K. S. SCHOOL OF ENGINEERING & MANAGEMENT

15, Mallasandra, Off Kanakapura Road, Bengaluru-560109

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Introduction to Python Programming

Lab Manual

(Practical Component of IPCC)

Sub Code: BPLCK205B

SEM: II

Prepared by:-

Mrs. Bindu K P
Assistant Professor
Dept. of CSE,KSSEM

Ms. Punitha M R
Assistant Professor
Dept. of CSE,KSSEM

Mrs. Prasanna N
Assistant Professor
Dept. of CSE,KSSEM

Vision

To emerge as a pioneer in the field of Electronics and Communication Engineering, through excellence in technical education and research

Mission

The Department of Electronics and Communication Engineering shall

- Provide a transformative educational experience focusing on disciplinary knowledge, problem solving techniques and innovative projects.
- Excel in research and promote Industry-Academia interaction.
  Inculcate entrepreneurial traits in the student community, by fostering.
- Managerial and leadership qualities.

Contents

	Course Title:	Introduction to Python Programming		
	Course Code:	**BPLCK105B/205B**	CIE Marks	50
	Course Type (Theory/Practical /Integrated)	Integrated	SEE Marks	50
			Total Marks	100
	Teaching Hours/Week (L:T:P: S)	2:0:2:0	Exam Hours	03
	Total Hours of Pedagogy	40 hours	Credits	03

Course objectives

- Learn the syntax and semantics of the Python programming language.
- Illustrate the process of structuring the data using lists, tuples
- Appraise the need for working with various documents like Excel, PDF, Word and Others.
- Demonstrate the use of built-in functions to navigate the file system.
- Implement the Object Oriented Programming concepts in Python.

Teaching-Learning Process

These are sample Strategies, which teacher can use to accelerate the attainment of the various course outcomes and make Teaching –Learning more effective

1. Use https://pythontutor.com/visualize.html#mode=edit in order to visualize the python code
2. Demonstrate and visualize basic data types (list, tuple, dictionary).
3. Chalk and talk
4. online and videos

Module-1 (08 hrs.)

Python Basics: Entering Expressions into the Interactive Shell, The Integer, Floating-Point, and String Data Types, String Concatenation and Replication, Storing Values in Variables, Your First Program, Dissecting Your Program, **Flow control:** Boolean Values, Comparison Operators, Boolean Operators, Mixing Boolean and Comparison Operators, Elements of Flow Control, Program Execution, Flow Control Statements, Importing Modules, Ending a Program Early with sys.exit(), **Functions:** def Statements with Parameters, Return Values and return Statements, The None Value, Keyword Arguments and print(), Local and Global Scope, The global Statement, Exception Handling, A Short Program: Guess the Number

Textbook 1: Chapters 1 – 3

Module-2 (08 hrs.)

Lists: The List Data Type, Working with Lists, Augmented Assignment Operators, Methods, Example Program: Magic 8 Ball with a List, List-like Types: Strings and Tuples, References,

Dictionaries and Structuring Data: The Dictionary Data Type, Pretty Printing, Using Data Structures to Model Real-World Things,

Textbook 1: Chapters 4 – 5

Module-3 (08 hrs.)

Manipulating Strings: Working with Strings, Useful String Methods, Project: Password Locker, Project: Adding Bullets to Wiki Markup
Reading and Writing Files: Files and File Paths, The os.path Module, The File Reading/Writing Process, Saving Variables with the shelve Module, Saving Variables with the print. Format() Function, Project: Generating Random Quiz Files, Project: Multiclipboard,
Textbook 1: Chapters 6 , 8

Module-4 (08 hrs.)

Organizing Files: The shutil Module, Walking a Directory Tree, Compressing Files with the zip file Module, Project: Renaming Files with American-Style Dates to European-Style Dates, Project: Backing

	Up a Folder into a ZIP File, **Debugging: Raising Exceptions, Getting the Traceback as a String, Assertions, Logging, IDLE‟s Debugger.** **Textbook 1: Chapters 9-10**
	Module-5 (08 hrs.)
	Classes and objects: Programmer-defined types, Attributes, Rectangles, Instances as return values, Objects are mutable, Copying, **Classes and functions: Time, Pure functions, Modifiers, Prototyping versus planning,** **Classes and methods: Object-oriented features, Printing objects, Another example, A more complicated example,Theinit method, The str method, Operator overloading, Type-based dispatch, Polymorphism, Interface and implementation,** **Textbook 2: Chapters 15 – 17**

Course outcome (Course Skill Set)

At the end of the course the student will be able to:

CO1	Demonstrate proficiency in handling loops and creation of functions.
CO2	Identify the methods to create and manipulate lists, tuples and dictionaries.
CO3	Develop programs for string processing and file organization
CO4	Interpret the concepts of Object-Oriented Programming as used in Python.

Programming Exercises:

1. a. Develop a program to read the student details like Name, USN, and Marks in three subjects. Display the student details, total marks and percentage with suitable messages.

   b. Develop a program to read the name and year of birth of a person. Display whether the person is a senior citizen or not.

2. a. Develop a program to generate Fibonacci sequence of length (N). Read N from the console.

   b. Write a function to calculate factorial of a number. Develop a program to compute binomial coefficient (Given N and R).

3. Read N numbers from the console and create a list. Develop a program to print mean, variance and standard deviation with suitable messages.

4. Read a multi-digit number (as chars) from the console. Develop a program to print the frequency of each digit with suitable message.

5. 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 the reverse order of frequency and display dictionary slice of first 10 items]

6. Develop a program to sort the contents of a text file and write the sorted contents into a separate text file. [Hint: Use string methods strip(), Len(), list methods sort(), append(), and file methods open(), readlines(), and write()].

7. 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.

8. 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 suitable program which reads two values from the console and calls a function DivExp.

9. 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.

10. Develop a program that uses class Student which prompts the user to enter marks in three subjects and calculates total marks, percentage and displays the score card details. [Hint: Use list to store the marks in three subjects and total marks. Use __init__() method to initialize name, USN and the lists to store marks and total, Use getMarks() method to read marks into the list, and display() method to display the score card details.]

