|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PYTHON PROGRAMMING LABORATORY (21CSL46)** | | | | |
| **Course Code** | | **21CSL46** | **CIE Marks** | 50 |
| **Number of Contact Hours/Week** | | 0:0:2:0 | **SEE Marks** | 50 |
| **Total Number of Lab Contact Hours** | | 24 | **Exam Hours** | 03 |
| **Credits –1** | | | | |
| **Descriptions(if any):** | | | | |
|  | | | | |
| **Programs List:** | | | | |
|  | | | | |
| 1. | a) Write a python program to find the best of two test average marks out of three test’s marks accepted from the user.  b) Develop a Python program to check whether a given number is palindrome or not and also count the number of occurrences of each digit in the input number. | | | |
| 2. | a) Defined as a function F as Fn = Fn-1 + Fn-2. Write a Python program which accepts a value for N (where N >0) as input and pass this value to the function. Display suitable error message if the condition for input value is not followed.  b) Develop a python program to convert binary to decimal, octal to hexadecimal using functions. | | | |
| 3 | a) Write a Python program that accepts a sentence and find the number of words, digits, uppercase letters and lowercase letters.  b) Write a Python program to find the string similarity between two given strings   |  |  | | --- | --- | | **Sample Output:**  Original string:  Python Exercises  Python Exercises  Similarity between two said strings:  1.0 | **Sample Output:**  Original string:  Python Exercises  Python Exercise  Similarity between two said strings:  0.967741935483871 | | | | |
| 4 | a) Write a python program to implement insertion sort and merge sort using lists  b) Write a program to convert roman numbers in to integer values using dictionaries. | | | |
| 5. | a) Write a function called isphonenumber () to recognize a pattern 415-555-4242without using regular expression and also write the code to recognize the same pattern using regular expression.  b) Develop a python program that could search the text in a file for phone numbers(+919900889977) and email addresses (sample@gmail.com) | | | |
| 6 | a) Write a python program to accept a file name from the user and perform the following operations  1. Display the first N line of the file  2. Find the frequency of occurrence of the word accepted from the user in the file  b) Write a python program to create a ZIP file of a particular folder which contains several files inside it. | | | |
| 7. | a) By using the concept of inheritance write a python program to find the area of triangle, circle and rectangle.  b) Write a python program by creating a class called Employee to store the details of Name, Employee ID, Department and Salary, and implement a method to update salary of employees belonging to a given department. | | | |
| 8 | a) Write a python program to find the whether the given input is palindrome or not (for both string and integer) using the concept of polymorphism and inheritance. | | | |
| 9 | a) Write a python program to download the all XKCD comics  b) Demonstrate python program to read the data from the spreadsheet and write the data in to the spreadsheet | | | |
| 10 | a) Write a python program to combine select pages from many PDFs  b) Write a python program to fetch current weather data from the JSON file | | | |

**PART B – Practical Based Learning**

A problem statement for each batch is to be generated in consultation with the co-examiner and student should develop an algorithm, program and execute the program for the given problem with appropriate outputs.

**Python Introduction**

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991

## **What is Python**

Python is a general-purpose, dynamic, high-level, and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.

Python is an easy-to-learn yet powerful and versatile scripting language, which makes it attractive for Application Development.With its interpreted nature, Python's syntax and dynamic typing make it an ideal language for scripting and rapid application development.

# Python Data Types

## **Built-in Data Types**

In programming, data type is an important concept.

Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default, in these categories:

|  |  |
| --- | --- |
| Text Type: | str |
| Numeric Types: | int, float, complex |
| Sequence Types: | list, tuple, range |
| Mapping Type: | dict |
| Set Types: | set, frozenset |
| Boolean Type: | bool |
| Binary Types: | bytes, bytearray, memoryview |
| None Type: | NoneType |

## **Getting the Data Type**

You can get the data type of any object by using the type() function:

### **Example**

Print the data type of the variable x:

x = 5  
print(type(x))

# Python Lists

List items are ordered, changeable, and allow duplicate values.

List items are indexed, the first item has index [0], the second item has index [1] etc.

List items can be of any data type

mylist = ["apple", "banana", "cherry", True, 1, 4.5]

## **Access Items**

List items are indexed and you can access them by referring to the index number:

### **Example**

Print the second item of the list:

thislist = ["apple", "banana", "cherry"]  
print(thislist[1])

## **Characteristics of Lists**

* The lists are in order.
* The list element can be accessed via the index.
* The mutable type of List is
* The rundowns are changeable sorts.
* The number of various elements can be stored in a list.

# Python Set

A Python set is the collection of the unordered items. Each element in the set must be unique, immutable, and the sets remove the duplicate elements. Sets are mutable which means we can modify it after its creation.

Unlike other collections in Python, there is no index attached to the elements of the set, i.e., we cannot directly access any element of the set by the index. However, we can print them all together, or we can get the list of elements by looping through the set.

Python Dictionary

Dictionaries are a useful data structure for storing data in Python because they are capable of imitating real-world data arrangements where a certain value exists for a given key.

The data is stored as key-value pairs using a Python dictionary.

* This data structure is mutable
* The components of dictionary were made using keys and values.
* Keys must only have one component.
* Values can be of any type, including integer, list, and tuple.

**Syntax:**

1. Dict = {"Name": "Gayle", "Age": 25}

In the above dictionary **Dict**, The keys **Name** and **Age** are the strings which comes under the category of an immutable object

**PYTHON PROGRAMMING LABORATORY (21CSL46)**

**Lab Program 1-10**

**1.a) Write a python program to find the best of two test average marks out of three test’s marks accepted from the use**

m1 = int(input("Enter marks for test1 : "))

m2 = int(input("Enter marks for test2 : "))

m3 = int(input("Enter marks for test3 : "))

if m1 <= m2 and m1 <= m3:

avgMarks = (m2+m3)/2

elif m2 <= m1 and m2 <= m3:

avgMarks = (m1+m3)/2

else:

avgMarks = (m1+m2)/2

print("Average of best two test marks out of three test’s marks is", avgMarks);

**output**

Enter the test1 Marks 25

Enter the test2 Marks 25

Enter the test3 Marks 12

Average of Best of two Test is 25.0

**or**

m1=int(input("Enter the test1 Marks"))

m2=int(input("Enter the test2 Marks"))

m3=int(input("Enter the test3 Marks"))

L1=[]

L1.append(m1)

L1.append(m2)

L1.append(m3)

L1.sort()

print(L1)

Avg=(L1[1]+L1[2])/2

print("Average of best two test marks out of three test’s marks is ",Avg)

**output**

Enter the test1 Marks 25

Enter the test2 Marks 25

Enter the test3 Marks 12

[12, 25, 25]

Average of Best of two Test is 25.0

**or**

m1=int(input("Enter the test1 Marks"))

m2=int(input("Enter the test2 Marks"))

m3=int(input("Enter the test3 Marks"))

Minimum=min(m1,m2,m3)

sum=m1+m2+m3-Minimum

Avg=sum/2

print("Average of best two test marks out of three test’s marks is ",Avg)

