# PYTHON PROGRAMMING LABORATORY

Course Code : 21CSL46

Credits : 01

CIE Marks : 50

SEE Marks : 50

Teaching Hours/Weeks (L: T: P: S) : 0: 0: 2: 0

Course Objectives:

CLO 1. Demonstrate the use of IDLE or PyCharm IDE to create Python Applications

CLO 2. Using Python programming language to develop programs for solving real-world problems

CLO 3. Implement the Object-Oriented Programming concepts in Python.

CLO 4. Appraise the need for working with various documents like Excel, PDF, Word and Others

CLO 5. Demonstrate regular expression using python programming

Aim: Introduce the Python fundamentals, data types, operators, flow control and exception handling in Python

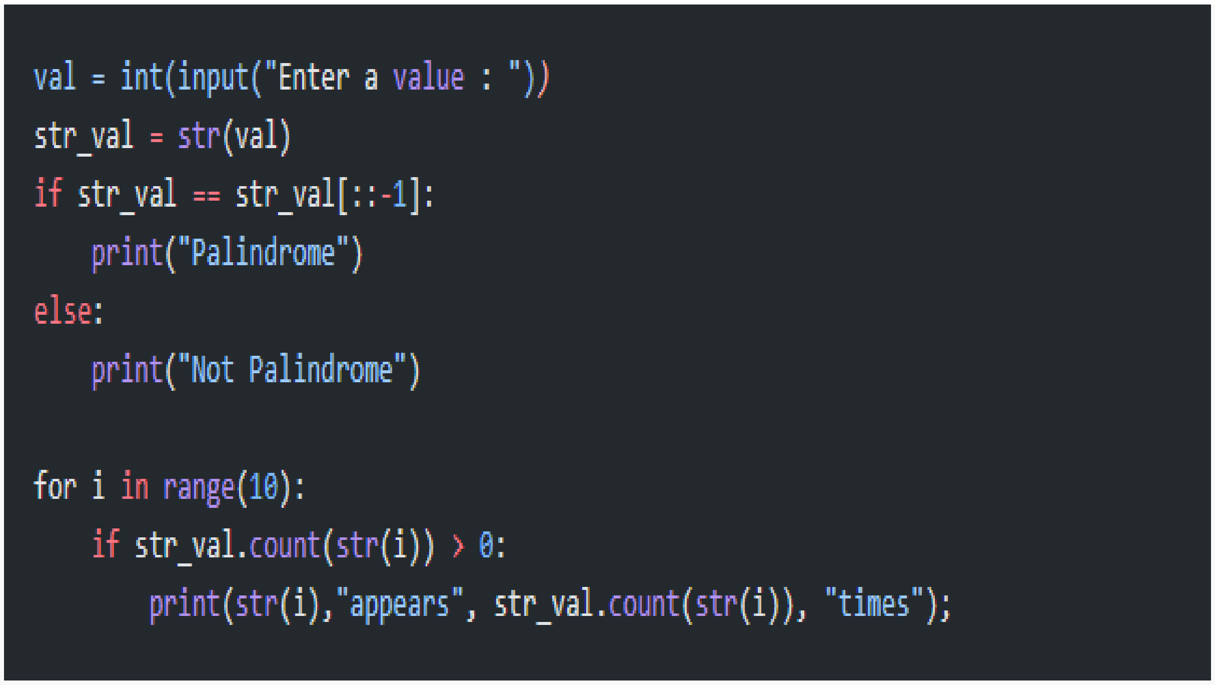
1. a) Write a python program to find the best of two test average marks out of three test's marks accepted from the user.

```
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
elif m3 <= m1 and m2 <= m2:
    avgMarks = (m1+m2)/2

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

Output

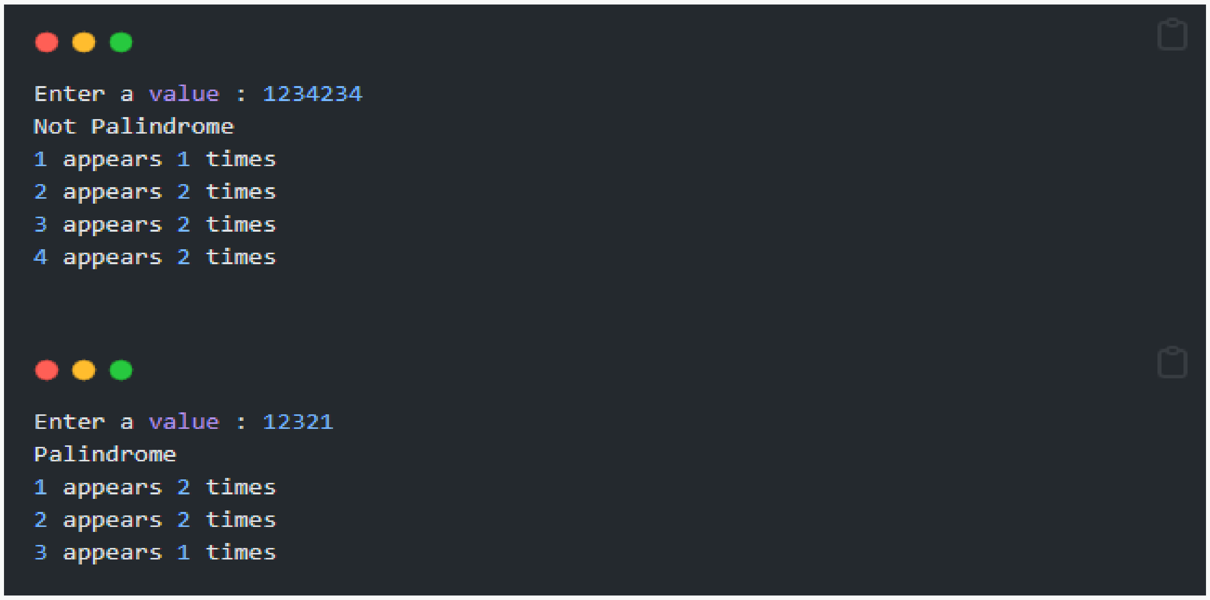
```
Enter marks for test1 : 45
Enter marks for test2 : 39
Enter marks for test3 : 48
Average of best two test marks  out of three test's marks is 46.5
```

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.



```python
val = int(input("Enter a value : "))
str_val = str(val)
if str_val == str_val[::-1]:
    print("Palindrome")
else:
    print("Not Palindrome")

for i in range(10):
    if str_val.count(str(i)) > 0:
        print(str(i),"appears", str_val.count(str(i)), "times");
```

Output

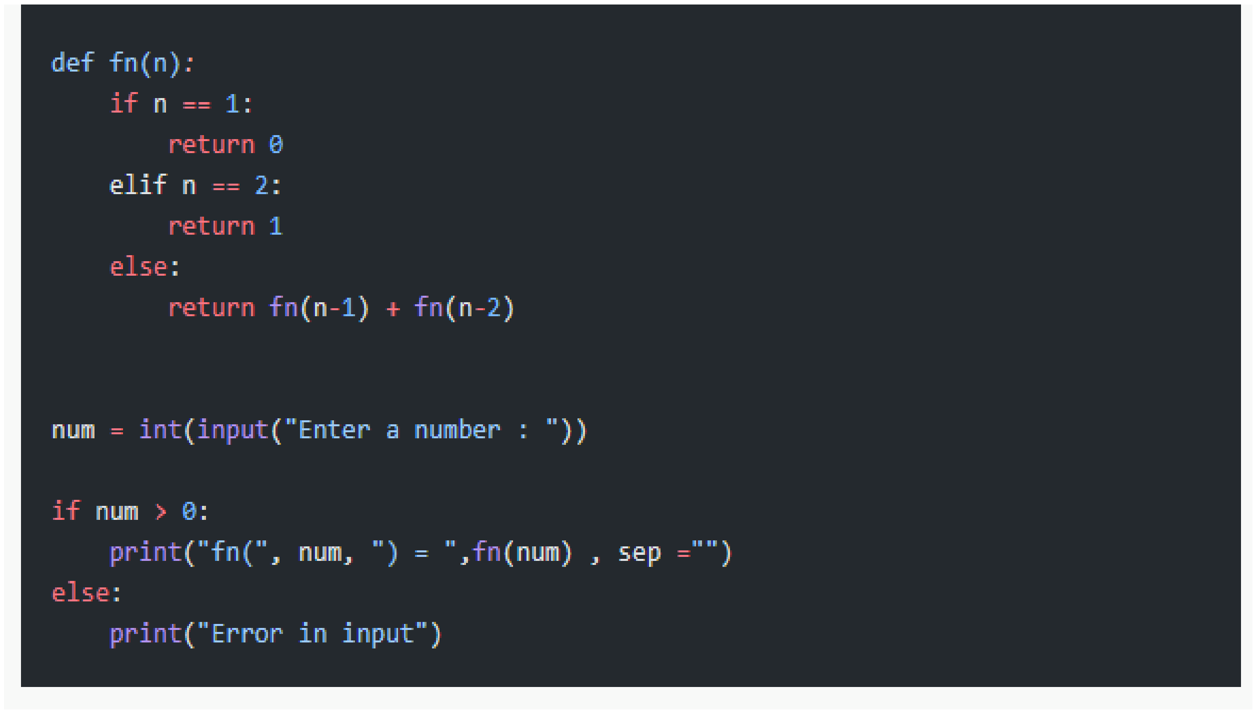


```
Enter a value : 1234234
Not Palindrome
1 appears 1 times
2 appears 2 times
3 appears 2 times
4 appears 2 times
```

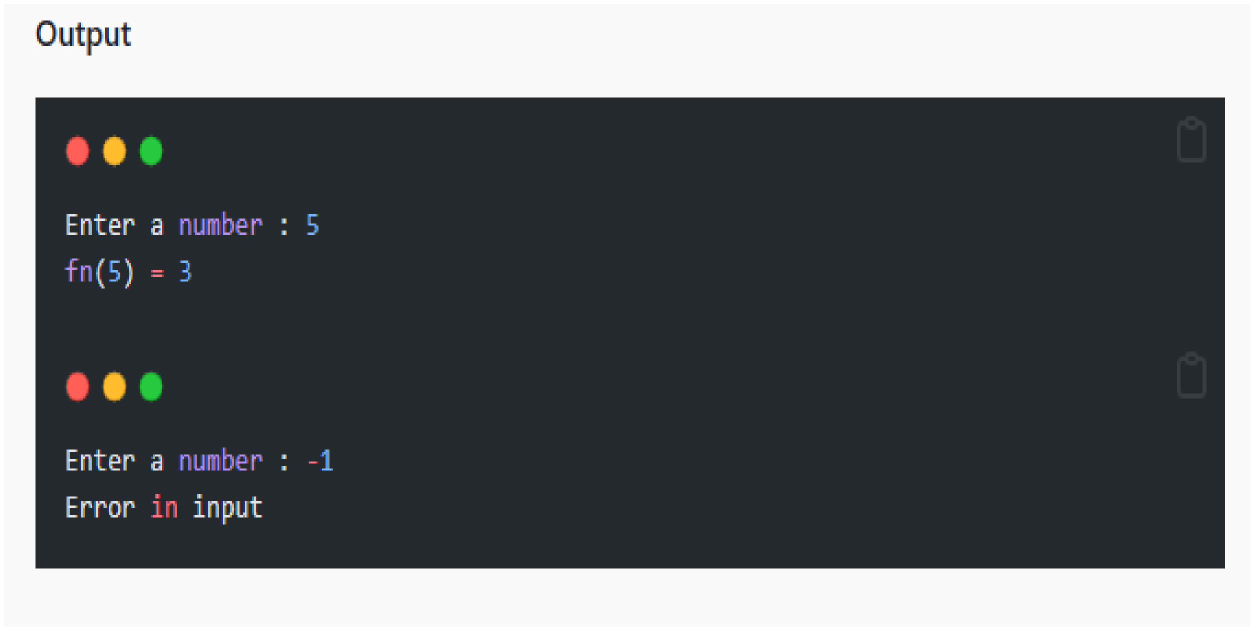
```
Enter a value : 12321
Palindrome
1 appears 2 times
2 appears 2 times
3 appears 1 times
```

Aim: Demonstrating creation of functions, passing parameters and return values

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.



```python
def fn(n):
    if n == 1:
        return 0
    elif n == 2:
        return 1
    else:
        return fn(n-1) + fn(n-2)


num = int(input("Enter a number : "))

if num > 0:
    print("fn(", num, ") = ",fn(num) , sep ="")
else:
    print("Error in input")
```

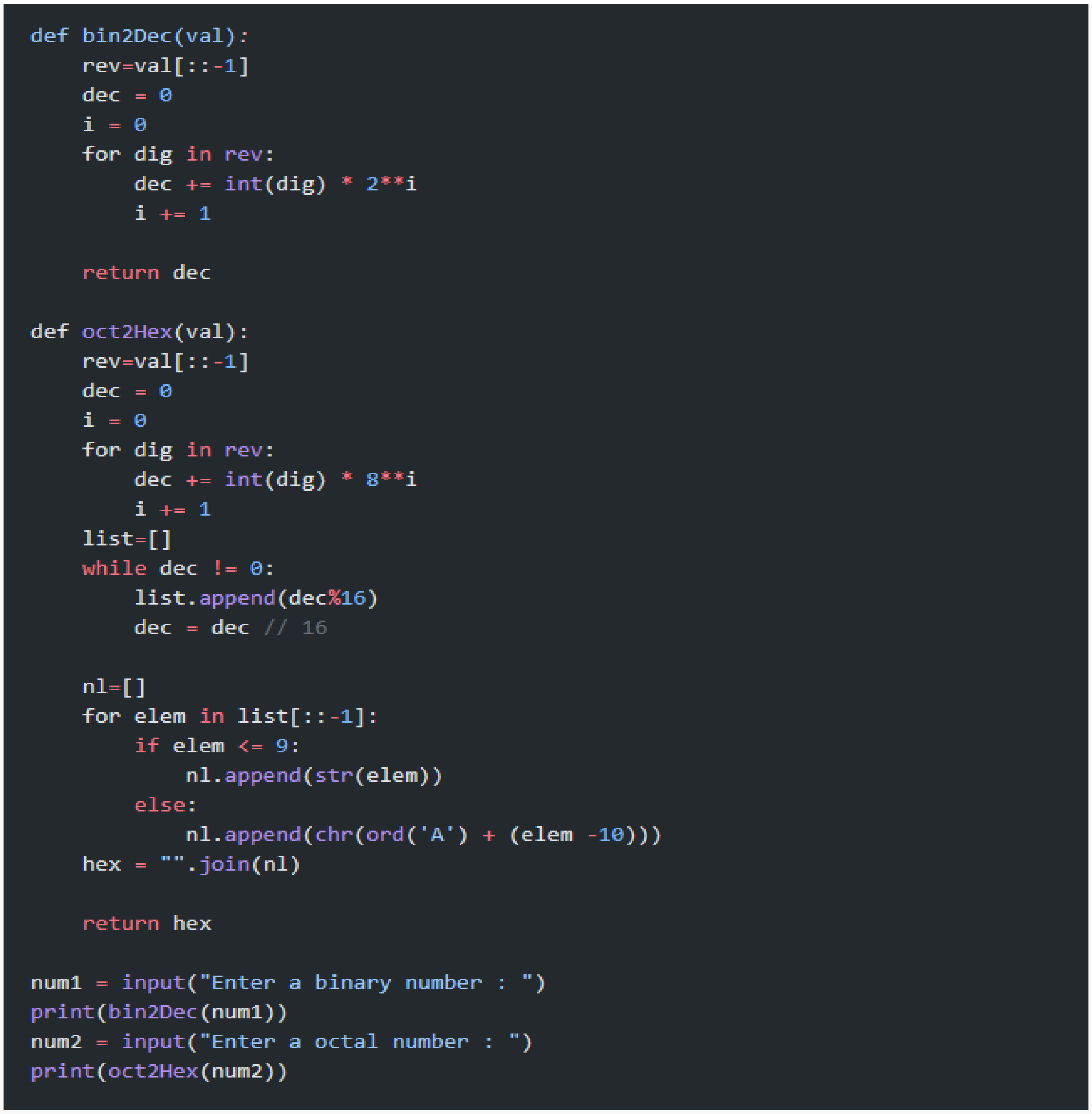


Output