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50). The minimum passing mark for the SEE is 35% of the maximum marks (18 marks out of 50). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures not less than 35% (18 Marks out of 50) in the semester-end examination (SEE), and a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together.

Continuous Internal Evaluation (CIE):

The CIE marks for the theory component of the IC shall be **30 marks** and for the laboratory component **20 Marks**.

CIE for the theory component of the IC

- Three Tests each of 20 Marks**;** after the completion of the syllabus of 35-40%, 65-70%, and 90-100% respectively.
- Two Assignments/two quizzes/ seminars/one field survey and report presentation/one-course project totaling 20 marks.

Total Marks scored (test + assignments) out of 80 shall be scaled down to **30 marks CIE**

for the practical component of the IC

- On completion of every experiment/program in the laboratory, the students shall be

evaluated and marks shall be awarded on the same day. The **15 marks** are for conducting the experiment and preparation of the laboratory record, the other **05 marks shall be for the test** conducted at the end of the semester.

- The CIE marks awarded in the case of the Practical component shall be based on the continuous evaluation of the laboratory report. Each experiment report can be evaluated for 10 marks. Marks of all experiments' write-ups are added and scaled down to 15 marks.
- The laboratory test **(duration 03 hours)** at the end of the 15th week of the semester /after completion of all the experiments (whichever is early) shall be conducted for 50 marks and scaled down to **05 marks.**

Scaled-down marks of write-up evaluations and tests added will be CIE marks for the laboratory component of IC/IPCC for **20 marks**.

- The minimum marks to be secured in CIE to appear for SEE shall be 12 (40% of maximum marks) in the theory component and 08 (40% of maximum marks) in the practical component. The laboratory component of the IC/IPCC shall be for CIE only. However, in SEE, the questions from the laboratory component shall be included**.** The maximum of 05 questions is to be set from the practical component of IC/IPCC, the total marks of all questions should not be more than 25 marks.

The theory component of the IC shall be for both CIE and SEE.

Semester End Examination (SEE):

SEE for IC

Theory SEE will be conducted by University as per the scheduled time table, with common question papers for the course (duration 03 hours)

1. The question paper will have ten questions. Each question is set for 20 marks.
2. There will be 2 questions from each module. Each of the two questions under a module (with a maximum of 3 sub-questions), **should have a mix of topics** under that module.
3. The students have to answer 5 full questions, selecting one full question from each module.

The theory portion of the Integrated Course shall be for both CIE and SEE, whereas the practical portion will have a CIE component only. Questions mentioned in the SEE paper shall include questions from the practical component).

Passing standard:

- The minimum marks to be secured in CIE to appear for SEE shall be 12 (40% of maximum marks-30) in the theory component and 08 (40% of maximum marks -20) in the practical component. The laboratory component of the IPCC shall be for CIE only. However, in SEE, the questions from the laboratory component shall be included**.** The maximum of 04/05 questions to be set from the practical component of IPCC, the total marks of all questions should not be more than 30 marks.

- SEE will be conducted for 100 marks and students shall secure 35% of the maximum marks to qualify for the SEE. Marks secured will be scaled down to 50.

Suggested Learning Resources:

Text Books

1. Al Sweigart,**"Automate the Boring Stuff with Python",**1st Edition, No Starch Press, 2015. (Available under CC-BY-NC-SA license at https://automatetheboringstuff.com/)

   (Chapters 1 to 18, except 12) for lambda functions use this link:

   https://www.learnbyexample.org/python-lambda-function/
2. Allen B. Downey, **"Think Python: How to Think Like a Computer Scientist",** 2nd Edition, Green Tea Press, 2015. (Available under CC-BY-NC license at http://greenteapress.com/thinkpython2/thinkpython2.pdf

(Chapters 13, 15, 16, 17, 18) (Download pdf/html files from the above link)

Web links and Video Lectures (e-Resources):

- https://www.learnbyexample.org/python/
- https://www.learnpython.org/
- https://pythontutor.com/visualize.html#mode=edit

Activity Based Learning (Suggested Activities in Class)/ Practical Based learning

- Quizzes for list, tuple, string dictionary slicing operations using below link https://github.com/sushantkhara/Data-Structures-And-Algorithms-with-Python/raw/main/Python%203%20_%20400%20exercises%20and%20solutions%20for%20beginners.pdf

COs and POs Mapping (Individual teacher has to fill up)

COs	POs						
	1	2	3	4	5	6	7
CO1							
CO2							
CO3							
CO4							
CO5							

Level 3- Highly Mapped, Level 2-Moderately Mapped, Level 1-Low Mapped, Level 0- Not Mapped

Program 1(a): Basic Input Output Operations

Develop a program to read the student details like Name, USN, and Marksin three subjects. Display the student details, total marks and percentage with suitable messages.

Aim: To understand the basic input-output and if-else statements inpython

Theory:

Programs may use the input function to obtain information from the user. The simplest use of the input function assigns a string to a variable. The only parameter passed to input function is a string which is prompt message to the user. The input function returns the stringand can be assigned to a variable and used in program. The string value returned by input can be converted to number either an integer or real number by using int() or float() function. The print function enables a Python program to display textual information to the user. It prints the values to a stream or to sys.stdout by default. The format of print is:
print (*args, sep=' ', end='\n', file=None, flush=False) where sep: string inserted between values, default a space.

end: string appended after the last value, default a newline.
file: a file-like object (stream); defaults to the current sys.stdout.flush: whether to forcibly flush the stream.

