Muhammad Zain

19F-0228

Lab Manual

Assignment 1

# TASK 1

## Source code:

# printing the sequence of string  
Name = "Muhammad Zain"  
RollNo = "19F-0228"  
print("My name is", Name)  
print("My roll number is ", RollNo)  
print("I Like AI Subject :D")

# Screenshot: Text Description automatically generated

# TASK 2

## Source code:

#Arithmatic Functions  
  
def Sum():  
 var1=float(input("Input 1st operand "))  
 var2 = float(input("Input 2nd operand "))  
 print("The sum of two numbers is ", var1+var2)  
  
def Sub():  
 var1=float(input("Input 1st operand "))  
 var2 = float(input("Input 2nd operand "))  
 print("The difference of two numbers is ", var1-var2)  
def Multiply():  
 var1=float(input("Input 1st operand "))  
 var2 = float(input("Input 2nd operand "))  
 print("The Product of two numbers is ", var1\*var2)  
def Divide():  
 var1=float(input("Input 1st operand "))  
 var2 = float(input("Input 2nd operand "))  
 print("The Quotient is ", var1/var2)  
 print("The Remainder is ", var1 % var2)  
def Square():  
 var = float(input("Input operand "))  
 print("The Square is ",var\*\*2)  
def Cube():  
 var = float(input("Input operand "))  
 print("The Cube is ",var\*\*3)  
  
print("\*\*Airthmatic Calculator\*\*")  
print(" \*Test all operations\*")  
  
print("=> Addition\n=> Subtraction\n=> Multiplication\n=> Division\n=> Square\n=> Cube\n")

# Screenshot:Text Description automatically generated

Text

Description automatically generated

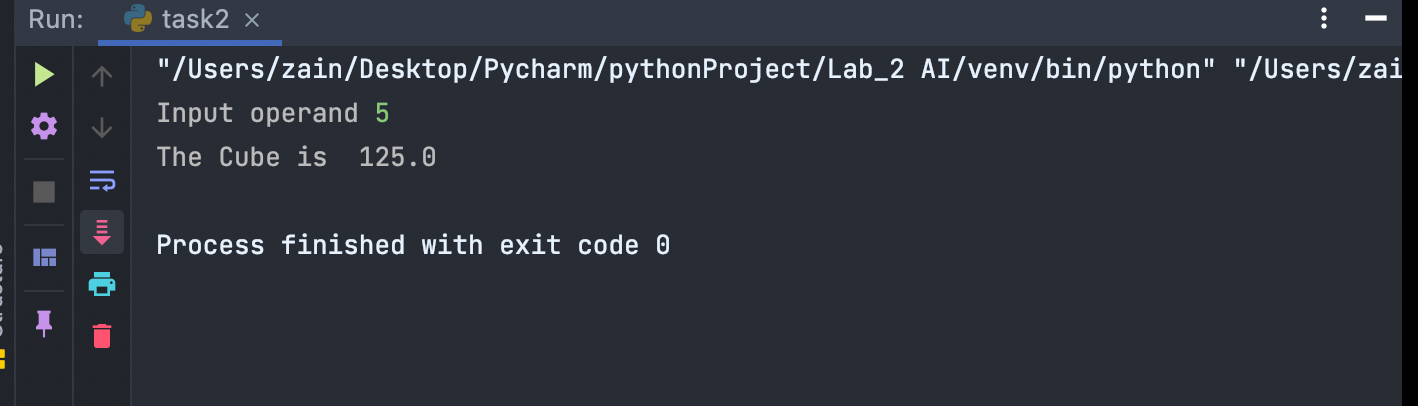
Text

Description automatically generatedGraphical user interface, text

Description automatically generated

Text

Description automatically generated



# TASK 3

## Source code:

#Value Manipulation  
  
def manipulation(one,two,three):  
 one=one+5  
 two=two+10  
 three=three+15  
 return one, two, three  
  
var1 = float(input("Input 1st operand "))  
var2 = float(input("Input 2nd operand "))  
var3 = float(input("Input 3rd operand "))  
#we can show the returning values in couple of ways  
# 1) store it in respective variables & print each  
# 2) Direct print the returning values  
var1,var2,var3=manipulation(var1, var2, var3)  
print(var1,var2,var3)  
# lets test both ways  
print(manipulation(var1, var2, var3))

