Q.1 Write a python program to greet users with welcome messages using print () method.

**CODE:**

#Enrollment No: 202203103510097

#Name: Angat Shah

#Branch: B.Tech Computer Science and Engineering

student\_name = input("Enter your Name: ")

enroll\_num = input("Enter your Enrollment Number: ")

student\_branch = input("Enter your Branch: ")

student\_college = input("Enter your College name: ")

print("Welcome dear {0} having enrollment number {1} to {2} in {3}\n".format(student\_name,enroll\_num,student\_branch,student\_college))

print("-\*-\*-\*-\*-\*-END OF PRACTICAL 1-\*-\*-\*-\*-\*-")

**Q.2 Write a python program to implement basic assignment, arithmetic, logical, relational ,bitwise, comparison, Identity, and Membership operations on user entered numbers..**

**CODE:**

#Enrollment No: 202203103510097

#Name: Angat Shah

#Branch: B.Tech Computer Science and Engineering

a = int(input("-->> Enter first number: "))

b = int(input("-->> Enter second number: "))

print("<--- Arithmetic Operation --->")

print("{0} + {1} = {2}".format(a,b,a+b))

print("{0} - {1} = {2}".format(a,b,a-b))

print("{0} \* {1} = {2}".format(a,b,a\*b))

print("{0} / {1} = {2}".format(a,b,a/b))

print("{0} % {1} = {2}".format(a,b,a%b))

print("{0} \*\* {1} = {2}".format(a,b,a\*\*b))

print("{0} // {1} = {2}".format(a,b,a//b))

print("")

print("<--- Relational Operators --->")

print("Is first number('{0}') same as second number('{1}'): {2}".format(a,b,a==b))

print("Is first number('{0}') greater than second number('{1}'): {2}".format(a,b,a>b))

print("Is first number('{0}') less than second number('{1}'): {2}".format(a,b,a<b))

print("Is first number('{0}') greater than or equal to second number('{1}'): {2}".format(a,b,a>=b))

print("Is first number('{0}') less than or equal to second number('{1}'): {2}".format(a,b,a<=b))

print("Is first number('{0}') not equal to second number('{1}'): {2}".format(a,b,a!=b))

print("")

print("<--- Logical Operators --->")

print("a = {0} & b = {1}".format(a,b))

print("DOES a>b or a<b:",a>b or a<b)

print("DOES a>b and a<b",a>b and a<b)

print("not a==b",not a==b)

print("")

print("<--- Bitwise Operator --->")

print("a = {0} & b = {1}".format(a,b))

print("Using Bitwise OR, a | b:",a|b)

print("Using Bitwise AND, a & b:",a&b)

print("Using Bitwise NOT, ~ a:",~a)

print("Using Bitwise NOT, ~ b:",~b)

print("Using Bitwise XOR, a ^ b:",a^b)

print("Using Bitwise RIGHT SHIFT, a << b:",a<<b)

print("Using Bitwise LEFT SHIFT, a >> b:",a>>b)

print("")

print("<--- Identity Operator --->")

print("First number('{0}') is same as second number('{1}'): {2} ".format(a,b,a is b))

print("First number('{0}') is not same as second number('{1}'): {2}".format(a,b,a is not b))

print("")

print("<--- Assignment Operators --->")

print("First number('{0}') & Second number('{1}')".format(a,b))

a += 2

print("Using '+=' assignment operator on the given two numbers: ",a)

a -= 2

print("Using '-=' assignment operator on the given two numbers: ",a)

a \*= 2

print("Using '\*=' assignment operator on the given two numbers: ",a)

a /= 2

print("Using '/=' assignment operator on the given two numbers: ",a)

a %= 2

print("Using '%=' assignment operator on the given two numbers: ",a)

a \*\*= 2

print("Using '\*\*=' assignment operator on the given two numbers: ",a)

a //= 2

print("Using '//=' assignment operator on the given two numbers: ",a)

print("")

print("<--- Membership Operator --->")

my\_list = [1,2,3,4,5,6,7]

print("Now we have a list for membership operator:",my\_list)

print("The element '3' is in list:",3 in my\_list)

print("The element '5' is not in the list:",5 not in my\_list)

print("")

print("-\*-\*-\*-\*-\*-END OF PRACTICAL 2-\*-\*-\*-\*-\*-")

**Q.3 Demonstrate the use of basic string methods in python - lower(), islower(), upper(), isupper(), join(), split(), find(), replace(), capitalize(), casefold(), and replace().**

**CODE:**

#Enrollment No: 202203103510097

#Name: Angat Shah

#Branch: B.Tech Computer Science and Engineering

my\_string = input("-->> Enter a string: ")

print("STRING <-----{0}----->\n".format(my\_string))

# lower()

lower\_str = my\_string.lower()

print("Lowercase of the given string: {0}\n".format(lower\_str))

#islower()

print("Is the given string in lowercase: {0}\n".format(my\_string.islower()))

# upper()

upper\_str = my\_string.upper()

print("Uppercase of the given string: {0}\n".format( upper\_str))

#isupper()

print("Is the given string in uppercase: {0}\n".format(my\_string.isupper()))

# join()

a = input("-->> Enter first word to join: ")

b = input("-->> Enter second word to join: ")

words = [a , b]

join\_str = " ".join(words)

print("Joining the words {0} and {1}: {2}\n".format(a,b,join\_str))

# split()

split\_str = my\_string.split()

print("Spliting the given string: {0}\n".format(split\_str) )

# find()

c = input("-->> Enter a word to find in the string: ")

find\_str = my\_string.find(c)

print("Index of the word '{0}': {1}\n".format(c,find\_str))

# replace()

d = input("-->> Enter the word which you want to replace from the string: ")

e = input("-->> Enter the word you want to replace in place of '{0}': ".format(d))

replace\_str = my\_string.replace(d, e)

print("Replacing '{0}' with '{1}': {2}\n".format(d,e,replace\_str))

# capitalize()

capitalize\_str = my\_string.capitalize()

print("Capitalizing the given string: {0}\n".format( capitalize\_str))

# title()

title\_str = my\_string.title()

print("Titlecasing the given string: {0}\n".format( title\_str))

# in

f = input("-->> Enter the word to check its availability in the given string: ")

print("Is '{0}' in the given string: {1}\n".format(f,f in my\_string))

print("-\*-\*-\*-\*-\*-END OF PRACTICAL 3-\*-\*-\*-\*-\*-")

Q.4 Write a python program to demonstrate the creation of List data structure along with its methods - append (), extend(), insert(), remove(), clear(), index(), count(), sort(), reverse(), and copy(),pop(),insert(),min(),max().

