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

Double-click (or enter) to edit

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

# Taking user input for length and width          (parameter) width: Any
length = float(input("Enter the length of the rectangle: "))
width = float(input("Enter the width of the rectangle: "))

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

```
Enter the length of the rectangle: 12
Enter the width of the rectangle: 16
The area of the rectangle with length 12.0 and width 16.0 is: 192.0
```

2.Write a program to convert miles to kilometers.

```
def miles_to_kilometers(miles):
    # Conversion factor: 1 mile = 1.60934 kilometers
    kilometers = miles * 1.60934
    return kilometers

# Taking user input for miles
miles = float(input("Enter distance in miles: "))

# Converting miles to kilometers and printing the result
kilometers = miles_to_kilometers(miles)
print(f"{miles} miles is equal to {kilometers} kilometers")
```

```
Enter distance in miles: 300
300.0 miles is equal to 482.802 kilometers
```

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

```
def is_palindrome(s):
    # Removing spaces and converting to lowercase for case-insensitive comparison
    s = s.replace(" ", "").lower()

    # Comparing the original string with its reverse
    return s == s[::-1]

# Taking user input for a string
input_string = input("Enter a string: ")

# Checking if the string is a palindrome and printing the result
if is_palindrome(input_string):
    print(f"{input_string} is a palindrome.")
else:
    print(f"{input_string} is not a palindrome.")
```

```
Enter a string: sai
sai is not a palindrome.
```

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

```
def find_second_largest(numbers):
    if len(numbers) < 2:
        return "List should have at least two elements."

    # Sorting the list in descending order
    sorted_numbers = sorted(numbers, reverse=True)

    # The second largest element is at index 1 in the sorted list
    second_largest = sorted_numbers[1]
    return second_largest

# Taking user input for a list of numbers
                                                   (parameter) width: Any
input_list = [float(x) for x in input("Enter a list of numbers separated by space: ").split()]

# Finding and printing the second largest element
result = find_second_largest(input_list)
print(f"The second largest element in the list is: {result}")
```

```
Enter a list of numbers separated by space: 34
The second largest element in the list is: List should have at least two elements.
```

5.Explain what indentation means in Python

In Python, indentation is a critical aspect of the language's syntax and structure. Unlike many other programming languages that use braces {} or other symbols to define blocks of code, Python uses indentation to indicate the grouping of statements. The level of indentation is used to determine which statements are part of a particular block of code, such as those within a function, loop, or conditional statement.

```
#example code for indentation
def example_function():
    print("This is inside the function")
```

6.Write a program to perform set difference operation.

```
def set_difference_example():
    # Creating two sets
    set1 = {1, 2, 3, 4, 5}
    set2 = {4, 5, 6, 7, 8}

    # Using the - operator for set difference
    difference_operator = set1 - set2

    # Using the difference() method for set difference
    difference_method = set1.difference(set2)

    # Printing the results
    print(f"Set difference using - operator: {difference_operator}")
    print(f"Set difference using difference() method: {difference_method}")

# Calling the function to demonstrate set difference
set_difference_example()
```

```
Set difference using - operator: {1, 2, 3}
Set difference using difference() method: {1, 2, 3}
```

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

```
# Initialize a counter
counter = 1

# Use a while loop to print numbers from 1 to 10
while counter <= 10:
    print(counter)
    counter += 1
```

```
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.

```
def calculate_factorial(n):
    # Initialize the result to 1
    result = 1

    # Use a while loop to calculate the factorial(parameter) width: Any
    while n > 1:
        result *= n
        n -= 1

    return result

# Taking user input for the number
number = int(input("Enter a number to calculate its factorial: "))

# Checking if the number is non-negative
if number < 0:
    print("Factorial is not defined for negative numbers.")
else:
    # Calculating and printing the factorial
    factorial_result = calculate_factorial(number)
    print(f"The factorial of {number} is: {factorial_result}")
```

```
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

```
# Taking user input for the number
number = float(input("Enter a number: "))

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

```
Enter a number: 34
The entered number is positive.
```

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

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

# Using if-elif-else statements to determine the largest number
if num1 >= num2 and num1 >= num3:
    largest = num1
elif num2 >= num1 and num2 >= num3:
    largest = num2
else:
    largest = num3

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

```
Enter the first number: 45
Enter the second number: 56
Enter the third number: 67
The largest number among 45.0, 56.0, and 67.0 is: 67.0
```

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

```
import numpy as np

def create_ones_array(shape):
    # Create a NumPy array filled with ones of the specified shape
    ones_array = np.ones(shape)
    return ones_array

# Taking user input for the shape of the array
rows = int(input("Enter the number of rows: "))
columns = int(input("Enter the number of columns: "))

# Creating and printing the ones array
shape = (rows, columns)                              (parameter) width: Any
ones_array = create_ones_array(shape)
print(f"Ones array of shape {shape}:\n{ones_array}")
```

```
Enter the number of rows: 3
Enter the number of columns: 4
Ones array 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.

```
import numpy as np

def create_random_array(rows, columns, low=0, high=10):
    # Create a 2D NumPy array initialized with random integers
    random_array = np.random.randint(low, high, size=(rows, columns))
    return random_array

# Taking user input for the shape of the array
rows = int(input("Enter the number of rows: "))
columns = int(input("Enter the number of columns:"))

# Creating and printing the random array
random_array = create_random_array(rows, columns)
print(f"2D NumPy array initialized with random integers:\n{random_array}")
```

```
Enter the number of rows: 4
Enter the number of columns:4
2D NumPy array initialized with random integers:
[[1 9 8 9]
 [5 5 3 5]
 [8 4 0 9]
 [4 0 5 4]]
```

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

```
import numpy as np

def generate_linspace(start, end, num_points):
    # Generate an array of evenly spaced numbers using linspace
    linspace_array = np.linspace(start, end, num_points)
    return linspace_array

# Taking user input for the range and number of points
start_value = float(input("Enter the start value: "))
end_value = float(input("Enter the end value: "))
num_points = int(input("Enter the number of points: "))

# Generating and printing the linspace array
linspace_array = generate_linspace(start_value, end_value, num_points)
print(f"Array of {num_points} evenly spaced numbers from {start_value} to {end_value}:\n{linspace_array}")
```

```
Enter the start value: 3
Enter the end value: 4
Enter the number of points: 5
Array of 5 evenly spaced numbers from 3.0 to 4.0:
[3.   3.25 3.5  3.75 4.  ]
```

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

```
import numpy as np

# Generate an array of 10 equally spaced values between 1 and 100 using linspace
linspace_array = np.linspace(1, 100, 10)

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

```
Array of 10 equally spaced values between 1 and 100:
[  1.  12.  23.  34.  45.  56.  67.  78.  89. 100.]
                                            (parameter) width: Any
```

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

```
import numpy as np

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

# Print the generated array
print("Array containing even numbers from 2 to 20:")
print(even_numbers_array)
```

```
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.

```
import numpy as np

# Create an array containing numbers from 1 to 10 with a step size of 0.5 using arange
numbers_array = np.arange(1, 10.5, 0.5)

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

```
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. ]
```