# Screenshot:Text Description automatically generated

# TASK 4

## Source code:

#String Manipulatioin  
string1="HELLO WORLD"  
string2="HOW,arE yOu"  
  
def concatenate():  
 print("After Concatenation ",string1+string2)  
def UpperLower():  
 print(string1.upper())  
 print(string2.upper())  
 print(string1.lower())  
 print(string2.lower())  
  
  
concatenate()  
UpperLower()  
  
print(string1)  
print(string2)  
  
print("The String 1 After Slicing: ", end="")  
print(string1[5:].replace(" ", ""))  
  
print("The String 2 After Slicing: ", end="")  
print(string2[-5:-1].replace(" ", ""))

# Screenshot:

# Text Description automatically generated

# TASK 5

## Source code:

#EVEN ODD using for and while  
  
print("Lets print even and odd Numbers")  
starting = int(input("\nInput a starting value : "))  
ending = int(input("Input a ending value : "))  
EvenArray=[]  
OddArray=[]  
print("\n Using For Loop")  
for x in range(starting,ending):  
 if(x%2==0):  
 EvenArray.append(x)  
 else:  
 OddArray.append(x)  
  
print("For Even : ", EvenArray)  
print("For ODD : ", OddArray)  
print("\n Using While Loop")  
EvenArray=[]  
OddArray=[]  
  
while starting < ending:  
 if(starting%2==0):  
 EvenArray.append(starting)  
 starting+=1  
 else:  
 OddArray.append(starting)  
 starting += 1  
  
print("While Even : ", EvenArray)  
print("While ODD : ", OddArray)

# Screenshot:

# Text Description automatically generated

# TASK 6

## Source code:

#sum of natural numbers  
print("\*\*\*Sum using loop\*\*\*")  
def sumNatural(limit):  
 sum=0  
 for x in range(1,limit+1):  
 sum=x+sum  
 print("The sum of natural numbers is from 1 till ",limit,"is = ",sum)  
print("Calculate the sum of natural numbers")  
limitVar=int(input("insert limit "))  
sumNatural(limitVar)  
  
#we can find th esum in multiple ways  
#using formula and loops  
  
print("\*\*\*Sum using formula\*\*\*")  
Result=limitVar\*(limitVar+1)/2  
print("The sum of natural numbers is from 1 till ",limitVar,"is = ",Result)

# Screenshot:

# Text Description automatically generated

# TASK 7

## Source code:

#Reverse a number  
  
def ReverseNumber(input):  
 Reverse=0  
 while(input>0):  
 Remainder = input%10  
 Reverse=(Reverse\*10)+Remainder  
 input= input//10  
  
 #The single division operator behaves abnormally generally for very large numbers.  
 #The Double Division operator in Python returns the floor value for both integer and floating-point arguments after division  
  
 print("Reversed Number = ", Reverse)  
print("lets reverse the number")  
number=int(input("\nInput a number : "))  
  
ReverseNumber(number)

# Screenshot:

# Text Description automatically generated

# TASK 8

## Source code:

#slicing  
line = "Master has failed more, than the beginner has tried"  
  
print ( "Before Slicing : ",line)  
print ( "AFTER SLICING : ", line[0:22])  
  
concat = line + ", so never give up!"  
print ( "After COntactenation :", line)