**output**

Enter the test1 Marks25

Enter the test2 Marks25

Enter the test3 Marks12

Average of Best of two Test is 25.0

1. **b ) Develop a Python program to check whether a given number is palindrome or not and also count the number of occurrences of each digit in the input number.**

num=input("Enter the Number")

rev=num

if (rev==rev[::-1]):

print("The Number is Palindrome", num)

else:

print("The Number is not Palindrome", num)

for i in range(9):

if (rev.count(str(i))>0):

print(i, "appears", rev.count(str(i)),"times")

**output**

Enter the Number12321

The Number is Palindrome 12321

1 appears 2 times

2 appears 2 times

3 appears 1 times

**or**

num=input("Enter the Number")

rev=num

if (rev==rev[::-1]):

print("The Number is Palindrome", num)

else:

print("The Number is not Palindrome", num)

for i in set(num):

print(i, "appears", num.count(str(i)),"times")

**output**

Enter the Number12321

The Number is Palindrome 12321

1 appears 2 times

3 appears 1 times

2 appears 2 times

**2. a) Defined as a function F as Fn = Fn-1 + Fn-2. Write a Python program which accepts a value for N (where N >0) as input and pass this value to the function. Display suitable error message if the condition for input value is not followed.**

def Fibonacci(n):

if n==1:

return 0

elif n==2:

return 1

else:

return (Fibonacci(n-1)+Fibonacci(n-2))

num=int(input("Enter the number\n"))

if num>0:

res=Fibonacci(num)

print("Fibonacci of ", num ,"is",res)

else:

print("Error in the input")

**Output**

Enter the number 5

Fibonacci of 5 is 3

Enter the number -9

Error in input

**2.b) Develop a python program to convert binary to decimal, octal to hexadecimal using functions**

def Bin2dec(bin):

l=len(oct)

dec=0

for i in range(l):

dec+=int(bin[i])\*((2\*\*(l-i-1))

return dec

def oct2hex(oct):

l=len(oct)

dec=0

for i in range(l):

dec+=int(oct[i])\*(8\*\*(l-i-1))

hexa=['0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F']

octhex=' '

while dec>0:

rem=dec%16

octhex=hexa[rem]+octhex

dec=dec//16

return octhex

bin=input("Enter the Binary Number")

print(“Binary to Decimal is “,Bin2dec(bin))

oct=input("Enter the octal Number")

print(“Octal to Decimal is “,oct2hex(oct))

**Output**

Enter a binary number : 100

Binary to Decimal is 4

Enter a octal number : 16

Octal to Decimal is E

**3. a) Write a Python program that accepts a sentence and find the number of words, digits, uppercase letters and lowercase letters.**

sentence = input("Enter a sentence : ")

digCnt = upCnt = loCnt =wordcnt=0

wordcont=sentence.split()

for ch in sentence:

if ch>='0' and ch<='9':

digCnt += 1

if ch>='A'and ch<='Z':

upCnt += 1

if ch>='a' and ch<='z':

loCnt += 1

print("This sentence has\n")

print("words: ", len(wordcont),"\n" "digits",digCnt, )

print("upper case letters",upCnt, "\n" "lower case letters ",loCnt)

***Output***

Enter a sentence : Hello RNSIT, 2001 &

This sentence has

words: 4

digits 4

upper case letters 6

lower case letters 4

**3.b) Write a Python program to find the string similarity between two given strings**

def compare(s,p):

count=0

n=min(len(s),len(p))

for i in range(n):

if s[i]==p[i]:

count+=1

return count

s1 = input("Enter String 1 \n")

s2 = input("Enter String 2 \n")

mx=max(len(s1),len(s2))

count=compare(s1,s2)

similarity=count/mx\*100

print ("Total number letter matched is",count)

print("simirality between two string is",similarity)

***Output***

Enter String 1

rnsit

Enter String 2

rnsit

Total number letter matched is 5

simirality between two string is 100.0

Enter String 1

welcome

Enter String 2

rnsit

Total number letter matched is 0

simirality between two string is 0.0

**4.a Write a python program to implement insertion sort and merge sort using lists**

def insertion\_sort(arr,n):

for i in range(1,n):

key = arr[i]

j = i-1

while j>=0 and key < arr[j]:

arr[j+1] = arr[j]

j -=1

arr[j+1] = key

return arr

def merge\_sort(arr):

n = len(arr)

if(n>1):

mid = n//2

L = arr[:mid]

R = arr[mid:]

merge\_sort(L)

merge\_sort(R)

i = j = k = 0

while i<len(L) and j<len(R):

if L[i]<=R[j]:

arr[k] = L[i]

i+=1

else:

arr[k] = R[j]

j+=1

k+=1

while(i<len(L)):

arr[k] = L[i]

k+=1

i+=1

while(j<len(R)):

arr[k] = R[j]

k+=1

j+=1

n = int(input("Enter the size of array"))

arr = [None]\*n

print("Enter the elements")

for i in range(n):

arr[i] = int(input())

print("Before Sorting : ",arr)

merge\_sort(arr)

print("After Merge Sort : ",arr)

insertion\_sort(arr,n)

print("After Insertion sort : ",arr)

**output**

Enter the size of array5

Enter the elements

3

6

7

1

9

Before Sorting : [3, 6, 7, 1, 9]

After Merge Sort : [1, 3, 6, 7, 9]

After Insertion sort : [1, 3, 6, 7, 9]

**4.b Write a program to convert roman numbers in to integer values using dictionaries**

def roman2integer(roman\_numeral):

roman\_dict={'I':1,'V':5,'X':10,'L':50,'C':100,'D':500,'M':1000}

res=0

pre\_val=0

for symbol in roman\_numeral[::-1]:

value=roman\_dict[symbol]

if(value>=pre\_val):

res=res+value

else:

res=res-value

pre\_val=value

return res

roman\_numeral=input("Enter a Roman numeral:")

integer\_value=roman2integer(roman\_numeral)

print("Integer value:", integer\_value)

**output**

Enter a Roman numeral:IV

Integer value: 4

**PYTHON File Operations**

To store data temporarily and permanently, we use files. A file is the collection of data stored on a disk in one unit identified by filename.

1. Create Files
2. Read Files
3. Write to files

**Create and Write**

with open(‘test.txt’,’w’) as fp:

fp.write(“new line”)

**#read**

with open(‘test.txt’,’w’) as fp:

fp.read()

## **Types of File**

* **Text File**: Text file usually we use to store character data. For example, test.txt
* **Binary File**: The binary files are used to store binary data such as images, video files, audio files, etc.

## **File Path**

A file path defines the location of a file or folder in the computer system. There are two ways to specify a file path.

1. **Absolute path**: which always begins with the root folder
2. **Relative path**: which is relative to the program's current working directory

The absolute path includes the complete directory list required to locate the file.

## **Read File**

To read or write a file, Python provides a built-in function open().

Syntax  open(file\_path, access\_mode) function. It returns the file object. This object is used to read or write the file according to the access mode.