CODE:

```
name = input ("Enter the name of the student: ")
USN = input ("Enter the USN of the student: ")
marks1 = int (input ("Enter marks in Subject 1: "))
marks2 = int (input ("Enter marks in Subject 2: "))
marks3 = int (input ("Enter marks in Subject 3: "))

total=marks1 + marks2 + marks3
percentage = (marks1 + marks2 + marks3) / 300 * 100


print("Student Details\n")
print("Name :", name)
print("USN :", USN)
print("Marks 1 :", marks1)
print("Marks 2 :", marks2)
print("Marks 3 :", marks3)
print("Total :", total)
print("Percentage :", percentage)
```

OUTPUT:

```
Enter the name of the student : aaa
Enter the USN of the student : 123
Enter marks in Subject 1 : 60
Enter marks in Subject 2 : 60
Enter marks in Subject 3 : 60
Student Details

Name : aaa
USN : 123
Marks 1 : 60
Marks 2 : 60
Marks 3 : 60
Total : 180
Percent : 60.0
```

Result:

A program was written to read the student details like Name, USN,and Marks in three subjects and display the entered details.

Program 1(b): Using Datetime module

Develop a program to read the name and year of birth of a person. Display whether the person is a senior citizen or not.

Aim:

To understand the basic input-output and if-else statements in python

Theory:

In Python, date and time are not data types of their own, but a modulenamed **datetime** can be imported to work with the date as well as time. Python Datetime module comes built into Python, so there is no needto install it externally. Python Datetime module supplies classes to work with date and time. These classes provide a number of functions to deal with dates, times, and time intervals. Date and DateTime are an object in Python, so when we manipulate them, we are actually manipulating objects and not strings or timestamps.

CODE:

```
from datetime import date

Name = input("Enter the name of the person : ")
DOB = int(input("Enter his year of birth : "))

curyear = date.today().year
age = curyear - DOB

if (age > 60):
    print(Name, "aged", age, "years is a Senior Citizen.")
else:
    print(Name, "aged", age, "years is not a Senior Citizen.")
```

OUTPUT 1:

```
Enter the name of the person : Akshay
Enter his year of birth : 1978
Akshay aged 44 years is not a Senior Citizen.
```

OUTPUT 2:

```
Enter the name of the person : Poornima
Enter his year of birth : 1957
Poornima  aged 65 years is a Senior Citizen.
```

Result:

A program was written to read the Name and birth year of user and builtin python module datetime was used to calculate user's current age.

Program 2(a): Fibonacci Series

Develop a program to generate Fibonacci sequence of length (N). Read Nfrom the console.

Aim: To understand the basic loop construct in python

Theory:

The Fibonacci sequence is a sequence in which each number is the sum of the two preceding ones. Numbers that are part of the Fibonacci sequence are known as Fibonacci numbers, commonly denoted Fn. The sequence commonly starts from 0 and 1, although some authors start the sequence from 1 and 1 or sometimes (as did Fibonacci) from 1 and 2. Starting from 0 and 1, the first few values in the sequence are:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144.

The Fibonacci numbers may be defined by the recurrence relation

$F_0 = 0$, $F_1 = 1$ and $F_N = F_{N-1} + F_{N-2}$ for $N > 1$.

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc. To loop through a set of code a specified number of times, usually the range () function is used. The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends ata specified number.

CODE:

```
num = int(inp  ut("Enter the Fibonacci sequence length to be generated : "))
firstTerm = 0
secondTerm = 1
print("The Fibonacci series with", num, "terms is :")
print(firstTerm, secondTerm, end=" ")
for i in range(2,num):
    curTerm = firstTerm + secondTerm
    print(curTerm, end=" ")
    firstTerm = secondTerm
    secondTerm = curTerm
```

OUTPUT 1:

```
Enter the Fibonacci sequence length to be generated: 8
The Fibonacci series with 8 terms is:
0 1 1 2 3 5 8 13
```

OUTPUT 2:

```
Enter the Fibonacci sequence length to be generated: 5
The Fibonacci series with 5 terms is:
0 1 1 2 3
```

Result:

A program was developed to generate first N elements of Fibonacciseries using both for and while constructs.

Program 2(b): Factorial and binomial coefficient

Write a function to calculate factorial of a number. Develop a program tocompute binomial coefficient (Given N and R).

Aim: To understand creation of user defined functions in python

Theory:

In Python, the **function is a block of code defined with a name**. The functions are used whenever we need to perform the same task multiple times without writing the same code again. It can take arguments and returns the value. Function improves efficiency and reduces errors because of the reusability of a code. Once we create a function, we can call it anywhere and anytime. The benefit of using afunction is reusability and modularity.

The following steps need to be followed to define a function inPython.

- Use the def keyword with the function name to define a function.
- Next, pass the number of parameters as per requirement.(Optional).
- Next, define the function body with a block of code. This block ofcode is nothing but the action to be performed.

In Python, there is no need to specify curly braces for the function body. The only **indentation** is essential to separate code blocks.

Factorial is a non-negative integer. It is the product of all positive integers less than or equal to that number you ask for factorial. It isdenoted by an exclamation sign (!).

Example:

n! = n* (n-1) * (n-2) *.....1

4! = 4x3x2x1 = 24

CODE:

```
def fact(num):
    if num == 0:
        return 1
    else:
        return num * fact(num-1)


n = int(input("Enter the value of N : "))
r = int(input("Enter the value of R (R cannot be negative or greater than N): "))
nCr = fact(n)/(fact(r)*fact(n-r))

print(n,'C',r," = ","%d"%nCr,sep="")
```

Output:

Enter the value of N : 7
Enter the value of R (R cannot be negative or greater than N): 5
7C5 = 21

Enter the value of N : 5
Enter the value of R (R cannot be negative or greater than N): 5
5C5 = 1

Enter the value of N : 3
Enter the value of R (R cannot be negative or greater than N): 1
3C1 = 3

Enter the value of N : 8
Enter the value of R (R cannot be negative or greater than N): 0
8C0 = 1

Program 3: List operations

Read N numbers from the console and create a list. Develop a program toprint mean, variance and standard deviation with suitable messages.

Aim: To understand list data type, its associated member functions and uses them to perform statistical analysis.

Theory:

Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage. List items are ordered, changeable, and allow duplicate values. List items are also indexed. Lists in Python can be created by just placing the sequence inside the square brackets [].