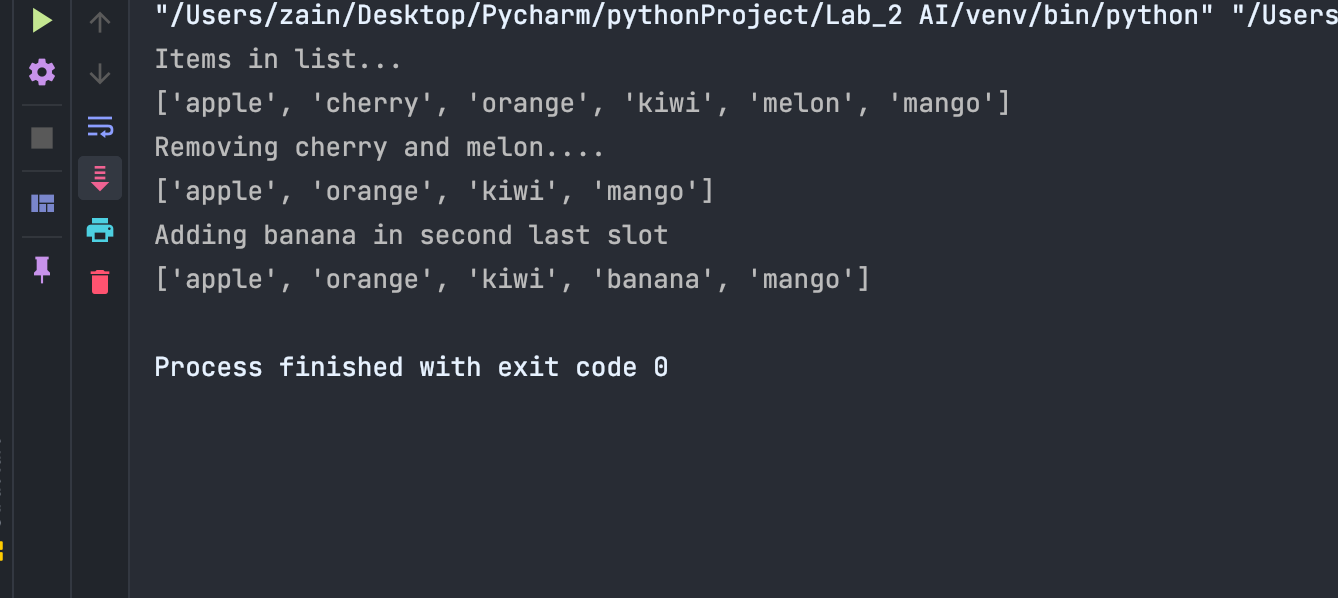
# Screenshot:Text Description automatically generated

# TASK 9

## Source code:

#list addition and removal  
  
list = ["apple", "cherry", "orange", "kiwi", "melon", "mango"]  
print("Items in list...")  
print(list)  
print("Removing cherry and melon....")  
list.remove("cherry")#removal by name \ value  
list.remove("melon")  
print(list)  
print("Adding banana in second last slot")  
list.insert(-1,"banana")  
#-1 indicates last element  
# we are adding infront of last element so second last  
print(list)

# Screenshot:



# TASK 10

## Source code:

#empty list using while loop  
list = [ 1, 4, 56, 2, 4 , 12, 6, 89 ,11, 0]  
counter=len(list)-1  
  
while counter>0:  
 print("Deleting ...", list.pop(counter)," ",len(list) ," ",counter)  
 counter=counter-1  
  
if counter == 0:  
 print("List is empty.....")

# Screenshot:

# Text Description automatically generated

# TASK 11

## Source code:

marks = {  
 'AI':74,  
 'CN':76,  
 'DS':42,  
 'PS':54  
 }  
sum=0  
for i in marks.values():  
 sum=sum+i  
print("Sum of Marks = ",sum)  
  
print("Subject\t Marks")  
for attr, value in marks.items():  
 print(' ',attr,' ',value)  
  
max=0  
counter=0  
for attr, value in marks.items():  
 if max < value: #checking on the basis of marks  
 max=value  
 objMarks=attr,value  
 counter=counter+1  
 if counter>4:  
 print("Max Marks\t= ", objMarks)  
 else:  
 counter = counter + 1  
min=max  
counter=0  
for attr, value in marks.items():  
 if min > value: #checking on the basis of marks  
 min=value  
 objMarks=attr,value  
 counter=counter+1  
 if counter>4:  
 print("Min Marks\t= ", objMarks)  
 else:  
 counter = counter + 1  
  
  
  
  
  
  
# Taken help from here  
# https://stackoverflow.com/questions/25150955/python-iterating-through-object-attributes

# Screenshot:

# Text Description automatically generated

# TASK 12

## Source code:

#armstrong number  
  
def checkArmstrong(input):  
 sum=0  
 temp=input  
 while temp > 0:  
 digit=temp%10#extracting each elemnent  
 sum= sum+(digit\*\*3)#taking cube of each digit and taking there sum  
 temp=temp//10#removing digit from number  
 return sum  
  
