1a). Students test marks for each course is considered as the best of two test average marks out of three test's marks, implement a python program to find the test average marks, take input from the user.

```
m1 = int(input("Enter the marks in the first test: "))
m2 = int(input("Enter the marks in the second test: "))
m3 = int(input("Enter the marks in the third test: "))

if m1 > m2:
    if m2 > m3:
        total = m1 + m2
    else:
        total = m1 + m3
else:
    total = m2 + m3

Avg = total / 2
print("The average of the best two test marks is:", Avg)
```

Output:

Enter the marks in the first test: 18

Enter the marks in second test: 19

Enter the marks in third test: 20

The average of the best two test marks is: 19.5

1b). Implement 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 = int(input("Enter a number:"))
digit = int(input("Enter a digit:"))
temp = num
rev = 0
count = 0

while num != 0:
    rem = num % 10
    rev = rev * 10 + rem
    if rem == digit:
        count += 1
    num = num // 10

if temp == rev:
    print("The number is a palindrome!")
else:
    print("The number isn't a palindrome!")
    print("{} occurred {} times in {}".format(digit, count, temp))
```

output:

Case1:
Enter number:2315132
Enter a Digit3
The number is a palindrome!
3 occurred 2 times in 2315132

Case2:
Enter number:1234356
Enter a Digit3
The number isn't a palindrome!
3 occurred 2 times in 1234356

2a) 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.

```
nterms = int(input("How many terms? "))
n1, n2 = 0, 1
count = 0

if nterms <= 0:
    print("Please enter a positive integer")
elif nterms == 1:
    print("Fibonacci sequence up to", nterms, ":")
    print(n1)
else:
    print("Fibonacci sequence:")
    while count < nterms:
        print(n1)
        nth = n1 + n2
        n1 = n2
        n2 = nth
        count += 1
```

output:

Case1:
How many terms? 5
Fibonacci sequence:
0
1
1
2
3

Case2:
How many terms? -3
Please enter a positive integer

2b) Demonstrate how to implement a python program to convert binary to decimal, octal to hexadecimal using functions.

Octal to Hexadecimal:

```
print("Enter the Octal Number:")
octnum = int(input())
chk = 0
i = 0
decnum = 0

while octnum != 0:
    rem = octnum % 10
    if rem > 7:
        chk = 1
        break
    decnum = decnum + (rem * (8**i))
    i = i + 1
    octnum = int(octnum / 10)

if chk == 0:
    i = 0
    hexdecnum = []
    while decnum != 0:
        rem = decnum % 16
        if rem < 10:
            rem = rem + 48
        else:
            rem = rem + 55
        rem = chr(rem)
        hexdecnum.insert(i, rem)
        i = i + 1
        decnum = int(decnum / 16)

    print("\nEquivalent Hexadecimal value is:")
    i = i - 1
    while i >= 0:
        print(end=hexdecnum[i])
        i = i - 1
    print()
else:
    print("\nInvalid Input:")
```

output:

Enter the Octal Number:
261

Equivalent Hexadecimal value is:
B1

Binary to Decimal

```
def BinaryToDecimal(binary):
    decimal, i = 0, 0
    while(binary != 0):
        dec = binary % 10
        decimal = decimal + dec * (2 ** i)  #
Use ** for exponentiation
        binary = binary // 10
        i += 1
    print(decimal)

BinaryToDecimal(100)
BinaryToDecimal(101)
BinaryToDecimal(1001)
```

Output:

4
5
9

3a) When an interpreter reads a line/ sentence from user, find the number of words, digits, uppercase letters
and lowercase letters in that sentence; demonstrate with the help of python programming

```
s = input("Enter a sentence: ")
w, d, u, l = 0, 0, 0, 0
l_w = s.split()
w = len(l_w)
for c in s:
    if c.isdigit():
        d = d + 1
    elif c.isupper():
        u = u + 1
    elif c.islower():
        l = l + 1
print("No of Words: ", w)
print("No of Digits: ", d)
print("No of Uppercase letters: ", u)
print("No of Lowercase letters: ", l)
```

output:

case1:
Enter a sentence: My name is Ram
No of Words: 4
No of Digits: 0
No of Uppercase letters: 2
No of Lowercase letters: 9

Case2:
Enter a sentence: I am 12 years old
No of Words: 5
No of Digits: 2
No of Uppercase letters: 1
No of Lowercase letters: 10

3b) Let us take two strings compare and find the string similarity between two given strings with the help of
python programming.