Mean is average of a given set of data.

Where μ is mean and x1, x2, x3…., xi are elements.

Variance is the sum of squares of differences between all numbers and means.

First, calculate the deviations of each data point from the mean, and squarethe result.

Where μ is Mean, N is the total number of elements or frequency of distribution, Standard Deviation is square root of variance. It is a measure of the extentto which data varies from the mean.

CODE:

```
from math import sqrt

myList = []

num = int(input("Enter the number of elements in your list : "))

for i in range(num):
    val = int(input("Enter the element : "))
    myList.append(val)

print('The length of list1 is', len(myList))

print('List Contents', myList)


total = 0
for elem in myList:
    total += elem

mean = total / num

total = 0
```

```
for elem in myList:
    total += (elem - mean) * (elem - mean)

variance = total / num

stdDev = sqrt(variance)

print("Mean =", mean)
print("Variance =", variance)
print("Standard Deviation =", "%.2f"%stdDev)
```

OUTPUT:

```
Enter the number of elements in your list : 5
Enter the element : 45
Enter the element : 34
Enter the element : 86
Enter the element : 92
Enter the element : 35

The length of list1 is 5
List Contents [45, 34, 86, 92, 35]

Mean = 58.4
Variance = 642.64
Standard Deviation = 25.35
```

Program 4: Frequency of digits in a number

Read a multi-digit number (as chars) from the console. Develop a programto print the frequency of each digit with suitable message.

Aim: To understand dictionary data type, its associated member functionsand uses them to find the frequency of each digit in a number.

Theory:

Method 1: Using if else ladder: Here we have to form an if elif structure forevery digit and update the corresponding count of digit. However there willhave to be checks for every possible digit (0 to 9) and as such the program becomes lengthy.

Method 2: Using set( ) function

set() method is used to convert any of the iterable to a sequence of iterable elements with distinct elements, commonly called Set. In Python, the set() function is a built-in constructor that is used to initialize a set or create an empty.

Count() is a python built-in function that returns the number of times an object appears in a list.

CODE:

```
num = input("Enter a number : ")
print("The number entered is :", num)

uniqDig = set(num)

for elem in uniqDig:
    print(elem, "occurs", num.count(elem), "times")
```

OUTPUT 1:

```
Enter a number : 234939
The number entered is : 234939
4 occurs 1 times
9 occurs 2 times
3 occurs 2 times
2 occurs 1 times
```

OUTPUT 2:

```
Enter a number : 7843338
The number entered is : 7843338
7 occurs 1 times
4 occurs 1 times
3 occurs 3 times
8 occurs 2 times
```

Program 5: Top 10 words in a text file

Develop a program to print 10 most frequently appearing words in a textfile.

Aim: To understand basic file read operation in python and apply the sortmethod of list class to find the top 10 frequently occurred words in file.

Theory:

A file can be opened with the open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None, closed=True, opener=None) file is a string which represents the path indicating the file to be opened, it can be either an absolute or relative path. Mode is an optional string that specifies the mode in which the file is opened. It defaults to 'r' which means open for reading in text mode. encoding is the name of the encoding used todecode or encode the file. errors is an optional string that specifies how encoding errors are to be handled newline controls how universal newlines works.

open() returns a file object whose type depends on the mode, and through which the standard file operations such as reading and writingare performed.

FileNotFoundError exception is thrown if mentioned file does not exist.

A File object represents a file on computer. Whenever we want to read from or write to the file, we can do so by calling methods on theFile object.

Reading the Contents of Files: Any of the following File object methods can be used to read the contents of opened file:

read(size=-1) :

Read at most size characters from stream. Read from underlying bufferuntil we have size characters or we hit EOF. If n is negative or omitted,read until EOF.

readline(size=-1) :

Read until newline or EOF. Returns an empty string if EOF is hitimmediately.

readlines(hint=-1) :

Return a list of lines from the stream. hint can be specified to control the number of lines read, no more lines will be read if the total size (in bytes/characters) of all lines so far exceeds hint.

CODE:

```
import sys
import string
import os.path

fname = input("Enter the filename : ")      #sample file text.txt also provided

if not os.path.isfile(fname):
    print("File", fname, "doesn't exists")
    sys.exit(0)

infile = open(fname, "r")

filecontents = "" for line in infile:
    for ch in line:
        if ch not in string.punctuation:
            filecontents = filecontents + ch
        else:
            filecontents = filecontents + ' '   #replace punctuations and newline with a space
```

```
wordFreq = {}

wordList = filecontents.split()
#Calculate word Frequency
for word in wordList:
    if word not in wordFreq.keys():
        wordFreq[word] = 1
    else:
        wordFreq[word] += 1

# print(wordFreq)

# for word, frequency in wordFreq.items():
#     print(word, "occurs", frequency, "times")

#Sort Dictionary based on values in descending order
sortedWordFreq = sorted(wordFreq.items(), key=lambda x:x[1], reverse=True )

# print(type(sortedWordFreq))

# print(sortedWordFreq)

#Display 10 most frequently appearing words with their count

print("\n=================================================")
print("10 most frequently appearing words with their count")
print("=================================================")
for i in range(10):
    print(sortedWordFreq[i][0], "occurs", sortedWordFreq[i][1], "times")
```

OUTPUT:

```
Enter the filename : text.txt
=================================================
10 most frequently appearing words with their count
=================================================
the occurs 45 times
of occurs 24 times
party occurs 12 times
part occurs 12 times
a occurs 9 times
and occurs 8 times
second occurs 7 times
to occurs 6 times
shall occurs 6 times
first occurs 5 times
```

Result: The contents of text file “sample.txt” were read using buitlin function of python and the top 10 frequently occurred words were displayedusing sorting. The sorting method of list class is used for thepurpose.

Program 6: Sorting contents of a file

Develop a program to sort the contents of a text file and write the sortedcontents into a separate text file. [Hint: Use string methods strip(), len(), list methods sort(), append(), andfile methods open(), readlines(), and write()].

