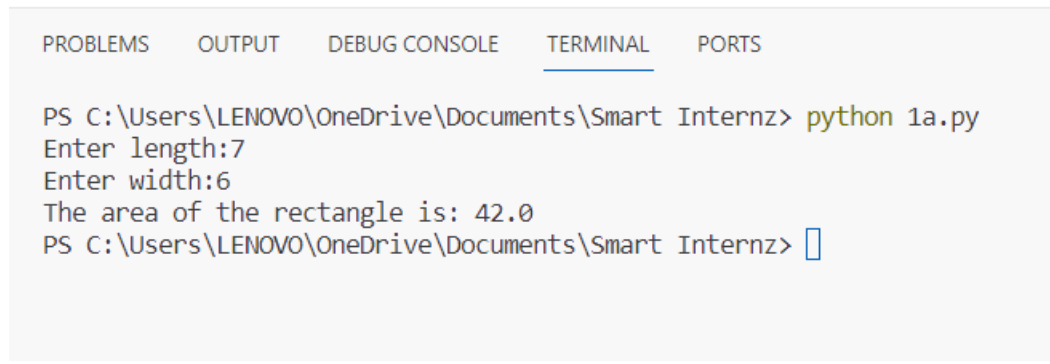


1. Write a Python program to calculate the area of a rectangle given its length and width.

```
def rect_area(l, w):  
    area = l * w  
    return area  
  
l = float(input("Enter length:"))  
w = float(input("Enter width:"))  
area = rect_area(l, w)  
  
print("Area of the rectangle:", area)
```

Output:



The screenshot shows a terminal window with a light gray background. At the top, there are five tabs: 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected and underlined), and 'PORTS'. The terminal content shows the command prompt 'PS C:\Users\LENOVO\OneDrive\Documents\Smart Internz>' followed by the command 'python 1a.py'. The program then prompts for 'Enter length:' with the input '7', and 'Enter width:' with the input '6'. The output is 'The area of the rectangle is: 42.0'. The prompt returns to 'PS C:\Users\LENOVO\OneDrive\Documents\Smart Internz>' with a blue cursor.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  
  
PS C:\Users\LENOVO\OneDrive\Documents\Smart Internz> python 1a.py  
Enter length:7  
Enter width:6  
The area of the rectangle is: 42.0  
PS C:\Users\LENOVO\OneDrive\Documents\Smart Internz> 
```

2. Write a program to convert miles to kilometres.

```
def miles_to_kilometers(miles):  
    kilometers = miles * 1.60934  
    return kilometers  
  
def main():  
    miles = float(input("Distance in miles: "))  
    kilometers = miles_to_kilometers(miles)  
    print(f"{kilometers} kilometers.")  
  
if __name__ == "__main__":  
    main()
```

## Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\LENOVO\OneDrive\Documents\Smart Internz> python 1a.py
Distance in miles: 9
14.48406 kilometers.
PS C:\Users\LENOVO\OneDrive\Documents\Smart Internz> █
```

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

```
def palindrome(s):
    s = s.replace(" ", "").lower()
    return s == s[::-1]

def main():
    test_string = input("Enter string:")
    if palindrome(test_string):
        print("String is palindrome.")
    else:
        print("String is not a palindrome.")

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

## Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\LENOVO\OneDrive\Documents\Smart Internz> python 1a.py
Enter string:fbvcfjvkfdnvk
String is not a palindrome.
PS C:\Users\LENOVO\OneDrive\Documents\Smart Internz> python 1a.py
Enter string:helloolleh
String is palindrome.
PS C:\Users\LENOVO\OneDrive\Documents\Smart Internz> █
```