Accesss mode represents R is for reading and W is for writing

**# Opening the file with absolute path**

fp = open(‘C:\\Users\\deepa\\OneDrive\\Desktop\\ex.txt’, 'r')

**# read file**

print(fp.read())

**# Closing the file after reading**

fp.close()

**output**

hn@gmail.com

+919743303225

+919686807518

hm46@yahoo.in

kffgh

## **Writing to a File**

To write content into a file, Use the access mode w to open a file in a write mode.

**Note**:

* If a file already exists, it truncates the existing content and places the filehandle at the beginning of the file. A new file is created if the mentioned file doesn’t exist.
* If you want to add content at the end of the file, use the access mode a to open a file in append mode

text = "This is new content"

**# writing new content to the file**

fp = open("C:\\Users\\deepa\\OneDrive\\Desktop\\ex.txt", 'w')

fp.write(text)

print('Done Writing')

fp.close()

**output**

Done Writing'

**Appending content to file**

import re

f=open("C://Users//deepa//OneDrive//Desktop//ex.txt",'+a')

print(f.write("hi"))

f1=open("C://Users//deepa//OneDrive//Desktop//ex.txt",'+r')

print(f1.read())

**Move File Pointer**

The seek() method is used to change or move the file's handle position to the specified location. The cursor defines where the data has to be read or written in the file.

The position (index) of the first character in files is zero, just like the string index.

Example

f = open("sample.txt", "r")

**# move to 11 character**

f.seek(11)

# read from 11th character

print(f.read())

**output**

content

**readlines() Read file into a list**

**# Single Line**

import re

f=open("C://Users//deepa//OneDrive//Desktop//ex.txt",'r')

print(f.readline())

**# Multiple Lines**

import re

f=open("C://Users//deepa//OneDrive//Desktop//ex.txt",'r')

print(f.readlines())

## **Python File Methods**

**Method Description**

read() Returns the file content.

readline() Read single line

readlines() Read file into a list

truncate(size) Resizes the file to a specified size.

write() Writes the specified string to the file.

writelines() Writes a list of strings to the file.

close() Closes the opened file.

seek() Set file pointer position in a file

tell() Returns the current file location.

fileno() Returns a number that represents the stream, from the operating system's perspective.

flush() Flushes the internal buffer.

# Python List Files in a Directory

[Home](https://pynative.com/) » [Python](https://pynative.com/python/) » [File Handling](https://pynative.com/python/file-handling/) » Python List Files in a Directory

**Python List Files in a Directory**

**To list all files in a directory using Python, you can use the built-in os module.**

Also, there are multiple ways to list files in a directory. In this article, We will use the following four methods.

* **os.listdir('dir\_path')**: Return the [list](https://pynative.com/python-lists/) of files and directories in a specified directory path.
* **os.walk('dir\_path')**: Recursively get the list of all files in a directory and subdirectories.
* **os.scandir('path')**: Returns directory entries along with file attribute information.
* **glob.glob('pattern')**: glob module to list files and folders whose names follow a specific pattern.

#### **List only files in a directory**

import os

# directory/folder path

dir\_path = r'E:\account'

# list to store files

res = []

# Iterate directory

for file\_path in os.listdir(dir\_path):

# check if current file\_path is a file

if os.path.isfile(os.path.join(dir\_path, file\_path)):

# add filename to list

res.append(file\_path)

print(res)

#### List both files and directories.

import os

# folder path

dir\_path = r'E:\\account\\'

# list file and directories

res = os.listdir(dir\_path)

print(res)

## **os.walk() to list all files in a directory and subdirectories**

from os import walk

# folder path

dir\_path = r'E:\\account\\'

# list to store files name

res = []

for (dir\_path, dir\_names, file\_names) in walk(dir\_path):

res.extend(file\_names)

print(res)

File handling is an important part of any web application.

To

Python has several functions for creating, reading, updating, and deleting files.

The key function for working with files in Python is the **open()** function. The open() function takes two parameters; filename, and mode.

There are four different methods (modes) for opening a file:

"r" - Read - Default value. Opens a file for reading, error if the file does not exist "a" - Append - Opens a file for appending, creates the file if it does not exist

"w" - Write - Opens a file for writing, creates the file if it does not exist "x" - Create - Creates the specified file, returns an error if the file exists

In addition you can specify if the file should be handled as binary or text mode "t" - Text - Default value. Text mode

"b" - Binary - Binary mode (e.g. images)

Syntax

To open a file for reading it is enough to specify the name of the file:

f = open("demofile.txt")

The code above is the same as:

f = open("demofile.txt", "rt")

Because "r" for read, and "t" for text are the default values, you do not need to specify them.

Example

# a file named "RNSIT", will be opened with the reading mode. file **=** open('RNSIT.txt', 'r')

# This will print every line one by one in the file

**for** each **in** file: print (each)

Example 2: In this example, we will extract a string that contains all characters in the file then we can use file.read().

# Python code to illustrate read () mode file = open ("RNSIT.txt", "r")

print (file.read())

Example 3: In this example, we will see how we can read a file using the with statement.

# Python code to illustrate with() with open("Example.txt") as file:

data = file.read() print(data)

**Example 4:** Another way to read a file is to call a certain number of characters like in the following code the interpreter will read the first five characters of stored data and return it as a string:

# Python code to illustrate read() mode character wise file **=** open("example.txt", "r")

print (file.read(5))

**Example 5:**

We can also split lines while reading files in Python. The split() function splits the variable when space is encountered. You can also split using any characters as you wish.

# Python code to illustrate split() function with open("example.txt", "r") as file:

data **=** file.readlines()

**for** line **in** data:

word **=** line.split() print (word)

OS module in Python provides functions for interacting with the operating

system. OS comes under Python’s standard utility modules. This module provides a portable way of using operating system dependent functionality.

os.path module is sub module of OS module in Python used for common path name manipulation.

os.path.isfile() method in Python is used to check whether the specified path is an existing regular file or not.

Syntax: os.path.isfile(path) Parameter:

path: A path-like object representing a file system path. A path-like object is either a string or bytes object representing a path.

Python program to explain os.getcwd() method # importing os module

**import** os

# Get the current working # directory (CWD)

cwd **=** os.getcwd()

# Print the current working # directory (CWD)

print("Current working directory:", cwd)

# PYTHON- REGULAR EXPRESSIONS

**What is a Regular Expression**

A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern. RegEx can be used to check if a string contains the specified search pattern.