```
Enter a number : 5
fn(5) = 3
```

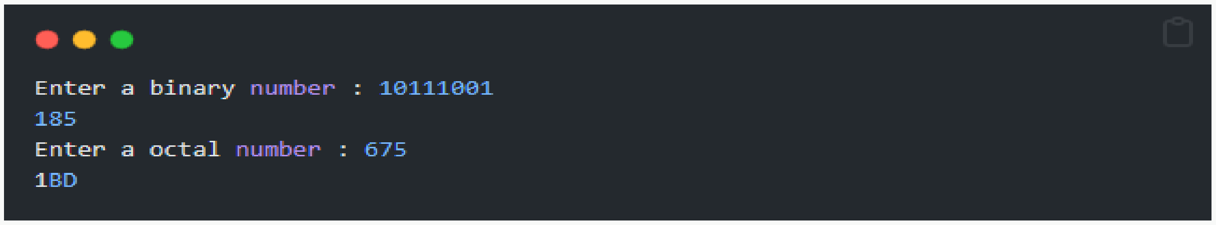
```
Enter a number : -1
Error in input
```

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



```python
def bin2Dec(val):
    rev=val[::-1]
    dec = 0
    i = 0
    for dig in rev:
        dec += int(dig) * 2**i
        i += 1

    return dec

def oct2Hex(val):
    rev=val[::-1]
    dec = 0
    i = 0
    for dig in rev:
        dec += int(dig) * 8**i
        i += 1
    list=[]
    while dec != 0:
        list.append(dec%16)
        dec = dec // 16

    nl=[]
    for elem in list[::-1]:
        if elem <= 9:
            nl.append(str(elem))
        else:
            nl.append(chr(ord('A') + (elem -10)))
    hex = "".join(nl)

    return hex

num1 = input("Enter a binary number : ")
print(bin2Dec(num1))
num2 = input("Enter a octal number : ")
print(oct2Hex(num2))
```

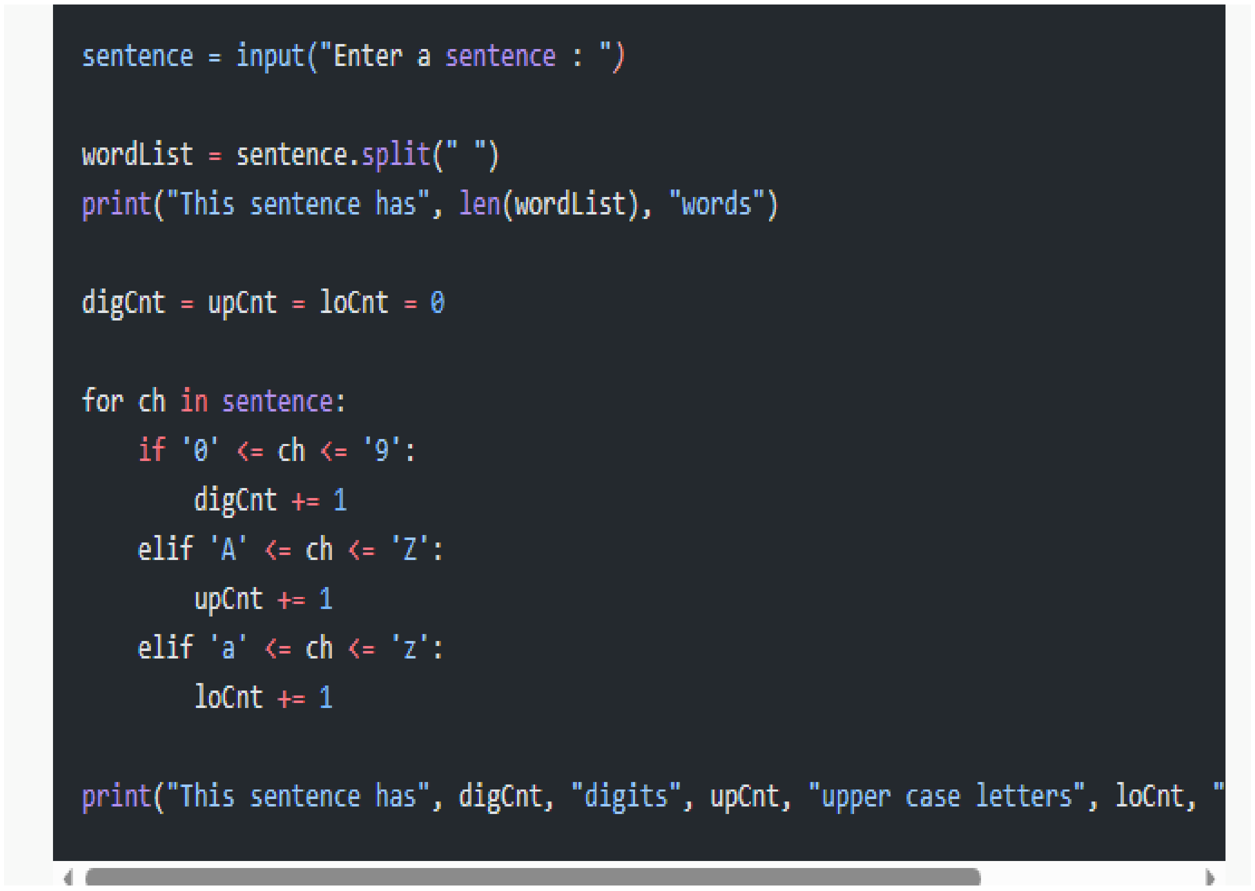
**Output**



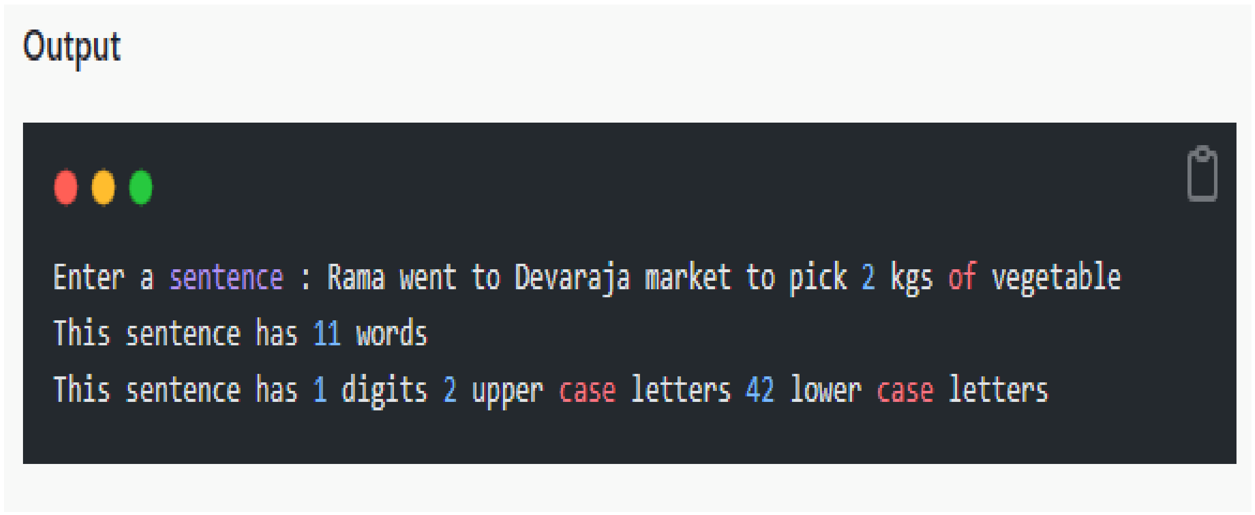
```
Enter a binary number : 10111001
185
Enter a octal number : 675
1BD
```

Aim: Demonstration of manipulation of strings using string methods

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



```python
sentence = input("Enter a sentence : ")

wordList = sentence.split(" ")
print("This sentence has", len(wordList), "words")

digCnt = upCnt = loCnt = 0

for ch in sentence:
    if '0' <= ch <= '9':
        digCnt += 1
    elif 'A' <= ch <= 'Z':
        upCnt += 1
    elif 'a' <= ch <= 'z':
        loCnt += 1

print("This sentence has", digCnt, "digits", upCnt, "upper case letters", loCnt, "
```



Output

```
Enter a sentence : Rama went to Devaraja market to pick 2 kgs of vegetable
This sentence has 11 words
This sentence has 1 digits 2 upper case letters 42 lower case letters
```

3. 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.

Sample Output:

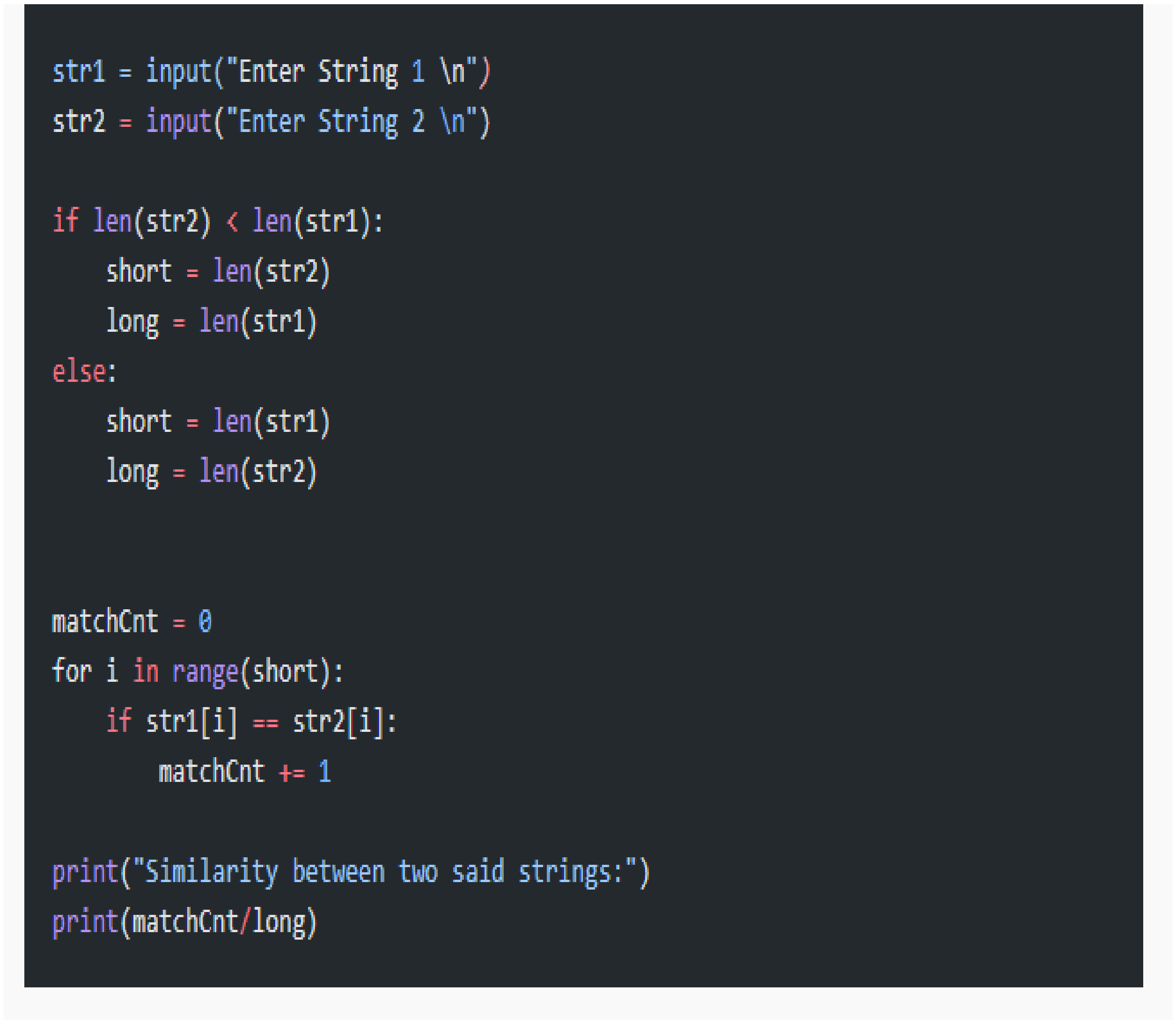
Original
string:

Python Exercises

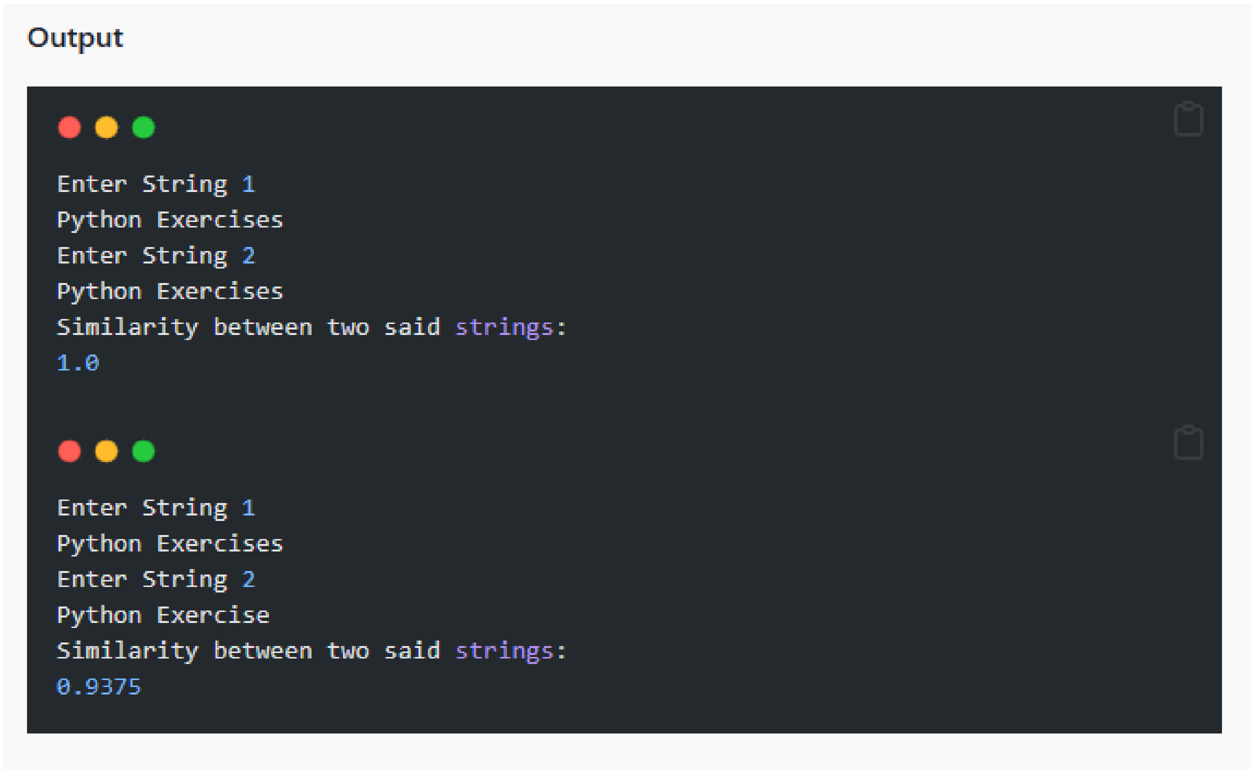
Python Exercise

Similarity between two said strings:

0.967741935483871