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

```
def second_large(nums):
    if len(nums) < 2:
        return "List should have at least 2 elements"

    largest = second_largest = float('-inf')
    for num in nums:
        if num > largest:
            second_largest = largest
            largest = num
        elif num > second_largest and num != largest:
            second_largest = num
    return second_largest

def main():
    nums = [int(x) for x in input("Enter elements in the
list:").split()]
    second_largest = second_large(nums)
    print(f"Second largest element in the list:
{second_largest}")

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

## Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\LENOVO\OneDrive\Documents\Smart Internz> python 1a.py
Enter elements in the list:8 65 42 36 71 2 9 45 32 6
Second largest element in the list: 65
PS C:\Users\LENOVO\OneDrive\Documents\Smart Internz> 
```

## 5. Explain what indentation means in Python.

- ✚ Indentation refers to the whitespace (spaces or tabs) that is used at the beginning of a line to define the grouping of statements. Indentation is not just for visual formatting; it is a fundamental part of Python's syntax and is used to indicate the structure and nesting of code blocks.

### Indentation Rules:

- Python uses four spaces as default indentation spaces. However, the number of spaces can be anything; it is up to the user. But a minimum of one space is needed to indent a statement.
- The first line of Python code cannot have an indentation.
- Indentation is mandatory in Python to define the blocks of statements.
- The number of spaces must be uniform in a block of code.
- It is preferred to use whitespaces instead of tabs to indent in Python. Also, either use whitespace or tabs to indent; intermixing of tabs and whitespaces in indentation can cause wrong indentation errors.

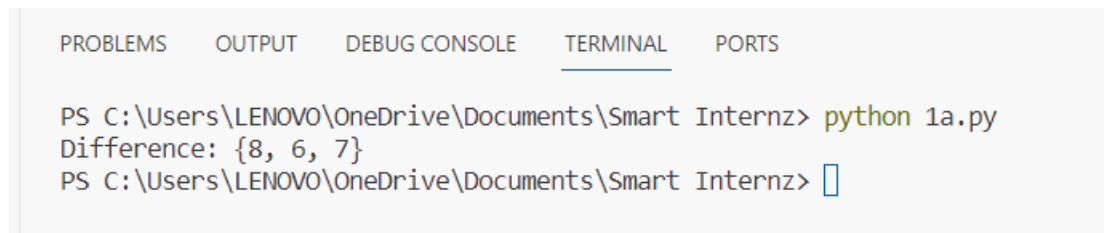
### Benefits of Indentation in Python:

- Indentation of code leads to better readability, although the primary reason for indentation in Python is to identify block structures.
- Missing, errors that sometimes pop up in C, and C++ languages can be avoided in Python and also the number of lines of code is reduced.

6. Write a program to perform a set difference operation.

```
def set_difference(set1, set2):  
    return set1.difference(set2)  
  
def main():  
    set1 = {4, 5, 6, 7, 8}  
    set2 = {1, 2, 3, 4, 5}  
    difference = set_difference(set1, set2)  
    print("Difference:", difference)  
  
if __name__ == "__main__":  
    main()
```

Output:

A screenshot of a code editor's terminal window. The terminal has tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected), and 'PORTS'. The terminal text shows a command prompt 'PS C:\Users\LENOVO\OneDrive\Documents\Smart Internz>' followed by the command 'python 1a.py'. The output of the program is 'Difference: {8, 6, 7}'. The prompt is then shown again with a cursor, 'PS C:\Users\LENOVO\OneDrive\Documents\Smart Internz> |'.

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

```
def print_numbers():  
    num = 1  
    while num <= 10:  
        print(num)  
        num += 1  
  
print_numbers()
```

## Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\LENOVO\OneDrive\Documents\Smart Internz> python 1a.py
1
2
3
4
5
6
7
8
9
10
PS C:\Users\LENOVO\OneDrive\Documents\Smart Internz> 
```

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

```
def factorial_num(n):
    factorial = 1
    while n > 1:
        factorial *= n
        n -= 1
    return factorial

def main():
    number = int(input("Enter num:"))
    if number < 0:
        print("Factorial not defined for -ve nums.")
    else:
        factorial = factorial_num(number)
        print(f"Factorial of {number} is: {factorial}")

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

## Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\LENOVO\OneDrive\Documents\Smart Internz> python 1a.py
Enter num:6
Factorial of 6 is: 720
PS C:\Users\LENOVO\OneDrive\Documents\Smart Internz> █
```

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

