# ASSESSMENT-1

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

A. Python program to calculate the area of a rectangle:

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

# Input length and width from the user
length = float(input("Enter the length of the rectangle: "))
width = float(input("Enter the width of the rectangle: "))

# Calculate and display the area
area = calculate_rectangle_area(length, width)
print(f"The area of the rectangle with length {length} and width {width} is: {area}")
```

Output:Enter the length of the rectangle: 9

Enter the width of the rectangle: 5

The area of the rectangle with length 9.0 and width 5.0 is: 45.0

2. Write a program to convert miles to kilometers.

A. Python program to convert miles to kilometers:

```
def miles_to_kilometers(miles):
kilometers = miles * 1.60934
return kilometers
# Input miles from the user
miles = float(input("Enter the distance in miles: “))
```

```
    # Convert miles to kilometers and display the result
    kilometers = miles_to_kilometers(miles)
    print(f"{miles} miles is equal to {kilometers:.2f} kilometers")
```

Output: Enter the distance in miles: 10

10.0 miles is equal to 16.09 kilometers

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

A. Python function to check if a given string is a palindrome:

```
    def is_palindrome(input_string):
    # Remove spaces and convert to lowercase for case-insensitive comparison
    cleaned_string = ''.join(input_string.split()).lower()

    # Compare the original string with its reverse
    return cleaned_string == cleaned_string[::-1]

   # Input a string from the user
    user_input = input("Enter a string: ")

   # Check if the input string is a palindrome and display the result
   if is_palindrome(user_input):
       print(f"{user_input} is a palindrome.")
  else:
       print(f"{user_input} is not a palindrome.")
```

Output: Enter a string: racecar

racecar is a palindrome.

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

A. Python program to find the second largest element in a list:

```
def second_largest_element(lst):
if len(lst) < 2:
    return "List must have at least two elements."

# Find the maximum element
max_element = max(lst)

# Remove the maximum element
lst.remove(max_element)

# Find the second maximum element in the modified list
second_largest = max(lst)

return second_largest

# Input a list of numbers from the user
user_input = input("Enter a list of numbers separated by spaces: ")
numbers = list(map(int, user_input.split()))

# Check if the list has at least two elements and find the second largest element
result = second_largest_element(numbers)

if isinstance(result, int):
    print(f"The second largest element in the list is: {result}")
```

```
else:
    print(result)
```

Output: Enter a list of numbers separated by spaces: 10 5 8 12 15

The second largest element in the list is: 12

5. Explain what indentation means in Python.

A. In Python, indentation is a critical aspect of the language's syntax and structure. Unlike many other programming languages that use braces `{}` or keywords like `begin` and `end` to define blocks of code, Python uses indentation to indicate the grouping and nesting of statements within blocks.

The Python interpreter determines the structure of code based on the indentation level. The common convention is to use four spaces for each level of indentation, but you can also use tabs or a different number of spaces as long as it is consistent throughout the code.

Here's an example to illustrate how indentation is used in Python:

```
if True:
  print("This is indented under the 'if' statement.")
  print("This is also part of the 'if' block.")

print("This is outside the 'if' block because it's not indented.")
```

In the example above:

- The two `print` statements are indented, indicating that they are part of the `if` block.
- The final `print` statement is not indented, indicating that it is outside the `if` block.

It's important to maintain consistent and proper indentation, as it directly affects the code's structure and logic. Incorrect indentation can lead to syntax errors or result

in code behaving differently than intended. The use of indentation in Python is not just a matter of style but a fundamental part of the language's syntax.

6. Write a program to perform set difference operation.

A. simple program to illustrate the set difference operation:

```
def set_difference_example(set1, set2):
# Using the "-" operator for set difference
difference1 = set1 - set2

# Using the difference() method
difference2 = set1.difference(set2)

return difference1, difference2

# Example sets
set_a = {1, 2, 3, 4, 5}
set_b = {3, 4, 5, 6, 7}

# Perform set difference
result1, result2 = set_difference_example(set_a, set_b)

# Display the results
print(f"Set difference using '-': {result1}")
print(f"Set difference using 'difference()': {result2}")
```

Output:Set difference using '-': {1, 2}

Set difference using 'difference()': {1, 2}

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

A. Python program that uses a `while` loop to print numbers from 1 to 10:

```
 # Initialize a variable to start from 1
number = 1

# Use a while loop to print numbers from 1 to 10
while number <= 10:
    print(number)
    number += 1  # Increment the number in each iteration
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

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

A. Python program that calculates the factorial of a number using a `while` loop:

```
    def calculate_factorial(n):
    if n < 0:
```

```
        return "Factorial is not defined for negative numbers."

    factorial_result = 1
    while n > 0:
        factorial_result *= n
        n -= 1

    return factorial_result

# Input a number from the user
user_input = int(input("Enter a number to calculate its factorial: "))

# Calculate and display the factorial
result = calculate_factorial(user_input)

if isinstance(result, int):
    print(f"The factorial of {user_input} is: {result}")
else:
    print(result)
```

Output:Enter a number to calculate its factorial: 5

The factorial of 5 is: 120

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

A. Python program that checks if a number is positive, negative, or zero using if-elif-else statements:

```
# Input a number from the user
user_input = float(input("Enter a number: "))

# Check if the number is positive, negative, or zero
if user_input > 0:
    print("The number is positive.")
elif user_input < 0:
    print("The number is negative.")
else:
    print("The number is zero.")
```

Output:Enter a number: -3.5

The number is negative.

