Python Loops

Loops are the tools that let you run code **repeatedly**. Python provides two main types of loops:

1. **for loop** – iterate over a sequence (like a list or string)
2. **while loop** – repeat while a condition is True

1. for Loop

Used to iterate over sequences (like list, tuple, string, range, etc.)

Example 1: Loop through a list

```
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

Example 2: Use with range()

```
for i in range(1, 6):
    print(i)
```

Output:

```
1
2
3
4
5
```

2. while Loop

Repeats as long as a condition is True.

Example:

```
count = 1
while count <= 5:
    print(count)
    count += 1
```

Output:

```
1
2
3
4
5
```

Loop Control Statements

Python provides statements to control the flow of loops:

Keyword	**Use Case**
`break`	Exits the loop early
`continue`	Skips to next iteration
`pass`	Placeholder, does nothing

- **break :** Stop the loop immediately.

```
for i in range(1, 10):
    if i == 5:
        break
  print(i)
```

Output:
1
2
3
4

- **continue :** Skip the current iteration and continue with the next.

```
for i in range(1, 6):
    if i == 3:
        continue
    print(i)
```

Output:
1
2
4
5

- **else with loops :** Runs **after the loop ends** (if not stopped by break):

```
for i in range(3):
    print(i)
else:
    print("Loop finished")
```

- **pass :** The pass statement in Python is used as a **placeholder** — it does **nothing** when executed, but allows your code to run **without errors** where a statement is syntactically required.

When to Use pass

- When you're **planning code** but haven't written it yet.
- Inside empty classes, functions, loops, or conditionals.

Examples of pass

Example 1: In a function definition

```
def my_function():
    pass  # Code will be written later

print("Function is defined without errors.")
```

Example 2: In an if statement

```
x = 5
if x > 0:
    pass  # Will add logic later
else:
    print("x is not positive")
```

Example 3: In a loop

```
for i in range(5):
    if i == 3:
        pass  # Placeholder for future logic
    print(i)
```

Without pass — This Would Give an Error:

```
if True:
    # Empty block — causes error!
```

```
✅ Fix it using pass:
if True:
    pass  # Now it's valid
```