Num=int(input("Input a number to check weather it si armstrong or not"))  
checker=checkArmstrong(Num)  
if checker==Num:  
 print("Yes This is Armstrong Number")  
else:  
 print("No this is not Armstrong Number")

# Screenshot:

# Text Description automatically generatedText Description automatically generated

# TASK 13

## Source code:

#Factorial Code  
def Factorial(input):  
#first we will define the base case  
 if input==1:  
 return 1  
#Recurssion ...main muda  
 else:  
 return (input\*Factorial(input-1))  
  
print("Factorial Calculator")  
Num=int(input("Input Number to get Factorial : "))  
if Num<0:  
 print("Factorial Not exist of negative number")  
else:  
 Ans=Factorial(Num)  
 print("The Factorial of ",Num," is ",Ans)

# Screenshot:

# Text Description automatically generated

# TASK 14

## Source code:

#list with random numbers  
import random  
  
list=[]  
def populateList(num):  
 for i in range(num):  
 list.append(random.randint(0,100))  
 print("After Populating with Random Numbers\n",list)  
  
def findMax(num):  
 max = 0  
 counter = 0  
 for i in range(num):  
 if max < list[i]:  
 max = list[i]  
 counter = counter + 1  
 else:  
 counter = counter + 1  
 print("Maximum Value\t= ", max)  
 return max  
def findMin(max,size):  
 min = max  
 counter = 0  
 for i in range(size):  
 if min > list[i]:   
 min = list[i]  
 counter = counter + 1  
 else:  
 counter = counter + 1  
 print("Minimum Value\t= ", min)  
  
# print("Random Number Generator")  
size=int(input("Input the Size of the list : "))  
populateList(size)  
max=findMax(size)  
findMin(max,size)

# Screenshoot:

# Text Description automatically generated

# TASK 15

## Source code:

# create a tuple  
# creation of tuple  
tuple\_ = ("john", "mark", 12, "14", "orange", 4.5)  
print(tuple\_)  
  
def updateTuple(value,tuple\_):  
 # there are two methods to add values to tuple say jugar system  
 # because tuple is immutable  
 # 1) add a value to new tuple then concatenate tuples  
 # 2) Convert tuple to list ,add value to list ,convert list to tuple  
 # adding value to tuple and then adding concatenating this  
 newtuple\_ = value,  
 tuple\_ = tuple\_+newtuple\_  
 print("After adding 6.5...")  
 print(tuple\_)  
  
def counterFun(tuple\_):  
 # Checking for int float and strings  
 stringCount = 0  
 intCount = 0  
 floatCount = 0  
 tupleCount = len(tuple\_)  
 for i in range(tupleCount):  
 if isinstance(tuple\_[i], int):  
 intCount = intCount + 1  
 elif isinstance(tuple\_[i], float):  
 floatCount = floatCount + 1  
 else:  
 stringCount = stringCount + 1  
 print("Integer Count : ", intCount)  
 print("Float Count : ", floatCount)  
 print("String Count : ", stringCount)  
  
updateTuple (6.5,tuple\_)  
counterFun (tuple\_)

# Screenshot:

# Text Description automatically generated

# TASK 16

## Source code:

# creating dictionary  
import random  
listAdd=[]  
Dictionary = {  
 "brand": "Samsung",  
 "os-type": "Oreo",  
 "color": "black",  
 "camera": "42 megapixels",  
 "year": 2012,  
 "sizes":[]  
 }  
def addList(size):  
 for i in range(size):  
 listAdd.append(random.randint(0,100))  
size=int(input("Input the size of the list : "))  
addList(size)  
print("Dictonary Before Addition...\n",Dictionary)  
Dictionary["sizes"].append(listAdd)  
print("Dictonary After Addition...\n",Dictionary)  
del Dictionary['year']  
print("Dictonary After Deletion...\n",Dictionary)  
print(sorted(Dictionary))

# Screenshot:

Text

Description automatically generated

# TASK 17

## Source code:

#list inside Functions  
def list2Sum(list\_1,list\_2):  
 counter= max(len(list\_2),len(list\_1))  
 list\_3=[]  
 for x in range(counter):  
 temp=(list\_1[x] + list\_2[x])  
 list\_3.append(temp)  
 print(list\_1[x], " + ", list\_2[x], " = ", list\_3[x])  
 return list\_3  
  
  
list1 = [11, 22, 33, 44, 21, 54, 67, 54, 33, 222, 4]  
list2 = [3, 4, 5, 32, 21, 33, 66, 75, 87, 97, 1]  
list3 = list2Sum(list1, list2)

