ASSIGNMNT

Write a Python program to create a tuple of numbers and print one item.

# Create a tuple containing a sequence of numbers

tuplex = 5, 10, 15, 20, 25

print(tuplex)

# Create a tuple with a single item (note the comma after the item)

tuplex = 5,

print(tuplex)

Write a Python program to unpack a tuple into several variables.

# Create a tuple containing three numbers

tuplex = 4, 8, 3

print(tuplex)

# Unpack the values from the tuple into the variables n1, n2, and n3

n1, n2, n3 = tuplex

print(n1 + n2 + n3)

# Attempt to unpack the tuple into more variables (n1, n2, n3, and n4)

n1, n2, n3, n4 = tuplex

Write a Python program to get the 4th element from the last element of a tuple.

# Create a tuple containing a sequence of items

tuplex = ("w", “e”, "C", "o", "m", "p", "u", "t", "e", "r")

print(tuplex)

item = tuplex[3]

print(item)

item1 = tuplex[-4]

print(item1)

Write a Python program to reverse a tuple.

x = ("DishaComputer")

y = reversed(x)

print(tuple(y))

x = (5, 10, 15, 20)

y = reversed(x)

print(tuple(y))

Write a Python program to convert a list of tuples into a dictionary..

l = [("x", 1), ("x", 2), ("x", 3), ("y", 1), ("y", 2), ("z", 1)]

d = {}

for a, b in l:

d.setdefault(a, []).append(b)

print(d)

**object oriented programming**

Write a Python program to create a class representing a Circle. Include methods to calculate its area and perimeter.

**# Import the math module to access mathematical functions like pi**

**import math**

**class Circle:**

**def \_\_init\_\_(self, radius):**

**self.radius = radius**

**def calculate\_circle\_area(self):**

**return math.pi \* self.radius\*\*2**

**def calculate\_circle\_perimeter(self):**

**return 2 \* math.pi \* self.radius**

**radius = float(input("Input the radius of the circle: "))**

**circle = Circle(radius)**

**area = circle.calculate\_circle\_area()**

**perimeter = circle.calculate\_circle\_perimeter()**

**print("Area of the circle:", area)**

**print("Perimeter of the circle:", perimeter)**

Write a Python program to create a person class. Include attributes like name, country and date of birth. Implement a method to determine the person's age.

# Import the date class from the datetime module to work with dates

from datetime import date

class Person:

def \_\_init\_\_(self, name, country, date\_of\_birth):

self.name = name

self.country = country

self.date\_of\_birth = date\_of\_birth

def calculate\_age(self):

today = date.today()

age = today.year - self.date\_of\_birth.year

if today < date(today.year, self.date\_of\_birth.month, self.date\_of\_birth.day):

age -= 1

return age

person1 = Person("Ferdi Odilia", "France", date(1962, 7, 12))

person2 = Person("Shweta Maddox", "Canada", date(1982, 10, 20))

person3 = Person("Elizaveta Tilman", "USA", date(2000, 1, 1))

print("Person 1:")

print("Name:", person1.name)

print("Country:", person1.country)

print("Date of Birth:", person1.date\_of birth)

print("Age:", person1.calculate\_age())

print("\nPerson 2:")

print("Name:", person2.name)

print("Country:", person2.country)

print("Date of Birth:", person2.date\_of birth)

print("Age:", person2.calculate\_age())

print("\nPerson 3:")

print("Name:", person3.name)

print("Country:", person3.country)

print("Date of Birth:", person3.date\_of birth)

print("Age:", person3.calculate\_age())

Write a Python program to create a calculator class. Include methods for basic arithmetic operations.

# Define a class called Calculator to perform basic arithmetic operations

class Calculator:

def add(self, x, y):

return x + y

def subtract(self, x, y):

return x - y

def multiply(self, x, y):

return x \* y

def divide(self, x, y):

if y != 0:

return x / y

else:

return ("Cannot divide by zero.")

calculator = Calculator()

result = calculator.add(7, 5)

print("7 + 5 =", result)

result = calculator.subtract(34, 21)

print("34 - 21 =", result)

result = calculator.multiply(54, 2)

print("54 \* 2 =", result)

result = calculator.divide(144, 2)

print("144 / 2 =", result)

result = calculator.divide(45, 0)

print("45 / 0 =", result)