Assignment – 1

1. Write a Python program to calculate the area of a rectangle given its length and width

def calculate\_area(length, width):

area = length \* width

return area

length = float(input("Enter the length of the rectangle: "))

width = float(input("Enter the width of the rectangle: "))

area = calculate\_area(length, width)

print("The area of the rectangle is:", area)

1. Write a program to convert miles to kilometers

conversion\_factor = 1.60934

miles = 10

kilometers = miles \* conversion\_factor

print(f"{miles} miles is equal to {kilometers:.2f} kilometers")

1. Write a function to check if a given string is a palindrome

def is\_palindrome(text):

text = ''.join(char.lower() for char in text if char.isalnum())

return text == text[::-1]

print(is\_palindrome("racecar")) # True

print(is\_palindrome("hello")) # False

print(is\_palindrome("A man, a plan, a canal: Panama")) # True (ignoring punctuation)

1. Write a Python program to find the second largest element in a list

def find\_second\_largest(numbers):

if len(numbers) < 2:

return None

largest = second\_largest = float('-inf')

for num in numbers:

if num > largest:

second\_largest = largest

largest = num

elif num > second\_largest and num != largest:

second\_largest = num

return second\_largest

numbers = [10, 20, 5, 8, 15]

second\_largest = find\_second\_largest(numbers)

if second\_largest is not None:

print("The second largest element is:", second\_largest)

else:

print("The list has less than 2 elements.")

1. Explain what indentation means in Python

Defining Code Blocks: Unlike most programming languages that use curly braces {} to define code blocks (like if statements, loops, etc.), Python relies solely on indentation to achieve the same functionality.

Adding spaces at the beginning of a line indicates that the line belongs to the code block started on the previous indented line.

Increasing indentation further creates nested blocks within the original block.

Decreasing indentation (usually by the same amount used for indentation) marks the end of the block.

For example:

if age >= 18:

print("You are eligible to vote.")

else:

print("You are not eligible to vote.")

Here, the indented lines under if and else belong to their respective code blocks, indicating the statements to be executed only when the condition is true or false.

Improving Readability: While Python enforces indentation for defining code blocks, it also significantly enhances the readability of the code.

Visual Cues: Indentation creates visual cues for the structure of your code, making it easier to understand the flow and hierarchy of your program.

Consistent Indentation: Maintaining consistent indentation throughout your code (usually 4 spaces is the standard) improves maintainability and helps you and others locate specific sections of code.

1. Write a program to perform set difference operation.

def difference\_of\_sets(set1, set2):

difference = set1.difference(set2)

return difference

# Example usage

set1 = {1, 2, 3, 4, 5}

set2 = {2, 4, 6, 8}

# Find the difference

difference = difference\_of\_sets(set1, set2)

# Print the difference set

print("The difference of sets:", difference)

1. Write a Python program to print numbers from 1 to 10 using a while loop

# Initialize a counter variable

counter = 1

# Loop until the counter reaches 11 (exclusive)

while counter <= 10:

# Print the current value of the counter

print(counter)

# Increment the counter

counter += 1

1. Write a program to calculate the factorial of a number using a while loop

def factorial(n):

if n < 0:

return None

# Initialize the factorial to 1

factorial = 1

# Use a while loop to calculate the factorial

counter = 1

while counter <= n:

factorial \*= counter

counter += 1

return factorial

# Calculate and print the factorial of 5

result = factorial(5)

print("The factorial of 5 is:", result)

1. Write a Python program to check if a number is positive, negative, or zero using if-elif-else statements

def check\_number(number):

if number > 0:

return "The number is positive."

elif number < 0:

return "The number is negative."

else:

return "The number is zero."

# Get user input for the number

number = float(input("Enter a number: "))

# Check the number and print the result

result = check\_number(number)

print(result)

1. Write a program to determine the largest among three numbers using conditional statements

def find\_largest(num1, num2, num3):

# Compare the first two numbers and store the larger one in largest

largest = num1 if num1 >= num2 else num2

# Compare the largest with the third number and update if necessary

largest = largest if largest >= num3 else num3

return largest

# Get user input for the numbers

num1 = float(input("Enter the first number: "))

num2 = float(input("Enter the second number: "))

num3 = float(input("Enter the third number: "))

# Find the largest number

largest\_number = find\_largest(num1, num2, num3)

# Print the result

print("The largest number among", num1, ",", num2, "and", num3, "is:", largest\_number)

1. Write a Python program to create a numpy array filled with ones of given shap

import numpy as np

# Define the desired shape of the array

shape = (3, 4)

# Create an array filled with ones using np.ones()

array\_of\_ones = np.ones(shape)

# Print the array

print(array\_of\_ones)

1. Write a program to create a 2D numpy array initialized with random integer

import numpy as np

# Define the shape of the 2D array

shape = (3, 4)

# Create a 2D array filled with random integers between 0 and 10 (exclusive)

random\_array = np.random.randint(0, 10, size=shape)

# Print the array

print(random\_array)

1. Write a Python program to generate an array of evenly spaced numbers over a specified range using linspace

import numpy as np

# Define the start, stop, and number of elements

start = 0

stop = 10

num\_elements = 5

# Generate an array of evenly spaced numbers using linspace

array\_of\_numbers = np.linspace(start, stop, num\_elements)

# Print the array

print(array\_of\_numbers)

1. Write a program to generate an array of 10 equally spaced values between 1 and 100 using linspace.

import numpy as np

# Define the starting and ending values

start = 1

stop = 100

# Set the number of elements (evenly spaced)

num\_elements = 10

# Generate the array of evenly spaced values using linspace

array\_of\_numbers = np.linspace(start, stop, num\_elements)

# Print the array

print(array\_of\_numbers)

1. Write a Python program to create an array containing even numbers from 2 to 20 using arange.

import numpy as np

# Define the starting and ending values for even numbers (inclusive)

start = 2

stop = 20

# Set the step size to 2 to select only even numbers

step = 2

# Create the array of even numbers using arange

array\_of\_even\_numbers = np.arange(start, stop + 1, step)

# Print the array

print(array\_of\_even\_numbers)

1. Write a program to create an array containing numbers from 1 to 10 with a step size of 0.5 using arrange

import numpy as np

start = 1

stop = 10

step = 0.5

array\_of\_numbers = np.arange(start, stop, step)

print(array\_of\_numbers)