I. Demonstrate positive and negative indexing with python List.

II. Demonstrate slicing operations on python List.

III. Demonstrate updation on List elements in python.

IV. Demonstrate deletion of a single python list element and multiple elements using slicing operators.

**CODE:**

#Enrollment No: 202203103510097

#Name: Angat Shah

#Branch: B.Tech Computer Science and Engineering

my\_list = [5,11,1,8,4]

another\_list = [1,2,3]

print("LIST <-----{0}----->".format(my\_list))

print("")

#append()

a = int(input("-->> Enter a number you want to append: "))

my\_list.append(a)

print("The given list after appending the element '{0}' --> {1}".format(a,my\_list))

print("")

#extend()

my\_list.extend(another\_list)

print("The given list after extending another list -->",my\_list)

print("")

#insert()

c = int(input("-->> Enter the index at which you want to enter the element: "))

d = int(input("-->> Enter the element you want to insert at position {0}: ".format(c)))

my\_list.insert(c,d)

print("The given list after inserting element '{0}' at position {1} --> {2}".format(d,c,my\_list))

print("")

#remove()

e = int(input("-->> Enter the element you want to remove: "))

my\_list.remove(e)

print("The given list after removing the element '{0}' from the list --> {1}".format(e,my\_list))

print("")

#index()

f = int(input("--> Enter the number for which you want to check the index: "))

index\_of\_11 = my\_list.index(f)

print("The index position of the element '{0}': {1}".format(f,index\_of\_11))

print("")

#count()

g = int(input("-->> Enter the number you want to check the repetition of: "))

count\_5 = my\_list.count(g)

print("Number of times element '{0}' is repeated in the given list: {1}".format(g,count\_5))

print("")

#sort()

my\_list.sort()

print("The sorted version of the given list -->",my\_list)

#reverse()

my\_list.reverse()

print("The reverse version of the given list -->",my\_list)

#copy()

my\_new\_list = my\_list.copy()

print("The copy version of the given list -->",my\_new\_list)

print("")

#pop()

popped\_list = my\_list.pop(1)

print("The element popped from the 1st position: ",popped\_list)

print("")

#min()

min\_my\_list = min(my\_list)

print("The minimum value in the given list:",min\_my\_list)

#max()

max\_my\_list = max(my\_list)

print("The maximum value in the given list:",max\_my\_list)

print("")

print("UPDATED LIST <-----{0}----->".format(my\_list))

print("")

#indexing

h = int(input("-->> Enter the positive index you want to check the number at: "))

print("The element at the '{0}' position is: {1}".format(h,my\_list[h]))

i = int(input("-->> Enter the negative index you want to check the number at: "))

print("The element at the '{0}' position is: {1}".format(i,my\_list[i]))

print("")

#updating

j = int(input("-->> Enter the index at which you want to change the element: "))

k = int(input("-->> Enter the element for the position {0}: ".format(j)))

my\_list[j] = k

print("After updating the given list -->",my\_list)

print("")

#slicing

l = int(input("-->> Enter the initial index for slicing: "))

m = int(input("-->> Enter the final index for slicing: "))

print("After slicing from '[{0}:{1}]' the list --> {2}".format(l,m,my\_list[l:m]))

print("")

print("-\*-\*-\*-\*-\*-END OF PRACTICAL 4-\*-\*-\*-\*-\*-")

**Q.5 Write a python program to demonstrate the creation of tuples along with its methods - count () and index ().**

**I. Demonstrate positive and negative indexing with python Tuple.**

**II. Demonstrate slicing operations on python Tuple.**

**CODE:**

#Enrollment No: 202203103510097

#Name: Angat Shah

#Branch: B.Tech Computer Science and Engineering

my\_tuple = (1,2,3,3,5)

print("TUPLE <-----{0}----->".format(my\_tuple))

print("")

#count()

a = int(input("-->> Enter the number for which you want to find the repetition: "))

count = my\_tuple.count(a)

print("'{0}' is repeated {1} times in the given tuple.".format(a,count))

print("")

#index()

b = int(input("-->> Enter the number for which you want to find the index: "))

index = my\_tuple.index(b)

print("The index of '{0}': {1}.".format(b,index))

print("")

#I. Demonstrate positive and negative indexing with python Tuple.

c = int(input("-->> Enter the positive index to find the element: "))

print("The Element at '{0}' positive position is: {1}".format(c,my\_tuple[c]))

d = int(input("-->> Enter the negative index to find the element: "))

print("The Element at '{0}' negative position is: {1}".format(d,my\_tuple[d]))

#II. Demonstrate slicing operations on python Tuple.

e = int(input("-->> Enter the initial value for slicing: "))

f = int(input("-->> Enter the final value for slicing: "))

slicing = my\_tuple[e:f]

print("The element from [{0}:{1}] in the given tuple: {2}".format(e,f,slicing))

print("")