# Screenshot:

# Text Description automatically generated

# TASK 18

## Source code:

#Dsorting  
  
def dsort(List):  
 counter = len(List)  
 temp = None  
 for x in range(counter):  
 for y in range(counter):  
 if (List[x] > List[y]):  
 temp = List[x]  
 List[x] = List[y]  
 List[y] = temp  
  
  
print("\t\t\*\*\*Descending Order\*\*\*")  
list\_ = [5, 6, 7, 23, 12, 3, 3, 4, 5, 12, 34]  
print("Before Sorting ......\n",list\_)  
dsort(list\_)  
print("After Sorting.........\n",list\_)

# Screenshot:

# Text Description automatically generated

# TASK 19

## Source code:

#stack and queue  
class Stack:  
 Counter = 0  
 StackList = []  
 def Insertion ( self , input ) :  
 self.StackList.insert(0,input)  
 self.Counter+=1  
 def Deletion(self):  
 self.StackList.pop(0)  
 self.Counter-=1  
 def isEmpty(self):  
 if self.Counter==0:  
 print("Stack Empty hay....")  
 def top(self):  
 if self.Counter != 0:  
 return self.StackList[0]  
 else:  
 print("Stack Empty hay....")  
 def printFun(self):  
 print(self.StackList)

class Queue:  
 Counter = 0  
 QueueList = []  
 def Insertion ( self , input ) :  
 self.QueueList.append(input)  
 self.Counter+=1  
 def Deletion(self):  
 self.QueueList.pop(0)  
 self.Counter -= 1  
 def isEmpty(self):  
 if self.Counter == 0:  
 print("Queue Empty hay....")  
 def top(self):  
 if self.Counter != 0:  
 return self.QueueList[0]  
 else:  
 print("Queue Empty hay....")  
 def printFun(self):  
 print(self.QueueList)

print ( " Testing Stack Class" )  
obj\_S = Stack()  
obj\_S.isEmpty()  
obj\_S.top()  
obj\_S.Insertion(2)  
obj\_S.Insertion(3)  
obj\_S.Insertion(4)  
obj\_S.Insertion(5)  
obj\_S.Insertion(6)  
print("After insertion.....\n")  
obj\_S.printFun()  
obj\_S.Deletion()  
print("After Deletion........\n")  
obj\_S.printFun()

print ( " Testing Queue Class" )  
obj\_Q = Queue()  
obj\_Q.isEmpty()  
obj\_Q.top()  
obj\_Q.Insertion(2)  
obj\_Q.Insertion(3)  
obj\_Q.Insertion(4)  
obj\_Q.Insertion(5)  
obj\_Q.Insertion(6)  
print("After insertion.....\n")  
obj\_Q.printFun()  
obj\_Q.Deletion()  
print("After Deletion........\n")  
obj\_Q.printFun()

# Screenshot:

# Text Description automatically generatedText Description automatically generated

# TASK 20

## Source code:

class Student:  
 def \_\_init\_\_ (self,firstName, lastName,age,cnic,course,gender,CGPA,SGPA,cred\_hour):  
 self.firstName = firstName  
 self.lastName = lastName  
 self.age=age  
 self.cnic=cnic  
 self.course = course  
 self.gender = gender  
 self.CGPA = CGPA  
 self.SGPA = SGPA  
 self.cred\_hour = cred\_hour  
  
 def set\_firstName(self,name):  
 self.firstName = name  
 def set\_lastName(self,name):  
 print("ast Name :- ",name)  
 self.lastName=name  
 def set\_age(self,age):  
 self.age=age  
 def set\_cnic(self,cnic):  
 self.cnic=cnic  
 def set\_course(self,course):  
 self.course=course  
 def set\_gender(self,gender):  
 self.gender=gender  
 def set\_CGPA(self,cgpa):  
 self.CGPA=cgpa  
 def set\_SGPA(self,sgpa):  
 self.SGPA = sgpa  
 def set\_cred\_hour(self,cred):  
 self.cred\_hour=cred  
 #GETTER FUNCTIONS  
 def get\_firstName(self):  
 return self.firstName  
 def get\_lastName(self):  
 return self.lastName  
 def get\_age(self):  
 return self.age  
 def get\_cnic(self):  
 return self.cnic  
 def get\_course(self):  
 return self.course  
 def get\_gender(self):  
 return self.gender  
 def get\_CGPA(self):  
 return self.CGPA  
 def get\_SGPA(self):  
 return self.SGPA  
 def get\_cred\_hour(self):  
 return self.cred\_hour  
  