Aim:
To further understand methods of string and list class and utilize themto process the contents of a text file and store the result in file.

Theory:

The built-in str and list classes provide many useful methods to help in processing strings which can be effectively used to process the contents oftext files. Few methods used in this program are:

i) strip(chars=None, /)
Returns a copy of the string with leading and trailing whitespacesremoved. If chars is given and not None, removes characters in chars instead.

ii) split(sep=None, maxsplit=-1) method of built-in. str instance Returns a list of substrings in the string, using sep as the separator
string where sep is the separator used to split string. When set to None (thedefault value), will split on any whitespace. Maxsplit is Maximum number of splits (starting from the left).-1 (the default value) means no limit.

iii) sort(*, key=None, reverse=False) method of built-in list instance Sort the list in ascending order and return None. The sort is in-place (i.e.the list itself is modified). If a key function is given, apply it once to each list item and sort them, ascending or descending, according to theirfunction values. The reverse flag can be set to sort in descending order.

A file can be opened with the open(file, mode='r', buffering=-1,encoding=None, errors=None, newline=None, closefd=True, opener=None).
There are various modes in which a file can be opened. Few modes are:
r : Opens a file for reading only. The file pointer is placed at the beg inningof the file. This is the default mode.
r+ : Opens a file for both reading and writing . The file pointer will be at thebeginning of the file.
rb+ : Opens a file for both reading and writing in binary format. The filepointer will be at the beg inning of the file.
w : Opens a file for writing only. Overwrites the file if the file exists. If thefile does not exist, creates a new file for writing.
w+ : Opens a file for both writing and reading . Overwrites the existing fileif the file exists. If the file does not exist, creates a new file for reading and writing.
a : Opens a file for appending . The file pointer is at the end of the file if thefile exists. That is, the file is in the append mode. If the file does not exist, itcreates a new file for writing.

CODE:

```
import os.path
import sys

fname = input("Enter the filename whose contents are to be sorted : ")   #sample file unsorted.txt also
provided

if not os.path.isfile(fname):
    print("File", fname, "doesn't exists")
    sys.exit(0)

infile = open(fname, "r")
myList = infile.readlines()
# print(myList)

#Remove trailing \n characters
lineList = []
for line in myList:
    lineList.append(line.strip())

lineList.sort()

#Write sorted contents to new file sorted.txt

outfile = open("sorted.txt","w")

for line in lineList:
    outfile.write(line + "\n")

infile.close()  # Close the input file
outfile.close() # Close the output file


if os.path.isfile("sorted.txt"):
    print("\nFile containing sorted content sorted.txt created successfully")
    print("sorted.txt contains", len(lineList), "lines")
    print("Contents of sorted.txt")
    print("================================================================")
    rdFile = open("sorted.txt","r")
    for line in rdFile:
        print(line, end="")
```

OUTPUT :

```
Enter the filename whose contents are to be sorted : unsorted.txt

File containing sorted content sorted.txt created successfully
sorted.txt contains 15 lines
Contents of sorted.txt
============================================================
A deep C diva.
All the troubles you have will pass away very quickly.
Beware of a tall black man with one blond shoe.
Don't read everything you believe.
Exercise caution in your daily affairs.
He changes the domain.
How does a hacker fix a function which doesn't work for all of the elements in its domain?
Lay on, MacDuff, and curs'd be him who first cries, "Hold, enough!".
People are beginning to notice you.  Try dressing before you leave the house.
The surest protection against temptation is cowardice.
To be or not to be.
Tuesday is the Wednesday of the rest of your life.
What is the square root of 4b^2?
You display the wonderful traits of charm and courtesy.
You may be recognized soon.  Hide.
```

Result: The contents of text file “sample1.txt” were read using buitlin function of python and sorted contents were stored in a new file sample2.txt.

Program 7: Backing up a folder using zipfile module

Develop a program for backing Up a given Folder (Folder in a current working directory) into a ZIP File by using relevant modules and suitablemethods.

Aim: To further understand methods of string and list class and utilize themto process the contents of a text file and store the result in file.

Theory:

ZIP files (with the .zip file extension), can hold the compressed contents of many other files. Compressing a file reduces its size, which is useful whentransferring it over the Internet. And since a ZIP file can also contain multiple files and subfolders, it's a handy way to package several files into one. This single file, called an archive file, can then be, say, attached to an email. Python programs can both create and open (or extract) ZIP files usingfunctions in the zipfile module.

The function backupToZip( ) function that takes just one parameter, folder. This parameter is a string path to the folder whose contents should be backed up. The function will determine what filename to use for the ZIP fileit will create; then the function will create the file, walk the folder folder, and add each of the subfolders and files to the ZIP file.

CODE:

```
import os
import sys
import pathlib
import zipfile

dirName = input("Enter Directory name that you want to backup : ")

if not os.path.isdir(dirName):
    print("Directory", dirName, "doesn't exists")
    sys.exit(0)

curDirectory = pathlib.Path(dirName)

with zipfile.ZipFile("myZip.zip", mode="w") as archive:
    for file_path in curDirectory.rglob("*"):
        archive.write(file_path, arcname=file_path.relative_to(curDirectory))

if os.path.isfile("myZip.zip"):
    print("Archive", "myZip.zip", "created successfully")
else:
    print("Error in creating zip archive")
```

OUTPUT:

```
Enter Directory name that you want to backup : zipDemo

Archive myZip.zip created successfully
```

Result: The contents of folder "E:\\Backup\\Resources\\Python_Resources" were backed up and stored as Python_Resources_1.zip.

Program 8: Assertion and Exception Handling

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 suitable program which reads two values from the console and calls a function DivExp.