# Regular Expression Module

Python has a built-in package called re, which can be used to work with Regular Expressions. Import the re module:

import re

RegEx Functions

# The re module offers a set of functions that allows us to search a string for a match:

Function Description

findall Returns a list containing all matches

search Returns a Match object if there is a match anywhere in the string split Returns a list where the string has been split at each match

sub Replaces one or many matches with a string

# Metacharacters

Metacharacters are characters with a special meaning:

Character Description Example

[] A set of characters "[a-m]"

\ Signals a special sequence (can also be used to escape special characters) "\d"

. Any character (except newline character) "he..o"

|  |  |  |  |
| --- | --- | --- | --- |
| ^ | Starts with | "^hello" | |
| $ | Ends with | "planet$" | |
| \* | Zero or more occurrences | | "he.\*o" |
| + | One or more occurrences | | "he.+o" |
| ? | Zero or one occurrences | | "he.?o" |

{} Exactly the specified number of occurrences "he.{2}o"

| Either or "falls|stays"

() Capture and group

# Special Sequences

A special sequence is a \ followed by one of the characters in the list below, and has a special meaning:

Character Description

\A Returns a match if the specified characters are at the beginning of the string

\b Returns a match where the specified characters are at the beginning or at the end of a word

\B Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word

\d Returns a match where the string contains digits (numbers from 0-9)

\D Returns a match where the string DOES NOT contain digits

\s Returns a match where the string contains a white space character

\S Returns a match where the string DOES NOT contain a white space character

\w Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore \_ character)

\W Returns a match where the string DOES NOT contain any word characters

\Z Returns a match if the specified characters are at the end of the string

# Sets

A set is a set of characters inside a pair of square brackets [] with a special meaning:

Set Description

[arn] Returns a match where one of the specified characters (a, r, or n) is present [a-n] Returns a match for any lower case character, alphabetically between a and n [^arn] Returns a match for any character EXCEPT a, r, and n

[0123] Returns a match where any of the specified digits (0, 1, 2, or 3) are present [0-9] Returns a match for any digit between 0 and 9

[0-5][0-9] Returns a match for any two-digit numbers from 00 and 59

[a-zA-Z] Returns a match for any character alphabetically between a and z, lower case OR upper case

[+] In sets, +, \*, ., |, (), $,{} has no special meaning, so [+] means: return a match for any

+ character in the string

# Examples

The findall() Function

The findall() function returns a list containing all matches

1. import re

txt = "The rain in Spain" x = re.findall("ai", txt) print(x)

1. import re

txt = "The rain in Spain"

x = re.findall("Portugal", txt) print(x)

The search() Function

The search() function searches the string for a match, and returns a Match object if there is a match.

If there is more than one match, only the first occurrence of the match will be returned: Example

Search for the first white-space character in the string:

1. import re

txt = "The rain in Spain" x = re.search("\s", txt)

print("The first white-space character is located in position:", x.start())

If no matches are found, the value None is returned:

1. import re

txt = "The rain in Spain"

x = re.search("Portugal", txt)

# print(x)

**The split() Function**

The split() function returns a list where the string has been split at each match:

Example

Split at each white-space character:

1. import re

txt = "The rain in Spain" x = re.split("\s", txt) print(x)

Example

Split the string only at the first occurrence:

1. import re

txt = "The rain in Spain" x = re.split("\s", txt, 1) print(x)

# The sub() Function

The sub() function replaces the matches with the text of your choice:

Example

Replace every white-space character with the number 9:

1. import re

txt = "The rain in Spain" x = re.sub("\s", "9", txt) print(x)

Example

Replace the first 2 occurrences:

1. import re

txt = "The rain in Spain" x = re.sub("\s", "9", txt, 2) print(x)

**5 a) Write a function called isphonenumber () to recognize a pattern 415-555-4242without using regular expression and also write the code to recognize the same pattern using regular expression.**

def isphoneNo(num):

if(len(num)==12

and num[0:3].isdigit()

and num[4:7].isdigit()

and num[9:12].isdigit()

and num[3]=='-'and num[7]=='-'):

print("Valid Phone Number")

else:

print("Invalid Phone Number")

def checkphoneNo(num):

pattern=('\d{3}-\d{3}-\d{4}')

result=re.match(pattern,num)

if result:

print("Valid Phone Number")

else:

print("Invalid Phone Number")

n=input("Enter Number\n")

print("Without Using Regular Expression")

isphoneNo(n)

print("Using Regular Expression")

res1=checkphoneNo(n)

**output**

Enter Number

159-897-6596

Without Using Regular Expression

Valid Phone Number

Using Regular Expression

Valid Phone Number

**5b) Develop a python program that could search the text in a file for phone numbers(+919900889977) and email addresses (sample@gmail.com)**

import re

phone\_pat=re.compile(r'\+\d{12}$')

email\_pat=re.compile(r'[0-9a-zA-Z.\_]+@gmail.com')

f=open("C://Users//deepa//OneDrive//Desktop//ex.txt",'r')

for line in f:

matches=phone\_pat.findall(line)

for match in matches:

print(match)

matches=email\_pat.findall(line)

for match in matches:

print(match)

**output**

+919743303226

hh@gmail.com

**6 a) Write a python program to accept a file name from the user and perform the following operations**

**1. Display the first N line of the file**

**2. Find the frequency of occurrence of the word accepted from the user in the file**

fname = input("Enter the filename : ")

infile = open(fname, "r")

line= int(input("Enter the first N line "))

for x in range(line):

a = infile.readline()

print(x+1,":",a)

infile.seek(0)

word = input("Enter a word : ")

cnt = 0

for line in infile:

r=line.split()

cnt += r.count(word)

print("The word", word, "appears", cnt, "times in the file")

**Output**

Enter the filename : C://Users//deepa//OneDrive//Desktop//ex.txt

Enter the first N line 5

1 : hellooo

2 : +919743303225

3 : hh@gmail.com

4 : ghhh@yahho.com

5 : 975-859-7895

Enter a word : hellooo

The word hellooo appears 3 times in the file

**6 b. Python program to create a ZIP file of a particular folder which contains several files inside it.**

import os **# provides functions for interacting with the operating system**

import sys

import pathlib **# classes representing filesystem paths**

import zipfile **# provides tools to create, read, write, append, and list a ZIP file**

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

**# check whether entered directory exist or not**

if not os.path.isdir(dirName):

print("Directory", dirName, "doesn't exists")

sys.exit(0)

curDirectory = pathlib.Path(dirName)

**# open the new zip folder write all the files from the given directory**

with zipfile.ZipFile("ex.zip","w") as f:

**# rglob (“\*”) is a function that's used to search for files that match a specific file pattern or name**

for file\_path in curDirectory.rglob("\*"):

f.write(file\_path, arcname=file\_path.relative\_to(curDirectory))

print(file\_path)

**# to check entered directory is zipped or not**

if os.path.isfile("ex.zip"):

print(“created successfully")

else:

print("Error in creating zip archive")

**output-1**

Enter Directory name that you want to backup : C:\Users\deepa\OneDrive\Desktop\student.xlsx

Directory C:\Users\deepa\OneDrive\Desktop\student.xlsx doesn't exists

An exception has occurred, use %tb to see the full traceback.

**SystemExit:** 0

**Output -2**

Enter Directory name that you want to backup : C:\Users\deepa\OneDrive\Desktop\RNS

C:\Users\deepa\OneDrive\Desktop\RNS\output.pdf

created zip folder successfully

**Python Classes/Objects**

The class is a user-defined data structure that binds the data members and methods into a single unit. Class is a blueprint or code template for object creation

**Some points on Python class:**

* Classes are created by keyword class.
* Attributes are the variables that belong to a class.
* Attributes are always public and can be accessed using the dot (.) operator.