 def SetStudent(obj,firstName,lastName,age,cnic,course,gender,CGPA,SGPA,cred\_hour ):  
 obj.set\_firstName(firstName)  
 obj.set\_lastName(lastName)  
 obj.set\_age(age)  
 obj.set\_cnic(cnic)  
 obj.set\_course(course)  
 obj.set\_gender(gender)  
 obj.set\_CGPA(CGPA)  
 obj.set\_SGPA(SGPA)  
 obj.set\_cred\_hour(cred\_hour)  
  
def FetchStudent(obj):  
 print ("First Name :- ",obj.get\_firstName()," Last Name :- ",obj.get\_lastName(),  
 " AGE :- ",obj.get\_age()," CNIC :- ",obj.get\_cnic()," Course :- ",obj.get\_course()," Gender :- ",obj.get\_gender(),  
 " CGPA :- ",obj.get\_SGPA()," Credit Hour :- ",obj.get\_cred\_hour())  
  
List=[]  
List.append(Student("Zain", "Ahsan", 22, "123421", list, "Male", 4, 4, 100))  
List.append(Student("Jahanzaib","Ahsan",18,"234234234234",list,"Male",4,4,100))  
List.append(Student("Fahad","Ahsan",12,"234234234234",list,"Male",4,4,100))  
  
print ("\*\*\*List\*\*\*")  
  
for obj in List:  
 FetchStudent(obj)

# Screenshot:

Graphical user interface, text

Description automatically generated

# TASK 21

## Source code:

class Student:  
 def \_\_init\_\_ (self,firstName, lastName,age,cnic,course,gender,CGPA,SGPA,cred\_hour):  
 self.firstName = firstName  
 self.lastName = lastName  
 self.age=age  
 self.cnic=cnic  
 self.course = course  
 self.gender = gender  
 self.CGPA = CGPA  
 self.SGPA = SGPA  
 self.cred\_hour = cred\_hour  
  
 def set\_firstName(self,name):  
 self.firstName = name  
 def set\_lastName(self,name):  
 print("ast Name :- ",name)  
 self.lastName=name  
 def set\_age(self,age):  
 self.age=age  
 def set\_cnic(self,cnic):  
 self.cnic=cnic  
 def set\_course(self,course):  
 self.course=course  
 def set\_gender(self,gender):  
 self.gender=gender  
 def set\_CGPA(self,cgpa):  
 self.CGPA=cgpa  
 def set\_SGPA(self,sgpa):  
 self.SGPA = sgpa  
 def set\_cred\_hour(self,cred):  
 self.cred\_hour=cred  
 #GETTER FUNCTIONS  
 def get\_firstName(self):  
 return self.firstName  
 def get\_lastName(self):  
 return self.lastName  
 def get\_age(self):  
 return self.age  
 def get\_cnic(self):  
 return self.cnic  
 def get\_course(self):  
 return self.course  
 def get\_gender(self):  
 return self.gender  
 def get\_CGPA(self):  
 return self.CGPA  
 def get\_SGPA(self):  
 return self.SGPA  
 def get\_cred\_hour(self):  
 return self.cred\_hour  
  
 def SetStudent(obj,firstName,lastName,age,cnic,course,gender,CGPA,SGPA,cred\_hour ):  
 obj.set\_firstName(firstName)  
 obj.set\_lastName(lastName)  
 obj.set\_age(age)  
 obj.set\_cnic(cnic)  
 obj.set\_course(course)  
 obj.set\_gender(gender)  
 obj.set\_CGPA(CGPA)  
 obj.set\_SGPA(SGPA)  
 obj.set\_cred\_hour(cred\_hour)  
  
def FetchStudent(obj):  
 print ("First Name :- ",obj.get\_firstName()," Last Name :- ",obj.get\_lastName(),  
 " AGE :- ",obj.get\_age()," CNIC :- ",obj.get\_cnic()," Course :- ",obj.get\_course()," Gender :- ",obj.get\_gender(),  
 " CGPA :- ",obj.get\_SGPA()," Credit Hour :- ",obj.get\_cred\_hour())  
  