#updating()

g = int(input("-->> Enter the index at which you want to change the number: "))

h = int(input("-->> Enter the number to change at index '{0}': ".format(g)))

my\_list = list(my\_tuple)

my\_list[g] = h

my\_tuple = tuple(my\_list)

print("After updating the given tuple --> {0}".format(my\_tuple))

print("")

print("-\*-\*-\*-\*-\*-END OF PRACTICAL 5-\*-\*-\*-\*-\*-")

**Q.6 Write a python program to demonstrate the creation of set along with its methods: add(),clear(),copy(),difference(),intersection(),union(),isdisjoint,issubset, issuperset()**

**CODE:**

#Enrollment No: 202203103510097

#Name: Angat Shah

#Branch: B.Tech Computer Science and Engineering

my\_set = {1,2,3,4,5}

print("SET <-----{0}----->".format(my\_set))

print("")

#add()

a = int(input("-->> Enter the number you want to add in the given set: "))

my\_set.add(a)

print("After adding '{0}', the given set --> {1}".format(a,my\_set))

print("")

#update()

my\_list = [11,10,12,5]

print("Now we have a LIST <-----{0}----->".format(my\_list))

my\_set.update(my\_list)

print("The given set after updating the given list in it --> {0}".format(my\_set))

print("")

#copy()

copy\_set = my\_set.copy()

print("Copy of the given set -->",copy\_set)

#discard()

b = int(input("-->> Enter the number you want to discard from the set: "))

my\_set.discard(b)

print("After discarding '{0}', the given set --> {1}".format(b,my\_set))

print("")

set\_1 = {1,2,3,4,5,6,7,8,9,0}

set\_2 = {2,4,6,8,0}

print("SET1 <-----{0}----->".format(set\_1))

print("SET2 <-----{0}----->".format(set\_2))

print("")

#union()

union\_set = set\_1.union(set\_2)

print("After taking union of the given two set\nUNION --> {0}".format(union\_set))

print("")

#intresection()

intersection\_set = set\_1.intersection(set\_2)

print("After intersecting the given two set\nINTERACTION --> {0}".format(intersection\_set))

print("")

#isdisjoint()

isdisjoint\_set = set\_1.isdisjoint(set\_2)

print("After using isdisjoint on the given two set, we get:",isdisjoint\_set)

print("")

#difference()

diff\_set = set\_1.difference(set\_2)

print("After taking difference of the given two set\nDIFFERENCE --> {0}".format(diff\_set))

print("")

#issubset() & issuperset()

print("Does set1 is a subset of set2:",set\_1.issubset(set\_2))

print("Does set1 is a superset of set2:",set\_1.issuperset(set\_2))

print("")

print("Does set2 is a subset of set1:",set\_2.issubset(set\_1))

print("Does set2 is a superset of set1:",set\_2.issuperset(set\_1))

print("")

#clear()

set\_1.clear()

print("After clearing the set1 --> {0}".format(set\_1))

set\_2.clear()

print("After clearing the set2 --> {0}".format(set\_2))

print("")

print("-\*-\*-\*-\*-\*-END OF PRACTICAL 6-\*-\*-\*-\*-\*-")

**Q.7 Write a python program to demonstrate the creation of a Dictionary student with**

**the name, age and branch of a student.**

**I. Demonstrate the updation of python dictionary.**

**II. Demonstrate the removal of elements from the python dictionary.**

**III. Demonstrate the use of following dictionary methods - clear(), copy(), get(), items(), keys(), popitem() and values()**

**CODE:**

#Enrollment No: 202203103510097

#Name: Angat Shah

#Branch: B.Tech Computer Science and Engineering

student = {"std\_name":"Angat Shah" , "std\_branch":"B.tech" , "std\_age":17}

print("DICTIONARY <-----{0}----->".format(student))

print("")

#I. Demonstrate the updation of python dictionary.

a = input("-->> Enter the key you want to update: ")

b = input("-->> Enter the value of the key({0}): ".format(a))

student[a] = b

print("After Updating the Dictionary --> {0}".format(student))

print("")

#II. Demonstrate the removal of elements from the python dictionary.

c = input("-->> Enter the key you want to remove: ")

del student[c]

print("After removing '{0}' from the given dictionary -->{1}".format(c,student))

print("")

"""III. Demonstrate the use of following dictionary methods- clear(), copy(), get(), items(), keys(), popitem() and values()."""

#keys()

keys = student.keys()

print("The keys of the given Dictionary are:",keys)

print("")

#values()

values = student.values()

print("The values of the keys in the given Dictionary are:",values)

print("")

#get()

d = input("-->> Enter the key for the value you want: ")

get = student.get(d)

print("The value of std\_name is:" ,get)

print("")

#copy()

copy\_dict = student.copy()

print("The copy of given Dictionary -->",copy\_dict)

print("")

#items()

items = student.items()

print("All the key-value pairs in the given Dictionary:",items)

print("")

#popitem()

student.popitem()

print("After using popitem on the given Dictionary, we get: ",student)

print("")

#clear()

student.clear()

print("After clearing the given Dictionary --> {0}".format(student))

print("-\*-\*-\*-\*-\*-END OF PRACTICAL 7-\*-\*-\*-\*-\*-")

**Q.8 Write a python program to demonstrate the use of**

**i.if-else:- Python program to check whether the string is Symmetrical or Palindrome**

CODE:

*#Enrollment No: 202203103510097*

*#Name: Angat Shah*

*#Branch: B.Tech Computer Science and Engineering*

*##I if-else:*

my\_string = input("-->> Enter a String: ")

half\_part = int(len(my\_string) / 2)

first\_half = my\_string[ :half\_part]

second\_half = my\_string[half\_part: ]

*#To check whether the given string is symmetrical or not.*

if len(my\_string)%2 == 0 :

if first\_half == second\_half :

print("{0} --> SYMMETRICAL".format(my\_string))

else :

print("{0} --> NOT SYMMETRICAL".format(my\_string))

else :

print("{0} --> NOT SYMMETRICAL".format(my\_string))