```
import difflib

def string_similarity(str1, str2):
    result = difflib.SequenceMatcher(a=str1.lower(), b=str2.lower())
    return result.ratio()

str1 = 'Python Exercises'
str2 = 'Python Exercises'
print("Original string:")
print(str1)
print(str2)
print("Similarity between two said strings:")
print(string_similarity(str1, str2))

str1 = 'Python Exercises'
str2 = 'Python Exercise'
print("\nOriginal string:")
print(str1)
print(str2)
print("Similarity between two said strings:")
print(string_similarity(str1, str2))

str1 = 'Python Exercises'
str2 = 'Python Ex.'
print("\nOriginal string:")
print(str1)
print(str2)
print("Similarity between two said strings:")
print(string_similarity(str1, str2))

str1 = 'Python Exercises'
str2 = 'Python'
print("\nOriginal string:")
print(str1)
print(str2)
print("Similarity between two said strings:")
print(string_similarity(str1, str2))

str1 = 'Python Exercises'
str2 = 'Java Exercises'
print("\nOriginal strings:")
print(str1)
print(str2)
print("Similarity between two said strings:")
print(string_similarity(str1, str2))
```

output:

Original string:
Python Exercises
Python Exercises
Similarity between two said strings:
1.0

Original string:
Python Exercises
Python Exercise
Similarity between two said strings:
0.967741935483871

Original string:
Python Exercises
Python Ex.
Similarity between two said strings:
0.6923076923076923

Original string:
Python Exercises
Python
Similarity between two said strings:
0.5454545454545454

Original string:
Java Exercises
Python
Similarity between two said strings:
0.0

4a) User enters a list of random numbers, the programmer need to arrange these random numbers in
ascending order with sorting techniques such as insertion sort and merge sort using lists in python.

Merge sort:

```
def merge(arr, l, m, r):
    n1 = m - l + 1
    n2 = r - m

    L = [0] * (n1)
    R = [0] * (n2)

    for i in range(0, n1):
        L[i] = arr[l + i]

    for j in range(0, n2):
        R[j] = arr[m + 1 + j]

    i = 0
    j = 0
    k = l

    while i < n1 and j < n2:
        if L[i] <= R[j]:
            arr[k] = L[i]
            i += 1
        else:
            arr[k] = R[j]
            j += 1
        k += 1

    while i < n1:
        arr[k] = L[i]
        i += 1
        k += 1

    while j < n2:
        arr[k] = R[j]
        j += 1
        k += 1

def mergeSort(arr, l, r):
    if l < r:
        m = l + (r - l) // 2
        mergeSort(arr, l, m)
        mergeSort(arr, m + 1, r)
        merge(arr, l, m, r)

arr = [12, 11, 13, 5, 6, 7]
n = len(arr)
```

```
print("Given array is:")
for i in range(n):
    print("%d" % arr[i], end=" ")

mergeSort(arr, 0, n - 1)

print("\n\nSorted array is:")
for i in range(n):
    print("%d" % arr[i], end=" ")
```

output:

Given array is
12 11 13 5 6 7

Sorted array is
5 6 7 11 12 13

Insertion sort:

```
def insertionSort(arr):
    if (n := len(arr)) <= 1:
        return
    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

arr = [12, 11, 13, 5, 6]
insertionSort(arr)
print(arr)
```

output:

Sorted array is:
[5, 6, 11, 12, 13]

4b). Demonstrate with a python program to convert roman numbers into integer values using dictionaries, by taking inputs from user.

