**Python Loops**

* **Python Loops – For, While, Nested Loops With Examples**

**1. For Loop:**

The for loop in Python is used to iterate over a sequence (that can be a list, tuple, string, or other iterable objects).

The basic syntax is: for variable in sequence:

The range() function is often used to generate a sequence of numbers.

**2. While Loop:**

The while loop in Python is used to repeatedly execute a block of code as long as the given condition is true.

The basic syntax is: while condition:

Be cautious to ensure the condition becomes false at some point; otherwise, you might end up in an infinite loop.

**3. Nested Loops:**

A nested loop is a loop inside another loop.

They are used to iterate over elements in a multi-dimensional data structure like a matrix.

Each time the outer loop runs once, the inner loop runs completely.

**Examples:**

**For Loop:**

for i in range(5): print(i)

**While Loop:**

counter = 0 while counter < 5: print(counter) counter += 1

**Nested Loops:**

for i in range(3): for j in range(3): print(f"({i}, {j})", end=' ') print()

**Infinite Loops**:

Be cautious with while loops to prevent infinite loops.

Ensure the loop condition becomes false at some point.

* **Loop Control Statements:**

**break**: Terminates the loop.

continue: Skips the rest of the code inside the loop for the current iteration and moves to the next iteration.

**else with loops**: The else block is executed when the loop condition becomes false.

Understanding these concepts will help you effectively use loops in Python and design efficient and readable code.

### 

Find Word Count In A Text Using The for Loop

* The function word\_count takes a text as input.
* It initializes a variable count to keep track of the word count.
* The for loop iterates through each character in the text.
* If the character is a space (checked using isspace()), it increments the count.
* Finally, 1 is added to the count to account for the last word (since the last word might not be followed by a space).

## ***Nested Loop***

1. Nested for loop:

* A nested loop is a loop inside another loop. This construct is often used when you need to iterate over elements in a multi-dimensional data structure or when you need to perform a repetitive task within another repetitive task. Here's an example of a nested loop:
* runs from 1 to 5, representing the rows of the multiplication table.

Example:

# Nested for loops to create a multiplication table

for i in range(1, 6):

for j in range(1, 6):

print(i \* j, end='\t')

print()

* The inner loop runs from 1 to 5 for each iteration of the outer loop, representing the columns of the multiplication table.
* The print(i \* j, end='\t') statement prints the product of the current row and column with a tab space ('\t') as a separator.
* The print() statement is used to move to the next line after each row is printed.

1. **Nested while loop:**

Nesting while loops follows a similar concept to nesting for loops. You can have one while loop inside another to perform more complex iterations.

Example:

# Nested while loops to create a pattern

row = 1

while row <= 5:

col = 1

while col <= row:

print(col, end=' ')

col += 1

print() # Move to the next line after each row

row += 1

* The outer while loop controls the number of rows, iterating as long as row is less than or equal to 5.
* The inner while loop controls the columns within each row, iterating as long as col is less than or equal to row.
* The print(col, end=' ') statement prints the value of col with a space as a separator, and col is incremented in each iteration of the inner loop.
* The print() statement is used to move to the next line after each row is printed.

**Break statement:**

The break statement is used in Python to exit a loop prematurely. It is commonly used in both for and while loops when a certain condition is met, and you want to terminate the loop immediately. Here's a simple example using a for loop:

Example:

# Using break in a for loop

fruits = ["apple", "banana", "cherry", "date", "elderberry"]

for fruit in fruits:

print(fruit)

if fruit == "date":

break

* The for loop iterates over the elements in the fruits list.
* The print(fruit) statement prints each fruit.
* The if fruit == "date": condition checks if the current fruit is "date".
* If the condition is true, the break statement is executed, and the loop is terminated immediately.