Homework #10

01286121 Computer Programming

Software Engineering Program,

Department of Computer Engineering,

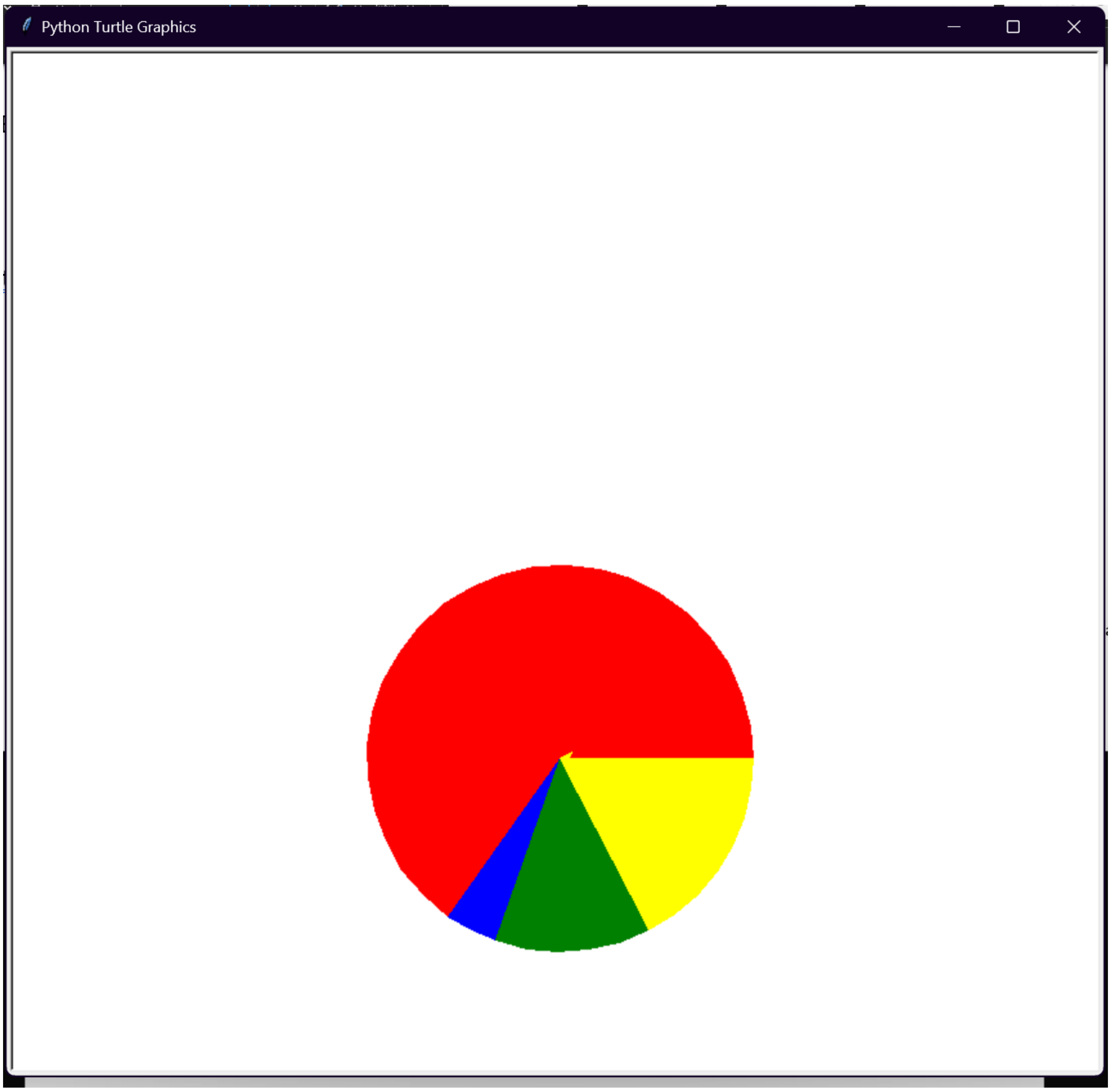
School of Engineering, KMITL

By

66011149 Phatthadon Sornplang

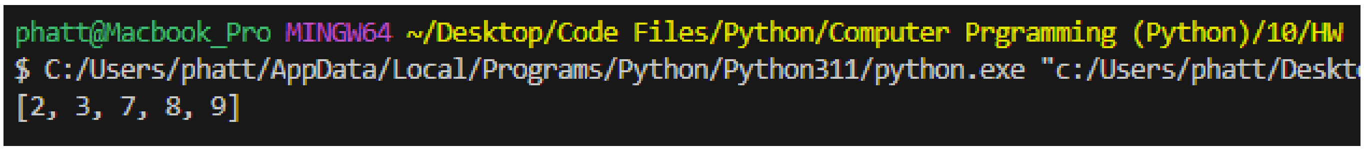
1.Write a python function to draw a Piechart according to the number of occurence in each integer lists.

```
import turtle

def draw_pie_chart(list1):
    count_dict = {}

    for num in list1:
        if num in count_dict:
            count_dict[num] += 1
        else:
            count_dict[num] = 1

    counts = list(count_dict.values())
    total = sum(counts)

    pie_turtle = turtle.Turtle()
    pie_turtle.speed(10)
    colors = ['red', 'blue', 'green', 'yellow', 'purple', 'orange', 'cyan']

    pie_turtle.penup()
    pie_turtle.goto(0, -150)
    pie_turtle.pendown()

    start_angle = 0
    for i, count in enumerate(counts):
        angle = (count/total) * 360
        pie_turtle.color(colors[i % len(colors)])
        pie_turtle.begin_fill()
        pie_turtle.setheading(start_angle)
        pie_turtle.forward(150)
        pie_turtle.left(90)
        pie_turtle.circle(150, angle)
