Break pass and continue in python with easy example

Here are simple examples of `break`, `pass`, and `continue` in Python:

1. `break` (Stops the loop)

```python
for num in range(1, 6):
    if num == 3:
        break  # Stops the loop when num is 3
    print(num)
```

Output:

```
1
2
```

The loop stops when `num == 3`.

---

2. `continue` (Skips the current iteration)

```python
for num in range(1, 6):
    if num == 3:
        continue  # Skips 3 and moves to the next iteration
    print(num)
```

Output:

```
1
2
4
5
```

It skips printing `3`.

---

3. `pass` (Does nothing, used as a placeholder)

```python
for num in range(1, 6):
    if num == 3:
        pass  # Does nothing, just a placeholder
    print(num)
```

Output:

```
1
2
3
4
5
```

The `pass` statement doesn't affect the loop execution.

Would you like more detailed examples?

Explain me what is this and how it is used and why it Is used

Understanding `break`, `continue`, and `pass` in Python

These three keywords are control flow statements used inside loops (`for` and `while`). They help modify how a loop behaves.

---

1. `break` Statement

- **What it does?**
  The `break` statement **immediately stops the loop** and exits it.
- **Why is it used?**
  When we want to exit a loop as soon as a specific condition is met.
- **Example Usage:**

```python
for num in range(1, 6):
    if num == 3:
        break  # Loop stops completely when num is 3
    print(num)
print("Loop ended")
```

Output:

```vbnet
1
2
Loop ended
```

Explanation:
When `num == 3`, `break` stops the loop immediately. So `4` and `5` are never printed.

---

2. `continue` Statement

- **What it does?**
  The `continue` statement **skips the current iteration** and moves to the next one.
- **Why is it used?**
  When we want to ignore specific values without stopping the loop.
- **Example Usage:**

```python
for num in range(1, 6):
    if num == 3:
        continue  # Skips 3 and moves to the next iteration
    print(num)
```

Output:

```
1
2
4
5
```

Explanation:

When `num == 3`, `continue` skips that iteration. So `3` is not printed, but the loop continues.

---

3. `pass` Statement

- **What it does?**
  The `pass` statement **does nothing**. It is used as a placeholder.
- **Why is it used?**
  When we need to define a loop, function, or condition but don't want to write logic yet.
- **Example Usage:**

```python
for num in range(1, 6):
    if num == 3:
        pass  # Placeholder; loop continues normally
    print(num)
```

Output:

```
1
2
3
4
5
```

Explanation:

The `pass` statement does nothing, so the loop runs normally.

Comparison Table

Statement	Purpose	Effect
`break`	Stops the loop completely	Loop ends immediately
`continue`	Skips the current iteration	Moves to the next loop cycle
`pass`	Does nothing (placeholder)	Loop continues as normal

When to Use Each?

- Use `break` **when you need to exit a loop early** (e.g., searching for an item in a list and stopping once found).
- Use `continue` **when you want to skip a particular iteration** (e.g., filtering specific values).
- Use `pass` **as a placeholder for future code** (e.g., when defining a function or loop that you will implement later).

Would you like more examples or a practical use case?

Are they called as control statements

Yes, `break`, `continue`, and `pass` are called **control statements** in Python. They help in **controlling the flow of execution** inside loops and conditional statements.

Types of Control Statements in Python

Control statements are used to **change the normal execution flow** of loops and conditional structures.

1. `break` → **Terminates** the loop completely.
2. `continue` → **Skips** the current iteration and moves to the next one.
3. `pass` → **Does nothing** (used as a placeholder).

Are they called Loop Control Statements?

Yes, specifically, `break` and `continue` are called **Loop Control Statements** because they directly affect how loops work.

- `break` **exits the loop** entirely.
- `continue` **skips** an iteration but keeps the loop running.

However, `pass` is not a Loop Control Statement. It is a **null statement**, mainly used in **functions, classes, or loops as a placeholder**.

Other Control Statements in Python

Besides `break`, `continue`, and `pass`, Python has:

- **Conditional control statements** (`if`, `elif`, `else`) → Control execution based on conditions.
- **Looping control statements** (`for`, `while`) → Manage iteration.

Would you like more examples or a deeper explanation of control flow in Python?