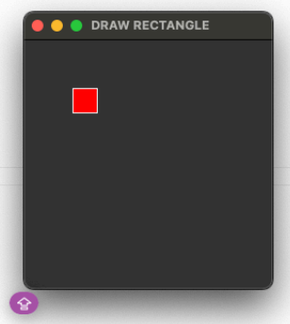
Week - 6

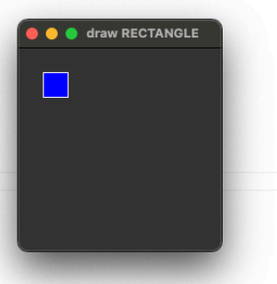
1. Write a function called draw_rectangle that takes a Canvas and Rectangle as arguments and draws a representation of the Rectangle on the Canvas.

```python
import tkinter as tk

class Rectangle:
    def __init__(self,x1,y1,x2,y2):
        self.x1=x1
        self.y1=y1
        self.x2=x2
        self.y2=y2

'''The function that draws a representation of the Rectangle on the
Canvas which takes Canvas and Rectangle as arguments'''
def draw_rectangle(canvas,rectangle):
    canvas.create_rectangle(rectangle.x1, rectangle.y1, rectangle.x2, rectangle.y2,
                            outline="white",fill="red")

# Main
root = tk.Tk()
root.title("DRAW RECTANGLE")
canvas = tk.Canvas(root, width='250', height='250')
canvas.pack()
rect = Rectangle(75,75,50,50)
draw_rectangle(canvas,rect)
root.mainloop()
```



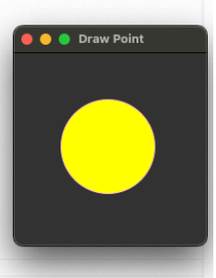
2. Add an attribute named color to your Rectangle objects and modify draw_rectangle so that it uses the color attribute as the fill color.

```python
import tkinter as tk

class Rectangle:
    def __init__(self,x1,y1,x2,y2,  color="blue"):
        self.x1=x1
        self.y1=y1
        self.x2=x2
        self.y2=y2
        self.color = color # NEW CLASS VARIABLE

def draw_rectangle(canvas,rectangle):
    canvas.create_rectangle(rectangle.x1,rectangle.y1,rectangle.x2,rectangle.y2,
                            outline="white",fill=rectangle.color) # CLASS VARIABLE IS CALLED

# MAIN
root = tk.Tk()
root.title("draw RECTANGLE")
canvas2 = tk.Canvas(root, width='200', height='200')
canvas2.pack()
rect2 = Rectangle(25,50,50,25)
draw_rectangle(canvas2,rect2)
root.mainloop()
```



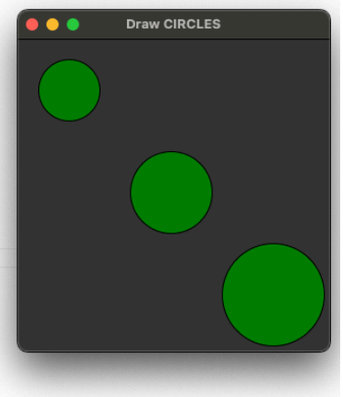
3. Write a function called draw_point that takes a Canvas and a Point as arguments and draws a representation of the Point on the Canvas.

```python
import tkinter as tk

class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

''' The function takes in canvas, point and radius as arguments
Points are A (x1, y1) and B (x2, y2).
The create_oval() method gives oval, but gives a circle if given equal coordinates.
The oval will be drawn between the top (x1) left (y1) and bottom (x2) right (y2) coordinates.
If the difference between the top to bottom and left to right
is the same then a circle will be drawn.'''
def draw_point(canvas, point, radius=50):
    x1, y1 = point.x - radius, point.y - radius
    x2, y2 = point.x + radius, point.y + radius
    # For a circle, ensure (x2 - x1) == (y2 - y1).
    canvas.create_oval(x1, y1, x2, y2, outline="pink", fill="yellow")

# MAIN
root = tk.Tk()
root.title("Draw Point")
canvas = tk.Canvas(root, width=200, height=200)
canvas.pack()
point = Point(100, 100)
draw_point(canvas, point)
root.mainloop()
```