```
str1 = input("Enter String 1 \n")
str2 = input("Enter String 2 \n")

if len(str2) < len(str1):
    short = len(str2)
    long = len(str1)
else:
    short = len(str1)
    long = len(str2)


matchCnt = 0
for i in range(short):
    if str1[i] == str2[i]:
        matchCnt += 1

print("Similarity between two said strings:")
print(matchCnt/long)
```



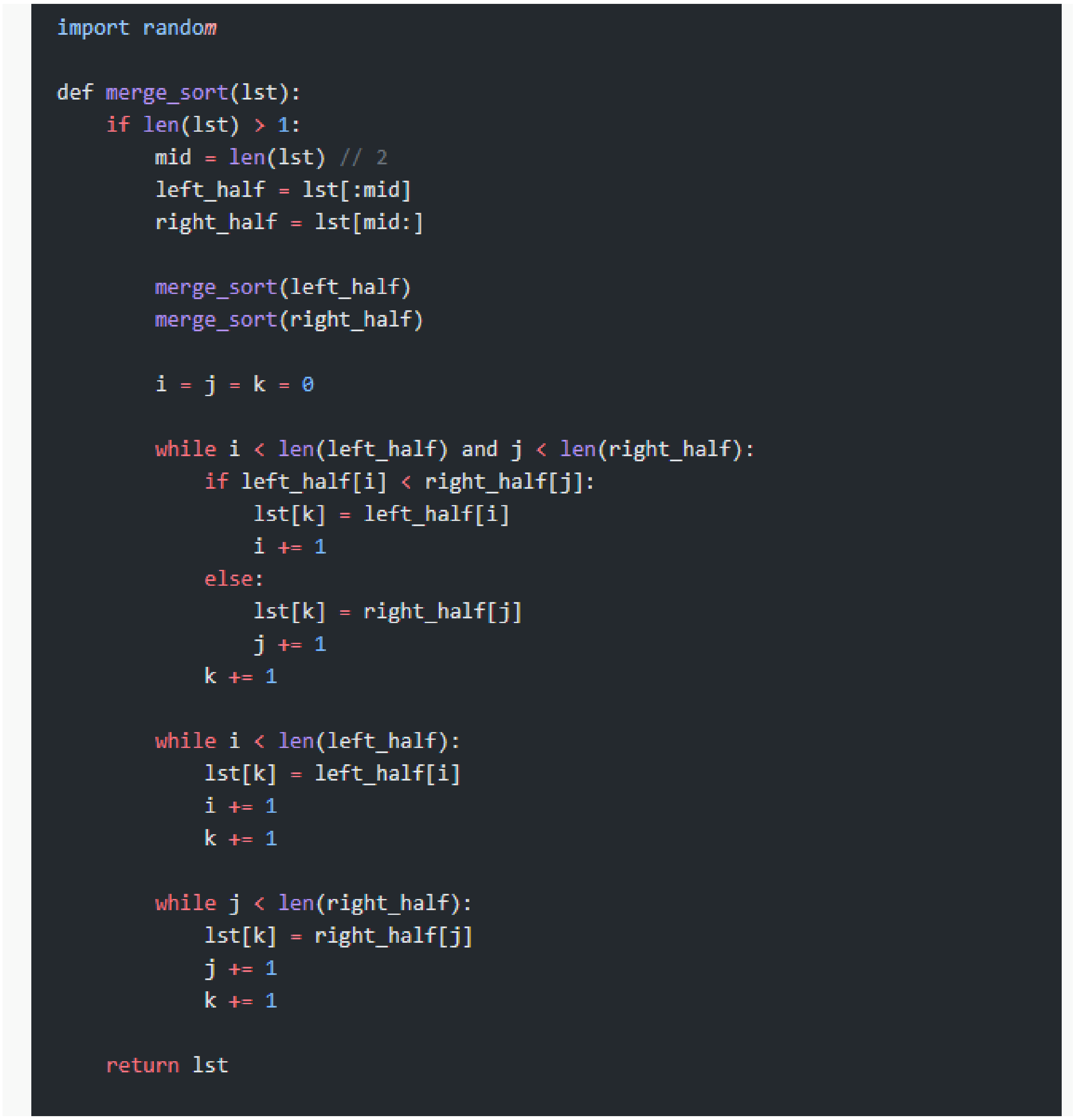
Output

```
Enter String 1
Python Exercises
Enter String 2
Python Exercises
Similarity between two said strings:
1.0
```

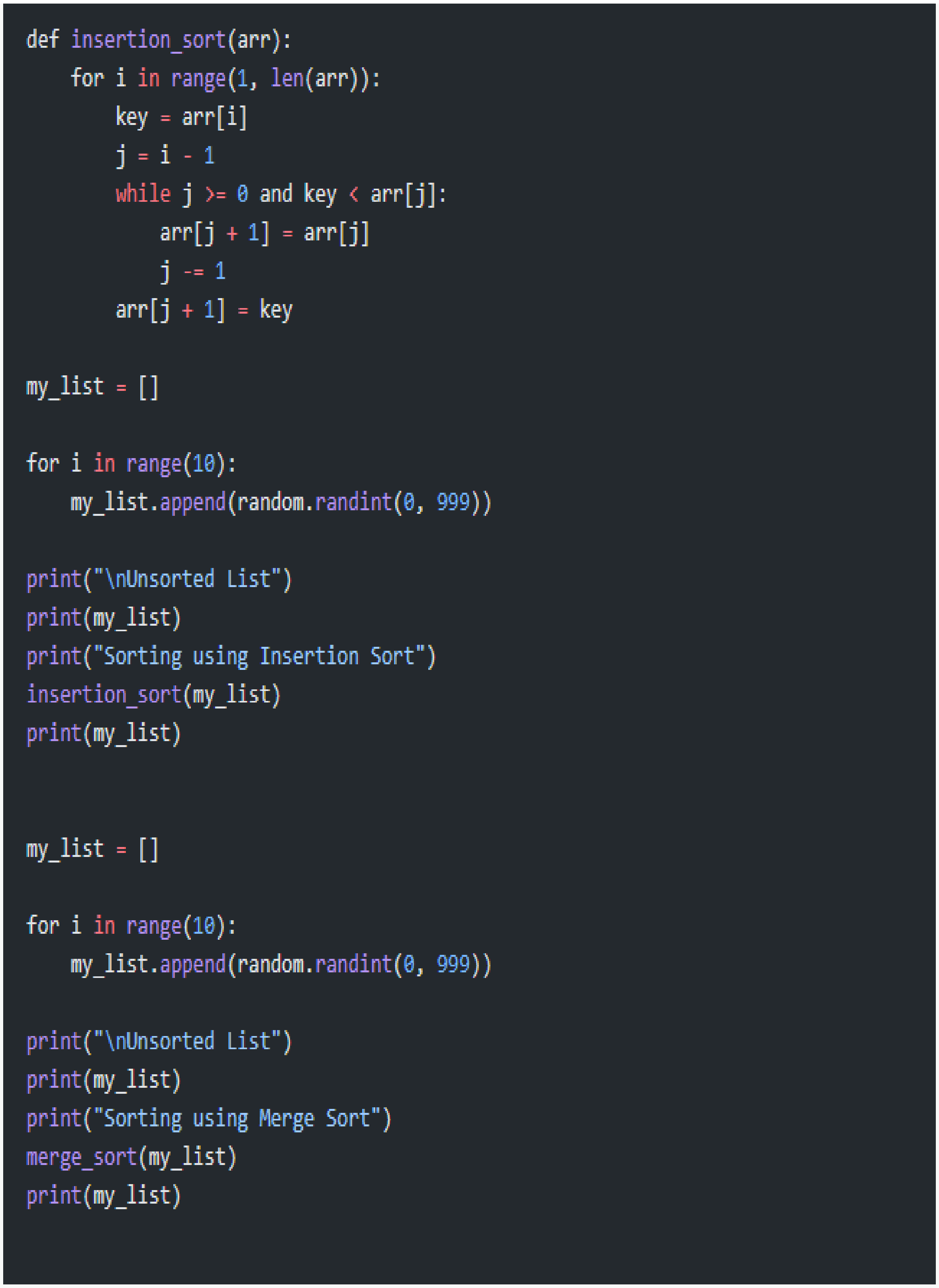
```
Enter String 1
Python Exercises
Enter String 2
Python Exercise
Similarity between two said strings:
0.9375
```

Aim: Discuss different collections like list, tuple and dictionary

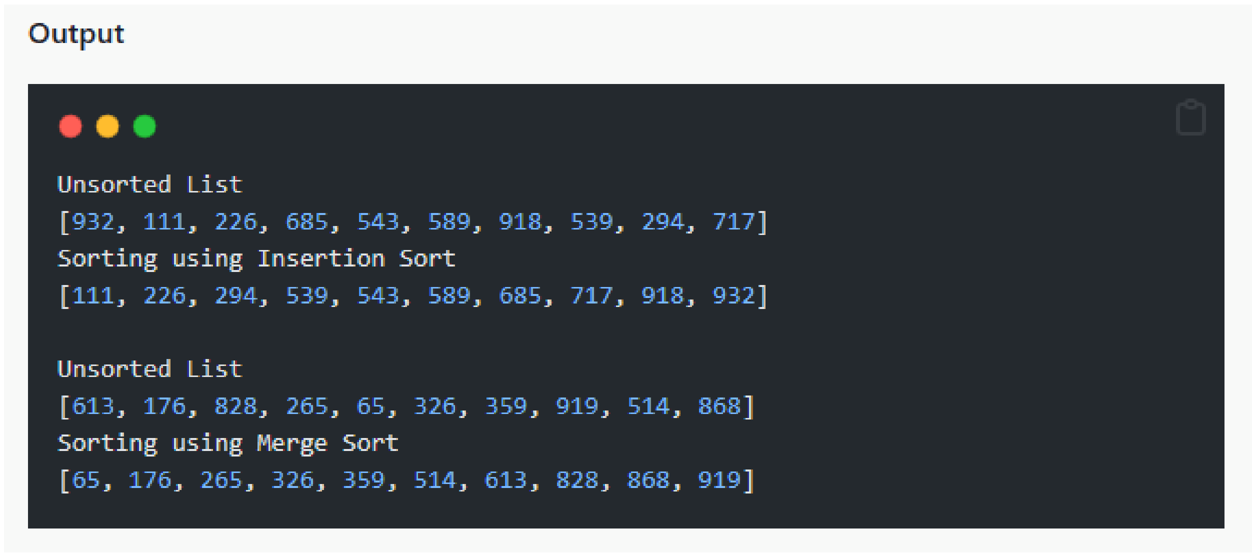
4. a) Write a python program to implement insertion sort and merge sort using lists



```python
import random

def merge_sort(lst):
    if len(lst) > 1:
        mid = len(lst) // 2
        left_half = lst[:mid]
        right_half = lst[mid:]

        merge_sort(left_half)
        merge_sort(right_half)

        i = j = k = 0

        while i < len(left_half) and j < len(right_half):
            if left_half[i] < right_half[j]:
                lst[k] = left_half[i]
                i += 1
            else:
                lst[k] = right_half[j]
                j += 1
            k += 1

        while i < len(left_half):
            lst[k] = left_half[i]
            i += 1
            k += 1

        while j < len(right_half):
            lst[k] = right_half[j]
            j += 1
            k += 1

    return lst
```

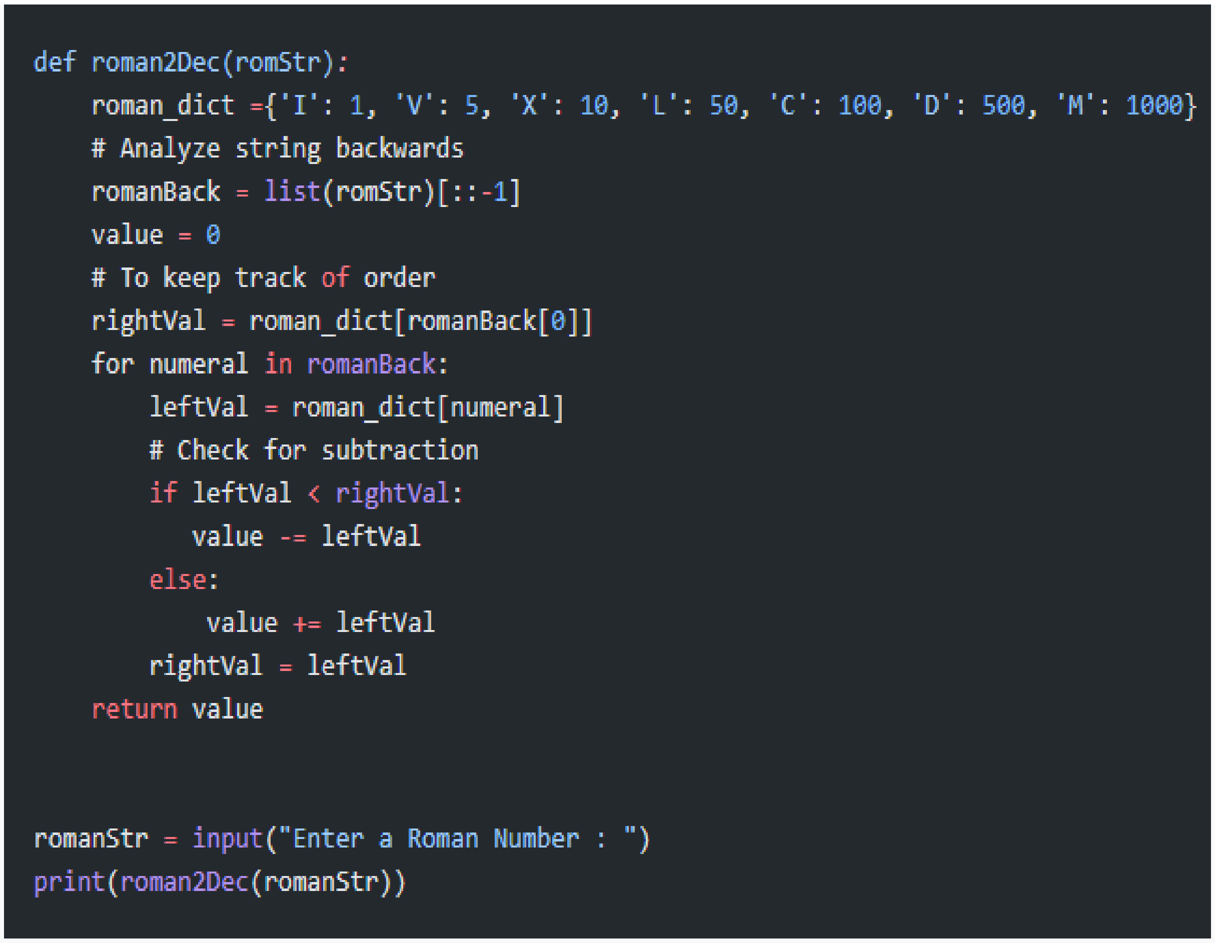


```python
def insertion_sort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i - 1
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key

my_list = []

for i in range(10):
    my_list.append(random.randint(0, 999))

print("\nUnsorted List")
print(my_list)
print("Sorting using Insertion Sort")
insertion_sort(my_list)
print(my_list)


my_list = []

for i in range(10):
    my_list.append(random.randint(0, 999))

print("\nUnsorted List")
print(my_list)
print("Sorting using Merge Sort")
merge_sort(my_list)
print(my_list)
```

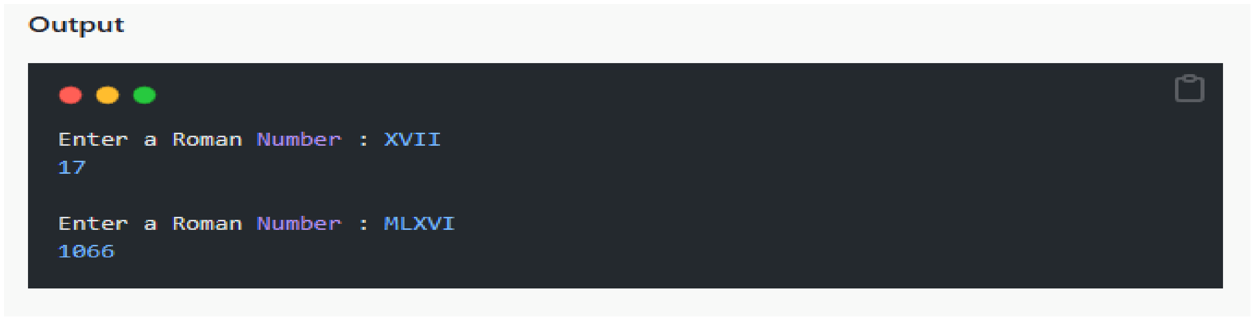

Output