Aim: To understand user defined function and exception handling feature ofpython

Theory:

An assertion is a (debugging) boolean expression to self check an assumption about the internal state of a function or object. If true, nothing happens otherwise, execution halts and an error is thrown. Assertion is a way to declare a condition that is impossible to occur. Assertions are typically used to do sanity checking that is too expensive for production use.The feedback given by an assertion is for the development team as opposed to the user. Assertions are used in following cases:

- Parameter types for a private function (ex. the power has to be aninteger in exponential function)
- Precondition (ex. positive number in a square root function)
- Can not happen situations (ex. duplicates in a list, zero divisor)
- Corner cases that are hard to occur but possible (depends on theapplication)

An exception is similar to an assertion but the meaning is different. An exception is an unexpected execution event that disrupts the normal flow of the program. Exceptions are used to signal an error condition where a corrective action can be taken (ex. file not found). Exceptions can be caught and recovered from as opposed to assertions.

When the program encounters an error during the course of execution, an exception should be raised with a meaningful error message. Here are a fewexamples…

- Invalid input from user or external source
- System failure such as file not found or network errors

Definition of an assertion

The syntax for assertion in Python is: assert (condition), “optional messageif not met”

To use a method that declares an exception in its signature, you must either: provide exception handling codes in a " try-except " construct, or. not handling the exception in the current method.

CODE:

```
import sys

def DivExp(a,b):
    assert a>0, "a should be greater than 0"
    try:
        c = a/b
    except ZeroDivisionError:
        print("Value of b cannot be zero")
        sys.exit(0)
```

```
    else:
        return c

val1 = int(input("Enter a value for a : "))
val2 = int(input("Enter a value for b : "))

val3 = DivExp(val1, val2)

print(val1, "/", val2, "=", val3)
```

OUTPUT 1:
Enter a value for a : 7
Enter a value for b : 6
7 / 6 = 1.1666666666666667

OUTPUT 2:
Enter a value for a : 0
Enter a value for b : 5
AssertionError: a should be greater than 0

OUTPUT 3:
Enter a value for a : -3
Enter a value for b : 6
AssertionError: a should be greater than 0

OUTPUT 4:
Enter a value for a : 6
Enter a value for b : 0
Value of b cannot be zero

Result: User defined function was developed and used to calculate divisionof two numbers along with proper assertion failure and exception handling.

Program 9:Operator overloading

Define a function which takes TWO objects representing complex numbersand 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 theaddition of N complex numbers.

Aim: To understand the concept of object oriented programming

Theory: Classes are a way of combining similar data and functions. A class is basically a scope inside which various code (especially function definitions) is executed, and the locals to this scope become attributes of the class and of any objects constructed by this class. An object constructed by a class is called an instance of that class.
User-defined objects can be made to work with all of Python's built-in operators by adding implementations of the special methods to a class. For example, if you wanted to add a new kind of number to Python, you could define a class in which special methods such as add( )were defined to make instances work with the standard mathematical operators.

CODE:

```
class Complex:
    def __init__(self, realp = 0, imagp=0):
        self.realp = realp
        self.imagp = imagp

    def setComplex(self, realp, imagp):
        self.realp = realp
        self.imagp = imagp

    def readComplex(self):
        self.realp = int(input("Enter the real part : "))
        self.imagp = int(input("Enter the real part : "))

    def showComplex(self):
        print('(',self.realp,')','+i','(',self.imagp,')',sep="")

    def addComplex(self, c2):
        c3 = Complex()
        c3.realp = self.realp + c2.realp
        c3.imagp = self.imagp + c2.imagp
        return c3

def add2Complex(a,b):
    c = a.addComplex(b)
    return c

def main():
    c1 = Complex(3,5)
    c2 = Complex(6,4)
```

```
 print("Complex Number 1")
    c1.showComplex()
    print("Complex Number 2")
    c2.showComplex()

    c3 = add2Complex(c1, c2)

    print("Sum of two Complex Numbers")
    c3.showComplex()

#Addition of N (N >=2) complex numbers

    compList = []

    num = int(input("\nEnter the value for N : "))

    for i in range(num):
        print("Object", i+1)
        obj = Complex()
        obj.readComplex()
        compList.append(obj)

    print("\nEntered Complex numbers are : ")
    for obj in compList:
        obj.showComplex()

    sumObj = Complex()
    for obj in compList:
        sumObj = add2Complex(sumObj, obj)


    print("\nSum of N complex numbers is", end = " ")
    sumObj.showComplex()

main()
```

OUTPUT:

```
Complex Number 1
(3)+i(5)
Complex Number 2
(6)+i(4)
Sum of two Complex Numbers
(9)+i(9)

Enter the value for N : 5
Object 1
Enter the real part : 1
Enter the real part : 9
Object 2
```

```
Enter the real part : 2
Enter the real part : 8
Object 3
Enter the real part : 3
Enter the real part : 7
Object 4
Enter the real part : 4
Enter the real part : 6
Object 5
Enter the real part : 5
Enter the real part : 5

Entered Complex numbers are :
(1)+i(9)
(2)+i(8)
(3)+i(7)
(4)+i(6)
(5)+i(5)

Sum of N complex numbers is (15)+i(35)
```

Result: User defined class Complex was created to handle complex numbers and + operator was overloaded for the class and verified.

Program 10: Student Class and its Methods

Develop a program that uses class Student which prompts the user to enter marks in three subjects and calculates total marks, percentage and displays the score card details.
[Hint: Use list to store the marks in three subjects and total marks. Use init() method to initialize name, USN and the lists to store marks and total,Use getMarks() method to read marks into the list, and display() method todisplay the score card details.]

Aim: To understand the concept of classes, it's attributes and objects.

Theory:

A class defines a set of attributes that are associated with, and shared by, a collection of objects known as instances. A class is most commonly a collection of functions (known as methods), variables (which are known as class variables), and computed attributes(which are known as properties).
The functions defined inside a class are known as instance methods. An instance method is a function that operates on an instance of the class, which is passed as the first argument. By convention, this argument is called self. In the program written below, getMarks ( ) and display are examples of instance methods. Instances of a class are created by calling a class object as a function. This creates a new instance that is then passed to the__init__() method of the class. The arguments to __init__() consist of the newly created instance self along with the arguments supplied when calling the class object. Inside__init__(), attributes are saved in the instance by assigning to self. For example, self.name = name is saving a name attribute in the instance. Once the newly created instance has been returned to the user, these attributes as well as attributes of the class are accessed using the dot (.) operator.