4. Define a new class called Circle with appropriate attributes and instantiate a few Circle objects. Write a function called draw_circle that draws circles on the canvas.

```python
import tkinter as tk

class Circle:
    def __init__(self, x, y, radius):   # Class attributes/variables
        self.x = x
        self.y = y
        self.radius = radius

def draw_circle(canvas, circle):    # Drawing a circle
    x1, y1 = circle.x - circle.radius, circle.y - circle.radius
    x2, y2 = circle.x + circle.radius, circle.y + circle.radius
    canvas.create_oval(x1, y1, x2, y2, outline="black", fill="green")

def draw_circles(canvas, circles):  # Drawing all the three circles
    for circle in circles:
        draw_circle(canvas, circle)

# Main
root = tk.Tk()
root.title("Draw CIRCLES")
canvas = tk.Canvas(root, width=300, height=300)
canvas.pack()
circles = [ Circle(50, 50, 30),
            Circle(150, 150, 40),
            Circle(250, 250, 50) ]   # Circle objects
draw_circles(canvas, circles)
root.mainloop()
```



5. Write a Python program to demonstrate the usage of Method Resolution Order (MRO) in multiple levels of Inheritances.

```python
'''The super() function is used to give access to
methods and properties of a parent or sibling class.
It returns an object that represents the parent class and
allows method resolution following the MRO.
'''

class A:    # Define class A
    def method(self):
        print("method in A")  # class A function

class B(A): # Define class B that inherits from A
    def method(self):
        print("method in B")  # class B function
        super().method()      # Call method from superclass (A)

class C(A): # Define class C that also inherits from A
    def method(self):
        print("method in C")  # class C function
        super().method()      # Call method from superclass (A)

class D(B, C):  # Define class D that inherits from both B and C
    def method(self):
        print("method in D")  # class D function
        super().method()      # Call next method in MRO (B -> C -> A)

d = D()  # Object/Instance of class D is created

d.method()  # Executes D.method() → B.method() → C.method() → A.method()

# Shows the order in which Python resolves method calls for class D
print("MRO in D: ", D.mro())

''' EXPECTED OUPUT
method in B
method in C
method in A
MRO in D:  [<class '__main__.D'>, <class '__main__.B'>, <class '__main__.C'>, <class '__main__.A'>, <class 'object'>]
'''
```

6. Write a python code to read a phone number and email-id from the user and validate it for correctness.

```python
import re

def validate_phone_number(phone_number):
    # regular expression matching a phone number
    pattern = r'^\d{10}$'
    if re.match(pattern, phone_number):
        return True
    else:
        return False

def validate_email(email):
    # regular expression matching a phone number
    pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
    if re.match(pattern, email):
        return True
    else:
        return False

# Main
phone_number = input("Enter your Phone number? ")
print(f"PHONE NUMBER entered: {phone_number}")

if validate_phone_number(phone_number):
    print("Phone number is valid.")
else:
    print("Phone number is invalid.")

email = input("Enter your email address? ")
print(f"EMAIL ADDRESS entered: {email}")
if validate_email(email):
    print("Email address is valid.")
else:
    print("Email address is invalid.")
```

Week - 7

7. Write a Python code to merge two given file contents into a third file.

```python
with open("file1", "w") as fp1: # Write to file1
    data1 = "Hello"
    fp1.write(data1)

with open("file2", "w") as fp2: # Write to file2
    data2 = "Aneetta"
    fp2.write(data2)

with open("file1", "r") as fp1: # Read and print contents of file1
    print("Content in file 1: ", fp1.read())

with open("file2", "r") as fp2: # Read and print contents of file2
    print("Content in file 2: ", fp2.read())

# Merge into file3
with open("file1", "r") as fp1, open("file2", "r") as fp2, open("file3", "w") as fp3:
    fp3.write(fp1.read() + " " + fp2.read())

with open("file3", "r") as fp3: # Print merged content
    print("Merged Content in file 3: ", fp3.read())

''' EXPECTED OUTPUT:
Content in file 1:  Hello
Content in file 2:  Aneetta
Merged Content in file 3:  Hello Aneetta
'''
```