```
Unsorted List
[932, 111, 226, 685, 543, 589, 918, 539, 294, 717]
Sorting using Insertion Sort
[111, 226, 294, 539, 543, 589, 685, 717, 918, 932]

Unsorted List
[613, 176, 828, 265, 65, 326, 359, 919, 514, 868]
Sorting using Merge Sort
[65, 176, 265, 326, 359, 514, 613, 828, 868, 919]
```

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



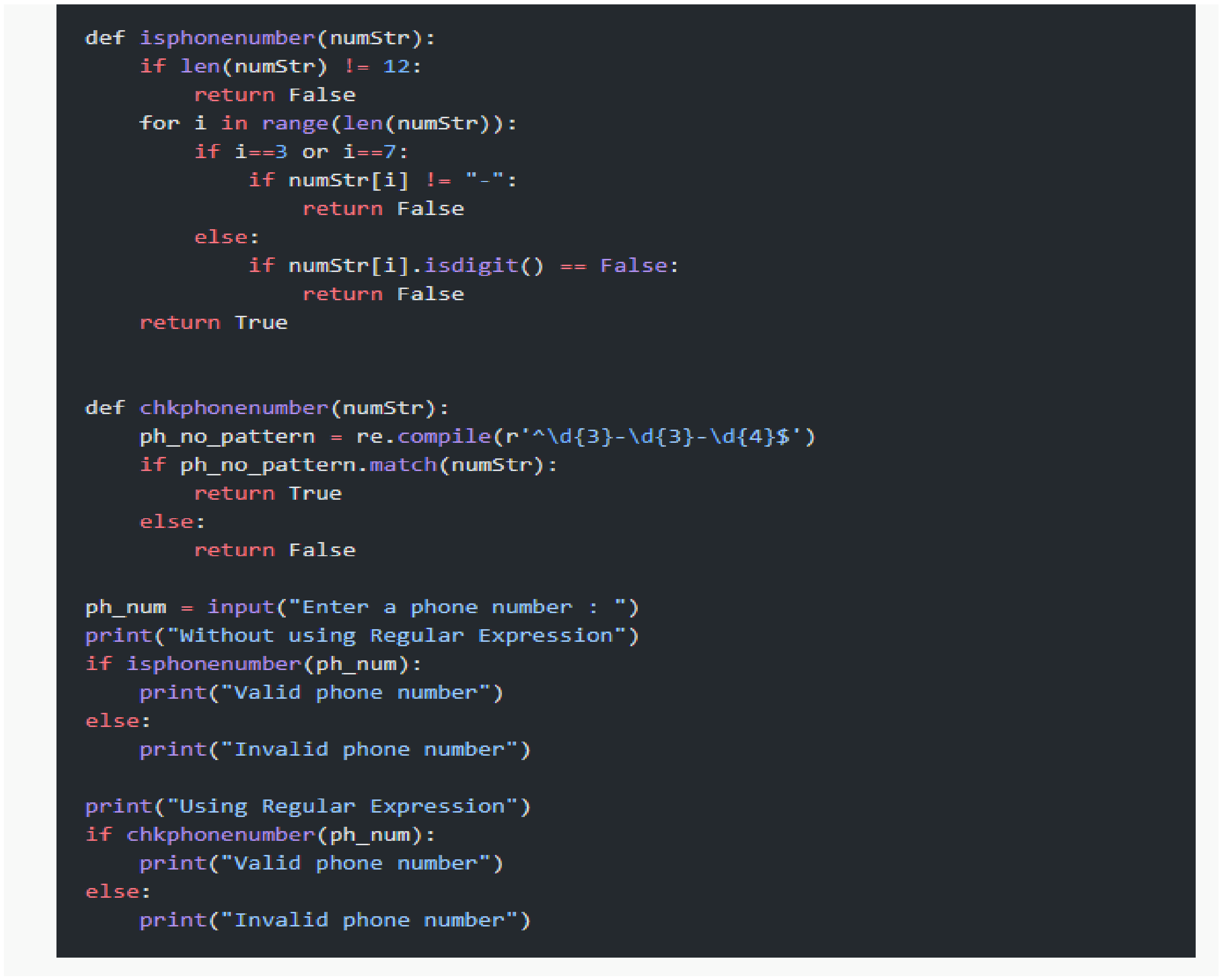
```
def roman2Dec(romStr):
    roman_dict ={'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000}
    # Analyze string backwards
    romanBack = list(romStr)[::-1]
    value = 0
    # To keep track of order
    rightVal = roman_dict[romanBack[0]]
    for numeral in romanBack:
        leftVal = roman_dict[numeral]
        # Check for subtraction
        if leftVal < rightVal:
           value -= leftVal
        else:
            value += leftVal
        rightVal = leftVal
    return value


romanStr = input("Enter a Roman Number : ")
print(roman2Dec(romanStr))
```


Output

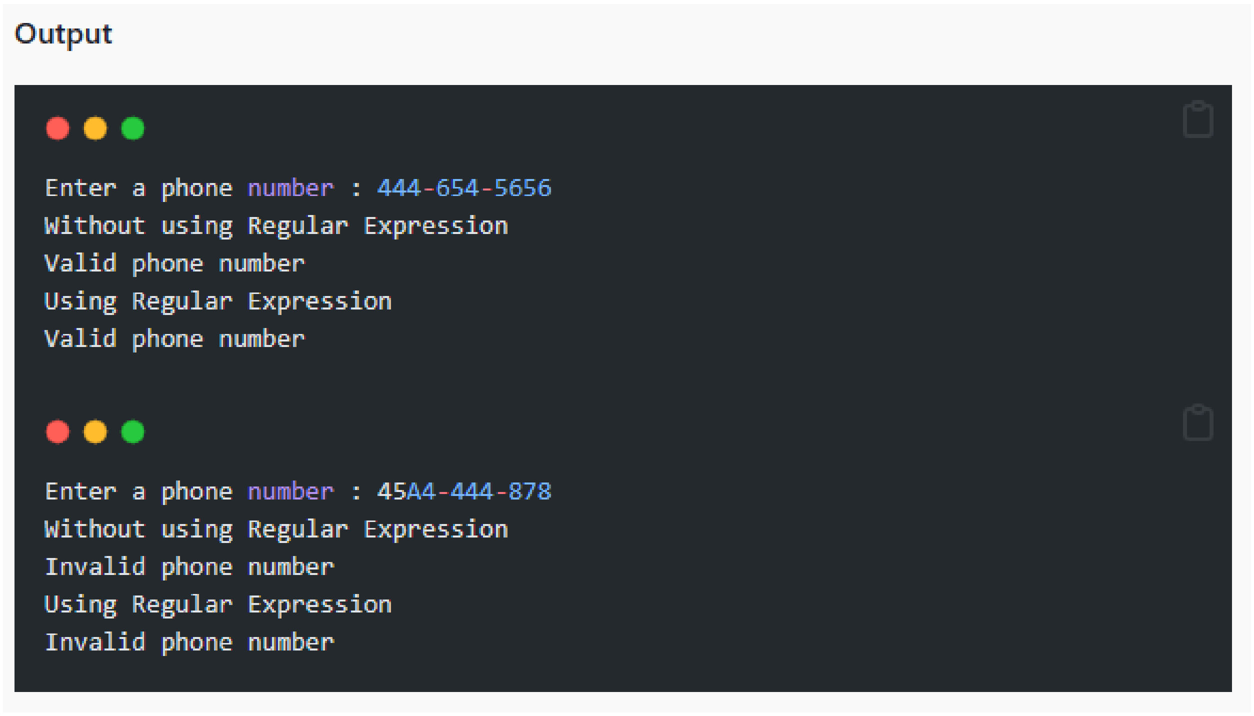
```
Enter a Roman Number : XVII
17

Enter a Roman Number : MLXVI
1066
```

Aim: Demonstration of pattern recognition with and without using regular expressions

5. a) Write 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.



```python
def isphonenumber(numStr):
    if len(numStr) != 12:
        return False
    for i in range(len(numStr)):
        if i==3 or i==7:
            if numStr[i] != "-":
                return False
        else:
            if numStr[i].isdigit() == False:
                return False
    return True


def chkphonenumber(numStr):
    ph_no_pattern = re.compile(r'^\d{3}-\d{3}-\d{4}$')
    if ph_no_pattern.match(numStr):
        return True
    else:
        return False

ph_num = input("Enter a phone number : ")
print("Without using Regular Expression")
if isphonenumber(ph_num):
    print("Valid phone number")
else:
    print("Invalid phone number")

print("Using Regular Expression")
if chkphonenumber(ph_num):
    print("Valid phone number")
else:
    print("Invalid phone number")
```

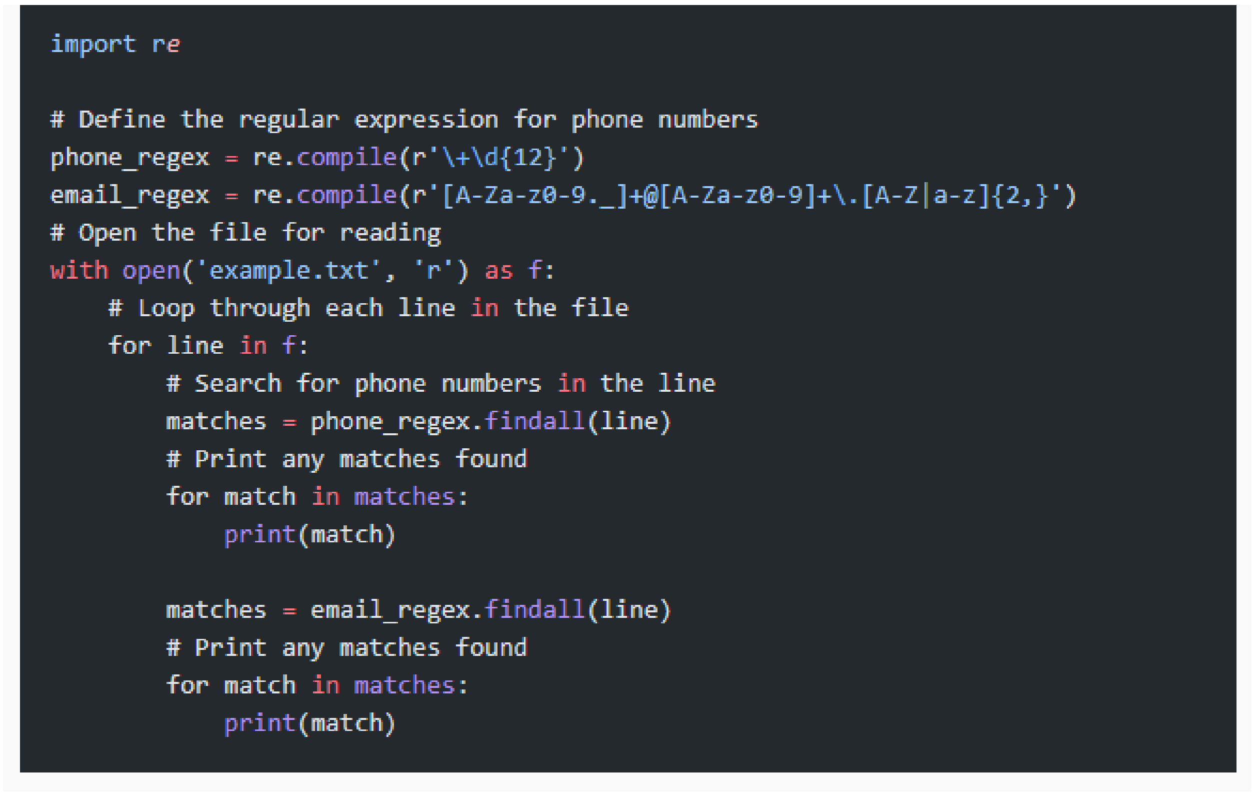


Output

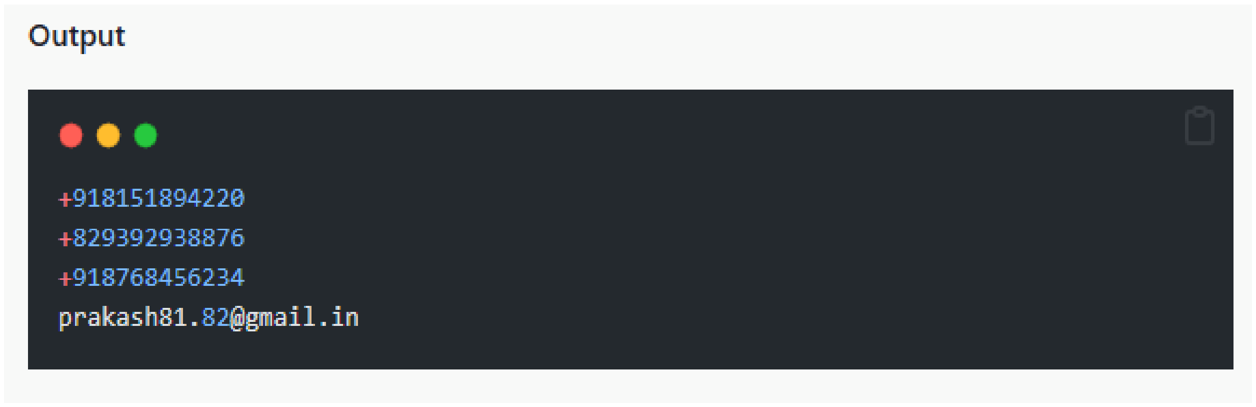
```
Enter a phone number : 444-654-5656
Without using Regular Expression
Valid phone number
Using Regular Expression
Valid phone number
```

```
Enter a phone number : 45A4-444-878
Without using Regular Expression
Invalid phone number
Using Regular Expression
Invalid phone number
```

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



```python
import re

# Define the regular expression for phone numbers
phone_regex = re.compile(r'\+\d{12}')
email_regex = re.compile(r'[A-Za-z0-9._]+@[A-Za-z0-9]+\.[A-Z|a-z]{2,}')
# Open the file for reading
with open('example.txt', 'r') as f:
    # Loop through each line in the file
    for line in f:
        # Search for phone numbers in the line
        matches = phone_regex.findall(line)
        # Print any matches found
        for match in matches:
            print(match)