```
class Solution(object):
    def romanToInt(self, s):
        roman = {'I': 1, 'V': 5, 'X': 10, 'L': 50,
'C': 100, 'D': 500, 'M': 1000, 'IV': 4, 'IX': 9,
'XL': 40, 'XC': 90, 'CD': 400, 'CM': 900}
        i = 0
        num = 0
        while i < len(s):
            if i + 1 < len(s) and s[i:i + 2] in
roman:
                num += roman[s[i:i + 2]]
                i += 2
            else:
                num += roman[s[i]]
                i += 1
        return num

ob1 = Solution()
print(ob1.romanToInt("III"))
print(ob1.romanToInt("CDXLIII"))
```

output:

3
443

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

Without regular expression:

```
def isPhoneNumber(text):
    if len(text) != 12:
        return False
    for i in range(0, 3):
        if not text[i].isdecimal():
            return False
    if text[3] != '-':
        return False
    for i in range(4, 7):
        if not text[i].isdecimal():
            return False
    if text[7] != '-':
        return False
    for i in range(8, 12):
        if not text[i].isdecimal():
            return False
    return True

print('Is 415-555-4242 a phone number?')
print(isPhoneNumber('415-555-4242'))
print('Is Moshi moshi a phone number?')
print(isPhoneNumber('Moshi moshi'))
```

output:
Is 415-555-4242 a phone number?
True
Is Moshi moshi a phone number?
False

With regular expression:

```
import re
phoneNumRegex = re.compile(r'\d\d\d-\d\d\d-\d\d\d\d')
mo = phoneNumRegex.search('My number is 415-555-4242.')
print('Phone number found: ' + mo.group())
```

output:

```
Phone number found: 415-555-4242
```

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_regex = re.compile(r'\+\d{12}')
email_regex = re.compile(r'[A-Za-z0-9._]+@[A-Za-z0-9]+\.[A-Za-z]{2,}')

with open('example.txt', 'r') as f:
    for line in f:
        matches = phone_regex.findall(line)
        for match in matches:
            print("Phone:", match)

        matches1 = email_regex.findall(line)
        for match1 in matches1:
            print("Email:", match1)
```

output:

6a) **Demonstrate how files are read in python by considering myfile.txt as an example file name which isentered by the user to 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**

1. Display the first N line of the file

```
  try:
    file_name = input("Enter the file name (e.g., myfile.txt): ")
    num_lines = int(input("Enter the number of lines to display: "))

    with open(file_name, 'r') as file:
        lines = file.readlines()

    if num_lines <= 0:
        print("Please enter a positive number of lines to display.")
    else:
        for line in lines[:num_lines]:
            print(line.strip())

except FileNotFoundError:
    print("File not found. Please make sure the file exists in the specified path.")
except ValueError:
    print("Invalid input. Please enter a valid number of lines to display.")
except Exception as e:
    print(f"An error occurred: {str(e)}")
```

Output:

Enter the file name (e.g., myfile.txt):
myfile.txtEnter the number of lines to
display: 2
this is my file
Display the first N line of the file

Enter the file name (e.g., myfile.txt):
myfile.txtEnter the number of lines to
display: 1
this is my file

6 a)

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

```
try:
    file_name = input("Enter the file name (e.g., myfile.txt): ")
    word = input("Enter the word to find its frequency: ")

    with open(file_name, 'r') as file:
        frequency = 0
        for line in file:
            words = line.strip().split()
            frequency += words.count(word)

    print(f"The word '{word}' appeared {frequency} times in the file.")
except FileNotFoundError:
    print("File not found. Please make sure the file exists in the specified path.")
except Exception as e:
    print(f"An error occurred: {str(e)}")
```

Output:

Enter the file name (e.g., myfile.txt):
myfile.txtEnter the word to find its
frequency: file
The word 'file' appeared 2 times in the file.

Enter the file name (e.g., myfile.txt):
myfile.txtEnter the word to find its
frequency: my
The word 'my' appeared 1 times in the file.