8. Write a Python code to open a given file and construct a function to check for given words present in it and display on found.

```python
def checkWordInFile(file, word: str) -> bool:
    try:
        with open(file, "r") as file1:
            data = file1.read()
            return word in data
    except FileNotFoundError:
        print("File not found")
        return False

# Create sample file
with open("file1", "w") as fp1:
    data = '''And now these three remain: faith, hope and love.
    But the greatest of these is love.'''
    fp1.write(data)

# Check for the word 1
if checkWordInFile("file1", "love"):
    print("Word 'love' found")
else:
    print("Word 'love' not found")

# Check for the word 2
if checkWordInFile("file1", "hate"):
    print("Word 'hate' found")
else:
    print("Word 'hate' not found")

''' EXPECTED OUTPUT:
Word 'love' found
Word 'hate' not found
'''
```

9. Write a Python code to Read text from a text file, find the word with the most number of occurrences.

```python
def mostOccurences(file):
    try:
        with open(file, "r") as fp1:
            data = fp1.read()
            occurrences = dict()
            for word in data.split():
                occurrences[word] = occurrences.get(word, 0) + 1
            most_common = max(occurrences, key=occurrences.get)
            print("Occurrences:", occurrences)
            print("Most frequent word:", most_common)
            return most_common
    except FileNotFoundError:
        print("File not found")

# Create sample file
with open("file1", "w") as fp1:
    data = '''A teacher works tirelessly to help students succeed.
            A great teacher listens, understands, and encourages.
            Without a teacher, learning would be difficult.
            Every student remembers at least one teacher who made a difference.'''
    fp1.write(data)

# Call function
mostOccurences("file1")

'''EXPECTED OUTPUT:
Occurrences: {'A': 2, 'teacher': 3, 'works': 1, 'tirelessly': 1,
'to': 1, 'help': 1, 'students': 1, 'succeed.': 1, 'great': 1,
'listens,': 1, 'understands,': 1, 'and': 1, 'encourages.': 1,
'Without': 1, 'a': 2, 'teacher,': 1, 'learning': 1, 'would': 1,
'be': 1, 'difficult.': 1, 'Every': 1, 'student': 1, 'remembers': 1,
'at': 1, 'least': 1, 'one': 1, 'who': 1, 'made': 1, 'difference.': 1}

Most frequent word: teacher
'''
```

10. Write a function that reads a file file1 and displays the number of words, number of vowels, blank spaces, lower case letters and uppercase letters.

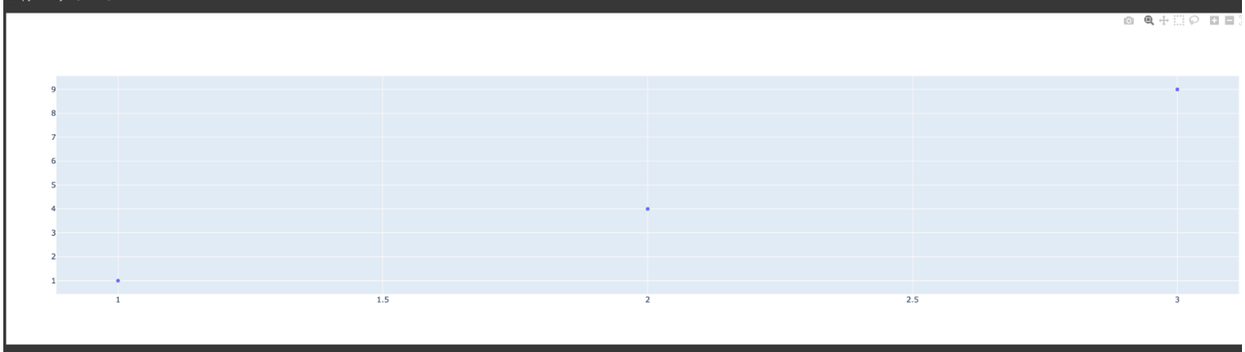
```python
def analyzeFile(file) -> list[int]:
    try:
        with open(file, "r") as fp1:
            data = fp1.read()
            noOfWords = len(data.split())
            noOfVowels = len([char for char in data.lower() if char in "aeiou"])
            blankSpaces = data.count(" ")
            lowerCaseLetters = len([char for char in data if char.islower()])
            upperCaseLetters = len([char for char in data if char.isupper()])
            return [noOfWords, noOfVowels, blankSpaces, lowerCaseLetters, upperCaseLetters]
    except FileNotFoundError:
        print("File not Found")
        return []

# Create sample file
with open("file1", "w") as fp1:
    data = '''Over the past decade, IITs Bombay and Delhi, and
    Indian Institute of Science (IISc) Bangalore, have stood in
    the top three positions among Indian institutions.'''
    fp1.write(data)

# Main
result = analyzeFile("file1")
print("Words:", result[0])
print("Vowels:", result[1])
print("Spaces:", result[2])
print("Lowercase Letters:", result[3])
print("Uppercase Letters:", result[4])

'''EXPECTED OUTPUT:
Words: 25
Vowels: 55
Spaces: 32
Lowercase Letters: 115
Uppercase Letters: 14
'''
```

