**CONDITION AND BRANCHING**

If (condition):

Execution of code if the condition is true

elif (condition):

Execution of code if the condition is true

Else:

Execution of code if the condition is false

**LOOP**

colors = ["red", "orange", "yellow", "green", "blue", "indigo", "violet"]

for color in colors:

print(color)

for number in range(1, 11):

print(number)

**Range Function**

The range function in Python generates an ordered sequence that can be used in loops. It takes one or two arguments:

* If given one argument (e.g., range(11)), it generates a sequence starting from 0 up to (but not including) the given number.

1. for number in range(11):
2. print(number)

* If given two arguments (e.g., range(1, 11)), it generates a sequence starting from the first argument up to (but not including) the second argument.

1. for number in range(1, 11):
2. print(number)

**The Enumerated For Loop**

Have you ever needed to keep track of both the item and its position in a list? An enumerated for loop comes to your rescue. It's like having a personal assistant who not only hands you the item but also tells you where to find it.

Consider this example:

1. fruits = ["apple", "banana", "orange"]
2. for index, fruit in enumerate(fruits):
3. print(f"At position {index}, I found a {fruit}")

With this loop, you not only get the fruit but also its position in the list. It's as if you have a magical guide pointing out each fruit's location!

**While Loops**

While loops are like a sleepless night at a friend's sleepover. Imagine you and your friends keep telling ghost stories until someone decides it's time to sleep. As long as no one says, "Let's sleep" you keep telling stories.  
A while loop works similarly - it repeats a task as long as a certain condition is true. It's like saying, "Hey computer, keep doing this until I say stop!"

**Basic syntax of While Loop.**

1. while condition:
2. # Code to be executed while the condition is true
3. # Indentation is crucial to indicate the scope of the loop

For example, here's how you might use a while loop to count from 1 to 10:

1. count = 1
2. while count <= 10:
3. print(count)
4. count += 1

here's a breakdown of the above code.

1. There is a variable named **count** initialized with the value 1.
2. The while loop is used to repeatedly execute a block of code as long as a given condition is True. In this case, the condition is **count <= 10**, meaning the loop will continue as long as count is less than or equal to 10.
3. Inside the loop:
   * The **print(count)** statement outputs the current value of the count variable.
   * The **count += 1** statement increments the value of count by 1. This step ensures that the loop will eventually terminate when count becomes greater than 10.
4. The loop will continue executing as long as the condition count <= 10 is satisfied.
5. The loop will print the numbers 1 to 10 in consecutive order since the print statement is inside the loop block and executed during each iteration.
6. Once count reaches 11, the condition count <= 10 will evaluate to False, and the loop will terminate.
7. The output of the code will be the numbers 1 to 10, each printed on a separate line.

**The Loop Flow**

Both for and while loops have their special moves, but they follow a pattern:

* **Initialization:** You set up things like a starting point or conditions.
* **Condition:** You decide when the loop should keep going and when it should stop.
* **Execution:** You do the task inside the loop.
* **Update:** You make changes to your starting point or conditions to move forward.
* **Repeat:** The loop goes back to step 2 until the condition is no longer true.

**When to Use Each**

**For Loops:** Use for loops when you know the number of iterations in advance and want to process each element in a sequence. They are best suited for iterating over collections and sequences where the length is known.

**While Loops:** Use while loops when you need to perform a task repeatedly as long as a certain condition holds true. While loops are particularly useful for situations where the number of iterations is uncertain or where you're waiting for a specific condition to be met.