6b) Python is termed as secure language, demonstrate a simple method of securing data by creating a ZIP file of a particular folder which contains several files inside it.

```
import zipfile
import os

folder_path = input("Enter the folder path to zip: ")
zip_file_name = input("Enter the name of the ZIP file to create: ")

try:
    with zipfile.ZipFile(zip_file_name, 'w', zipfile.ZIP_DEFLATED) as zipf:
        for root, _, files in os.walk(folder_path):
            for file in files:
                file_path = os.path.join(root, file)
                zipf.write(file_path, arcname=file)
        print(f"The folder '{folder_path}' has been securely zipped into the file
'{zip_file_name}'.")
except FileNotFoundError:
    print("Folder not found. Please make sure the folder exists in the specified
path.")
```

Output:

Enter the folder path to zip:
C:\Users\SAM_LALI\Desktop\Temp22062023Enter the name
of the ZIP file to create: tt

The folder 'C:\Users\SAM_LALI\Desktop\Temp22062023' has been securely zipped into the file 'tt'.

7a) Inheritance is one of the main pillars of OOPs concept. By using inheritance, a child class acquires all properties and behaviors of parent class. Referring the above inheritance concept write a python program to find the area of triangle, circle and rectangle.

```
import math

class Shape:
    def __init__(self):
        pass

    def calculate_area(self):
        pass

class Triangle(Shape):
    def __init__(self, base, height):
        super().__init__()
        self.base = base
        self.height = height

    def calculate_area(self):
        return 0.5 * self.base * self.height

class Circle(Shape):
    def __init__(self, radius):
        super().__init__()
        self.radius = radius

    def calculate_area(self):
        return math.pi * self.radius**2

class Rectangle(Shape):
    def __init__(self, length, width):
        super().__init__()
        self.length = length
        self.width = width

    def calculate_area(self):
        return self.length * self.width

triangle = Triangle(5, 7)
circle = Circle(3)
rectangle = Rectangle(4, 6)

print("Area of Triangle:", triangle.calculate_area())
print("Area of Circle:", circle.calculate_area())
print("Area of Rectangle:", rectangle.calculate_area())
```

Output:

Area of Triangle: 17.5
Area of Circle: 28.274333882308138
Area of Rectangle: 24

7 b) Implement 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.

```
class Employee:
    def __init__(self, name, employee_id, department, salary):
        self.name = name
        self.employee_id = employee_id
        self.department = department
        self.salary = salary

    def update_salary(self, new_salary, department):
        if self.department == department:
            self.salary = new_salary

    def __str__(self):
        return f"Name: {self.name}\nEmployee_ID: {self.employee_id}\nDepartment:
{self.department}\nSalary: {self.salary}"

employee1 = Employee("John Doe", 1, "Finance", 5000)
employee2 = Employee("Jane Smith", 2, "Engineering", 6000)
employee3 = Employee("Tom Wilson", 3, "Engineering", 5500)

print("Before salary update:")
print(employee1)
print(employee2)
print(employee3)

new_salary = 6500
department = "Engineering"
employee2.update_salary(new_salary, department)
employee3.update_salary(new_salary, department)

print("\nAfter salary update:")
print(employee1)
print(employee2)
print(employee3)
```

Output:

Before salary update:

Name: John Doe
Employee_ID: 1
Department: Finance
Salary: 5000

Name: Jane Smith
Employee_ID: 2

Department: Engineering
Salary: 6000

Name: Tom Wilson
Employee_ID: 3
Department: Engineering
Salary: 5500

After salary update:

