self.attribute2 = attribute2

    def method(self):
        Print("This is a method")
```

Objects

An Object in python is an instance of a class. once a class is defined you create objects from it to actualy use the data and behavior defined in the class.

Example

```
class Car:
    def __init__(self, brand, Color):
        self.brand = brand
        self.Color = Color

    def start_engine(self):
        Print("Car engine started")

Car1 = Car("Toyota", "Red")
Car2 = Car("Honda", "Blue")

Print(Car1.brand)
```

o/p → Toyota

```
print(Car2.Color)
```

o/p → Blue.

(Q9) Explain purefunction and modifier.

b) *) Purefunction

→ Purefunction Creates anew Time object, intiliazes Its attributes and returns a reference to the new object.

→ This is called a pure function bez it does not modify any of the objects passed to it as arguments and it has no effect, like displaying a value or getting user input, other than returning a value

Ex:-

```
def add_time(t1, t2):
    sum = Time()
    sum.hour = t1.hour + t2.hour
    sum.minute = t1.minute + t2.minute
    sum.second = t1.second + t2.second
    return sum
```

→ To test this function, let us create two Time objects start time & duration time.

```
>>> start = Time()
>>> start.hour = 9
>>> start.minute = 45
>>> start.second = 0.

>>> duration = Time()
>>> duration.hour = 1
>>> duration.minute = 35
>>> duration.second = 0.
>>> done = add_time(start, duration)
>>> print_time(done)
```

10:80:00.

4) Modifiers

→ Sometimes it is useful for a function to modify the objects it gets as parameters.

→ In that case, the changes are visible to the caller, functions that work this way are called modifiers.

→ Increment, which adds a given number of seconds to a Time object, can be written naturally as a modifier.

```
def increment(time, seconds):
time.second += seconds
if time.second >= 60:
    time.second -= 60
    time.minute += 1
if time.minute >= 60:
    time.minute -= 60
    time.hour += 1
```

(Q10)

a) Illustrate Operator overloading and Polymorphism in python with an example.

Operator overloading

* By defining other special methods, you can specify the behavior of operators on programmer defined types.
* Ex:- if we define a method named __add__ for the Time class, you can use the + operator on Time objects.

Ex:-

```
def __add__(self, other):
    seconds = self.time_to_int() + other
                .time_to_int()
    return int_to_time(seconds)
```

```
>>> start = Time(9, 45)
>>> duration = Time(1, 35)
>>> print(start + duration)
```

11:20:00.

Polymorphism

Many of the function ~~also works~~ we wrote for strings also work for other sequence types for example, we used histogram to count the number of times each letter appears in a word.

```
def
histogram(s):
    d = dict()
    for c in s:
        if c not in d:
            d[c] = 1
        else:
            d[c] = d[c] + 1
    return d.
```

* This function also works for lists, tuples, and even dictionaries, as long as the elements of s are hashable, so they can be used as keys in d:

10) b) Explain __init__ and __str__ method with Example.

1) __init__ method

* The init method is a Special method gets invoked when an object is instantiated.
* An init method for the Time class might look like this.

```
def __init__(self, hour=0, minute=0, second=0)
    self.hour = hour
    self.minute = minute
    self.second = second
```

* the parameters are optional so if you call Time with no arguments you get default value.

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

* If we provide one argument

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

* Two argument

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

2) The __str__ method

* __str__ is a Special method like __init__ that is supposed to return a string representation of an object.
* Ex: str method for Time objects.

```
def __str__(self)
    return '%.2d:%.2d:%.2d' % (self.hour, self.minute, self.second)
```

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

[Prof. Santosh Saraswadi]