Key Concepts: Looping Constructs in Python

What is Looping?

- **Looping** refers to executing the same set of steps repeatedly.
- Python offers **two main looping constructs**:
  - **For loop**: Used to iterate over a sequence (e.g., strings, arrays).
  - **While loop**: Executes as long as a specified condition is true.

1. The For Loop

Basic Syntax:

```
for variable in sequence:
    # Code to execute
```

Example: Iterating Over a String

- Strings in Python are sequences, so you can loop through each character:

```
my_string = "looping"
for char in my_string:
    print(char)
```

Output:

```
l
o
o
p
i
n
g
```

Iterating Over a Range:

- Use the `range()` function to specify a sequence of numbers:

```
for i in range(10):
    print(f"Iteration {i}")
```

Output:

```
Iteration 0
Iteration 1
...
Iteration 9
```

Iterating Over an Array:

- Example with an array of desserts:

```
favorites = ["Cake", "Ice Cream", "Pie", "Brownie", "Pudding"]
for item in favorites:
    print(f"I like this dessert: {item}")
```

Output:

```
I like this dessert: Cake
I like this dessert: Ice Cream
...
```

2. The While Loop

Basic Syntax:

```
while condition:
    # Code to execute
```

Example: Iterating Over an Array with a Counter

- A `while` loop requires an explicit counter:

```
favorites = ["Cake", "Ice Cream", "Pie", "Brownie", "Pudding"]
count = 0
while count < len(favorites):
    print(f"I like this dessert: {favorites[count]}")
    count += 1
```

Output:

```
I like this dessert: Cake
I like this dessert: Ice Cream
...
```

Key Points:

- **Infinite Loop**:
  - If you forget to increment the counter, the loop will run indefinitely.
  - Always ensure the loop condition eventually becomes false.

3. Accessing Index in a For Loop

Using `enumerate()`:

- To access both the index and value in a `for` loop:

```
favorites = ["Cake", "Ice Cream", "Pie", "Brownie", "Pudding"]
for idx, item in enumerate(favorites):
    print(f"{idx}: {item}")
```

Output:

```
0: Cake
1: Ice Cream
...
```

Key Differences Between For and While Loops

Aspect	For Loop	While Loop
Use Case	When iterating over a sequence or range.	When repeating until a condition is false.
Index Management	Automatic.	Manual (with a counter).
Risk of Infinite Loop	None (unless explicitly written).	Exists if the condition is not updated.

Summary

- **For loops**: Ideal for iterating over sequences or ranges.
- **While loops**: Suitable for conditions where the number of iterations is unknown.
- Use **`enumerate()`** in `for` loops to access indices.

Congratulations! You've learned how to implement looping constructs in Python with both `for` and `while` loops.