Week - 8

11. Import Numpy, Plotpy and Scipy and explore their functionalities.

```python
import numpy as np
import plotly.graph_objects as go
from scipy import stats

# Numpy
array = np.array([1, 2, 3])
print("Numpy array:", array)
print()

# Plotly
fig = go.Figure(data=go.Scatter(x=[1, 2, 3], y=[1, 4, 9], mode='markers'))
fig.show()
print()

# Scipy
data = [1, 2, 3, 4, 5]
mean = stats.tmean(data)
print("Mean of data using Scipy:", mean)
```

```
Numpy array: [1 2 3]
```




```
Mean of data using Scipy: 3.0
```

12. Install Numpy package with pip and explore it.

```python
!pip install numpy

import numpy as np

array = np.array([1, 2, 3])
print("Numpy array:", array)
print("Array mean:", np.mean(array))
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (2.0.2)
Numpy array: [1 2 3]
Array mean: 2.0
```

13. Write a program to implement Digital Logic Gates –AND, OR, NOT, EX-OR.

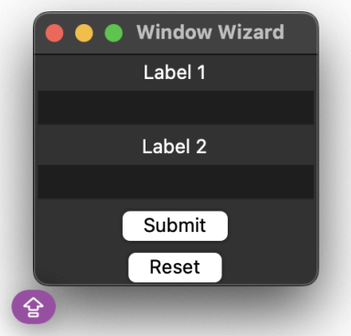
```python
def AND(a, b):  #AND gate: Returns 1 only when both inputs are 1
    return int(a and b)

def OR(a, b):   #OR gate: Returns 1 when at least one input is 1
    return int(a or b)

def NOT(a): #NOT gate: Returns opposite of input
    return int(not a)

def XOR(a, b):  #XOR gate: Returns 1 when inputs are different
    return int(a ^ b)

def NAND(a, b): #NAND gate: NOT AND - opposite of AND gate
    return int(not (a and b))

def NOR(a, b):  #NOR gate: NOT OR - opposite of OR gate
    return int(not (a or b))

        # Main
print("DIGITAL LOGIC GATES \n")
print("Enter two binary inputs to see all gate outputs")
        # Get user inputs
A = int(input("Enter A (0 or 1): "))
B = int(input("Enter B (0 or 1): "))
        # Calculate all gate outputs
and_result = AND(A, B)
or_result = OR(A, B)
xor_result = XOR(A, B)
nand_result = NAND(A, B)
nor_result = NOR(A, B)
not_a = NOT(A)
not_b = NOT(B)
        # Print single row truth table format
print()
print("| A | B | AND | OR | XOR | NAND | NOR | NOT A | NOT B |")
print(f"| {A} | {B} |  {and_result}  |  {or_result} |  {xor_result}  |   {nand_result}  |  {nor_result}  |   {not_a}   |   {not_b}   |")

'''EXPECTED OUTPUT
DIGITAL LOGIC GATES

Enter two binary inputs to see all gate outputs
Enter A (0 or 1): 1
Enter B (0 or 1): 1

| A | B | AND | OR | XOR | NAND | NOR | NOT A | NOT B |
| 1 | 1 |  1  |  1 |  0  |   0  |  0  |   0   |   0   |

DIGITAL LOGIC GATES

Enter two binary inputs to see all gate outputs
Enter A (0 or 1): 0
Enter B (0 or 1): 1

| A | B | AND | OR | XOR | NAND | NOR | NOT A | NOT B |
| 0 | 1 |  0  |  1 |  1  |   1  |  0  |   1   |   0   |
'''
```

