ASSIGNMENT-01

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

Sol:
```
def calculate_rectangle_area(length, width):
    area = length * width
    return area

def main():
    length = float(input("Enter the length of the rectangle: "))
    width = float(input("Enter the width of the rectangle: "))
    area = calculate_rectangle_area(length, width)
    print("The area of the rectangle is:", area)

if __name__ == "__main__":
    main()
```

2. **Write a program to convert miles to kilometre?**

Sol:
```
def miles_to_kilometers(miles):
    kilometers = miles * 1.60934
    return kilometers

def main():
    miles = float(input("Enter the distance in miles: "))
    kilometers = miles_to_kilometers(miles)
    print(f"{miles} miles is equal to {kilometers} kilometers.")
```

3. **Write a function to check if a given string is a palindrome**?

Sol:
```
def is_palindrome(s):
    s = ''.join(char.lower() for char in s if char.isalnum())
    return s == s[::-1]

# Example usage:
string = input("Enter a string: ")
if is_palindrome(string):
    print("The string is a palindrome.")
else:
    print("The string is not a palindrome.")
```

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

Sol.
```
def find_second_largest(numbers):
    # Ensure the list has at least two elements
    if len(numbers) < 2:
        return "List should have at least two elements."

    # Initialize the maximum and second maximum variables
    max_num = max(numbers[0], numbers[1])
    second_max = min(numbers[0], numbers[1])

    # Iterate through the list to find the second largest element
    for num in numbers[2:]:
        if num > max_num:
            second_max = max_num
            max_num = num
        elif num > second_max and num != max_num:
            second_max = num

    return second_max
```

5. **Explain what indentation means in Python.**

Sol; Block Structure: Indentation is used to define the block structure of code such as loops, conditional statements, function definitions, class definitions, etc. Blocks of code at the same hierarchical level must have the same indentation.

Consistency is Key: Python relies on consistent indentation to determine the structure of the code. Mixing tabs and spaces for indentation or using inconsistent levels of indentation will result in syntax errors.

No Braces: Unlike languages like C, Java, or JavaScript, Python does not use braces {} to delineate blocks of code. Instead, it uses indentation to specify where a block of code begins and ends.

Whitespace: Python code is sensitive to whitespace, particularly at the beginning of lines. Indentation errors are common sources of syntax errors in Python programs.

Recommended Style: PEP 8, the official Python style guide, recommends using 4 spaces for each level of indentation. This is the most widely adopted convention in the Python community.

Readability: Indentation enhances code readability by making the structure of the code visually clear. It helps developers quickly understand the logical flow of the program.

6. Write a program to perform set difference operation.

Sol;

```
def set_difference(set1, set2):
    return set1 - set2

# Example usage:
set1 = {1, 2, 3, 4, 5}
set2 = {3, 4, 5, 6, 7}
result = set_difference(set1, set2)
print("Set difference:", result)
```

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

Sol;

```
# Initialize the counter
num = 1
# Iterate using a while loop
while num <= 10:
print(num)
num += 1
```

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

Sol;

```
def factorial(n):

 if n < 0:
return "Factorial is not defined for negative numbers"
elif n == 0:
return 1
else:
factorial_result = 1
while n > 0:
factorial_result *= n
   n -= 1
return factorial_result

# Example usage:
number = int(input("Enter a number to calculate its factorial: "))
result = factorial(number)
 print("Fact
```

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

Sol;

```
def check_number(num):
    if num > 0:
        return "Positive"
    elif num < 0:
        return "Negative"
    else:
        return "Zero"

# Example usage:
number = float(input("Enter a number: "))
result = check_number(number)
print("The number is", result)
```

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

Sol;

```
def find_largest(num1, num2, num3):
    if num1 >= num2 and num1 >= num3:
        return num1
    elif num2 >= num1 and num2 >= num3:
        return num2
    else:
        return num3

# Example usage:
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))
```

```
largest = find_largest(num1, num2, num3)
print("The largest number among", num1, ",", num2, ", and", num3, "is", largest)
```

11. . **Write a Python program to create a numpy array filled with ones of given shape**.

Sol;

```
import numpy as np

def create_ones_array(shape):
return np.ones(shape)

# Example usage:
shape = tuple(int(x) for x in input("Enter the shape of the array (space-separated integers): ").split())
ones_array = create_ones_array(shape)
print("Array filled with ones of shape", shape, ":\n", ones_array)
```

12. . **Write a program to create a 2D numpy array initialized with random integers**.

sol;

```
import numpy as np

def create_random_int_array(rows, cols, min_val, max_val):
return np.random.randint(min_val, max_val+1, size=(rows, cols))

# Example usage:
rows = int(input("Enter the number of rows: "))
cols = int(input("Enter the number of columns: "))
min_val = int(input("Enter the minimum value for random integers: "))
max_val = int(input("Enter the maximum value for random integers: "))

random_array = create_random_int_array(rows, cols, min_val, max_val)
print("2D array initialized with random integers:\n", random_array)
```

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

Sol;

```
import numpy as np


def generate_linspace(start, stop, num):
 return np.linspace(start, stop, num)


# Example usage:
start = float(input("Enter the starting value of the range: "))
stop = float(input("Enter the ending value of the range: "))
num = int(input("Enter the number of evenly spaced samples to generate: "))
result_array = generate_linspace(start, stop, num)
print("Array of evenly spaced numbers over the range", start, "to", stop, ":\n", result_array)
```

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

Sol:

```
import numpy as np


# Using linspace to generate the array
result_array = np.linspace(1, 100, 10)


# Printing the generated array
print("Array of 10 equally spaced values between 1 and 100:")
print(result_array)
```

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

Sol;

```
import numpy as np
even_numbers_array = np.arange(2,21,2)
print(even_ numbers)
```

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

Sol;

```
import numpy as np

# Using arange to generate the array with step size of 0.5
result_array = np.arange(1, 10.5, 0.5)

# Printing the generated array
print("Array containing numbers from 1 to 10 with a step size of 0.5:")
print(result_array)

(2, 21, 2)
print("Array containing even numbers from 2 to 20:")
print(even_numbers_array)
```