```
def check_num(n):
    if n > 0:
        print("Positive.")
    elif n < 0:
        print("Negative.")
    else:
        print("Zero.")

def main():
    number = float(input("Enter number: "))
    check_num(number)

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

## Output:

```
PS C:\Users\LENOVO\OneDrive\Documents\Smart Internz> python 1a.py
Enter number: 46
Positive.
PS C:\Users\LENOVO\OneDrive\Documents\Smart Internz> python 1a.py
Enter number: -8
Negative.
PS C:\Users\LENOVO\OneDrive\Documents\Smart Internz> python 1a.py
Enter number: 0
Zero.
PS C:\Users\LENOVO\OneDrive\Documents\Smart Internz> █
```

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

```
def largest_num(n1, n2, n3):  
    if n1 >= n2 and n1 >= n3:  
        largest = n1  
    elif n2 >= n1 and n2 >= n3:  
        largest = n2  
    else:  
        largest = n3  
    return largest  
  
def main():  
    n1 = float(input("Enter num1: "))  
    n2 = float(input("Enter num2: "))  
    n3 = float(input("Enter num3: "))  
    largest = largest_num(n1, n2, n3)  
    print(f"Largest among {n1}, {n2}, {n3} is: {largest}")  
  
if __name__ == "__main__":  
    main()
```

Output:

```
PS C:\Users\LENOVO\OneDrive\Documents\Smart Internz> python 1a.py  
Enter num1: 6  
Enter num2: 3  
Enter num3: 9  
Largest among 6.0, 3.0, 9.0 is: 9.0  
PS C:\Users\LENOVO\OneDrive\Documents\Smart Internz> 
```

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



```

import numpy as np
def create_ones_array(shape):
    return np.ones(shape)

def main():
    shape = tuple(map(int, input("Enter the shape of the array:
").split()))
    ones_array = create_ones_array(shape)
    print("Array of ones:")
    print(ones_array)

if __name__ == "__main__":
    main()

```

### Output:

```

Enter the shape of the array: 5 8
Array of ones:
[[1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 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 array(rows, cols):
    return np.random.randint(low=0, high=100, size=(rows, cols))

def main():
    rows = int(input("Enter no.of rows:"))
    cols = int(input("Enter no.of columns:"))
    random_array = array(rows, cols)
    print("Random 2D array:")
    print(random_array)

if __name__ == "__main__":
    main()

```

## Output:

```
Enter no.of rows:3
Enter no.of columns:4
Random 2D array:
[[65 71 75 85]
 [85 33  9 11]
 [ 5 26 82 42]]
```

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

```
import numpy as np
def linspace(start, stop, num):
    return np.linspace(start, stop, num)

def main():
    start = float(input("Enter start value:"))
    stop = float(input("Enter stop value:"))
    num = int(input("Enter no.of values:"))

    linspace_array = linspace(start, stop, num)
    print("Final Array:")
    print(linspace_array)

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

## Output:

```
Enter start value:5
Enter stop value:6
Enter no.of values:10
Final Array:
[5.          5.11111111 5.22222222 5.33333333 5.44444444 5.55555556
 5.66666667 5.77777778 5.88888889 6.          ]
```

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

```
import numpy as np
def array():
    return np.linspace(1, 100, 10)

def main():
    final_array = array()
    print("Final Array:")
    print(final_array)

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

Output:

```
➞ Final Array:
[ 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.

```
import numpy as np
def array():
    return np.arange(2, 21, 2)

def main():
    even_array = array()
    print("Final Array:")
    print(even_array)

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

## Output:

```
➞ Final Array:  
[ 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  
def create_array():  
    return np.arange(1, 10.5, 0.5)  
  
def main():  
    n_array = create_array()  
    print("Final Array:")  
    print(n_array)  
  
if __name__ == "__main__":  
    main()
```

## Output:

```
➞ Final Array:  
[ 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. ]
```