        matches = email_regex.findall(line)
        # Print any matches found
        for match in matches:
            print(match)
```


Output

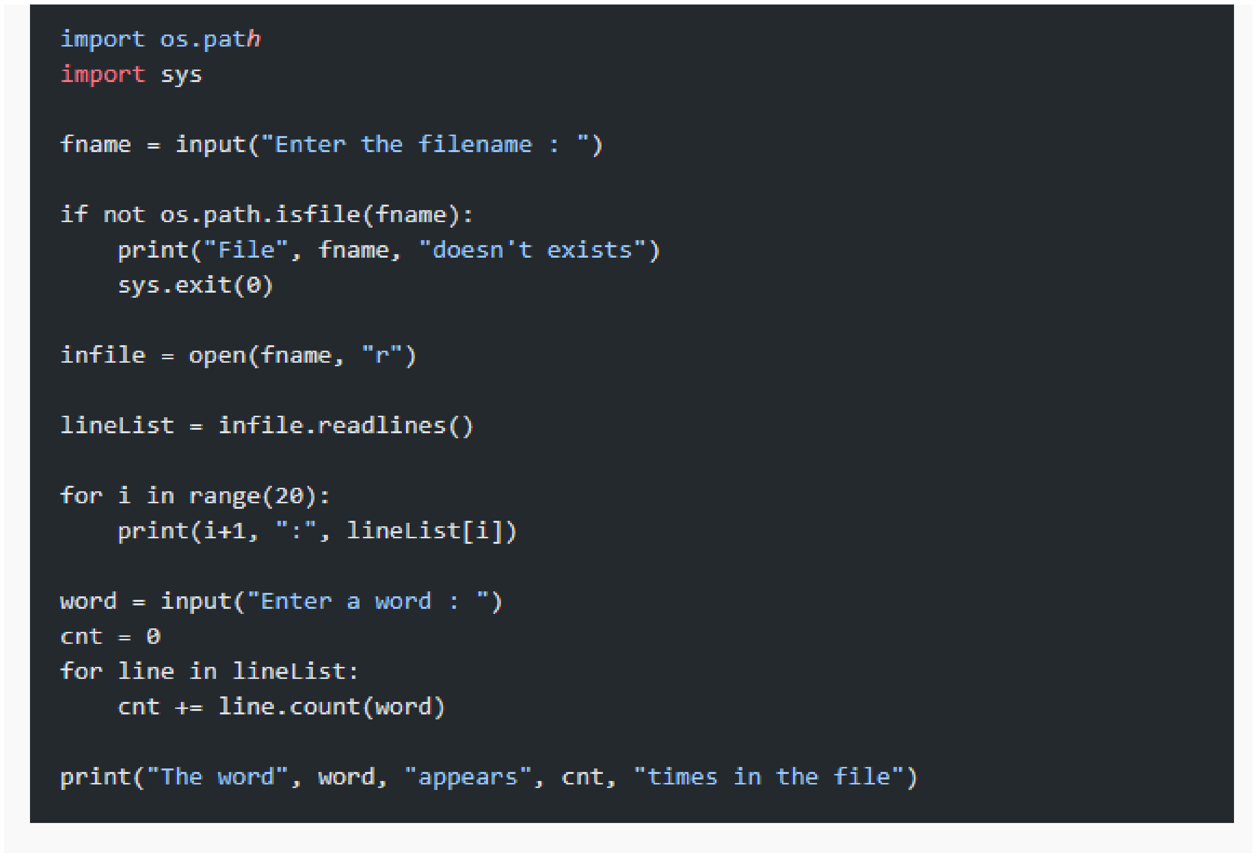
```
+918151894220
+829392938876
+918768456234
prakash81.82@gmail.in
```

Aim: Demonstration of reading, writing and organizing files.

6. a) Write a python program to accept a file name from the user and perform the following Operations:

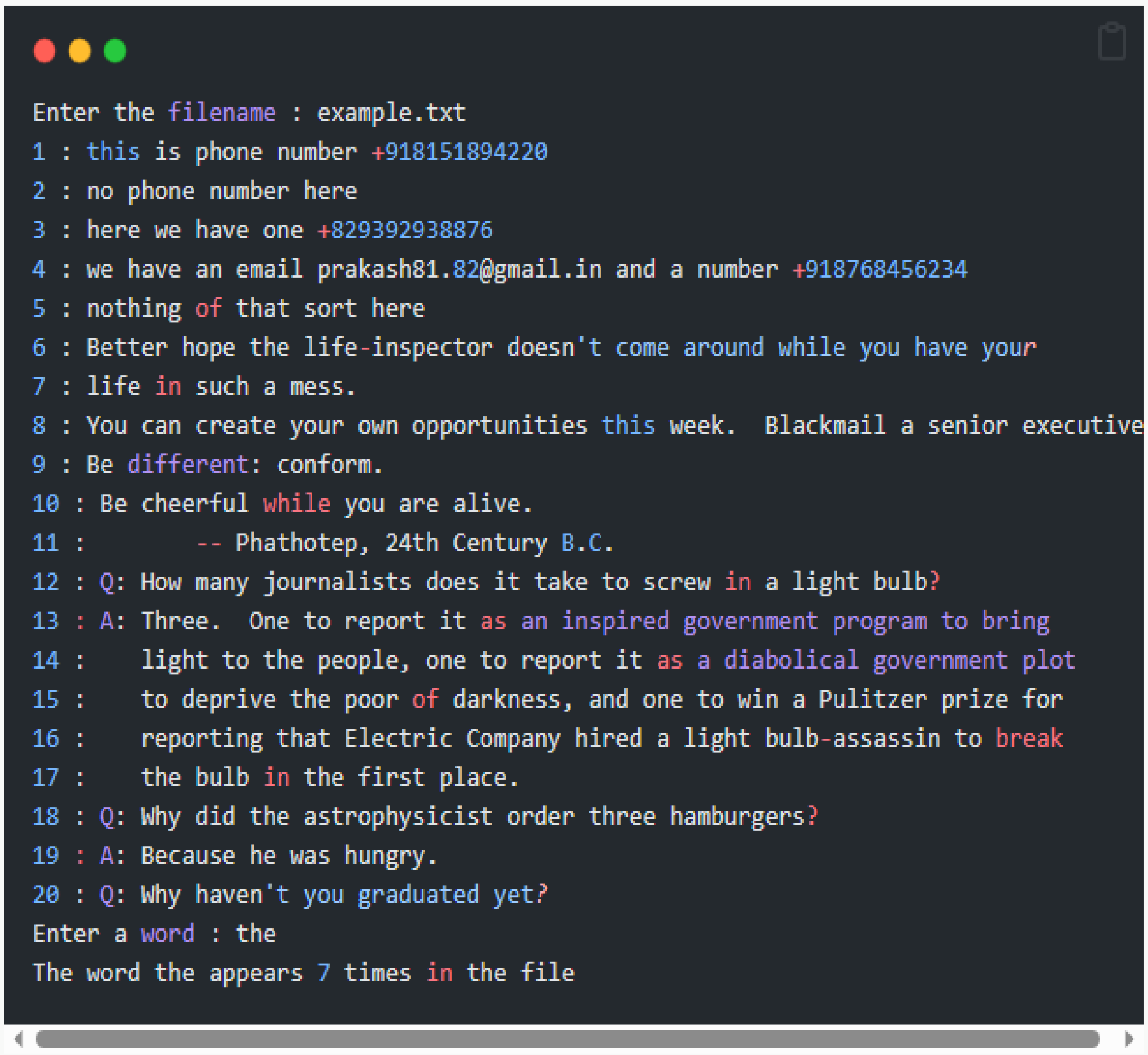
1. Display the first N line of the file

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



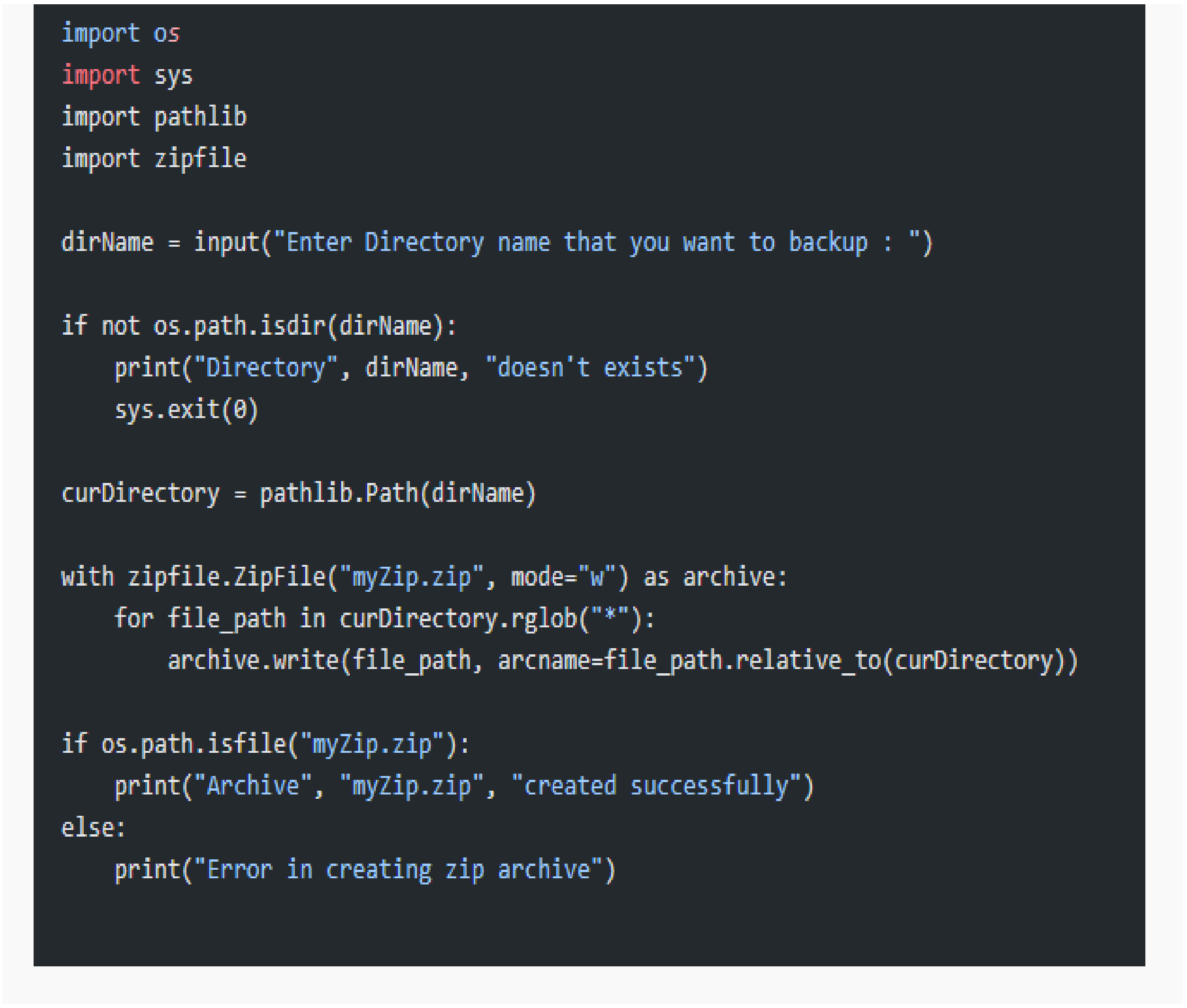
```python
import os.path
import sys

fname = input("Enter the filename : ")

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

infile = open(fname, "r")

lineList = infile.readlines()

for i in range(20):
    print(i+1, ":", lineList[i])

word = input("Enter a word : ")
cnt = 0
for line in lineList:
    cnt += line.count(word)

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

Output

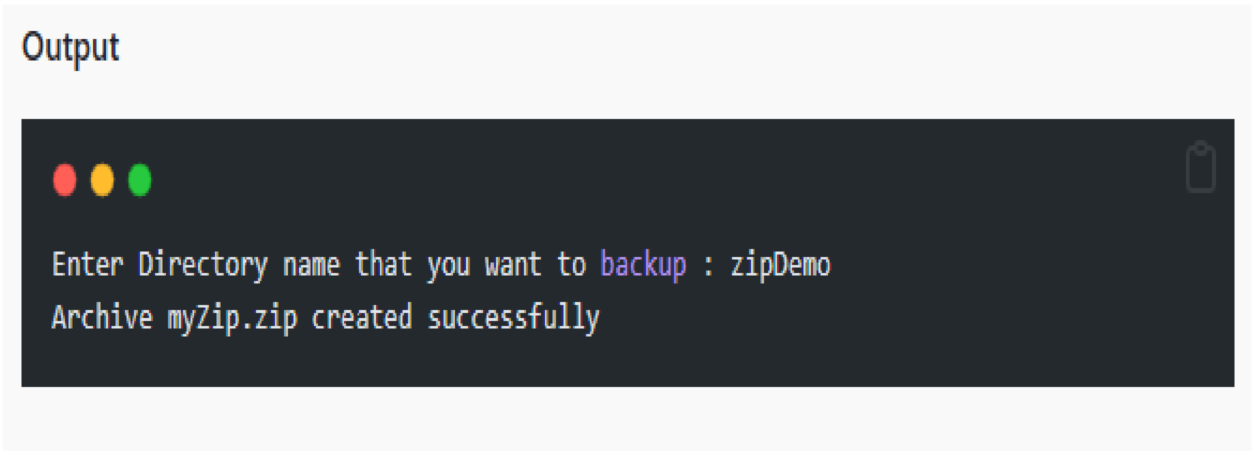


```
Enter the filename : example.txt
1 : this is phone number +918151894220
2 : no phone number here
3 : here we have one +829392938876
4 : we have an email prakash81.82@gmail.in and a number +918768456234
5 : nothing of that sort here
6 : Better hope the life-inspector doesn't come around while you have your
7 : life in such a mess.
8 : You can create your own opportunities this week.  Blackmail a senior executive
9 : Be different: conform.
10 : Be cheerful while you are alive.
11 :        -- Phathotep, 24th Century B.C.
12 : Q: How many journalists does it take to screw in a light bulb?
13 : A: Three.  One to report it as an inspired government program to bring
14 :    light to the people, one to report it as a diabolical government plot
15 :    to deprive the poor of darkness, and one to win a Pulitzer prize for
16 :    reporting that Electric Company hired a light bulb-assassin to break
17 :    the bulb in the first place.
18 : Q: Why did the astrophysicist order three hamburgers?
19 : A: Because he was hungry.
20 : Q: Why haven't you graduated yet?
Enter a word : the
The word the appears 7 times in the file
```

6. b) Write a python program to create a ZIP file of a particular folder which contains several files inside it.



```python
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")
```

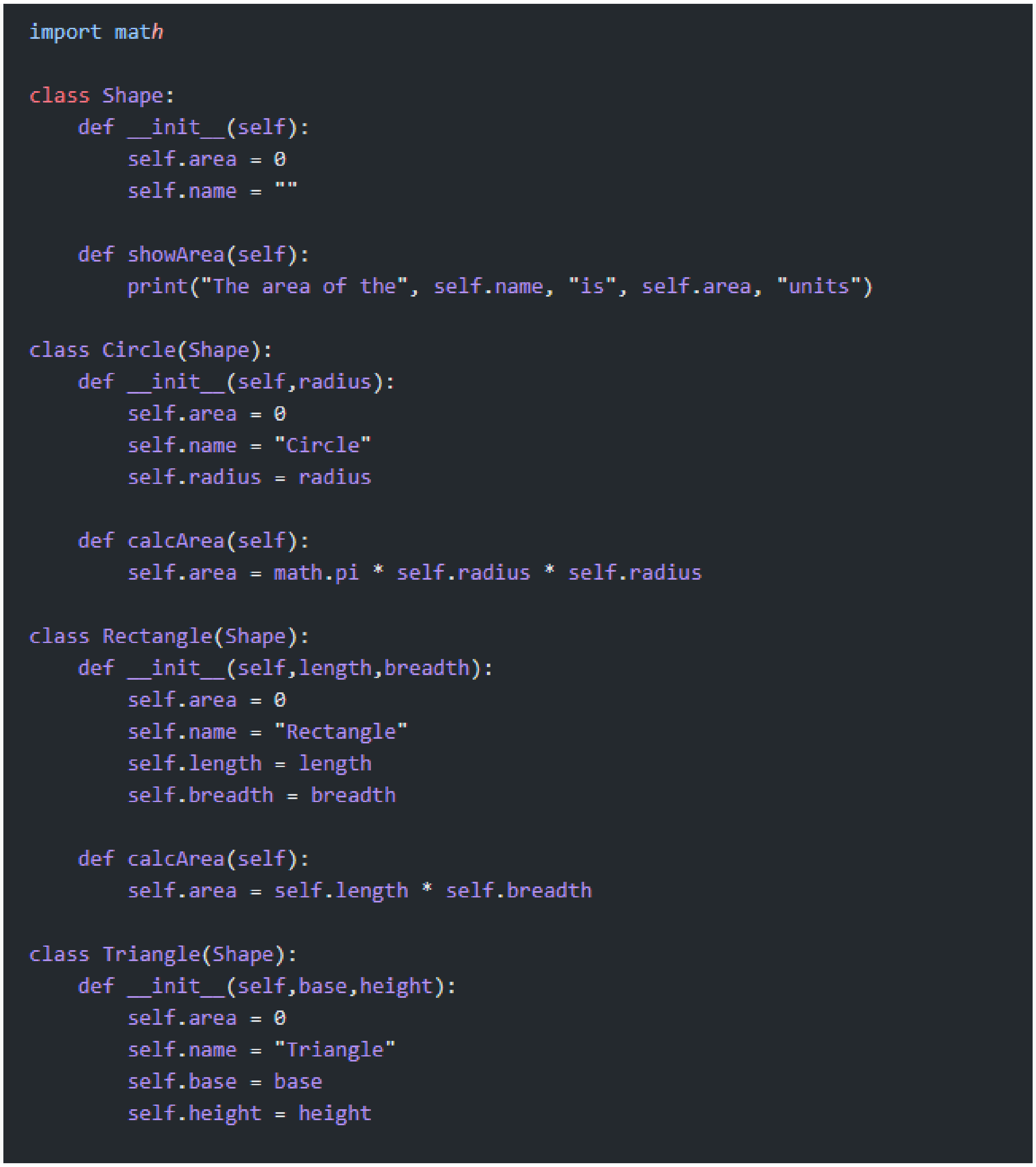


Output

```
Enter Directory name that you want to backup : zipDemo
Archive myZip.zip created successfully
```

Aim: Demonstration of the concepts of classes, methods, objects and inheritance

7. a) By using the concept of inheritance write a python program to find the area of triangle, circle and rectangle.