List=[]  
List.append(Student("Zain", "Ahsan", 22, "123421", list, "Male", 4, 4, 100))  
List.append(Student("Jahanzaib","Ahsan",18,"234234234234",list,"Male",4,4,100))  
List.append(Student("Fahad","Ahsan",12,"234234234234",list,"Male",4,4,100))  
  
print ("\*\*\*List\*\*\*")  
  
for obj in List:  
 FetchStudent(obj)  
  
print ("\*\*\* Tuple \*\*\*")  
obj = Student("Burhan", "Shakir", 20, "2342342", list, "Male", 3.8, 4, 100)  
obj1 = Student("Kashif","Aziz",20,"234234234234",list,"Male",3.8,4,100)  
tup = (obj,obj1)  
print(tup)  
print ("Dicrionary")  
obj = Student("Burhan", "Shakir", 20, "2342342", list, "Male", 3.8, 4, 100)  
obj1 = Student("Kashif","Aziz",20,"234234234234",list,"Male",3.8,4,100)  
dict = {"Obj 1 ":obj,"obj2:- ":obj1}  
print(dict)

# Screenshot:Text Description automatically generated

# TASK 22

## Source code:

class Shape():  
 def \_\_init\_\_(self):  
 print("Shape Class Called")  
 def printRect(self):  
 print("\*\*\*Rectangle Class \*\*\*")  
 def printCircle(self):  
 print("\*\*\* Circle Class Called \*\*\*")  
  
class Rectangle(Shape):  
 def \_\_init\_\_(self, l, w):  
 super().\_\_init\_\_()#ya mena nai kea compiler na suggestion d ha k super class add kro agr opar access krna ha  
 self.length = l  
 self.width = w  
 def rectangle\_area(self):  
 return self.length\*self.width  
 def rectangle\_perimeter(self):  
 return self.length\*self.width  
class Circle(Shape):  
 def \_\_init\_\_(self, r):  
 self.radius = r  
 def circle\_area(self):  
 return 3.14\*self.radius\*self.radius  
 def circle\_perimeter(self):  
 return 3.14 \*2\* self.radius  
rect = Rectangle(10,10)  
rect.printRect()  
  
print ("Rectangle Area :- ",rect.rectangle\_area())  
print ("Circle Area :- ",rect.rectangle\_perimeter())  
circ = Circle(10)  
circ.printCircle()  
print ("Circle Area :- ",circ.circle\_area())  
print ("Circle Perimeter :- ",circ.circle\_perimeter())

# Screenshot:

Text

Description automatically generated

# TASK 23

## Source code:

import pandas as pd  
import networkx as nx  
import matplotlib.pyplot as plt  
df = pd.DataFrame({'from': ['Spider Man', 'Civil War', 'Iron Man', 'Thor','Avengers','Ragnarok','Captain America','Ant Man','Hulk'],  
'to': ['Avengers', 'Spider Man', 'Spider Man', 'Avengers','Thor','Thor','Avengers','Avengers','Avengers']})  
# Build your graph  
G = nx.from\_pandas\_edgelist(df, 'from', 'to')  
# Plot it  
nx.draw(G, with\_labels=True)  
plt.show()

Chart

Description automatically generated

# TASK 24

## Source code:

def append\_Vertex(vertex):  
 global graph  
 global vertices\_no  
 if vertex in graph:  
 print("\*\*\*", vertex, " Vertex already exists \*\*\*")  
 else:  
 vertices\_no = vertices\_no + 1  
 graph[vertex] = []  
def append\_edge(vertex1, vertex2):  
 global graph  
 if vertex1 not in graph:  
 print("\*\*\* ", vertex1, " Vertex not exist \*\*\*")  
 elif vertex2 not in graph:  
 print("\*\*\* ", vertex2, " Vertex not exist \*\*\*")  
 else:  
 temp = [vertex2]  
 graph[vertex1].append(temp)  
  