*#To check whether the given string is palindrome or not.*

i=0

for i in range(len(my\_string)):

if my\_string[i] == my\_string[-1-i]:

print("{0} --> PALINDROME\n".format(my\_string))

break

else:

print("{0} --> Not PALINDROME\n".format(my\_string))

print("-\*-\*-\*-\*-\*-END OF PRACTICAL 8\_1-\*-\*-\*-\*-\*-")

**ii.for() - Program to multiply two matrices using nested loops.**

**CODE:**

*#Enrollment No: 202203103510097*

*#Name: Angat Shah*

*#Branch: B.Tech Computer Science and Engineering*

matrix\_1 = []

matrix\_2 = []

result=[]

rows = int(input("-->> Enter the number of rows for the matrices: "))

columns = int(input("-->> Enter the number of columns for the matrices: "))

print()

print("--->>> Enter the values for the First Matrix")

for i in range(rows):

a =[]

for j in range(columns):

a.append(int(input("-->> Enter the elements for the {0} row of first matrix: ".format(i+1))))

print()

matrix\_1.append(a)

print()

print("--->>> Enter the values for the Second Matrix")

for i in range(rows):

b =[]

for j in range(columns):

b.append(int(input("-->> Enter the elements for the {0} row of second matrix: ".format(i+1))))

print()

matrix\_2.append(b)

print()

print("--> FIRST MATRIX")

for i in range(rows):

for j in range(columns):

print(matrix\_1[i][j], end = " ")

print()

print("--> SECOND MATRIX")

for i in range(rows):

for j in range(columns):

print(matrix\_2[i][j], end = " ")

print()

print()

print("-->> MULTIPLICATION OF THE MATRICES")

for i in range(rows):

c = []

for j in range(columns):

c.append(0)

result.append(c)

for i in range(rows):

for j in range(columns):

result[i][j] += matrix\_1[i][j] \* matrix\_2[i][j]

for i in range(rows):

for j in range(columns):

print(result[i][j], end = " ")

print()

print()

print("-\*-\*-\*-\*-\*-END OF PRACTICAL 8\_2-\*-\*-\*-\*-\*-")

**iii. while() :- Addition of even numbers**

CODE:

*#Enrollment No: 202203103510097*

*#Name: Angat Shah*

*#Branch: B.Tech Computer Science and Engineering*

elements = []

num = int(input("-->> Enter the number of elements you want to add: "))

for i in range(num):

a = int(input("-->> Enter {0} Elemment: ".format(i+1)))

elements.append(a)

print("")

print("ELEMENTS <-----{0}----->".format(elements))

i=0

sum=0

while i < num:

if elements[i]%2 == 0:

sum += elements[i]

i += 1

print("")

print("Addition of even numbers from the given elements:",sum)

print("")

print("-\*-\*-\*-\*-\*-END OF PRACTICAL 8\_3-\*-\*-\*-\*-\*-")

**Q.9 Write a python program to display the above patterns.**

**CODE:**

#Enrollment No: 202203103510097

#Name: Angat Shah

#Branch: B.Tech Computer Science and Engineering

#PATTERN I

rows\_1 = 5

for i in range(1,rows\_1+1) :

for j in range(1,i+1) :

print(j,end=" ")

print("")

print('\n')

#PATTERN II

n = 1

for i in range(0,5):

for j in range(0,i+1):

print(n , end=" ")

n = n + 1

print("")

print('\n')

#PATTERN III

k = 5

for i in range(0, 5):

for j in range(0, k):

print(end=" ")

k = k - 1

for j in range(0, i+1):

print(" \*", end="")

print("\r")

print('\n')

#PATTERN IV

c = 65

for i in range(0,5):

for j in range(0,i+1):

ch = chr(c)

print(ch, end=" ")

c = c + 1

print("")

print('\n')

print("-\*-\*-\*-\*-\*-END OF PRACTICAL 9-\*-\*-\*-\*-\*-")

**Q.10 Write a python program to demonstrate the use of user-defined functions with single, multiple and arbitrary arguments.**

**i.WAP to design simple calculator**

*#Enrollment No: 202203103510097*

*#Name: Angat Shah*

*#Branch: B.Tech Computer Science and Engineering*

*# SINGLE ARGUMENT*

def fac(a):

print("SINGLE ARGUMENTS")

fact = 1

for i in range(1,a+1):

fact \*= i

print("{0}! --> {1}".format(a,fact))

print()

*# MULTIPLE ARGUMENTS*

def compare(n1, n2):

print("MULTIPLE ARGUMENTS")

if n1 > n2 :

print("{0} Is grater than {1}".format(n1,n2))

elif n2 > n1 :

print("{0} Is grater than {1}".format(n2,n1))

else :

print("{0} Is equal to {1}".format(n1,n2))

print()

*# ARBITARY ARGUMENTS*

def num\_add(\*args):

print("ARBITARY ARGUMENTS")

args = []

num = int(input("-->> Enter the number of elements you want to add: "))

for i in range(num):

a = int(input("-->> Enter {0} Elemment: ".format(i+1)))

args.append(a)

sum = 0

for i in args :

sum += i

print("The sum of all numbers: ",sum)

print()

*# CALCULATOR*

def calculator():

print("CALCULATOR")

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

op = input("Enter the operator(+,-,\*,/): ")

b = int(input("Enter second number: "))

if op=='+' :

print("{0} {1} {2} = {3}".format(a,op,b,a+b))

elif op=='-' :

print("{0} {1} {2} = {3}".format(a,op,b,a-b))

elif op=='\*' :

print("{0} {1} {2} = {3}".format(a,op,b,a\*b))

elif op=='/' :

print("{0} {1} {2} = {3}".format(a,op,b,a/b))

else :

print("ERROR")

fac(5)

compare(11,30)

num\_add()

calculator()

print()

print("-\*-\*-\*-\*-\*-END OF PRACTICAL 10-\*-\*-\*-\*-\*-")

