HomeWork_2 (02/Nov./2022)

1. Problem1_A_Loops.py

```python
while True:
    try:
        input_list=list(input("input a list of integers").split(" "))
        print(input_list)
        int_list=[int(i) for i in input_list]
        print(int_list)
        break
    except:
        print("your input must be integer numbers")


#return the list of integers that accept the division by 5:
out_list=[i for i in int_list if i%5==0]

print("The integers that accept division by 5 =",out_list)
```

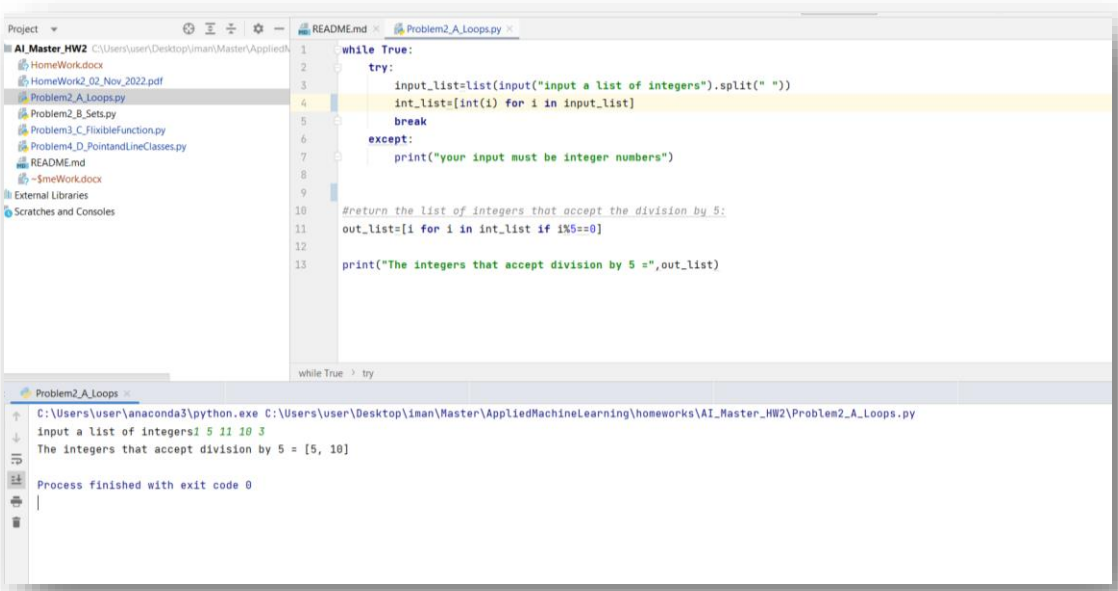

Figure 1: Output Sample of Problem1_A

2. Problem2_B_Sets.py

```python
no_of_lines = 2
lines =[]
try:
    #let the user to input multiple list of integers, convert the list to set find the unique values
    for i in range(no_of_lines):
        input_list=set(input(f"inter list {i+1}").split(" "))
        lines.append(input_list)

except:
    print("please try a gain!")

print(lines)
output=lines[0]
#apply intersection on the all sets entered by the user and find the duplicated keywords:
for n in range(no_of_lines-1):
    output &= lines[n+1]
print("common words",output)
```