Name: John Doe
Employee_ID: 1
Department: Finance
Salary: 5000

Name: Jane Smith
Employee_ID: 2
Department: Engineering
Salary: 6500

Name: Tom Wilson
Employee_ID: 3
Department: Engineering
Salary: 6500

8. **a) Inheritance applies to classes, whereas polymorphism applies to methods, using these concepts implement a python program to find whether the given input is palindrome or not (for both string and integer).**

```
class PalindromeChecker:
    def __init__(self, input_str):
        self.input_str = input_str

    def is_palindrome(self):
        pass

class StringPalindromeChecker(PalindromeChecker):
    def __init__(self, input_str):
        super().__init__(input_str)

    def is_palindrome(self):
        reversed_str = self.input_str[::-1]
        return self.input_str.lower() == reversed_str.lower()

class IntegerPalindromeChecker(PalindromeChecker):
    def __init__(self, input_int):
        super().__init__(str(input_int))

    def is_palindrome(self):
        reversed_str = self.input_str[::-1]
        return self.input_str == reversed_str

user_input = input("Enter a string or number to check palindromicity: ")
str_palindrome = StringPalindromeChecker(user_input)

if str_palindrome.is_palindrome():
    print("The entered string is a palindrome.")
else:
    print("The entered string is not a palindrome.")

try:
    int_input = int(user_input)
    int_palindrome = IntegerPalindromeChecker(int_input)

    if int_palindrome.is_palindrome():
        print("The entered number is a palindrome.")
    else:
        print("The entered number is not a palindrome.")
except ValueError:
    pass
```

Output:

Enter a string or number to check palindromicity: noonThe entered string is a palindrome.

Enter a string or number to check palindromicity: 124The entered string is not a palindrome.

Enter a string or number to check palindromicity: jamThe entered string is not a palindrome.

9. a) XKCD is a webcomic website consists of many curious comics and sometimes user wants to save that comic image on their local devices, a user has to visit every page of the comic website. instead implement a python program to download the all XKCD comics.

```
import requests
import os

folder_path = 'xkcd_comics'
os.makedirs(folder_path, exist_ok=True)

response = requests.get('https://xkcd.com/info.0.json')
latest_comic_number = response.json()['num']

for comic_number in range(1, 6 + 1):
    response = requests.get(f'https://xkcd.com/{comic_number}/info.0.json')

    if response.status_code == 200:  # Check if the request was successful
        comic_info = response.json()
        image_url = comic_info['img']

        response = requests.get(image_url)

        if response.status_code == 200:  # Check if the image request was successful
            file_path = os.path.join(folder_path, f'comic{comic_number}.png')

            with open(file_path, 'wb') as file:
                file.write(response.content)

            print(f"Downloaded comic {comic_number}.")
        else:
            print(f"Failed to download comic {comic_number}.")
    else:
        print(f"Failed to fetch comic info for comic {comic_number}.")

print("All XKCD comics downloaded successfully!")
```

first need to install requests for importing

pip install requests

9 b) In python programming how to read the data from the spreadsheet and write the data in to thespreadsheet , by using the load_workbook() method, demonstrate with code snippet.

```
import openpyxl

workbook = openpyxl.load_workbook('exp.xlsx')

worksheet = workbook['Sheet1']

cell_value = worksheet.cell(row=1, column=1).value

worksheet.cell(row=1, column=2, value='rohith')
workbook.save('exp.xlsx')
```

10 a) Demonstrate with a python program the possible ways to combine select pages from many PDFs.

```
from PyPDF2 import PdfFileMerger

merger = PdfFileMerger()
merger.append(open('file1.pdf', 'rb'))
merger.append(open('file2.pdf', 'rb'))

merger.addBookmark('Page 1', 0)
merger.addBookmark('Page 2', 1)

merger.write(open('output.pdf', 'wb'))
merger.close()
```

OUTPUT:

PDFs merged successfully!

10. b) Accessing current weather data for any location on Earth, We collect and process weather data from different sources such as global and local weather models, satellites, radars and a vast network of weather stations and in different format. Implement a python program to fetch current weather data from the JSON file format.

```python
import json

def fetch_weather_data(file_path):
    try:
        with open(file_path, 'r') as file:
            data = json.load(file)
        return data
    except FileNotFoundError:
        print(f"File '{file_path}' not found.")
    except json.JSONDecodeError as e:
        print(f"Invalid JSON format in '{file_path}': {e}")

file_path = 'weather_data.json'

try:
    weather_data = fetch_weather_data(file_path)
    if weather_data:
        print(weather_data)
except Exception as e:
    print(f"An error occurred: {str(e)}")
```

OUTPUT:

{'city': 'New York',
 'temperature': 25,
 'weather_condition': 'Sunny'}