**Q.11 Create a class named employee having attributes - emp\_name, emp\_age, and emp\_city. Create a method named get\_data() in employee class that takes user input for these attributes. Derive a class named emp\_derived() from the employee class, having an \_\_init\_\_() method that displays the attributes of the employee class upon instantiation.**

**CODE:**

#Enrollment No: 202203103510097

#Name: Angat Shah

#Branch: B.Tech Computer Science and Engineering

class Employee :

def get\_data(self) :

self.emp\_name = input("-->> Enter Employee Name: ")

self.emp\_age = int(input("-->> Enter Employee Age: "))

self.emp\_city = input("-->> Enter Employee City: ")

class Emp\_Derived(Employee) :

def \_\_init\_\_(self) :

self.get\_data()

print("")

print(f"EMPLOYEE NAME: {self.emp\_name}")

print(f"EMPLOYEE AGE: {self.emp\_age}")

print(f"EMPLOYEE CITY: {self.emp\_city}")

obj = Emp\_Derived()

print("")

print("-\*-\*-\*-\*-\*-END OF PRACTICAL 11-\*-\*-\*-\*-\*-")

**Q.12**

**CODE:**

*#Enrollment No: 202203103510097*

*#Name: Angat Shah*

*#Branch: B.Tech Computer Science and Engineering*

class University:

def \_\_init\_\_(self, name, year\_of\_estd, city):

*self*.name = name

*self*.year\_of\_estd = year\_of\_estd

*self*.city = city

def display(self):

print("Name:", *self*.name)

print("Year of establishment:", *self*.year\_of\_estd)

print("City:", *self*.city)

class professor(University):

def \_\_init\_\_(self, name, year\_of\_estd, city, designation, highest\_qualification, area\_of\_research, year\_of\_joining, year\_of\_experience, name\_of\_institute):

super().\_\_init\_\_(name, year\_of\_estd, city)

*self*.designation = designation

*self*.highest\_qualification = highest\_qualification

*self*.area\_of\_research = area\_of\_research

*self*.year\_of\_joining = year\_of\_joining

*self*.year\_of\_experience = year\_of\_experience

*self*.name\_of\_institute = name\_of\_institute

def display(self):

super().display()

print("Designation:", *self*.designation)

print("Highest qualification:", *self*.highest\_qualification)

print("Area of research:", *self*.area\_of\_research)

print("Year of joining:", *self*.year\_of\_joining)

print("Year of experience:", *self*.year\_of\_experience)

print("Name of institute:", *self*.name\_of\_institute)

class LabAssistant(University):

designation = "Lab Assistant"

def \_\_init\_\_(self, name, year\_of\_estd, city, highest\_qualification, additional\_skills, year\_of\_joining, name\_of\_institute):

super().\_\_init\_\_(name, year\_of\_estd, city)

*self*.highest\_qualification = highest\_qualification

*self*.additional\_skills = additional\_skills

*self*.year\_of\_joining = year\_of\_joining

*self*.name\_of\_institute = name\_of\_institute

def display(self):

super().display()

print("Designation:", *self*.designation)

print("Highest qualification:", *self*.highest\_qualification)

print("Additional skills:", *self*.additional\_skills)

print("Year of joining:", *self*.year\_of\_joining)

print("Name of institute:", *self*.name\_of\_institute)

class OfficeAssistant(University):

designation = "Office Assistant"

def \_\_init\_\_(self, name, year\_of\_estd, city, highest\_qualification, year\_of\_joining, name\_of\_institute):

super().\_\_init\_\_(name, year\_of\_estd, city)

*self*.highest\_qualification = highest\_qualification

*self*.year\_of\_joining = year\_of\_joining

*self*.name\_of\_institute = name\_of\_institute

def display(self):

super().display()

print("Designation:", *self*.designation)

print("Highest qualification:", *self*.highest\_qualification)

print("Year of joining:", *self*.year\_of\_joining)

print("Name of institute:", *self*.name\_of\_institute)

class Peon(University):

job\_role = "Office Peon"

def \_\_init\_\_(self, name, year\_of\_estd, city, qualification, year\_of\_joining, name\_of\_institute):

super().\_\_init\_\_(name, year\_of\_estd, city)

*self*.qualification = qualification

*self*.year\_of\_joining = year\_of\_joining

*self*.name\_of\_institute = name\_of\_institute

def display(self):

super().display()

print("Job role:", *self*.job\_role)

print("Qualification:", *self*.qualification)

print("Year of joining:", *self*.year\_of\_joining)

print("Name of institute:", *self*.name\_of\_institute)

print("Choose an option:")

print("1. Professor")

print("2. Lab Assistant")

print("3. Office Assistant")

print("4. Peon")

print("5. Exit")

ch = int(input("Enter the choice to get data of? "))

print()

print("ABOUT UNIVERSITY: ")

uni = University("Uka Tarsadia University", 2011, "Bardoli")

uni.display()

print()

if ch == 1:

print("INFORMATION OF PROFESSOR: ")

prof = professor("John Wick", 1990, "New York", "Assistant professor", "PhD", "Artificial Intelligence", 2015, 6, "MIT University")

prof.display()

elif ch == 2:

print("Information of Lab Assistant: ")

la = LabAssistant("Ethan Hunt", 1987, "New York", "B.Tech", "AI and Machine Learning", 2016, "Harvard University")

la.display()

elif ch == 3:

print("Information of Office Assistant: ")

oa = OfficeAssistant("Stephen Strange", 1987, "Manhattan", "MBA", 2019, "Standford")

oa.display()

elif ch == 4:

print("Information of Peon: ")

pn = Peon("Finn Mikaelson", 1981, "Texas", "BBA", 2020, "Harvard University")

pn.display()

else :

print("Invalid Error")

print()

print("-\*-\*-\*-\*-\*-END OF PRACTICAL 12-\*-\*-\*-\*-\*-")

class University:

def \_\_init\_\_(self, name, year\_of\_estd, city):