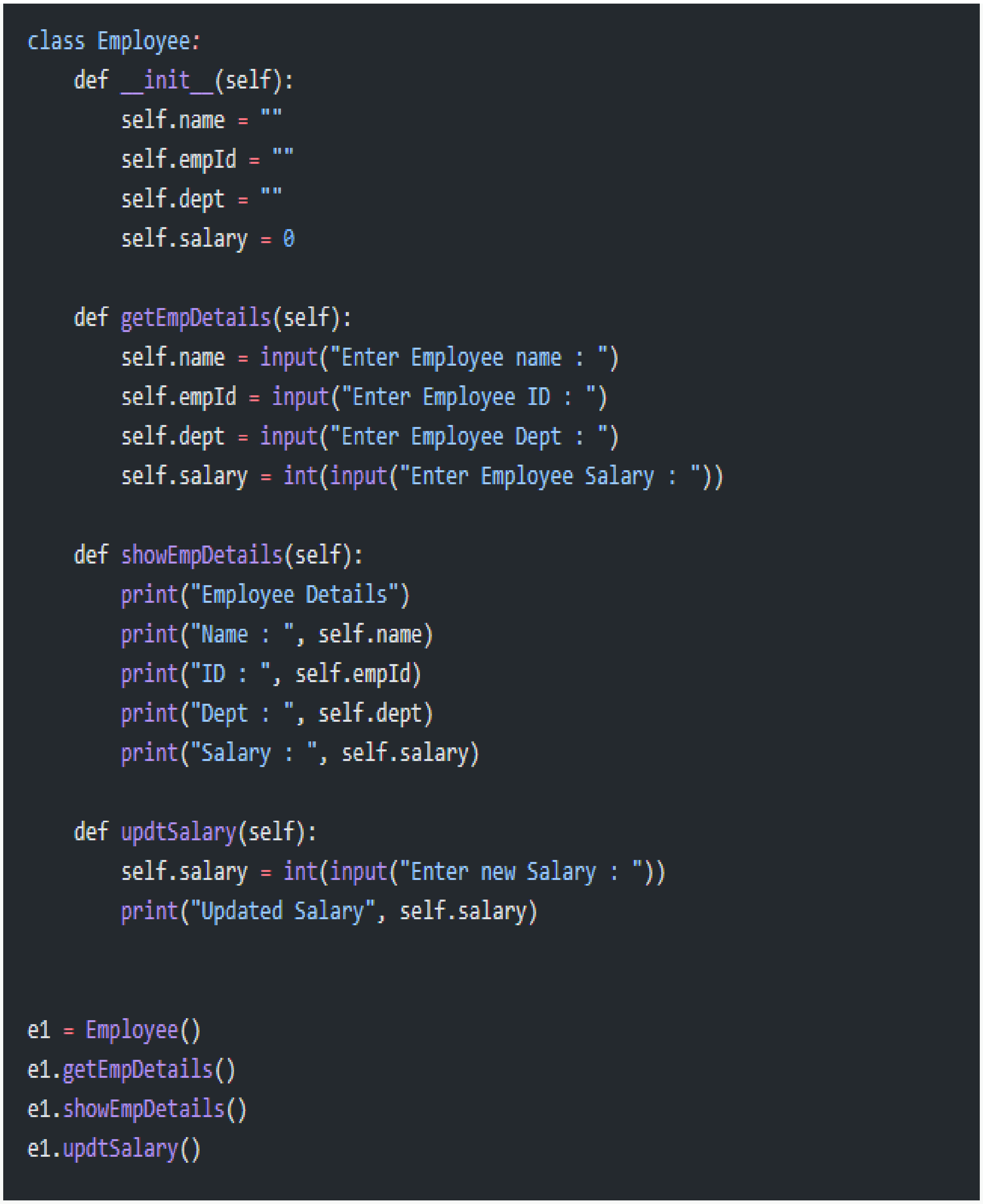


```python
import math

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()
```

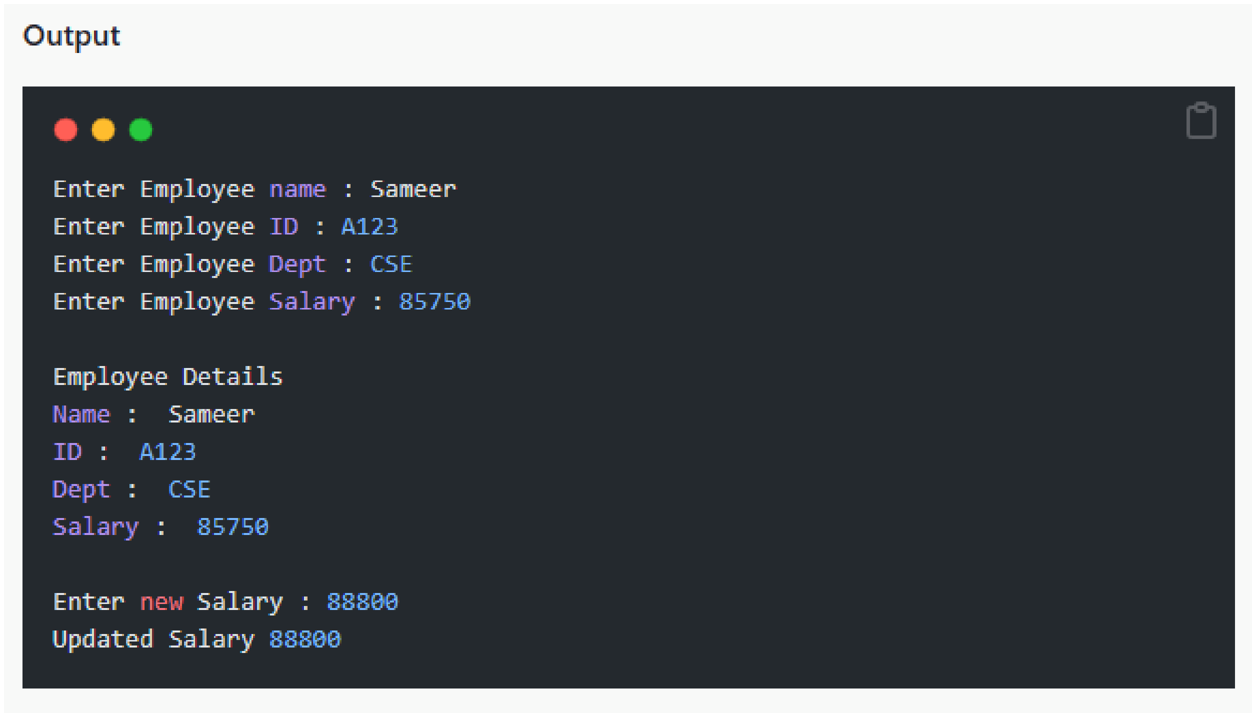
Output

```
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
```

7. 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.



```python
class Employee:
    def __init__(self):
        self.name = ""
        self.empId = ""
        self.dept = ""
        self.salary = 0

    def getEmpDetails(self):
        self.name = input("Enter Employee name : ")
        self.empId = input("Enter Employee ID : ")
        self.dept = input("Enter Employee Dept : ")
        self.salary = int(input("Enter Employee Salary : "))

    def showEmpDetails(self):
        print("Employee Details")
        print("Name : ", self.name)
        print("ID : ", self.empId)
        print("Dept : ", self.dept)
        print("Salary : ", self.salary)

    def updtSalary(self):
        self.salary = int(input("Enter new Salary : "))
        print("Updated Salary", self.salary)


e1 = Employee()
e1.getEmpDetails()
e1.showEmpDetails()
e1.updtSalary()
```

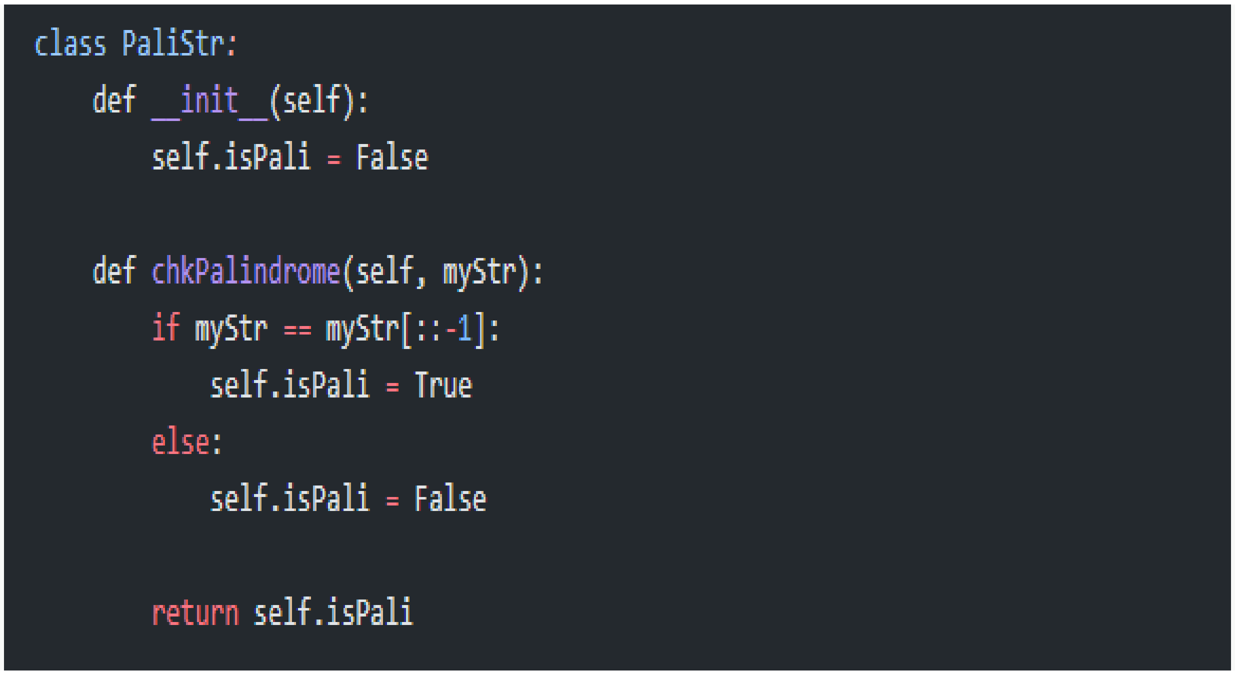


Output

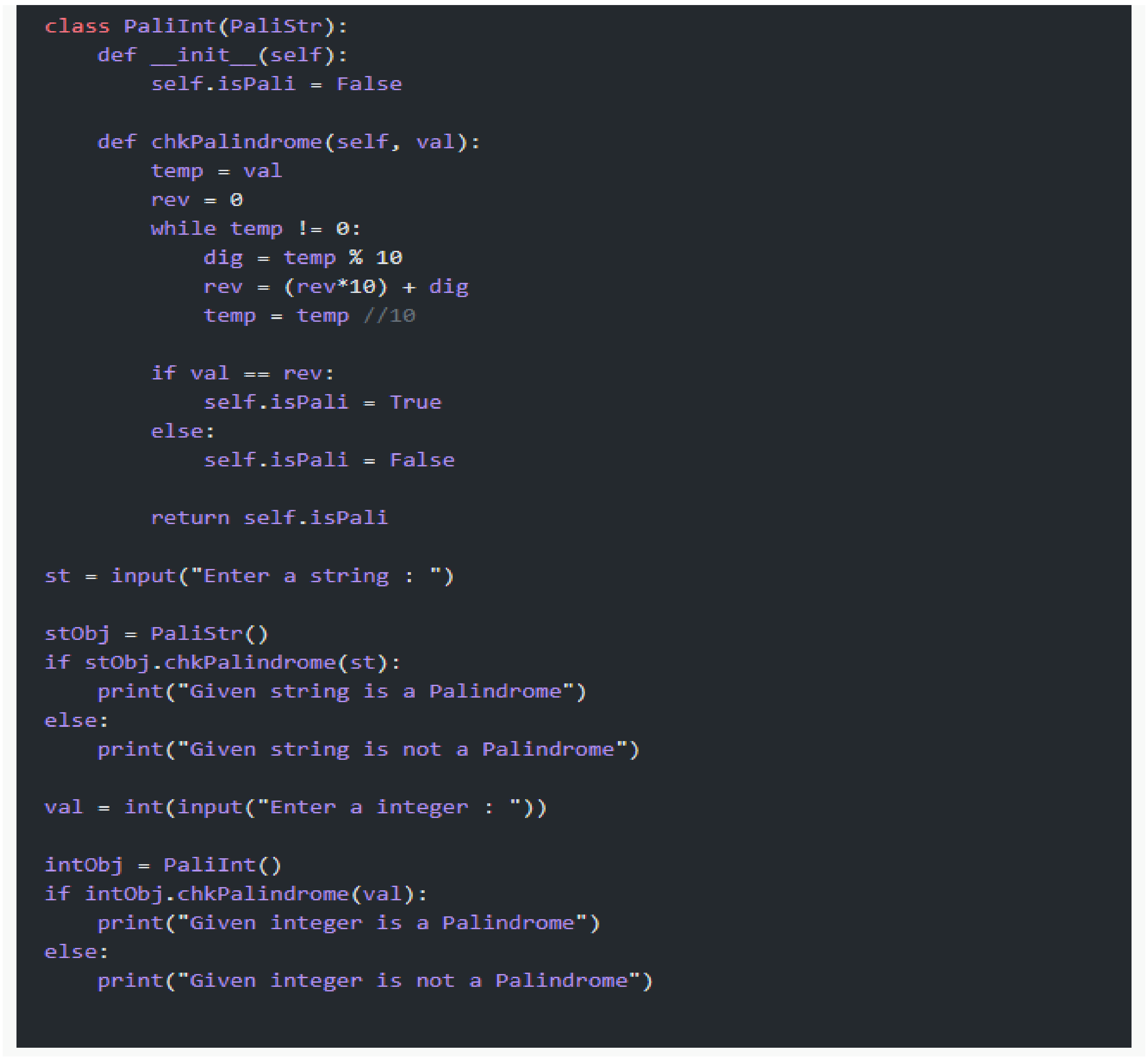
```
Enter Employee name : Sameer
Enter Employee ID : A123
Enter Employee Dept : CSE
Enter Employee Salary : 85750

Employee Details
Name :  Sameer
ID :  A123
Dept :  CSE
Salary :  85750

Enter new Salary : 88800
Updated Salary 88800
```