def print\_graph():  
 global graph  
 for vertex in graph:  
 for edges in graph[vertex]:  
 print (vertex, " -> ", edges[0])  
graph = {}  
vertices\_no = 0  
append\_Vertex('A')  
append\_Vertex('B')  
append\_Vertex('C')  
append\_Vertex('D')  
append\_Vertex('E')  
append\_Vertex('F')  
append\_Vertex('G')  
append\_Vertex('H')  
append\_edge('A', 'B')  
append\_edge('A', 'C')  
append\_edge('A', 'D')  
append\_edge('B', 'E')  
append\_edge('B', 'F')  
append\_edge('E', 'G')  
append\_edge('F', 'H')  
print("\*\*\* Ajency List Using Dictionay \*\*\*")  
print("\*\*\* Print Ajency List \*\*\*")  
print\_graph()

# Screenshot:Text Description automatically generated

# TASK 25

## Source code:

def append\_edge(vertex1, vertex2,weight):  
 global graph  
 global vertices\_no  
 global vertices  
  
 if vertex1 not in vertices:  
 print("==> ", vertex1, " Vertex not exist <==")  
 elif vertex2 not in vertices:  
 print("==> ", vertex2, " Vertex not exist <==")  
 else:  
 index1 = vertices.index(vertex1)  
 index2 = vertices.index(vertex2)  
 graph[index1][index2] = weight  
  
  
def append\_Vertex(vertex):  
 global graph  
 global vertices\_no  
 global vertices  
 if vertex in vertices:  
 print("==>", vertex, " Vertex already exists <==")  
 else:  
 vertices\_no = vertices\_no + 1  
 vertices.append(vertex)  
 if vertices\_no > 1:  
 for vertex in graph:  
 vertex.append(0)  
 temp = []  
 for i in range(vertices\_no):  
 temp.append(0)  
 graph.append(temp)  
  
def print\_graph():  
 global graph  
 global vertices\_no  
 for i in range(vertices\_no):  
 for j in range(vertices\_no):  
 if graph[i][j] != 0:  
 print(vertices[i], " -> ", vertices[j]," weight :- ", graph[i][j])  
def appendVer():  
 append\_Vertex(5)  
 append\_Vertex(6)  
 append\_Vertex(3)  
 append\_Vertex(4)  
 append\_Vertex(1)  
 append\_Vertex(2)  
def appendEdg():  
 append\_edge(5, 6, 9)  
 append\_edge(5, 4, 6)  
 append\_edge(6, 3, 2)  
 append\_edge(6, 5, 9)  
 append\_edge(6, 1, 14)  
 append\_edge(3, 6, 2)  
 append\_edge(3, 1, 9)  
 append\_edge(3, 4, 11)  
 append\_edge(3, 2, 10)  
 append\_edge(4, 3, 11)  
 append\_edge(4, 2, 15)  
 append\_edge(2, 3, 10)  
 append\_edge(2, 4, 15)  
 append\_edge(2, 1, 7)  
 append\_edge(1, 6, 14)  
 append\_edge(1, 3, 9)  
 append\_edge(1, 2, 7)  
vertices = []  
vertices\_no = 0  
graph = []  
appendVer()  
appendEdg()  
print("\t\*\*\* Ajency Matrix Using Dictionay \*\*\*")  
print("\t\*\*\*Print Ajency List\*\*\*")  
print\_graph()  
print("\t Print Ajency Matrix")  
print(" Matrix :- ", graph)

# Screenshot:

A picture containing graphical user interface

Description automatically generated

# TASK 26

# BASICS SHORTCUTS OF PYCHARM

## Command + /

Comments all selected Portion

## CTRL + T

Refactoring

## CTRL + R

Runs the code

## Command + N

Creates new file questionnaire

## Option + up arrow

Selects all the portion above from the cursor placed

## Option + down arrow

Selects all the portion down from the cursor placed

# TASK 27

## In Terminal

* File>Setting>project Select project
* Select Python Interpreter Click + to add Library

## Without terminal:

* By using Import keyword
* Py -m pip

# TASK 28

# How to zoom in PyCharm:

* File > setting > editor > font > increase size of font

# TASK 30

# How to change the Theme of PYCHARM

File > setting >Appearance and Behavior >Select Theme