*self*.name = name

*self*.year\_of\_estd = year\_of\_estd

*self*.city = city

def display(self):

print(f"Name: {*self*.name}\nYear of establishment: {*self*.year\_of\_estd}\nCity: {*self*.city}")

class Professor(University):

def \_\_init\_\_(self, name, year\_of\_joining, year\_of\_experience, highest\_qualification, area\_of\_research,

name\_of\_institute, designation):

super().\_\_init\_\_(name\_of\_institute, year\_of\_estd, city)

*self*.designation = designation

*self*.highest\_qualification = highest\_qualification

*self*.area\_of\_research = area\_of\_research

*self*.year\_of\_joining = year\_of\_joining

*self*.year\_of\_experience = year\_of\_experience

*self*.name\_of\_institute = name\_of\_institute

def display(self):

super().display()

print(f"Designation: {*self*.designation}\nHighest qualification: {*self*.highest\_qualification}\n"

f"Area of research: {*self*.area\_of\_research}\nYear of joining: {*self*.year\_of\_joining}\n"

f"Year of experience: {*self*.year\_of\_experience}\nName of institute: {*self*.name\_of\_institute}")

class LabAssistant(University):

designation = "Lab Assistant"

def \_\_init\_\_(self, name, year\_of\_joining, highest\_qualification, additional\_skills, name\_of\_institute):

super().\_\_init\_\_(name\_of\_institute, year\_of\_estd, city)

*self*.highest\_qualification = highest\_qualification

*self*.additional\_skills = additional\_skills

*self*.year\_of\_joining = year\_of\_joining

*self*.name\_of\_institute = name\_of\_institute

def display(self):

super().display()

print(f"Designation: {*self*.designation}\nHighest qualification: {*self*.highest\_qualification}\n"

f"Additional skills: {*self*.additional\_skills}\nYear of joining: {*self*.year\_of\_joining}\n"

f"Name of institute: {*self*.name\_of\_institute}")

class OfficeAssistant(University):

designation = "Office Assistant"

def \_\_init\_\_(self, name, year\_of\_joining, highest\_qualification, name\_of\_institute):

super().\_\_init\_\_(name\_of\_institute, year\_of\_estd, city)

*self*.highest\_qualification = highest\_qualification

*self*.year\_of\_joining = year\_of\_joining

*self*.name\_of\_institute = name\_of\_institute

def display(self):

super().display()

print(f"Designation: {*self*.designation}\nHighest qualification: {*self*.highest\_qualification}\n"

f"Year of joining: {*self*.year\_of\_joining}\nName of institute: {*self*.name\_of\_institute}")

class Peon(University):

job\_role = "Office Peon"

def \_\_init\_\_(self, name, year\_of\_joining, qualification, name\_of\_institute):

super().\_\_init\_\_(name\_of\_institute, year\_of\_estd, city)

*self*.qualification = qualification

*self*.year\_of\_joining = year\_of\_joining

*self*.name\_of\_institute = name\_of\_institute

def display(self):

super().display()

print(f"Job role: {*self*.job\_role}\nQualification: {*self*.qualification}\nYear of joining: {*self*.year\_of\_joining}\n"

f"Name of institute: {*self*.name\_of\_institute}")

*# Main program*

while True:

print("Choose an option:")

print("1. Professor")

print("2. Lab Assistant")

print("3. Office Assistant")

print("4. Peon")

print("5. Exit")

choice = input("Enter your choice: ")

if choice == '1':

name = input("Enter name: ")

year\_of\_joining = input("Enter year of joining: ")

year\_of\_experience = input("Enter years of experience: ")

highest\_qualification = input("Enter highest qualification: ")

area\_of\_research = input("Enter area of research: ")

name\_of\_institute = input("Enter name of institute: ")

designation = input("Enter designation: ")

professor = Professor(name, year\_of\_joining, year\_of\_experience, highest\_qualification, area\_of\_research,

name\_of\_institute, designation)

professor.display()

elif choice == '2':

name = input("Enter name: ")

year\_of\_joining = input("Enter year of joining: ")

highest\_qualification = input("Enter highest qualification: ")

additional\_skills = input("Enter additional skills: ")

name\_of\_institute = input("Enter name of institute: ")

lab\_assistant = LabAssistant(name, year\_of\_joining, highest\_qualification, additional\_skills, name\_of\_institute)

lab\_assistant.display()

elif choice == '3':

name = input("Enter name: ")

year\_of\_joining = input("Enter year of joining: ")

highest\_qualification = input("Enter highest qualification: ")

name\_of\_institute = input("Enter name of institute: ")

office\_assistant = OfficeAssistant(name, year\_of\_joining, highest\_qualification, name\_of\_institute)

office\_assistant.display()

elif choice == '4':

name = input("Enter name: ")

year\_of\_joining = input("Enter year of joining: ")

qualification = input("Enter qualification: ")

name\_of\_institute = input("Enter name of institute: ")

peon = Peon(name, year\_of\_joining, qualification, name\_of\_institute)

peon.display()

elif choice == '5':

break

else:

print("Invalid choice")

**Q.13 Write a python program to create a class named area. Define a class method find\_area() that can find areas of different shapes whose value is given by the user. Invoke the class method by instantiation and prove method overloading.**