Eg.: My class.Myattribute

**Object**: An **object is an instance of a class**. It is a collection of attributes (variables) and methods

**Create a Class and object in Python**

class is defined by using the class keyword. The syntax to create a class is given below

**Syntax:**Class Definition

class ClassName:

# Statement

**Syntax:**Object Definition

obj = ClassName()

print(obj.atrr)

**Example**

# a class

**class** Dog:

    # A sample attribute

    attr1 **=** "mammal"

    attr2 **=** "dog"

    # A sample method

**def** fun(self):

**print**("I'm a", self.attr1)

        print("I'm a", self.attr2)

 # Object instantiation

Rodger **=** Dog()

# Accessing class attributes and method through objects

print(Rodger.attr1)

Rodger.fun()

**Output:**

mammal

I'm a mammal

I'm a dog

**\_\_init\_\_()**

Constructors are generally used for instantiating an object. The task of constructors is to initialize(assign values) to the data members of the class when an object of the class is created. In Python the \_\_init\_\_() method is called the constructor and is always called when an object is created.

**Syntax of constructor declaration :**

def \_\_init\_\_(self):

# body of the constructor

### **Self Parameter**

When we call a method of this object as myobject.method(arg1, arg2), this is automatically converted by Python into MyClass.method(myobject, arg1, arg2) .

The [Self](https://www.geeksforgeeks.org/self-in-python-class/) Parameter does not call it to be Self, You can use any other name instead of it. Here we change the self to the word someone and the output will be the same

**Types of constructors :**

* **default constructor:** The default constructor is a simple constructor which doesn’t accept any arguments

# a class with default constructor

class Dog:

# A sample attribute

def \_\_init\_\_(self):

self.attr1 ="Mamal"

self.attr2 = "Cat"

# A sample method

def fun(self):

print("I'm a", self.attr1)

print("I'm a", self.attr2)

# Object instantiation

Rodger = Dog()

# Accessing class attributes and method through objects

Rodger.fun()

* **parameterized constructor:** constructor with parameters is known as parameterized constructor. The parameterized constructor takes its first argument as a reference to the instance being constructed known as self and the rest of the arguments are provided by the programmer.

# a class with parameterized constructor

class Dog:

# A sample attributes

def \_\_init\_\_(self, attr1,attr2):

self.attr1 = attr1

self.attr2 = attr2

# A sample method

def fun(self):

print("I'm a", self.attr1)

print("I'm a", self.attr2)

# Object instantiation

Rodger = Dog("mamal","dog")

# Accessing class attributes and method through objects

Rodger.fun()

# Inheritance in Python

It is a mechanism that allows you to create a hierarchy of classes that share a set of properties and methods by deriving a class from another class. Inheritance is the capability of one class to derive or inherit the properties from another class.

## Python Inheritance Syntax

Class BaseClass:

{Body}

Class DerivedClass(BaseClass):

{Body}

### **Creating a Parent Class**

class A:

def feature1(self):

print("Feature 1")

def feature2(self):

print("Feature 2")

### **Creating a Child Class**

To create a class that inherits the functionality from another class, send the parent class as a parameter when creating the child class:

class B(A):

def feature3(self):

print("Feature 3")

def feature4(self):

print("Feature 4")

Example

class A:

def feature1(self):

print("Feature 1")

def feature2(self):

print("Feature 2")

class B(A):

def feature3(self):

print("Feature 3")

def feature4(self):

print("Feature 4")

a=A()

a.feature1()

a.feature2()

b=B()

b.feature1()

b.feature2()

b.feature3()

b.feature3()

**What is Polymorphism:** The word polymorphism means having many forms. In programming, polymorphism means the same function name (but different signatures) being used for different types. The key difference is the data types and number of arguments used in function.

**Example of inbuilt polymorphic functions:**

Python program to demonstrate in-built polymorphic functions

 # len() being used for a string

print(len("xxyz"))

# len() being used for a list

**print**(len([10, 20, 30]))

output:

4

3

**Examples of user-defined polymorphic functions:**

# A simple Python function to demonstrate Polymorphism

**def** add(x, y, z **=** 0):

**return** x **+** y**+**z

# Driver code

**print**(add(7, 3))

print(add(6, 3, 4))

**output:**

**10**

**13**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**7A. Program to find the area of triangle, circle and rectangle using concept of inheritance.**

import math

class Shape:

def \_\_init\_\_(self):

self.area = 0

self.name = ""

def showArea(self):

print("The area of the", self.name, "is", self.area, "units")

class Circle(Shape):

def \_\_init\_\_(self,radius):

self.area = 0

self.name = "Circle"

self.radius = radius

def calcArea(self):

self.area = math.pi \* self.radius \* self.radius

class Rectangle(Shape):

def \_\_init\_\_(self,length,breadth):

self.area = 0

self.name = "Rectangle"

self.length = length

self.breadth = breadth

def calcArea(self):

self.area = self.length \* self.breadth

class Triangle(Shape):

def \_\_init\_\_(self,base,height):

self.area = 0

self.name = "Triangle"

self.base = base

self.height = height

def calcArea(self):

self.area = self.base \* self.height / 2

c1 = Circle(5)

c1.calcArea()

c1.showArea()

r1 = Rectangle(5, 4)

r1.calcArea()

r1.showArea()

t1 = Triangle(3, 4)

t1.calcArea()

t1.showArea()

**OUTPUT**

The area of the Circle is 78.53981633974483 units

The area of the Rectangle is 20 units

The area of the Triangle is 6.0 units

**Or**

class Shape:

def \_\_init\_\_(self,a,b,constant):

print(a\*b\*constant)

class triangle(Shape):

def \_\_init\_\_(self,b,h):

super().\_\_init\_\_(b,h,0.5)

class circle(Shape):

def \_\_init\_\_(self,r):

super().\_\_init\_\_(r,r,3.1415)

class rect(Shape):

def \_\_init\_\_(self,b,h):

super().\_\_init\_\_(b,h,1)

print("Area of Triangle")

triangle(20,10)

print("Area of Circle")

circle(5)

print("Area of Rectangle")

rect(4,5)

**7b. Program to store and update details of employees belonging to given department**

class Employee:

def \_\_init\_\_(self,Ename,Eid,Edept,Esal):

self.name=Ename

self.id=Eid

self.dept=Edept

self.sal=Esal

def display(self):

print(self.name, self.id,self.dept,self.sal)

def update\_sal(self,dept,sal):

if(self.dept==dept):

self.sal=sal

Emp=[]

n=int(input("enter the number of employee"))

for i in range(n):

name= input("Enter Employee name : ")

eid = input("Enter Employee ID : ")

dept = input("Enter Employee Dept : ")

sal= int(input("Enter Employee Salary : "))

emp1=Employee(name,eid,dept,sal)

Emp.append(emp1)

print("Employee Details are")

for i in range(n):

print(Emp[i].name," ",Emp[i].sal)

print("Update salary Particular Department")

dep=input("enter the department of the salary to be updated")

salary=int(input("Enter the salary to be upadated"))

for i in range(n):

Emp[i].update\_sal(dep,salary)

print(Emp[i].name," ",Emp[i].sal)