CODE:

```
class Student:
    def __init__(self, name = "", usn = "", score = [0,0,0,0]):
        self.name = name
        self.usn = usn
        self.score = score

    def getMarks(self):
        self.name = input("Enter student Name : ")
        self.usn = input("Enter student USN : ")
        self.score[0] = int(input("Enter marks in Subject 1 : "))
        self.score[1] = int(input("Enter marks in Subject 2 : "))
        self.score[2] = int(input("Enter marks in Subject 3 : "))
        self.score[3] = self.score[0] + self.score[1] + self.score[2]

    def display(self):
        percentage = self.score[3]/3
        spcstr = "=" * 81
        print(spcstr)
        print("SCORE CARD DETAILS".center(81))
        print(spcstr)
        print("%15s"%("NAME"), "%12s"%("USN"),
```

```
"%8s"%"MARKS1","%8s"%"MARKS2","%8s"%"MARKS3","%8s"%"TOTAL","%12s"%("PERCEN
TAGE"))
        print(spcstr)
        print("%15s"%self.name, "%12s"%self.usn,
"%8d"%self.score[0],"%8d"%self.score[1],"%8d"%self.score[2],"%8d"%self.score[3],"%12.2f"%percen
tage)
        print(spcstr)


def main():
    s1 = Student()
    s1.getMarks()
    s1.display()

main()
```

OUTPUT :

```
Enter student Name : Shivappa
Enter student USN : 1SI22CS065
Enter marks in Subject 1 : 87
Enter marks in Subject 2 : 79
Enter marks in Subject 3 : 92

=============================================================================
                              SCORE CARD DETAILS
=============================================================================
       NAME        USN        MARKS1   MARKS2   MARKS3    TOTAL   PERCENTAGE
=============================================================================
     Shivappa   1KG22CS065      87         79         92          258        86.00
=============================================================================
```

Result: User defined class Student was created and used to process elementary information about student class using instance variables andmethods.

Viva Question and Answers

1. What is Python?

Python is an interpreted scripting language that is known for its power, interactivity, and object-oriented nature. It utilizes English keywords extensively and has a simpler syntax compared to many other programming languages.

2. Python is an interpreted language. Explain.

An interpreted programming language refers to any language that executes its instructions sequentially, one line at a time. In the case of Python, programs are executed directly from the source code without the need for any intermediate compilation process.

3.What distinguishes lists from tuples?

Lists	**Tuples**
Lists are mutable, i.e., they can be edited	Tuples possess immutability, denoting their incapability of being modified like lists.
Lists are usually slower than tuples	Tuples are faster than lists
Lists consume a lot of memory	Tuples consume less memory when compared to lists

4. **What are the Key features of Python?**

The key features of Python are as follows: Interpreted, Dynamically typed, Object-oriented programming, Cross platform, General-purpose language.

5. **How is Memory managed in Python?**

Python makes use of automatic memory management through garbage collection. The garbage collector keeps track of objects and frees memory when they are no longer in use.

6. What are Python Modules?

Files containing Python codes are referred to as Python modules. This code can be of different types like classes, functions, or variables.

7. What are python namespaces?

A Python namespace ensures that the names assigned to objects within a program are unique and can be used without conflict.

8. Explain Inheritance in Python with an example?

Python embraces the principles of object-oriented programming and allows classes to acquire the characteristics of another class, a concept known as inheritance. This facilitates code reuse, promoting efficiency.

9. What is scope resolution?

In Python, a scope defines the region of code where an object remains valid and accessible. Every object in Python operates within its designated scope.

10. What is a dictionary in Python?

Python supports various data types, including dictionaries. A dictionary in Python is a collection of elements that are stored as key-value pairs.

11. What is the difference between / and // operator in Python?

/ : is a division operator and returns the value of the quotient.

// : is known as floor division operator and used to return the value of quotient before the decimal point.

12. What are the applications of Python?

The applications of Python are as follows:

- GUI-based desktop applications
- Image processing applications
- Business and Enterprise applications
- Prototyping
- Web and web framework applications

13. What are the two major loop statements?

The two major loop statements in python are for and while.

14. What are functions in Python?

A function is a segment of code that runs only when it is called. The “def” keyword is utilized to define a specific function,

15. What are the common built-in data types in Python?

Python supports the below-mentioned built-in data types:

Immutable data types: Number, String, Tuple

Mutable data types: List, Dictionary, Set

16. What are local variables and global variables in Python?

A local variable is a variable that is defined within a specific function and is only accessible within that function. It cannot be accessed by other functions within the program. A global variable is a variable that is declared outside of any function, allowing it to be accessed by all functions in the program

17. What is type conversion in Python?

Python offers a valuable feature that allows for the conversion of data types as needed. This process is referred to as type conversion in Python. Type conversion can be divided into two types:

Implicit Type Conversion: This type of conversion is automatically performed by the Python interpreter without requiring any user intervention.

Explicit Type Conversion: This type of conversion involves the user explicitly changing the data type to the desired type.

18. What is the difference between Python Arrays and lists?

Arrays are data structures that hold fixed-size elements of the same type. Lists are versatile data structures that can hold elements of different types and sizes.

19. Is python case sensitive?

Yes, Python is a case sensitive language. In Python, it is important to note that "Function" and "function" are distinct entities,

20. What are decorators?

In Python, decorators serve as essential functions that enable the addition of functionality to an already existing function without altering its structure. These decorators are denoted by the @decorator_name syntax in Python and are invoked in a bottom-up manner

21. Is indentation required in Python?

Indentation is an essential aspect of Python syntax, ensuring proper code structure. It is a method used by programming languages to determine the scope and extent of code blocks.

22. How does break, continue, and pass work?

The following statements assist in altering the course of execution from the regular flow, which categorizes them as loop control statements.