14. Write a program to implement Half Adder, Full Adder,and Parallel Adder.

```python
def half_adder(a, b):
    """Half Adder: XOR for sum, AND for carry"""
    sum_ = a ^ b  # XOR
    carry = a & b  # AND
    return sum_, carry

def full_adder(a, b, c):
    """Full Adder: Uses two half adders"""
    sum1, carry1 = half_adder(a, b)  # First half adder
    sum2, carry2 = half_adder(sum1, c)  # Second half adder
    carry_out = carry1 | carry2  # OR the carries
    return sum2, carry_out

# Main - Truth Table Testing

print("HALF ADDER TRUTH TABLE:")
print("A | B | Sum | Carry")
print("--|---|-----|------")

'''Half adder truth table entries :
[(0,0), (0,1), (1,0), (1,1)] '''
ah, bh = 0, 1
sum_, carry = half_adder(ah, bh)
print(f"{ah} | {bh} |  {sum_}  |   {carry}")

print("\nFULL ADDER TRUTH TABLE:")
print("A | B | C | Sum | Carry")
print("--|---|---|-----|------")

'''Full adder truth table entries :
[(0,0,0), (0,0,1), (0,1,0), (0,1,1),
(1,0,0), (1,0,1), (1,1,0), (1,1,1)] '''
a, b, c = 1, 0, 1
sum_, carry = full_adder(a, b, c)
print(f"{a} | {b} | {c} |  {sum_}  |   {carry}")

'''EXPECTED OUTPUT
HALF ADDER TRUTH TABLE:
A | B | Sum | Carry
--|---|-----|------
0 | 1 |  1  |   0

FULL ADDER TRUTH TABLE:
A | B | C | Sum | Carry
--|---|---|-----|------
1 | 0 | 1 |  0  |   1
'''
```

15. Write a GUI program to create a window wizard having two text labels, two text fields and two buttons as Submit and Reset.

```python
from tkinter import Tk, Label, Entry, Button

def submit():
    print("Submitted")
def reset():
    entry1.delete(0, 'end')
    entry2.delete(0, 'end')

root = Tk()
root.title("Window Wizard")
label1 = Label(root, text="Label 1")
label1.pack()
entry1 = Entry(root)
entry1.pack()
label2 = Label(root, text="Label 2")
label2.pack()
entry2 = Entry(root)
entry2.pack()
submit_button = Button(root, text="Submit", command=submit)
submit_button.pack()
reset_button = Button(root, text="Reset", command=reset)
reset_button.pack()
root.mainloop()
```



More similar examples in detail for revision:

Following are some important Linux/Unix commands which will be useful for you.

- ➔ cat -- Display File Contents
- ➔ cd -- Changes Directory to dirname
- ➔ chmod -- Changing Permissions
- ➔ cp -- Copy source file into destination
- ➔ file -- Determine file type
- ➔ find -- Find files
- ➔ grep -- Search files for regular expressions.
- ➔ head -- Display first few lines of a file
- ➔ ls -- Display information about file type.
- ➔ mkdir -- Create a new directory dirname
- ➔ mv -- Move (Rename) a oldname to newname.
- ➔ pwd -- Print current working directory.
- ➔ rm -- Remove (Delete) filename
- ➔ rmdir -- Delete an existing directory provided it is empty.
- ➔ tail -- Prints the last few lines in a file.
- ➔ touch -- Update access and modification time of a file.

★ **For TKINTER to be installed in your local system:**

- Go to the home directory.
  - Type `pwd` in terminal and press enter. It should give /home as result
- Check whether you have Python3 version:
  - Type `python3 -version` in terminal and press enter.
  - If not installed, `sudo apt upgrade` and press enter then `sudo apt update` and press enter and at last type `sudo apt install python3` and press enter.
- Install Tkinter for Python3
  - Type `sudo apt install python3-tk` in terminal and press enter.
  - If already installed, Type the below sample program in the terminal and press enter to see if a GUI window pops up. `python3 -c "import tkinter; tkinter.Tk().mainloop()"`