**OUTPUT**

enter the number of employee2

Enter Employee name : dd

Enter Employee ID : 3

Enter Employee Dept : ise

Enter Employee Salary : 4000

Enter Employee name : rr

Enter Employee ID : 6

Enter Employee Dept : cse

Enter Employee Salary : 56

Employee Details are

dd 4000

rr 56

Update salary Particular Department

enter the department of the salary to be updatedise

Enter the salary to be upadated590

dd 590

rr 56

**8. Program to find whether the given input is palindrome or not (for both string and integer)** **using the concept of polymorphism and inheritance.**

class PaliStr:

def \_\_init\_\_(self):

self.isPali = False

def chkPalindrome(self, myStr):

if myStr == myStr[::-1]:

self.isPali = True

else:

self.isPali = False

return self.isPali

class PaliInt(PaliStr):

def \_\_init\_\_(self):

self.isPali = False

def chkPalindrome(self, val):

temp = val

rev = 0

while temp != 0:

dig = temp % 10

rev = (rev\*10) + dig

temp = temp //10

if val == rev:

self.isPali = True

else:

self.isPali = False

return self.isPali

st = input("Enter a string : ")

stObj = PaliStr()

if stObj.chkPalindrome(st):

print("Given string is a Palindrome")

else:

print("Given string is not a Palindrome")

val = int(input("Enter a integer : "))

intObj = PaliInt()

if intObj.chkPalindrome(val):

print("Given integer is a Palindrome")

else:

print("Given integer is not a Palindrome")

**OUTPUT**

Enter a string : nnnnnn

Given string is a Palindrome

Enter a integer : 12321

Given integer is a Palindrome

**Or**

class A:

def Pal(self, val:str):

if(val!=val[::-1]):

print("Not palindrome")

else:

print("Is Palindrome")

class B(A):

def Pal(self, val:int):

super().Pal(str(val))

A().Pal(input("enter String"))

B().Pal(int(input("enter Number")))

**Downloading files from web using Python**

[Python](https://www.tutorialspoint.com/python/index.htm) provides different modules like [urllib](https://www.tutorialspoint.com/url-handling-python-modules-urllib), [requests](https://www.tutorialspoint.com/python_network_programming/python_http_requests.htm) etc to download files from the web

raise\_for\_status() Ensures that program halt if bad download occurs without crashing. Always call **raise\_for\_status()** after calling requests.get()

### The iter\_content() method returns “chunks” of the content on each iteration through the loop. Each chunk is of the bytes data type, and you get to specify how many bytes each chunk will contain. One hundred thousand bytes is generally a good size, so pass 100000 as the argument to iter\_content()

To review, here’s the complete process for downloading and saving a file:

1. Call requests.get() to download the file.
2. Call open() with 'wb' to create a new file in write binary mode.
3. Loop over the Response object’s iter\_content() method.
4. Call write() on each iteration to write the content to the file.
5. Call close() to close the file.

### **1. Import module**

**import requests**

### The requests module lets you easily download files from the web without having to worry about complicated issues such as network errors, connection problems, and data compression

### **2. Get the link or url**

url = 'https://www.facebook.com/favicon.ico'

r = requests.get(url, allow\_redirects=True)

### **3. Save the content with name.**

open('facebook.ico', 'wb').write(r.content)

save the file as facebook.ico.

**BeautifulSoup**

BeautifulSoup is a Python library that is used to pull data of HTML and XML files. It is mainly designed for web scrapping. It works with the parser to provide a natural way of navigating, searching, and modifying the parse tree

**Example**

import requests

res = requests.get('https://automatetheboringstuff.com/files/rj.txt')

res.raise\_for\_status()

playFile = open('RomeoAndJuliet.txt', 'wb')

for chunk in res.iter\_content(100000):

playFile.write(chunk)

**9 a. Write a python program to download the all XKCD comics**

import requests

import os

from bs4 import BeautifulSoup

**# Set the URL of the first XKCD comic**

url = 'https://xkcd.com/1/'

**# Create a folder to store the comics**

if not os.path.exists('xkcd\_comics'):

os.makedirs('xkcd\_comics')

**# Loop through all the comics**

while True:

**# Download the page content**

res = requests.get(url)

res.raise\_for\_status()

**# Parse the page content using BeautifulSoup**

soup = BeautifulSoup(res.text, 'html.parser')

**# Find the URL of the comic image**

comic\_elem = soup.select('#comic img')

if comic\_elem == []:

print('Could not find comic image.')

else:

comic\_url = 'https:' + comic\_elem[0].get('src')

**# Download the comic image**

print(f'Downloading {comic\_url}...')

res = requests.get(comic\_url)

res.raise\_for\_status()

**# Save the comic image to the xkcd\_comics folder**

image\_file = open(os.path.join('xkcd\_comics', os.path.basename(comic\_url)),'wb')

for chunk in res.iter\_content(100000):

image\_file.write(chunk)

image\_file.close()

**# Get the URL of the previous comic**

prev\_link = soup.select('a[rel="prev"]')[0]

if not prev\_link:

break

url = 'https://xkcd.com' + prev\_link.get('href')

print('All comics downloaded.')

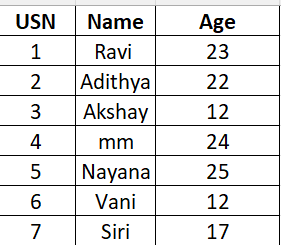
**Working with Excel Spreadsheets in Python**

First, let’s go over some basic definitions: an Excel spreadsheet document is called a workbook. A single workbook is saved in a file with the .xlsx extension. Each **workbook** can contain multiple sheets (also called **worksheets**). The sheet the user is currently viewing (or last viewed before closing Excel) is called the active sheet. Each sheet has columns (addressed by letters starting at A) and rows (addressed by numbers starting at 1). A box at a particular column and row is called a cell. Each cell can contain a number or text value. The grid of cells with data makes up a sheet.

**Openpyxl**is a Python library that provides various methods to interact with Excel Files using Python. It allows operations like reading, writing, arithmetic operations, plotting graphs, etc.

Getting Sheets from the Workbook Each sheet is represented by a **Worksheet object**, which you can obtain by using the square brackets with the sheet name string like a dictionary key. Finally, you can use the active attribute of a Workbook object to get the **workbook’s active sheet**. The active sheet is the sheet that’s on top when the workbook is opened in Excel. Once you have the Worksheet object, you can get its name from the title attribute.

***Example.xlsx***



**Reading from Spreadsheets**

To read an Excel file you have to open the spreadsheet using the **load\_workbook()** method.

**active**to select the first sheet available

the **cell** attribute to select the cell by passing the row and column parameter. The **value** attribute prints the value of the particular cell.

**Note:**The first row or column integer is 1, not 0.

import openpyxl

path = "demo.xlsx"

wb\_obj = openpyxl.load\_workbook(path)

#sheet\_obj = wb\_obj.active

cell\_obj = sheet\_obj.cell(row = 1, column = 2)

print(cell\_obj.value)

**output**

Name

### **Reading from Multiple Cells**

There can be two ways of reading from multiple cells.