- **Python break:** This statement aids in discontinuing the loop or the statement and transferring control to the subsequent statement.
- **Python continue:** This statement enforces the execution of the subsequent iteration when a particular condition is met, instead of terminating it.
- **Python pass:** This statement allows the syntactical writing of code while intending to bypass its execution. It is also recognized as a null operation, as no action is taken when the pass statement is executed.

23. How to comment with multiple lines in Python?

To include a multiline comment in Python, each line should begin with the # symbol. This practice ensures that the code is clear and easily understood.

24. What type of language is python? Programming or scripting?

Python is a versatile programming language that excels not only as a general-purpose language but also as a scripting tool.

25. What are indexes and why are they used?

To retrieve an item from a sequential collection, we can simply utilize its index, which represents the position of that specific item. Conventionally, the index commences at 0, implying that the initial element has an index of 0, the second element has an index of 1, and so forth.

26. what split(), sub(), subn() functions perform in Python?

- split(): This method is used to split a given string into a list.
- sub(): This method is used to find a substring where a regex pattern matches, and then it replaces the matched substring with a different string.
- subn(): This method is similar to the sub() method, but it returns the new string, along with the number of replacements.

27. What do you mean by Python literals?

In programming, literals are used to represent specific values assigned to variables or constants. Python offers various types of literals, including string literals, numeric literals, and boolean literals.

28. Do we need to declare variables with respective data types in Python?

No. Python is a dynamically typed language, i.e., the Python Interpreter automatically identifies the data type of a variable based on the type of value assigned.

29. How do you write comments in python?

Python comments are statements used by the programmer to increase the readability of the code. With the help of the #, you can define a single comment.

30. Is multiple inheritances supported in Python?

Yes, unlike Java, Python provides users with a range of support in terms of inheritance and its usage. Multiple inheritances refer to a scenario where a class is instantiated from more than one individual parent class. This provides a lot of functionality and advantages to users.

31. Is Python fully object oriented?

Python does follow an object-oriented programming paradigm and has all the basic OOPs concepts such

as inheritance, polymorphism, and more, with the exception of access specifiers. Python doesn't support strong encapsulation (adding a private keyword before data members). Although, it has a convention that can be used for data hiding, i.e., prefixing a data member with two underscores.

32. Explain all file processing modes supported in Python?

Python has various file processing modes.

For opening files, there are three modes:

- read-only mode (r)
- write-only mode (w)
- read–write mode (rw)

For opening a text file using the above modes, we will have to append 't' with them as follows:

- read-only mode (rt)
- write-only mode (wt)
- read–write mode (rwt)

Similarly, a binary file can be opened by appending 'b' with them as follows:

- read-only mode (rb)
- write-only mode (wb)
- read–write mode (rwb)

33. Explain the use of the 'with' statement and its syntax?

Using the 'with' statement, we can open a file and close it as soon as the block of code, where 'with' is used, exits. In this way, we can opt for not using the close() method.

34. What does len() do?

len() is an inbuilt function used to calculate the length of sequences like list, python string, and array.

35. How will you remove duplicate elements from a list?

To remove duplicate elements from the list we use the set () function.

36. How can files be deleted in Python?

You need to import the OS Module and use **os.remove()** function for deleting a file in python.

37. How will you read a random line in a file?

We can read a random line in a file using the random module.

38. What is the purpose of "is", "not" and "in" operators?

Operators are referred to as special functions that take one or more values (operands) and produce a corresponding result.

- **is**: returns the true value when both the operands are true (Example: "x" is 'x')
- **not**: returns the inverse of the boolean value based upon the operands (example:"1" returns "0" and vice-versa.
- **In**: helps to check if the element is present in a given Sequence or not.

39. How can the ternary operators be used in python?

The ternary operator is the operator that is used to show conditional statements in Python. This consists of the boolean true or false values with a statement that has to be checked.

40. How to add values to a python array?

In python, adding elements in an array can be easily done with the help of **extend(), append()** and **insert()**functions.

41. How to remove values from a python array?

Elements can be removed from a python array by using **pop()** or **remove()** methods.

pop (): This function will return the removed element .

remove (): It will not return the removed element.

42. How can you generate random numbers in Python?

This is achieved by importing the random module. It is the module that is used to generate random numbers.

43. How will you convert a string to all lowercase?

To convert a string to all lowercase in Python, you can use the built-in lower () method. The lower () method is available for strings and returns a new string with all characters converted to lowercase.

44. What is polymorphism in Python?

Polymorphism is the ability of the code to take multiple forms. Let's say, if the parent class has a method named XYZ then the child class can also have a method with the same name XYZ having its own variables and parameters.

45. Define encapsulation in Python?

Encapsulation in Python refers to the process of wrapping up the variables and different functions into a single entity or capsule. The Python class is the best example of encapsulation in python.

46. What is self in Python?

Self is an object or an instance of a class. This is explicitly included as the first parameter in Python. On the other hand, in Java it is optional. It helps differentiate between the methods and attributes of a class with local variables.

47. What is the difference between append () and extend() methods?

Both **append ()** and **extend ()** methods are methods used to add elements at the end of a list.

The primary differentiation between the append () and extend () methods in Python is that append () is used to add a single element to the end of a list. In contrast, open () is used to append multiple aspects, such as another list or an iterable, to the end of a list.

4**8. What is slicing in Python?**

Slicing is a technique employed to extract a specific range of elements from sequential data types, such

as lists, strings, and tuples. Slicing is beneficial and easy to extract out elements. It requires a : (colon) which separates the start index and end index of the field.

49. How do you create a Python function?

Functions are defined using the **def** statement.

50. What happens when a function doesn't have a return statement? Is this valid?

Yes, this is valid. The function will then return a none object. The end of a function is defined by the block of code that is executed (i.e., the indenting) not by any explicit keyword.

51. Name some of the built-in modules in Python?

The built-in modules in Python are:

- sys module
- OS module
- random module
- collection module
- JSON
- Math module