        pie_turtle.left(90)
        pie_turtle.forward(150)
        pie_turtle.end_fill()
        start_angle += angle


data_list = [3, 1, 3, 3, 2, 3, 3, 2, 3, 2, 4, 3, 3, 3, 3, 4, 3, 4, 3, 3, 3, 4, 3]
draw_pie_chart(data_list)

turtle.done()
```


Python Turtle Graphics

2.Write a Python function to perform a bubble sort of a list.

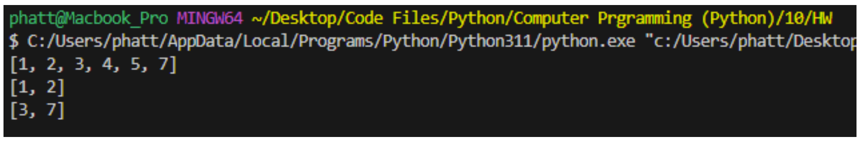
```
def bubble_sort(listin):
    length = len(listin)

    for i in range(length):
        swaps = 0
        for i in range(0, length - 1):
            if listin[i] > listin[i + 1]:
                listin[i], listin[i + 1] = listin[i + 1], listin[i]
                swaps += 1
        if swaps == 0:
            break
        length -= 1

    return listin

list = [3,2,9,7,8]
out = bubble_sort(list)
print(out)
```



```
phatt@Macbook_Pro MINGW64 ~/Desktop/Code Files/Python/Computer Prgramming (Python)/10/HW
$ C:/Users/phatt/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/phatt/Deskt
[2, 3, 7, 8, 9]
```

3.Write a Python three function which, given any two list, return a list which represents a union, intersection, and a difference of the two lists, respectively.