Aim: Demonstration of classes and methods with polymorphism and overriding

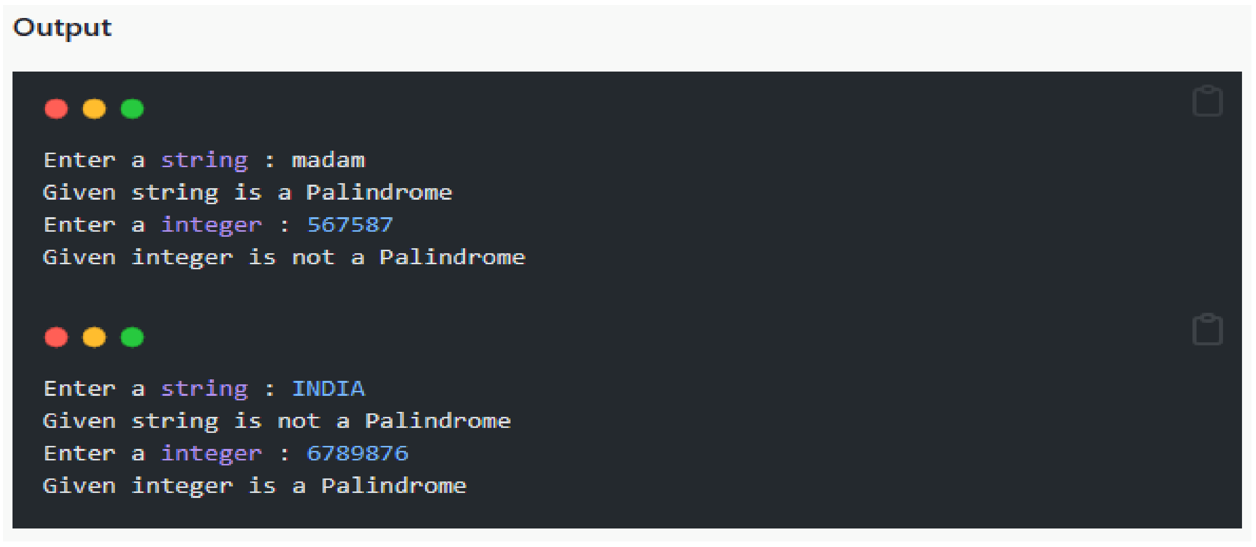
8. a) Write a python program to find the whether the given input is palindrome or not (for both string and integer) using the concept of polymorphism and inheritance.



```python
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")
```



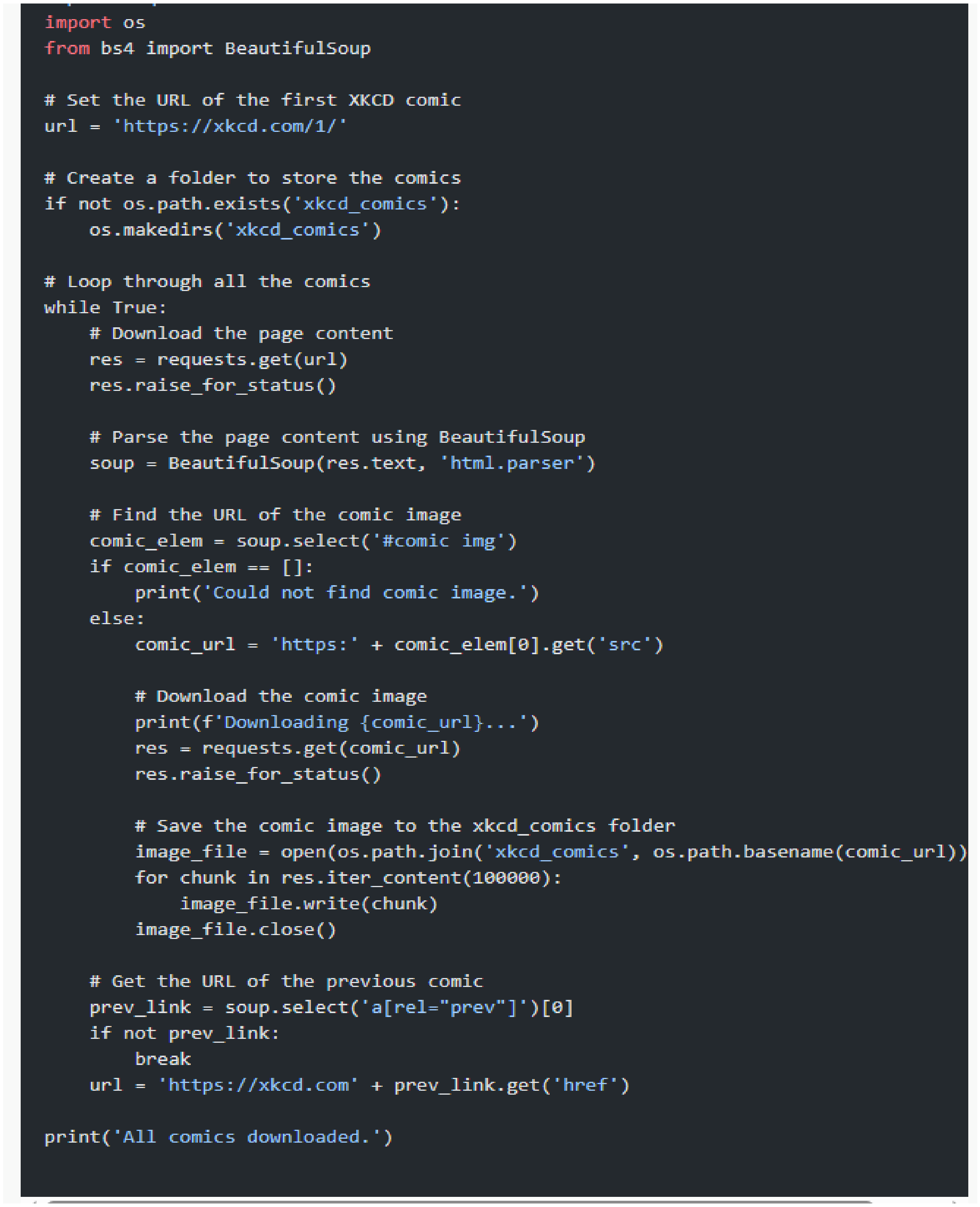
Output

```
Enter a string : madam
Given string is a Palindrome
Enter a integer : 567587
Given integer is not a Palindrome
```

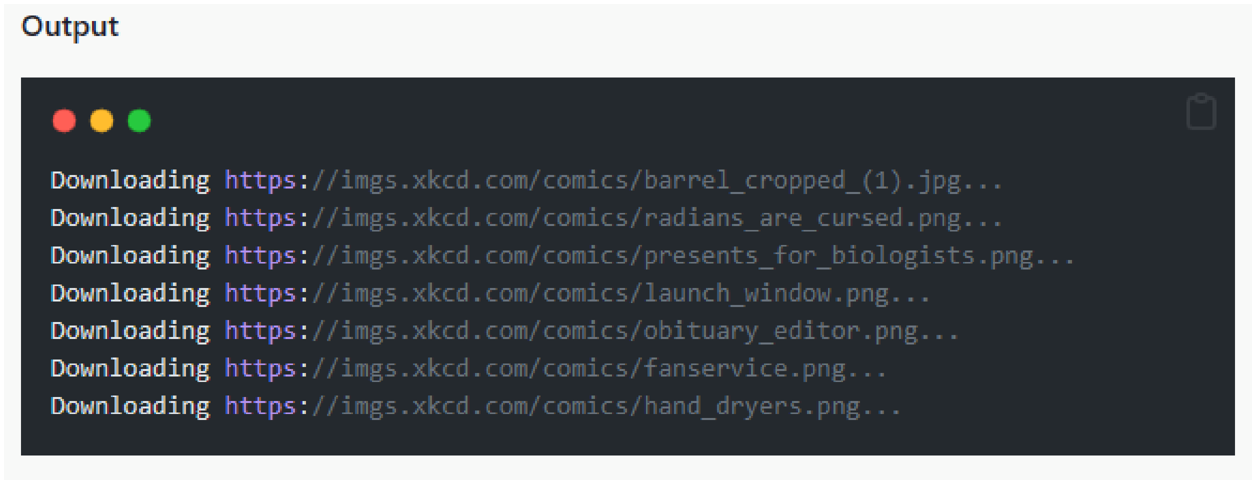
```
Enter a string : INDIA
Given string is not a Palindrome
Enter a integer : 6789876
Given integer is a Palindrome
```

Aim: Demonstration of working with excel spreadsheets and web scraping

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



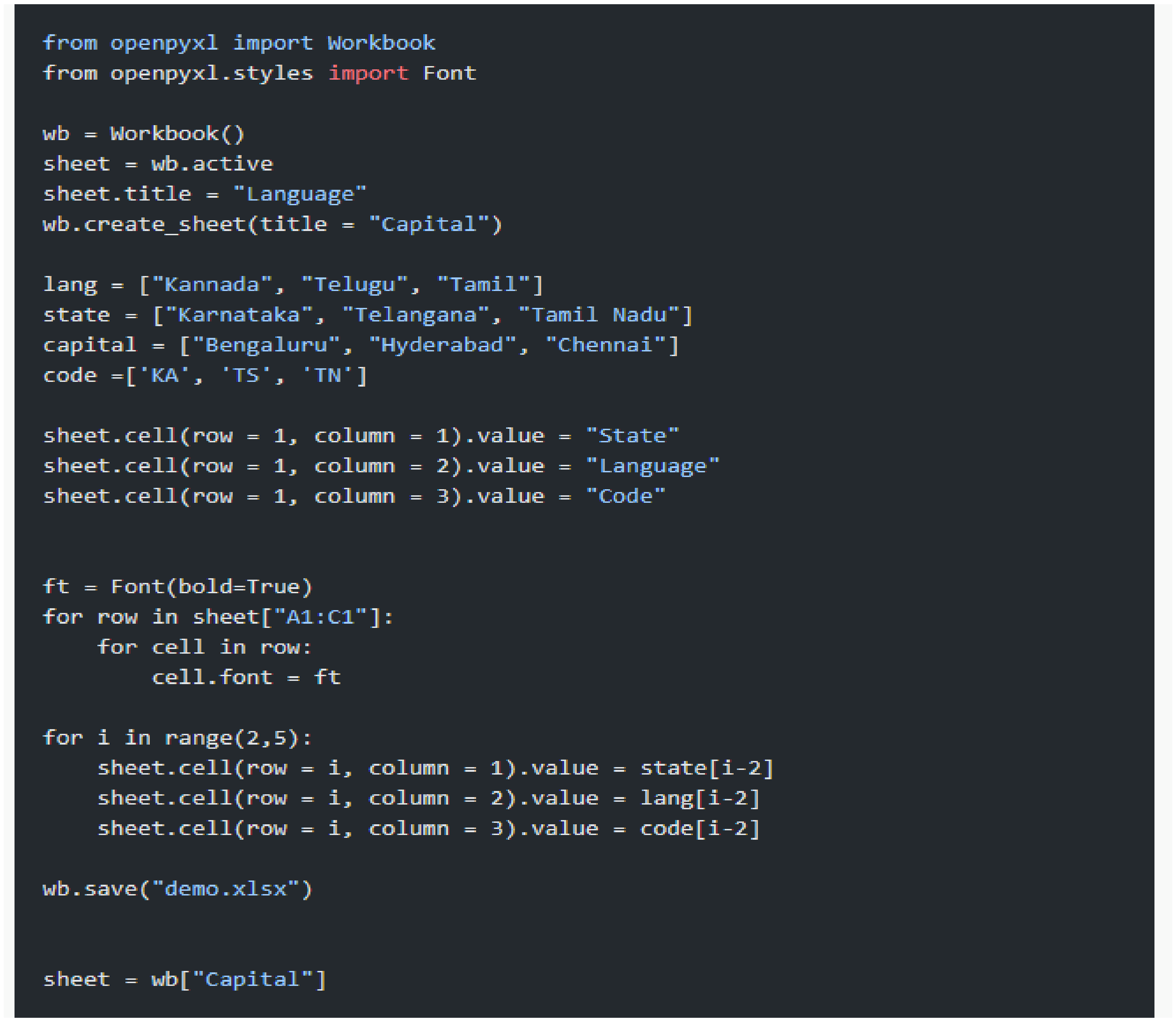
```python
import os
from bs4 import BeautifulSoup

# 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))
        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.')
```



Output

```
Downloading https://imgs.xkcd.com/comics/barrel_cropped_(1).jpg...
Downloading https://imgs.xkcd.com/comics/radians_are_cursed.png...
Downloading https://imgs.xkcd.com/comics/presents_for_biologists.png...
Downloading https://imgs.xkcd.com/comics/launch_window.png...
Downloading https://imgs.xkcd.com/comics/obituary_editor.png...
Downloading https://imgs.xkcd.com/comics/fanservice.png...
Downloading https://imgs.xkcd.com/comics/hand_dryers.png...
```

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

wb = Workbook()
sheet = wb.active
sheet.title = "Language"
wb.create_sheet(title = "Capital")

lang = ["Kannada", "Telugu", "Tamil"]
state = ["Karnataka", "Telangana", "Tamil Nadu"]
capital = ["Bengaluru", "Hyderabad", "Chennai"]
code =['KA', 'TS', 'TN']

sheet.cell(row = 1, column = 1).value = "State"
sheet.cell(row = 1, column = 2).value = "Language"
sheet.cell(row = 1, column = 3).value = "Code"


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("demo.xlsx")


sheet = wb["Capital"]
```

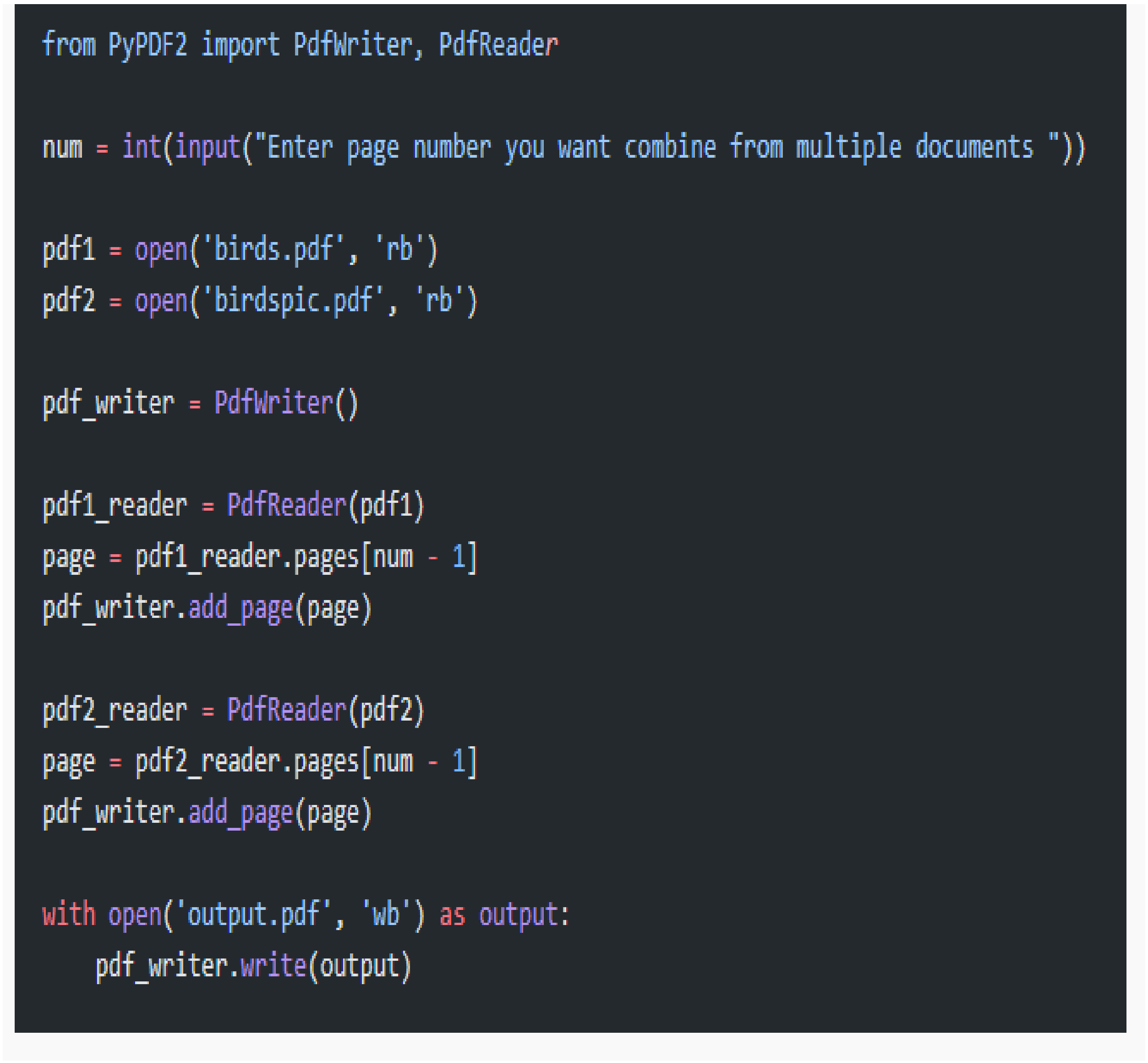
```
sheet.cell(row = 1, column = 1).value = "State"
sheet.cell(row = 1, column = 2).value = "Capital"
sheet.cell(row = 1, column = 3).value = "Code"

