Loop Control Statements in Python

Let’s loop you in on one of the coolest tricks in Python!

Recap: What Are Loops?

Loops repeat code to save time and avoid mistakes. They automate tasks with repeated actions.

Common loops: **for** and **while** loops help repeat instructions until a condition ends.

Why use loops?

Run tasks multiple times quickly

Types of loops

Repeat with **for** or **while** structures

Loops save effort

Avoid repetitive coding by automating actions

Types of Loops in Python

For Loop

Repeats over items in a list or range.

```
for i in range(3):
  print(i)
```

While Loop

Repeats as long as a condition is true.

```
count = 0
while count < 3:
  count += 1
print(count)
```

What Are Loop Control Statements?

Control Loop Flow

They adjust how loops run or stop.

Key Statements

Include **break**, **continue**, and **pass**.

Make loops smarter

Help avoid errors and manage tasks efficiently.

The ***break*** Statement

1

Purpose

Stops the loop immediately when needed.

2

Syntax

```
break
```

3

Example

```
for i in range(5):
    if i == 3:
        break
print(i)
```

The **continue** Statement

1

Purpose

Skips current loop step and moves to next.

2

Syntax

```
continue
```

3

Example

```
for i in range(5):
    if i == 2:
       continue
 print(i)
```

The *pass* Statement

1

Purpose

Does nothing, acts as a placeholder.

2

Syntax

```
pass
```

3

Example

```
for i in range(3):
    pass
```

In a Nutshell…

Loops help us repeat tasks efficiently.

Loop control statements like break, continue, and pass give us more control over how loops behave.

Use break to exit, continue to skip, and pass to do nothing (for now.)