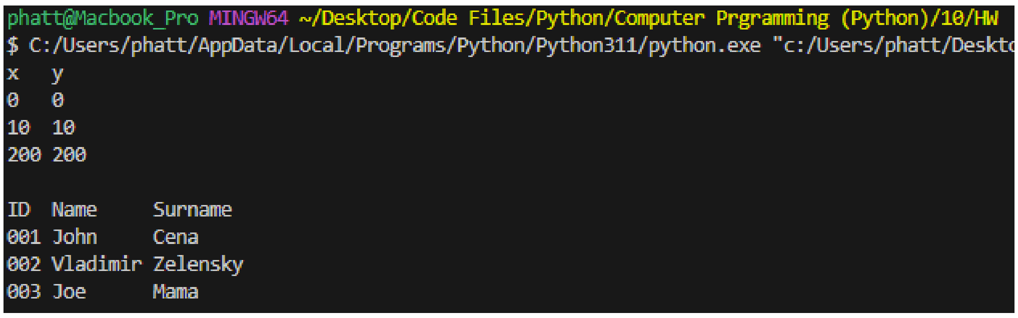
```
def my_union(list1, list2):

    return list(set(list1) | set(list2))


def my_intersection(list1, list2):

    return list(set(list1) & set(list2))


def my_difference(list1, list2):

    return list(set(list1) - set(list2))




list1 = [3, 1, 2, 7]

list2 = [4, 1, 2, 5]


out1 = (my_union(list1, list2))

print(out1)

out2 = my_intersection(list1, list2)

print(out2)

out3 = my_difference(list1, list2)

print(out3)
```



```
phatt@Macbook_Pro MINGW64 ~/Desktop/Code Files/Python/Computer Prgramming (Python)/10/HW
$ C:/Users/phatt/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/phatt/Desktop
[1, 2, 3, 4, 5, 7]
[1, 2]
[3, 7]
```

4.Write a Python function which given a list of a lists representing a table, prints that table on screen. The first member on the list contains a header row which gives the name for each column of the table. The other members represent the other rows of the table, each of them is matched with all the columns of the header.

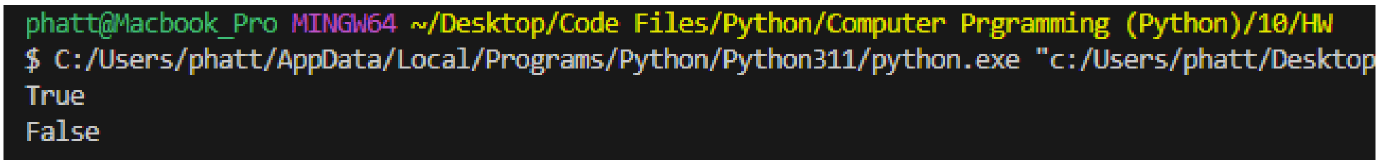
```
def print_table(table):
    if not table:
        print("Table is empty")
        return

    column_width = []
    for col in range(len(table[0])):
        max_width = 0
        for row in table:
            cell_width = len(str(row[col]))
            if cell_width > max_width:
                max_width = cell_width
        column_width.append(max_width)

    for i, header in enumerate(table[0]):
        print(header.ljust(column_width[i]), end=" ")
    print()
    for row in table[1:]:
        for i, cell in enumerate(row):
            print(str(cell).ljust(column_width[i]), end=" ")
        print()
table_data1 = [['x', 'y'], [0, 0], [10, 10], [200, 200]]
print_table(table_data1)
print()
table_data2 = [['ID', 'Name', 'Surname'], ['001', 'John', 'Cena'], ['002', 'Vladimir', 'Zelensky'], ['003', 'Joe', 'Mama']]
print_table(table_data2)
```



```
phatt@Macbook_Pro MINGW64 ~/Desktop/Code Files/Python/Computer Prgramming (Python)/10/HW
$ C:/Users/phatt/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/phatt/Deskt
x   y
0   0
10  10
200 200

ID  Name     Surname
001 John     Cena
002 Vladimir Zelensky
003 Joe      Mama
```

5.Write a Python function isAnagram[String1, String2] that decides whether two words (strings) are anagram. Some two words are anagrams if they contain the same letters regardless of the letters regardless of the letters' positions.

```
def isAnagram(string1, string2):
    count1 = {}
    for char in string1:
        count1[char] = count1.get(char, 0) + 1

    count2 = {}
    for char in string2:
        count2[char] = count2.get(char, 0) + 1

    if count1 == count2:
        return "True"
    else:
        return "False"

word1 = "listen"
word2 = "silent"
out1 = isAnagram(word1, word2)
print(out1)

word1 = "hello"
word2 = "helnaw"
out2 = isAnagram(word1, word2)
print(out2)
```



```
phatt@Macbook_Pro MINGW64 ~/Desktop/Code Files/Python/Computer Prgramming (Python)/10/HW
$ C:/Users/phatt/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/phatt/Desktop
True
False
```