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

wb.save("demo.xlsx")

srchCode = input("Enter state code for finding capital ")
for i in range(2,5):
    data = sheet.cell(row = i, column = 3).value
    if data == srchCode:
        print("Corresponding capital for code", srchCode, "is", sheet.cell(row = i


sheet = wb["Language"]

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 =

wb.close()
```

**Output**

```
Enter state code for finding capital KA
Corresponding capital for code KA is Bengaluru
Enter state code for finding language TS
Corresponding language for code TS is Telugu
```

Aim: Demonstration of working with PDF, word and JSON files

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

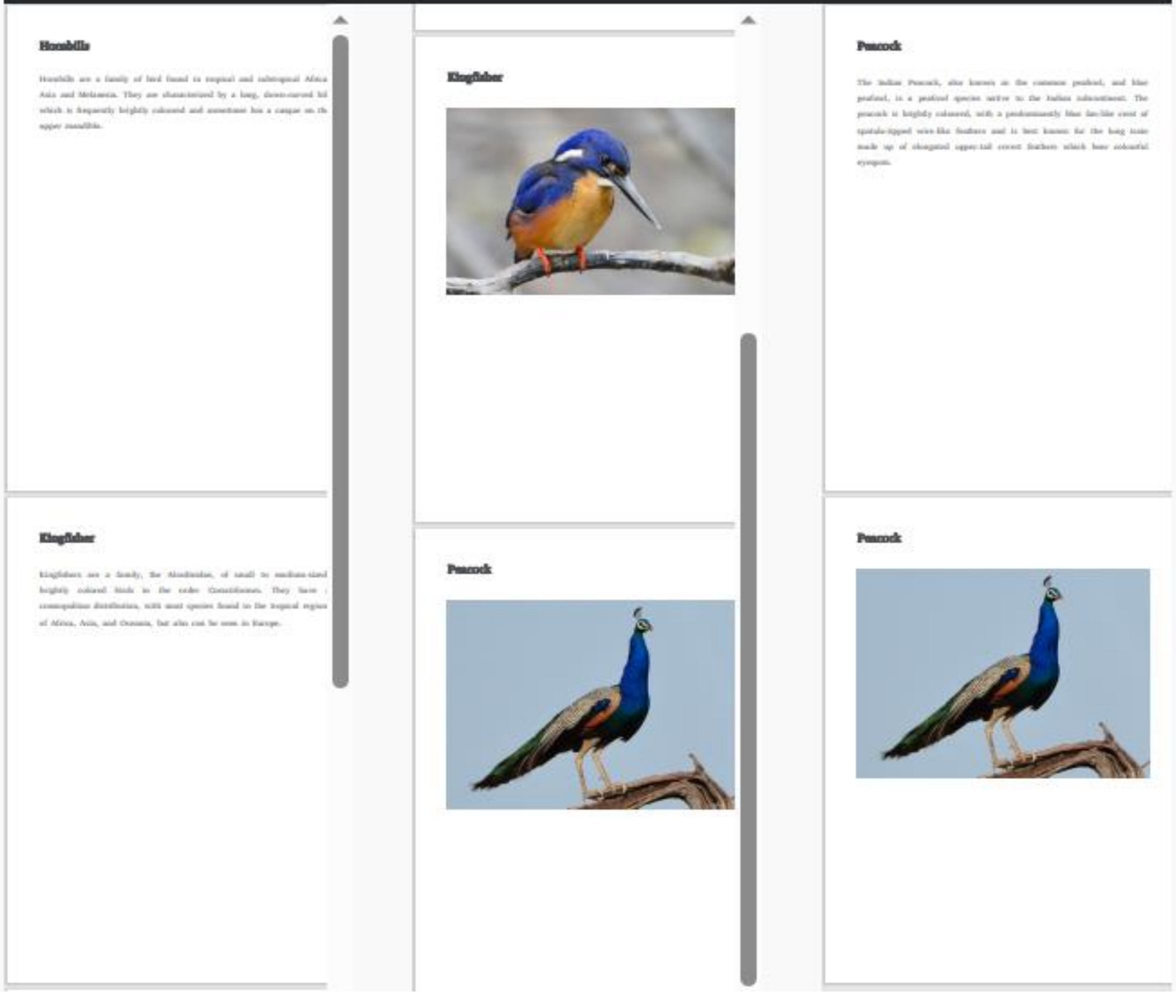


```python
from PyPDF2 import PdfWriter, PdfReader

num = int(input("Enter page number you want combine from multiple documents "))

pdf1 = open('birds.pdf', 'rb')
pdf2 = open('birdspic.pdf', 'rb')

pdf_writer = PdfWriter()

pdf1_reader = PdfReader(pdf1)
page = pdf1_reader.pages[num - 1]
pdf_writer.add_page(page)

pdf2_reader = PdfReader(pdf2)
page = pdf2_reader.pages[num - 1]
pdf_writer.add_page(page)

with open('output.pdf', 'wb') as output:
    pdf_writer.write(output)
```

**Output**

This program allows you to extract specific pages from two PDF files, “birds.pdf” and “birdspic.pdf,” by entering the page numbers as user input. Once you input the desired page numbers, the program fetches those pages from both PDF files and combines them into a new file called “output.pdf.” This way, you can easily compile the desired pages from multiple PDF files into one document for your convenience.

```
Enter page number you want combine from multiple documents 3
```



**Hornbills**

Hornbills are a family of bird found in tropical and subtropical Africa Asia and Melanesia. They are characterized by a long, down-curved bil which is frequently brightly coloured and sometimes has a casque on th upper mandible.

**Kingfisher**

Kingfishers are a family, the Alcedinidae, of small to medium-sized brightly colored birds in the order Coraciiformes. They have : cosmopolitan distribution, with most species found in the tropical region of Africa, Asia, and Oceania, but also can be seen in Europe.

**Kingfisher**

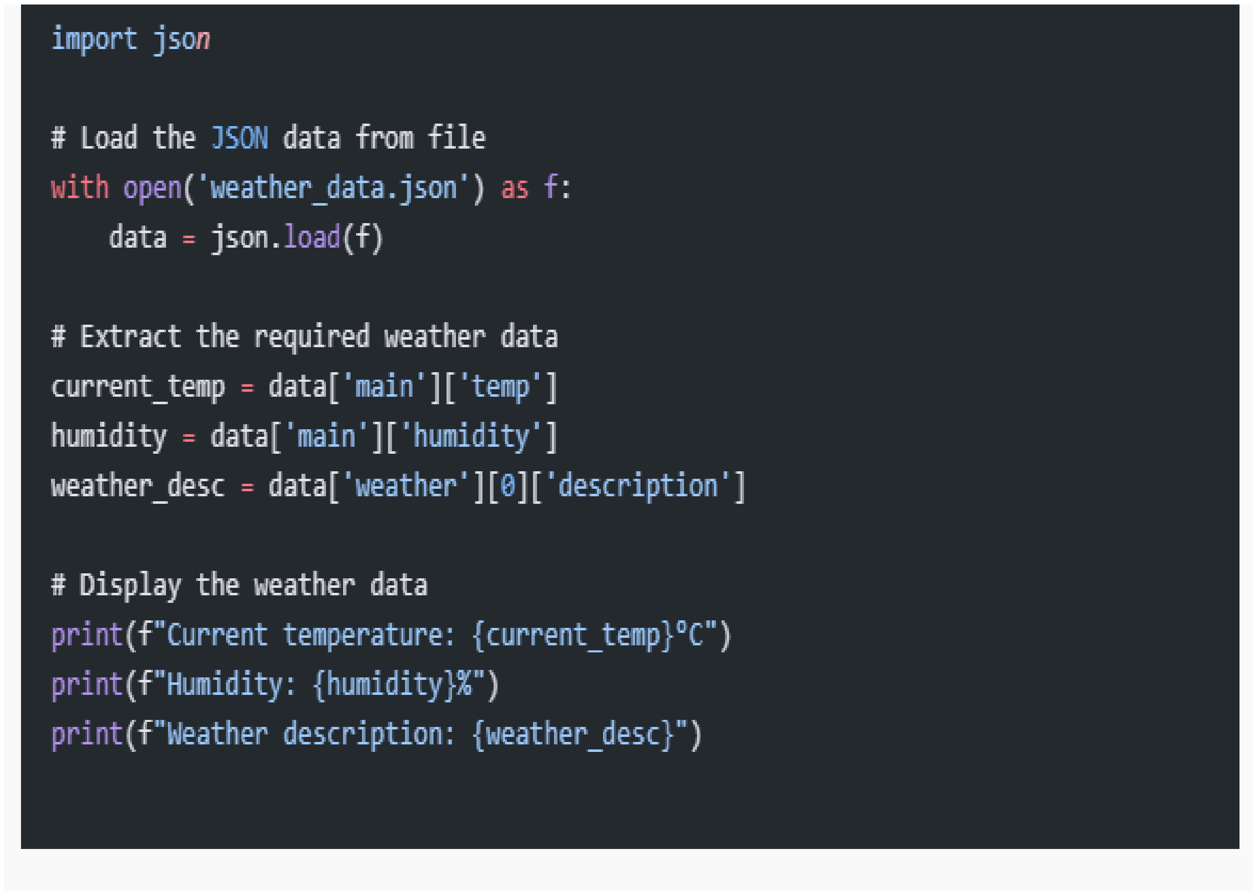
**Peacock**

**Peacock**

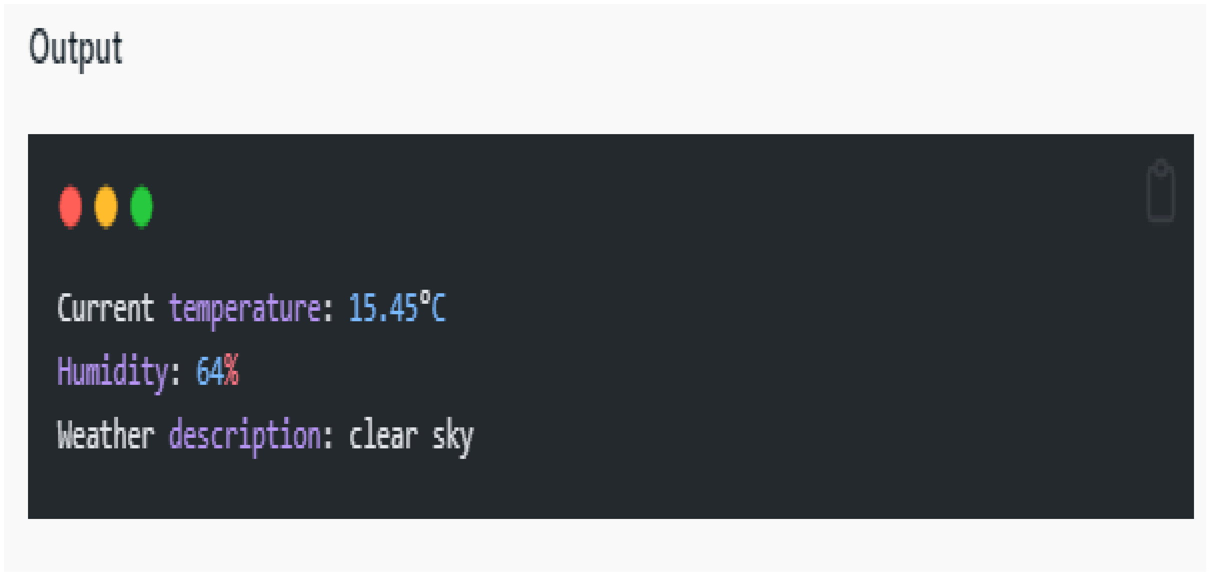
The Indian Peacock, also known as the common peafowl, and blue peafowl, is a peafowl species native to the Indian subcontinent. The peacock is brightly coloured, with a predominantly blue fan-like crest of spatula-tipped wire-like feathers and is best known for the long train made up of elongated upper-tail covert feathers which have colourful eyespots.

**Peacock**

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



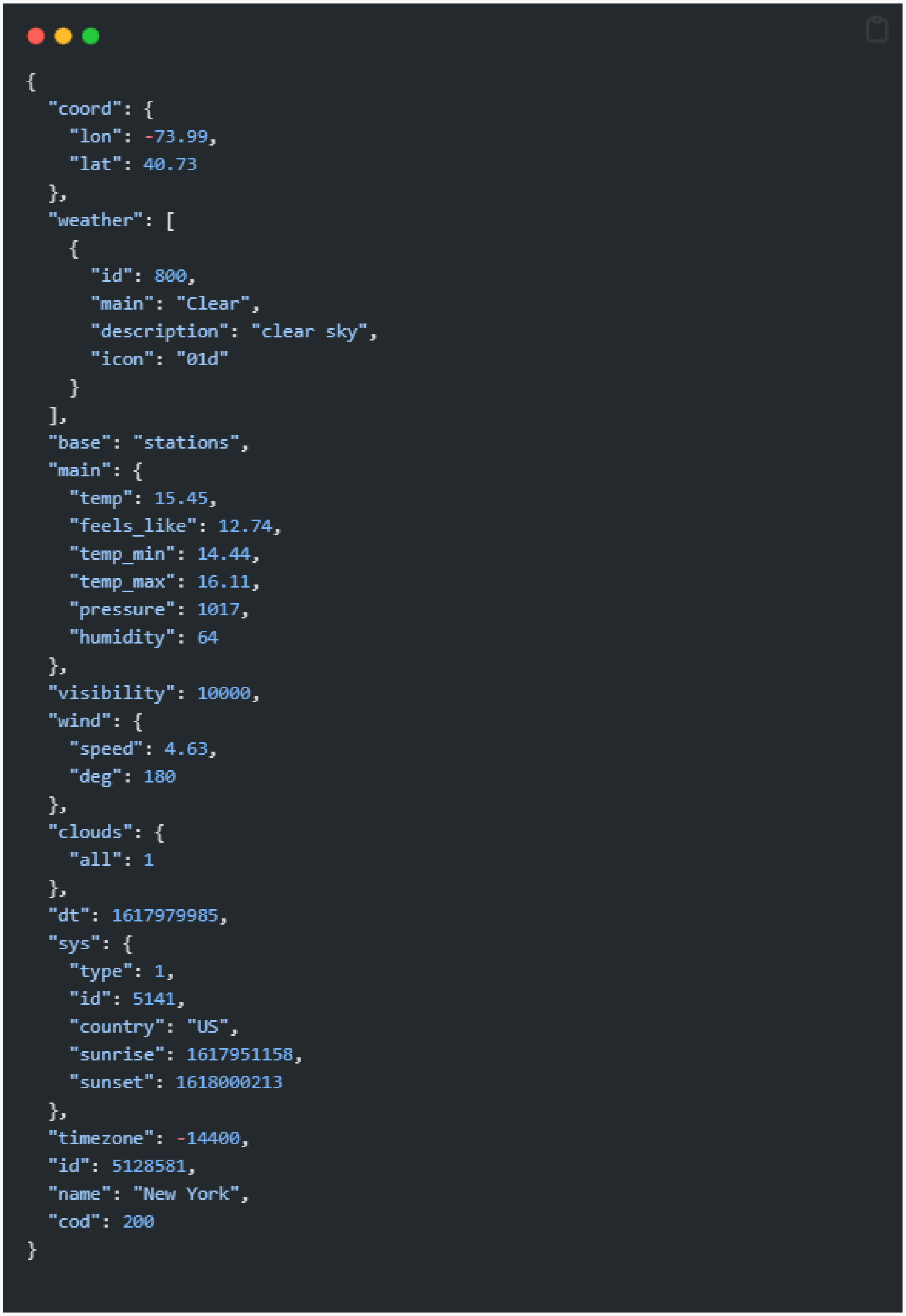
```python
import json

# Load the JSON data from file
with open('weather_data.json') 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
}
```