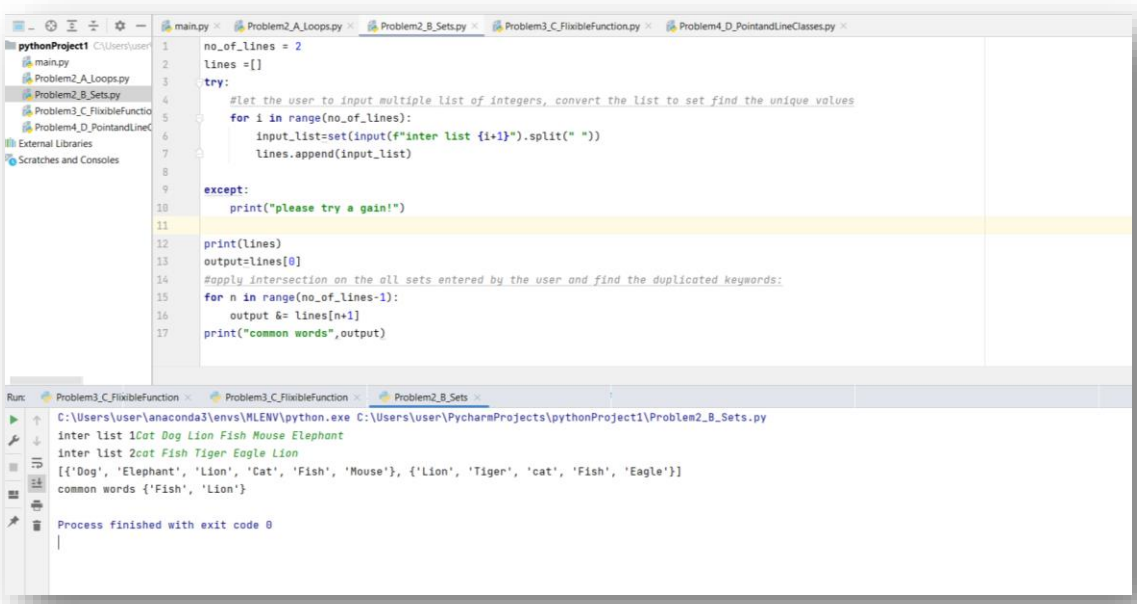

Figure 2:Output Sample of Problem2_B

3. Problem3_C_FlixibleFunction.py

```
def find_min_max(*args,**kwargs):
    #find the Max or Min value in the list

    res=[]
    for key in kwargs.keys():
        if kwargs[key][1]=="L":

            res.append(max(kwargs[key][0]))
        elif kwargs[key][1]=="S":
            res.append(min(kwargs[key][0]))

    return res


while True:
    try:
        no_of_lines = 4
        lines = ""
        for i in range(no_of_lines):
            lines += input() + "\n"

        #let the user to input the list of integers:
        input_list=lines.split("\n")[0:4]

        op1=input_list[1]

        op2 = input_list[3]
        #convert the list of strings to list of floats:
        int_list1 = [float(i) for i in input_list[0].split(" ")]
        int_list2 = [float(i) for i in input_list[2].split(" ")]


        #let the user to choose the operant L,Large,large,l=Max and
S,Small,small,s=Min:
        #input_op=input("tpye L to find the max or S to find the
min:")[0].upper()

    except:
        print("your input must be numbers")
        continue


    #return result to the user:

    output=find_min_max(int_list1=[int_list1,op1],int_list2=[int_list2,op2])
    print(output)
    status = list(input("press Q to quite or C to  continue"))[0].upper()
    if status=="Q":
        break
```

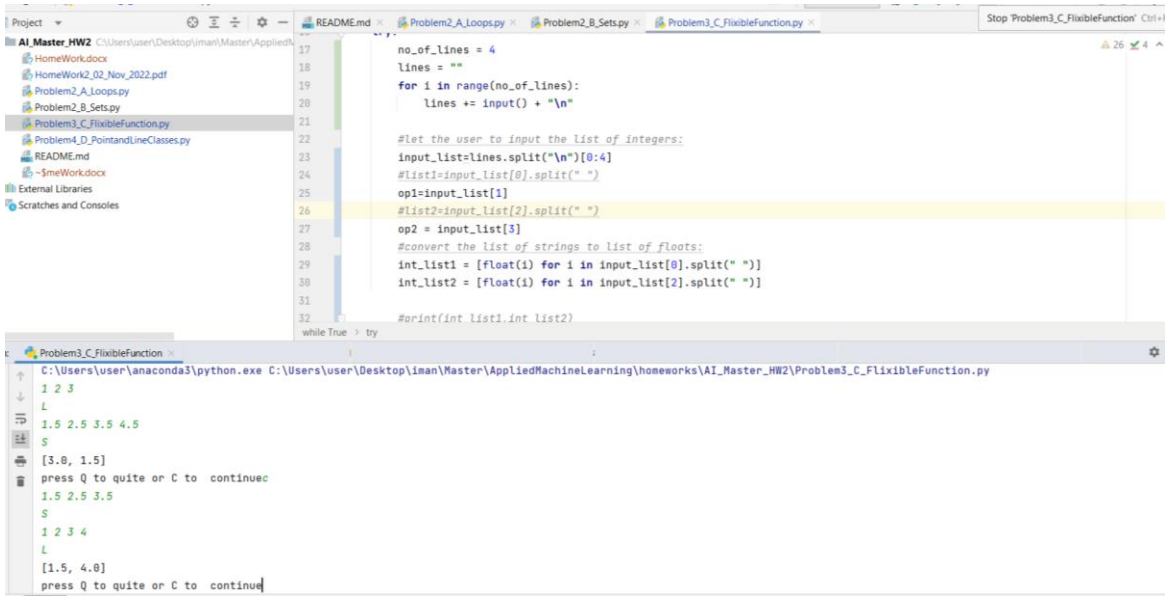

Figure 3:Output Sample of Problem3_C (Case_0)