**CODE:**

*#Enrollment No: 202203103510097*

*#Name: Angat Shah*

*#Branch: B.Tech Computer Science and Engineering*

class Area:

def find\_area(self, x = None, y = None):

if x != None and y != None:

print("Area of Rectangle:",x\*y)

elif x != None and y == None:

print("Area of Square:",x\*x)

obj = Area()

parameters = int(input("-->> Enter Number of Parameters: "))

if parameters == 1 :

a = int(input("-->> Enter The Parameter: "))

obj.find\_area(a)

elif parameters == 2 :

a = int(input("-->> Enter First Parameters: "))

b = int(input("-->> Enter Second Parameters: "))

obj.find\_area(a,b)

print("")

print("-\*-\*-\*-\*-\*-END OF PRACTICAL 13-\*-\*-\*-\*-\*-")

**Q.14 Write a python program to demonstrate the use of method overriding.**

**CODE:**

#Enrollment No: 202203103510097

#Name: Angat Shah

#Branch: B.Tech Computer Science and Engineering

class Super:

def display(self):

print("--> I am a Super Class")

class Sub(Super):

def display(self):

print("--> I am a Sub Class")

super().display()

obj = Sub()

obj.display()

print("")

print("-\*-\*-\*-\*-\*-END OF PRACTICAL 14-\*-\*-\*-\*-\*-")

**Q.15 Write a python program to perform basic matrix operations on user entered matrices.**

**CODE:**

#Enrollment No: 202203103510097

#Name: Angat Shah

#Branch: B.Tech Computer Science and Engineering

rows = int(input("-->> Enter the number of rows for the matrix: "))

columns = int(input("-->> Enter the number of columns for the matrix: "))

print()

matrix\_1 = []

matrix\_2 = []

result = []

print("--->>> Enter the values for the First Matrix")

for i in range(rows):

a =[]

for j in range(columns):

a.append(int(input("-->> Enter the elements for the {0} row of first matrix: ".format(i+1))))

print()

matrix\_1.append(a)

print()

print("--->>> Enter the values for the Second Matrix")

for i in range(rows):

b =[]

for j in range(columns):

b.append(int(input("-->> Enter the elements for the {0} row of second matrix: ".format(i+1))))

print()

matrix\_2.append(b)

print()

print("--> FIRST MATRIX")

for i in range(rows):

for j in range(columns):

print(matrix\_1[i][j], end = " ")

print()

print("--> SECOND MATRIX")

for i in range(rows):

for j in range(columns):

print(matrix\_2[i][j], end = " ")

print()

print()

print("1 -> ADDITION\n2 -> SUBTRATCTION\n3 -> MULTIPLICATION\n")

operation = int(input("-->> Enter the choice of operation you would like to perform on the matrix: "))

for i in range(rows):

c = []

for j in range(columns):

c.append(0)

result.append(c)

print()

if operation == 1 :

for i in range(rows) :

for j in range(columns) :

result[i][j] = matrix\_1[i][j] + matrix\_2[i][j]

print("--> ADDITION OF THE MATRIX")

for i in range(rows) :

for j in range(columns) :

print(result[i][j],end=" ")

print()

elif operation == 2 :

for i in range(rows) :

for j in range(columns) :

result[i][j] = matrix\_1[i][j] - matrix\_2[i][j]

print("--> SUBTRACTION OF THE MATRIX")

for i in range(rows) :

for j in range(columns) :

print(result[i][j],end=" ")

print()

elif operation == 3 :

for i in range(rows) :

for j in range(columns) :

result[i][j] += matrix\_1[i][j] \* matrix\_2[i][j]

print("--> MULTIPLICATION OF THE MATRIX")

for i in range(rows) :

for j in range(columns) :

print(result[i][j],end=" ")

print()

print()

print("-\*-\*-\*-\*-\*-END OF PRACTICAL 15-\*-\*-\*-\*-\*-")

**Q.16 Write a python program to demonstrate the use of data hiding.**

**CODE:**

*#Enrollment No: 202203103510097*

*#Name: Angat Shah*

*#Branch: B.Tech Computer Science and Engineering*

class Card():

def get\_data(self):

*self*.\_card\_number = int(input("-->> ENTER CARD NUMBER :"))

*self*.card\_exyear = int(input("-->> ENTER CARD EXPIRY YEAR : "))

*self*.card\_name = input("-->> ENTER NAME ON THE CARD: ")

*self*.\_\_card\_cvv = int(input("-->> ENTER CVV CODE : "))

class Owner(Card):

def \_\_init\_\_(self) :

*self*.get\_data()

print(f"CARD NUMBER : {*self*.\_card\_number}")

print(f"EXPIRY YEAR : {*self*.card\_exyear}")

print(f"NAME ON THE CARD : {*self*.card\_name}")

print(f"CVV CODE : {*self*.\_\_card\_cvv}")

print()

obj = Owner()

print("-\*-\*-\*-\*-\*-END OF PRACTICAL 16-\*-\*-\*-\*-\*-")

**Q.17**

**CODE:**

*#Enrollment No: 202203103510097*

*#Name: Angat Shah*

*#Branch: B.Tech Computer Science and Engineering*

*def try\_catch():*

*my\_tuple = (1,2,3,4)*

*try:*

*my\_tuple.append(5)*

*except :*

*print("We cannot append the tuple due to immutability")*

*def raise\_keyword():*

*x = -11*

*if x < 0:*

*raise ValueError("Value is invalid print\n-\*-\*-\*-\*-\*-END OF PRACTICAL 17-\*-\*-\*-\*-\*-")*

*def try\_finally(x, y):*

*try:*

*result = x // y*

*except ZeroDivisionError:*

*print("You are dividing by zero ")*

*else:*

*print("Your answer is --> ", result)*

*finally:*

*print('This is Floor Division')*

*print("TRY-CATCH BLOCK:")*

*try\_catch()*

*print()*

*print("TRY-FINALLY BLOCK:")*

*try\_finally(3, 0)*

*print()*

*print("RAISE KEYWORD:")*

*raise\_keyword()*

*print()*

*print("-\*-\*-\*-\*-\*-END OF PRACTICAL 17-\*-\*-\*-\*-\*-")*

Q.18

CODE:

*#Enrollment No: 202203103510097*

*#Name: Angat Shah*

*#Branch: B.Tech Computer Science and Engineering*

""" Regular Expression is a meaningful sequence of patterns that attempts to operate on input text.