***Method 1:*** We can get the count of the total rows and columns using the **max\_row** and **max\_column** respectively. We can use these values inside the for loop to get the value of the desired row or column or any cell depending upon the situation. Let’s see how to get the value of the first column and first row.

import openpyxl

path = "demo.xlsx"

wb\_obj = openpyxl.load\_workbook(path)

sheet\_obj = wb\_obj.active

row = sheet\_obj.max\_row

column = sheet\_obj.max\_column

print("Total Rows:", row)

print("Total Columns:", column)

print("\nValue of first column")

for i in range(1, row + 1):

cell\_obj = sheet\_obj.cell(row = i, column = 1)

print(cell\_obj.value)

print("\nValue of first row")

for i in range(1, column + 1):

cell\_obj = sheet\_obj.cell(row = 2, column = i)

print(cell\_obj.value, end = " "))

**output**

Total Rows: 8

Total Columns: 3

Value of first column

USN

1

2

3

4

5

6

7

Value of first row

1 Ravi 23

**Method 2:**We can also read from multiple cells using the cell name. This can be seen as the list slicing of Python.

import openpyxl

path = "demo.xlsx"

wb\_obj = openpyxl.load\_workbook(path)

sheet\_obj = wb\_obj.active

cell\_obj = sheet\_obj['A1': 'c4']

for cell1, cell2,cell3, cell4 in cell\_obj:

print(cell1.value, cell2.value, cell3.value, cell4.value)

**Output**

USN Name Age

1 Ravi 23

2 Adithya 22

3 Akshay 12

### **Writing to Spreadsheets**

First, let’s create a new spreadsheet, and then we will write some data to the newly created file. An empty spreadsheet can be created using the **Workbook()**method

from openpyxl import Workbook

wb=Workbook()

sheet=wb.active

sheet['A1']=200

sheet['A2']=100

sheet['A3']=300

sheet['A4']=400

sheet['A5']=500

sheet['A6']='=sum(A1:A5)'

wb.save("ex.xlsx")

### **Insert data or to write to an Excel sheet**

Call the **openpyxl.Workbook()** function to create a new, blank Workbook object. The workbook will start off with a single sheet named Sheet. You can change the name of the sheet by storing a new string in its title attribute. Any time you modify the Workbook object or its sheets and cells, the spreadsheet file will not be saved until you call the **save()** workbook method.

import openpyxl

my\_wb = openpyxl.Workbook()

my\_sheet = my\_wb.active

c1 = my\_sheet.cell(row = 1, column = 1)

c1.value = "Aadrika"

c2 = my\_sheet.cell(row= 1 , column = 2)

c2.value = "Adwaita"

c3 = my\_sheet['A2']

c3.value = "Satyajit"

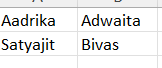
# B2 = column = 2 & row = 2.

c4 = my\_sheet['B2']

c4.value = "Bivas"

my\_wb.save("C:\\Users\\deepa\\OneDrive\\Desktop\\e.xlsx")

**output**

****

### **To add Sheets in the Workbook**

import openpyxl

my\_wb = openpyxl.Workbook()

my\_sheet = my\_wb.active

my\_wb.create\_sheet(index = 1 , title = "new sheet")

my\_wb.save("C:\Users\TP\Desktop\Book1.xlsx")

### **Display Total number of rows.**

import openpyxl

my\_path = "C:\Users\TP\Desktop\Book1.xlsx"

my\_wb\_obj = openpyxl.load\_workbook(my\_path)

my\_sheet\_obj = my\_wb\_obj.active

print(my\_sheet\_obj.max\_row)

### **Display total number of columns**

import openpyxl

**# Give the location of the file**

My\_path = "C:\Users\TP\Desktop\Book1.xlsx"

My\_wb\_obj = openpyxl.load\_workbook(path)

my\_sheet\_obj = my\_wb\_obj.active

print(sheet\_obj.max\_column)

### **Display all columns name**

import openpyxl

**# Give the location of the file**

my\_path = "C:\Users\TP\Desktop\Book1.xlsx"

**# workbook object is created**

my\_wb\_obj = openpyxl.load\_workbook(my\_path)

my\_sheet\_obj = my\_wb\_obj.active

my\_max\_col = my\_sheet\_obj.max\_column

for i in range(1, my\_max\_col + 1):

my\_cell\_obj = my\_sheet\_obj.cell(row = 1, column = i)

print(my\_cell\_obj.value)

### **Workbooks, Sheets, Cells**

As a quick review, here’s a rundown of all the functions, methods, and data types involved in reading a cell out of a spreadsheet file:

1. Import the openpyxl module.
2. Call the openpyxl.load\_workbook() function.
3. Get a Workbook object.
4. Use the active or sheetnames attributes.
5. Get a Worksheet object.
6. Use indexing or the cell() sheet method with row and column keyword arguments.
7. Get a Cell object.
8. Read the Cell object’s value attribute.

**9.b Demonstrate python program to read the data from the spreadsheet and write the data in to the spreadsheet**

from openpyxl import Workbook

from openpyxl.styles import Font

**# Writing Data to workbook**

wb = Workbook() **# Create a Workbook**

sheet = wb.active # **active the created workbook**

sheet.title = "Language" **# set name for created sheet**

wb.create\_sheet(title = "Capital") **# Create sheet and set name**

lang = ["Kannada", "Telugu", "Tamil"]

state = ["Karnataka", "Telangana", "Tamil Nadu"]

capital = ["Bengaluru", "Hyderabad", "Chennai"]

code =['KA', 'TS', 'TN']

**#header of the Excel**

sheet.cell(row = 1, column = 1).value = "State"

sheet.cell(row = 1, column = 2).value = "Language"

sheet.cell(row = 1, column = 3).value = "Code"

**# make bold header**

ft = Font(bold=True)

for row in sheet["A1:C1"]:

for cell in row:

cell.font = ft

for i in range(2,5):

sheet.cell(row = i, column = 1).value = state[i-2]

sheet.cell(row = i, column = 2).value = lang[i-2]

sheet.cell(row = i, column = 3).value = code[i-2]

wb.save("C:\\Users\\deepa\\OneDrive\\Desktop\\demo.xlsx")

srchCode = input("Enter state code for finding language ")

for i in range(2,5):

data = sheet.cell(row = i, column = 3).value

if data == srchCode:

print("Corresponding language for code", srchCode, "is", sheet.cell(row = i, column = 2).value)

wb.close()

# Working with PDF files in Python

PDF stands for Portable Document Format and uses the .pdf file extension. The module to work with PDFs is **PyPDF2**. To install it, run pip install PyPDF2 from the command line.

[PyPDF2](https://pypi.python.org/pypi/PyPDF2/1.26.0) is a python library built as a PDF toolkit

* Extracting document information (title, author, …)
* Splitting documents page by page
* Merging documents page by page
* Cropping pages
* Merging multiple pages into a single page
* Encrypting and decrypting PDF files

import PyPDF2

pdfObj=open("D:\\hi.pdf", 'rb') **# open pdf in read binary mode**

pdfReader=PyPDF2.PdfReader(pdfObj)

len(pdfReader.pages) **# get number of pages**

pageObj=pdfReader.pages[0] **# get the first page**

pageObj.extract\_text() **# extract the text content of the page**

Creating PDFs PyPDF2’s counterpart to **PdfReader** is **PdfWriter**, which can create **new PDF** files.