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

A. Python program that determines the largest among three numbers using conditional statements:

```
# Input three numbers from the user
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))

# Determine the largest among three numbers
if num1 >= num2 and num1 >= num3:
    largest = num1
```

```
elif num2 >= num1 and num2 >= num3:
    largest = num2
else:
    largest = num3

# Display the result
print(f"The largest among {num1}, {num2}, and {num3} is: {largest}")
```

Output:Enter the first number: 7

Enter the second number: 12

Enter the third number: 5

The largest among 7.0, 12.0, and 5.0 is: 12.0

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

A. use the `numpy.ones()` function to create a NumPy array filled with ones of a given shape.

```
import numpy as np

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

# Input the shape from the user
rows = int(input("Enter the number of rows: "))
columns = int(input("Enter the number of columns: "))

# Create a NumPy array filled with ones of the given shape
```

```
ones_array = create_ones_array((rows, columns))


# Display the result
print(f"NumPy array filled with ones of shape ({rows}, {columns}):\n{ones_array}")
```

Output:Enter the number of rows: 3

Enter the number of columns: 4

NumPy array filled with ones of shape (3, 4):

[[1. 1. 1. 1.]

[1. 1. 1. 1.]

[1. 1. 1. 1.]]

12. Write a program to create a 2D numpy array initialized with random integers.
A. use the `numpy.random.randint()` function to create a 2D NumPy array initialized with random integers.

```
import numpy as np


def create_random_int_array(rows, columns, low, high):
    random_int_array = np.random.randint(low, high, size=(rows, columns))
    return random_int_array


# Input the shape and range of random integers from the user
rows = int(input("Enter the number of rows: "))
columns = int(input("Enter the number of columns: "))
low_limit = int(input("Enter the low limit for random integers: "))
high_limit = int(input("Enter the high limit for random integers: "))
```

```
# Create a 2D NumPy array initialized with random integers
random_int_array = create_random_int_array(rows, columns, low_limit, high_limit)

# Display the result
print(f"2D NumPy array initialized with random integers:\n{random_int_array}”)
```

Output:Enter the number of rows: 3

Enter the number of columns: 4

Enter the low limit for random integers: 1

Enter the high limit for random integers: 10

2D NumPy array initialized with random integers:

[[ 7  2  9  1]

[ 4  6 10  2]

[ 8  6  4 10]]

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

A. use the `numpy.linspace()` function to generate an array of evenly spaced numbers over a specified range. Here's an example program:

```
import numpy as np

# Input the start, end, and number of elements from the user
start = float(input("Enter the start of the range: "))
end = float(input("Enter the end of the range: "))
num_elements = int(input("Enter the number of elements: "))

# Generate an array of evenly spaced numbers using linspace
```

```
evenly_spaced_array = np.linspace(start, end, num_elements)


# Display the result

print(f"Array of evenly spaced numbers over the range ({start}, {end}) with {num_elements} elements:\n{evenly_spaced_array}")
```

Output:

```
Enter the start of the range: 1

Enter the end of the range: 5

Enter the number of elements: 10

Array of evenly spaced numbers over the range (1.0, 5.0) with 10 elements:

[1.  1.4 1.8 2.2 2.6 3.  3.4 3.8 4.2 4.6 5. ]
```

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

A. Python program that uses `numpy.linspace()` to generate an array of 10 equally spaced values between 1 and 100:

```
import numpy as np


# Generate an array of 10 equally spaced values between 1 and 100

equally_spaced_array = np.linspace(1, 100, 10)


# Display the result

print(f"Array of 10 equally spaced values between 1 and 100:\n{equally_spaced_array}")
```

Output:

Array of 10 equally spaced values between 1 and 100:

[  1.  12.  23.  34.  45.  56.  67.  78.  89. 100.]

15.  Write a Python program to create an array containing even numbers from 2 to 20 using arange.
A.  use numpy.arange() to create an array containing even numbers from 2 to 20. Here's an example program:

```
import numpy as np

# Create an array containing even numbers from 2 to 20
even_numbers_array = np.arange(2, 21, 2)

# Display the result
print(f"Array containing even numbers from 2 to 20:\n{even_numbers_array}")
```

Output:

Array containing even numbers from 2 to 20:

[ 2  4  6  8 10 12 14 16 18 20]

16.  Write a program to create an array containing numbers from 1 to 10 with a step size of 0.5 using arange.
A.  use numpy.arange() to create an array containing numbers from 1 to 10 with a step size of 0.5. Here's an example program:

```
import numpy as np

# Create an array containing numbers from 1 to 10 with a step size of 0.5
```

```
array_with_step = np.arange(1, 10.5, 0.5)


# Display the result

print(f"Array containing numbers from 1 to 10 with a step size of
0.5:\n{array_with_step}")
```

Output:

```
Array containing numbers from 1 to 10 with a step size of 0.5:

[ 1.   1.5  2.   2.5  3.   3.5  4.   4.5  5.   5.5  6.   6.5  7.   7.5  8.   8.5  9.   9.5 10. ]
```