4. Problem4_D_PointandLineClasses.py

```
import math
#Point Class
class Point():
    def __init__(self,x,y):
        self.x=x
        self.y=y
#Line Class
class Lin):
    def __init__(self,line_start,line_end):
        #
super(Line,self).__init__([line_start[0],line_end[0]],[line_start[1],line_end
[1]])
        #create two points instances
        self.point1=Point(line_start[0],line_start[1])
        self.point2=Point(line_end[0],line_end[1])
        print(self.point1.x,self.point1.y)
        print(self.point2.x,self.point2.y)


    #claculate the length of line
    def line_length(self):
        length=math.sqrt((self.point2.x-self.point1.x)**2+(self.point2.y-
self.point1.y)**2)
        return length
while True:
    try:
        #take the coordinates from user as input
        input_coordinate=list(input("please input the coordination of teh
line x1,y1,x2,y2 respectively:").split(" "))
        #check the length of input values(must be 4)
        if len(input_coordinate)<4:
            print("please input 4 values of type number")
            continue
        #if the input values more that 4 elements it takes the first 4
elements:
        if len(input_coordinate)>=4:
            input_coordinate=input_coordinate[0:4]
        #convert the list of strings to list of floats:
        input_coordinate=[float(i) for i in input_coordinate]
    except:
        print("check the input to be 4 numbers")
        continue
    #create new line instance from the line class and find the length by
calling line_lingth function.

new_line=Line([input_coordinate[0],input_coordinate[1]],[input_coordinate[2],
input_coordinate[3]])
    length=new_line.line_length()
    print("the length of line =",length)
```

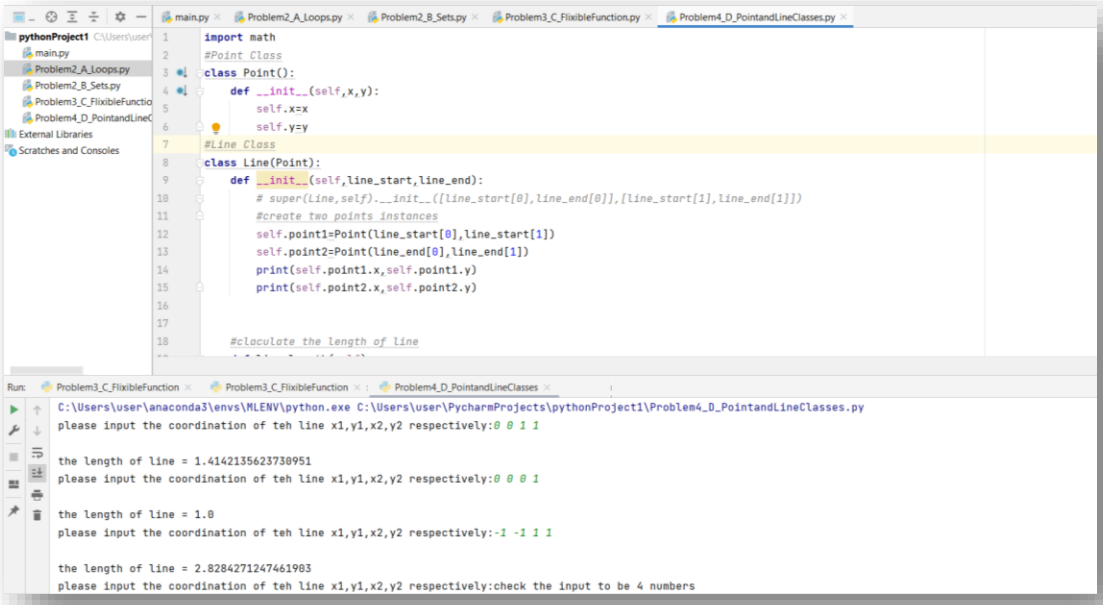

Figure 4: :Output Sample of Problem4_D