But **PyPDF2 cannot write** arbitrary text to a PDF like Python can do with plaintext files. Instead,

PyPDF2’s PDF-writing capabilities are limited to **copying pages** from other PDFs, **rotating pages,**

**overlaying pages, and encrypting files**.

PyPDF2 doesn’t allow you to directly edit a PDF. Instead,

you have to **create a new PDF** and then copy content over from an existing document.

The examples in this section will follow this general approach:

1. Open one or more existing PDFs (the source PDFs) into PdfReader objects.

2. Create a new **PdfWriter** object.

3. Copy pages from the **PdfReader** objects into the PdfWriter object.

4. Finally, use the PdfWriter object to write the output PDF.

Creating a **PdfWriter** object creates only a value that represents a PDF document in Python. It doesn’t create the actual PDF file. For that, you must call the PdfWriter’s **write()** method. The write() method

takes a regular File object that has been opened in write-binary mode. You can get such a File object

by calling Python’s open() function with two arguments: the string of what you want the **PDF’s**

**filename** to be and 'wb' to indicate the file should be opened in write-binary mode.

**Copying Pages**

You can use PyPDF2 to copy pages from one PDF document to another. This allows you to combine multiple PDF files, cut unwanted pages, or reorder pages.

import PyPDF2

pdfFile1=open("D:\\Even Sem 2023\\f2.pdf",'rb')

pdfFile2=open("D:\\Even Sem 2023\\f1.pdf", 'rb')

pdfReader1=PyPDF2.PdfReader(pdfFile1)

pdfReader2=PyPDF2.PdfReader(pdfFile2)

pdfWriter=PyPDF2.PdfWriter() #represents a blank pdf document

**#copy all the pages from the two source PDFs and add them to the PdfWriter object.**

for pageNum in range(len(pdfReader1.pages)):

pageObj=pdfReader1.pages[pageNum]

pdfWriter.add\_page(pageObj)

for pageNum in range(len(pdfReader2.pages)):

pageObj=pdfReader2.pages[pageNum]

pdfWriter.add\_page(pageObj)

**#write a new PDF called combinedminutes.pdf by passing a File object to the PdfWriter’s write() method**

pdfOut=open("D:\\Even Sem 2023\\combinedminutes.pdf", 'wb')

pdfWriter.write(pdfOut)

pdfOut.close()

pdfFile1.close()

pdfFile2.close()

PyPDF2 cannot insert pages in the middle of a **PdfFileWriter** object; the **add\_Page()** method will

only add pages to the end.

We have now created a **new PDF file** that **combines** the pages from file1.pdf and file2.pdf into a single document. Remember that the File object passed to **PyPDF2.PdfReader()** needs to be opened in read-binary mode by passing **'rb'** as the second argument to open(). Likewise, the File object passed to **PyPDF2.PdfWriter()** needs to be opened in write-binary mode with '**wb'**.

***10a) Write a python program to combine select pages from many PDFs***

from PyPDF2 import PdfWriter, PdfReader

num1 = int(input("Enter page number from file1 "))

num2 = int(input("Enter page number from file2 "))

pdf1 = open("D:\\Even Sem 2023\\ex1.pdf ", 'rb')

pdf2 = open("D:\\Even Sem 2023\\ex1.pdf ", 'rb')

pdf\_writer = PdfWriter()

pdf1\_reader = PdfReader(pdf1)

page = pdf1\_reader.pages[num1 - 1]

pdf\_writer.add\_page(page)

pdf2\_reader = PdfReader(pdf2)

page = pdf2\_reader.pages[num2 - 1]

pdf\_writer.add\_page(page)

with open('D:\\ Even Sem 2023\\output.pdf', 'wb') as output:

pdf\_writer.write(output)

print("Combined successfully")

**output**

Enter page number from file1 you want combine from multiple documents 4

Enter page number from file2 you want combine from multiple documents 5

Combined successfully

**WORKING WITH JSON DATA**

**JSON** (JavaScript Object Notation) is the most widely used data format for data interchange

on the web

**JSON supports mainly 6 data types:**

1. String
2. Number
3. Boolean
4. Null
5. Object
6. Array

JSON (pronounced “JAY-sawn” or “Jason”) is a format that stores information as JavaScript source code in plaintext files. **(JSON is short for JavaScript Object Notation.)**

**JSON Module**

Python’s json module handles all the details of translating between a string with **JSON data** **and**

**Python values** for the **json.loads() and json.dumps()** functions. JSON can’t store every kind of

Python value. It can contain values of only the following data types: strings, integers, floats, Booleans, lists, dictionaries, and None Type. JSON cannot represent Python-specific objects, such as File objects, CSV reader or writer objects, Regex objects, or Selenium WebElement objects.

StrofJsonData='{"name":"Virat", "isCaptain":true, "MatchPlayed":120, "CatchDropped":null}'

import json

jsonasPython=json.loads(StrofJsonData)

print(jsonasPython)

**output**

{'name': 'Virat', 'isCaptain': True, 'MatchPlayed': 120, 'CatchDropped': None}

After you import the json module, you can call **loads()** and pass it a string of **JSON** data. Note that JSON

strings always use double quotes. It will **return** that data as a Python dictionary.

**Writing JSON with the dumps() Function**

The **json.dumps()** function (which means “dump string,” not “dumps”) will translate a Python value into a string of JSON-formatted **data**.

convert from Python to JSON, Python objects are converted into the JSON (JavaScript) equivalent



10 b) Write a python program to fetch current weather data from the JSON file

import json

**# Load the JSON data from file**

with open("C:\\Desktop\\weather.json.txt") as f:

data = json.load(f)

**# Extract the required weather data**

current\_temp = data['main']['temp']

humidity = data['main']['humidity']

weather\_desc = data['weather'][0]['description']

**# Display the weather data**

print(f"Current temperature: {current\_temp}°C")

print(f"Humidity: {humidity}%")

print(f"Weather description: {weather\_desc}")

**output**

Current temperature: 15.45°C

Humidity: 64%

Weather description: clear sky

**JSON File :**

{

"coord": {

"lon": -73.99,

"lat": 40.73

},

"weather": [

{

"id": 800,

"main": "Clear",

"description": "clear sky",

"icon": "01d"

}

],

"base": "stations",

"main": {

"temp": 15.45,

"feels\_like": 12.74,

"temp\_min": 14.44,

"temp\_max": 16.11,

"pressure": 1017,

"humidity": 64

},

"visibility": 10000,

"wind": {

"speed": 4.63,

"deg": 180

},

"clouds": {

"all": 1

},

"dt": 1617979985,

"sys": {

"type": 1,

"id": 5141,

"country": "US",

"sunrise": 1617951158,

"sunset": 1618000213

},

"timezone": -14400,

"id": 5128581,

"name": "New York",

"cod": 200

}