Regular Expressions are useful in wide variety of text processing tasks. They even used for validation,

data scraping, parsing, etc. """

""" Python provide in-built module named "re" that can be imported in python as "import re"

For more detail study on re you can visit https://docs.python.org/3/library/re.html """

*#Note: Line 1-8 are for your study, it should be removed in final practical.*

import re

*#We want to search for a 6 letters string startswith (^) A and endswith ($) S. Any string qualifies the criteria will be considered valid.*

re\_pattern = "^A....S$"

my\_string = input("-->> Enter your Institute Name: ")

check = re.match(re\_pattern, my\_string) *#Method-1*

if check:

print("--> You have entered correct data for your institute.\n")

else:

print("--> Invalid institute name.\n")

*#We want to match for all (+) decimal digits (\d) in input.*

re\_pattern = "\d+"

my\_string = input("-->> Enter your address:")

decimals = re.findall(re\_pattern, my\_string) *#Method-2*

print(decimals)

*#We want to search for specific character (\A) in user given input:*

re\_pattern = "\AComputer"

my\_string = input("-->> Enter your course name:")

success = re.search(re\_pattern, my\_string)

if success:

print("--> You have enter a correct course name.\n")

else:

print("--> Invalid course name.\n")

*#We want to validate the use entered Indian mobile number.*

re\_pattern = "^[6789]\d{9}$" *#starting with (^) having starting digits 6-9[6789] is decimal (\d) remaining size of {9} and end($)*

my\_string = input("-->> Enter your Mobile Number: ")

match = re.match(re\_pattern, my\_string)

if match:

print("--> You have entered valid mobile number.\n")

else:

print("--> Invalid mobile number:\n")

re\_pattern = "^[a-z0-9]+[\.\_]\*[a-z0-9]+@[a-z]+\.[a-z]{2,3}[\.]?([a-z]{2,3})?$"

my\_email = input("-->> Enter your e-mail ID: ")

match = re.match(re\_pattern, my\_email)

if match:

print("--> You have entered valid e-mail ID.\n")

else:

print("--> Invalid e-mail.\n")

print("-\*-\*-\*-\*-\*-END OF PRACTICAL 18-\*-\*-\*-\*-\*-")

Q.19

CODE:

*#Enrollment No: 202203103510097*

*#Name: Angat Shah*

*#Branch: B.Tech Computer Science and Engineering*

def count\_words(file\_path):

with open(file\_path, 'r') as f:

content = f.read()

words = content.split()

return len(words)

file\_path = 'words.txt'

word\_count = count\_words(file\_path)

print(f'The file {file\_path}contains {word\_count} words.\n')

print("-\*-\*-\*-\*-\*-END OF PRACTICAL 19-\*-\*-\*-\*-\*-")

Q.20

CODE:

*#Enrollment No: 202203103510097*

*#Name: Angat Shah*

*#Branch: B.Tech Computer Science and Engineering*

def copy\_file(source\_file, destination\_file):

with open(source\_file, 'r') as f1, open(destination\_file, 'w') as f2:

content = f1.read()

f2.write(content)

source\_file = 'words.txt'

destination\_file = 'words\_copy.txt'

copy\_file(source\_file, destination\_file)

print(f'--> Content of {source\_file} has been copied to {destination\_file}.')

print("")

print("-\*-\*-\*-\*-\*-END OF PRACTICAL 20-\*-\*-\*-\*-\*-")

Q.21

CODE:

import string

import os

alphabet = string.ascii\_uppercase

for letter in alphabet:

filename = letter + '.txt'

with open(filename, 'w') as f:

f.write('This is file ' + filename)

print(f'File {filename} has been created in the current directory.')

Q.22

CODE:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

*# Example usage of the imported packages*

x = np.array([0, 1, 2, 3, 4])

y = np.array([1, 3, 7, 13, 21])

plt.plot(x, y, 'ro')

plt.show()

df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})

print(df)

model = LinearRegression()

X = [[0], [1], [2], [3], [4]]

y = [1, 3, 7, 13, 21]

model.fit(X, y)

print('Intercept:', model.intercept\_)

print('Coefficient:', model.coef\_)

Q.23

CODE:

*#Enrollment No: 202203103510097*

*#Name: Angat Shah*

*#Branch: B.Tech Computer Science and Engineering*

import requests

from bs4 import BeautifulSoup

url = 'https://en.wikipedia.org/wiki/Python\_(programming\_language)'

response = requests.get(url)

content = response.content

soup = BeautifulSoup(response.content, 'html.parser')

title = soup.find('h1', id='firstHeading')

print('TITLE:', title.text)

print("")

heading\_2 = soup.find\_all('h2')

for subtitle\_1 in heading\_2:

print("SUBTITLE\_1:",subtitle\_1.text)

print("")

heading\_3 = soup.find\_all('h3')

for subtitle\_2 in heading\_3:

print("SUBTITLE\_2:",subtitle\_2.text)

print("")

print("-\*-\*-\*-\*-\*-END OF PRACTICAL 23-\*-\*-\*-\*-\*-")

Q.24

CODE:

from selenium import webdriver

*# Create a new instance of the Chrome driver*

driver = webdriver.Chrome()

*# Navigate to the web page you want to automate*

driver.get('https://www.example.com')

*# Find the button element on the page and click it*

button = driver.find\_element\_by\_css\_selector('button')

button.click()

*# Close the web driver*

driver.quit()