Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @dataengineeringtamil

🎬 Topic: Loops & Control Flow in Python

☑️ 1. `for` Loop – Looping through sequences

Used when you know **how many times** to loop (e.g., lists, strings, ranges)

```
for i in range(5):
    print(i)
```

🖨️ Output:

```
0
1
2
3
4
```

✅ Also works with lists:

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

☑️ 2. `while` Loop – Repeat while condition is True

Used when the **ending condition depends on logic**

```
i = 1
while i <= 5:
```

```
    print(i)
    i += 1
```

🖨️ Output:

```
1
2
3
4
5
```

☑️ 3. `break` – Stop the loop immediately

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

🛑 Loop stops when `i == 5`

☑️ 4. `continue` – Skip the current iteration

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

🚫 Skips printing 2

☑ 5. `pass` – Do nothing (placeholder)

```
for i in range(3):
    pass  # reserved for future code
```

🧠 Used when Python **requires a statement**, but you don't want to do anything (like in unfinished functions or loops)

🧪 Mini Real-Life Example:

💡 Login Attempt System

```
attempts = 0
while attempts < 3:
    password = input("Enter password: ")
    if password == "admin123":
        print("Login successful")
        break
    else:
        print("Wrong password")
        attempts += 1
```

🧠 Recap Table:

Keyword	Purpose
`for`	Loop over known items/range
`while`	Loop while condition is True

`break`	Exit the loop
`continue`	Skip current iteration and continue
`pass`	Placeholder that does nothing

✅ Difference Between `for` and `while` Loops

Feature	`for` Loop	`while` Loop
Usage	Used when **number of iterations is known**	Used when **condition-based repetition** is needed
Syntax	Loops over a **sequence** (like `range`, list)	Loops **while a condition is True**
Control Flow	Automatically stops after sequence ends	Needs a condition that eventually becomes False
Example Use Case	Looping over a list, range, or string	Waiting for a user to enter correct input

✅ Use Case of `continue`: Skip unwanted data

▧ Real-life Scenario:

You're processing a list of numbers, but you want to **skip the negative values**.

🥷 Python Example:

```
numbers = [10, -5, 20, -3, 15]

for num in numbers:
    if num < 0:
        continue  # skip the negative numbers
    print(f"Processing: {num}")
```

🖨️ Output:

```
Processing: 10
Processing: 20
Processing: 15
```

🤔 When to Use `continue`:

Scenario	Why use `continue`?
Skip processing invalid inputs	Don't exit loop, just skip that data
Skip weekends while looping over days	Only process weekdays
Filter values during iteration	Example: Skip even numbers, skip blanks

Here's how you can take a list of names and convert each one to **uppercase** using a `for` loop.

🥷 Python Code:

```
names = ["saravana", "gowtham", "nandini", "nila", "rahul"]

for name in names:
    print(name.upper())
```

🖨️ **Output:**

```
SARAVANA
GOWTHAM
NANDINI
NILA
RAHUL
```

🧠 **Explanation:**

- `for name in names:` → Loops through each name in the list
- `name.upper()` → Converts the name to uppercase
- `print()` → Displays the result

🎯 Real-Time `while` Loop Examples

🔸 1. ATM PIN Authentication System

📘 **Logic:**
Keep asking the user for a PIN until they enter the correct one.

```
correct_pin = "1234"
entered_pin = ""

while entered_pin != correct_pin:
    entered_pin = input("Enter your PIN: ")

print("Access Granted ☑")
```

✅ Keeps looping until user enters the correct PIN.

◇ 2. Countdown Timer

📘 **Logic:**
Countdown from 5 to 1 using a `while` loop.

```
count = 5

while count > 0:
    print(f"Countdown: {count}")
    count -= 1

print("Time's up! ⏰")
```

✅ Runs until count reaches 0.

◇ 3. Shopping Cart – Add Items Until 'done'

📘 **Logic:**
Keep asking for items until user types `"done"`.

```
items = []

while True:
    item = input("Add item (type 'done' to finish): ")
    if item.lower() == "done":
        break
    items.append(item)

print("Items in cart:", items)
```

✅ Realistic use of infinite loop with a `break`.

About the Author

Gowtham SB is a **Data Engineering expert, educator, and content creator** with a passion for **big data technologies, as well as cloud and Gen AI** . With years of experience in the field, he has worked extensively with **cloud platforms, distributed systems, and data pipelines**, helping professionals and aspiring engineers master the art of data engineering.

Beyond his technical expertise, Gowtham is a **renowned mentor and speaker**, sharing his insights through engaging content on **YouTube and LinkedIn**. He has built one of the **largest Tamil Data Engineering communities**, guiding thousands of learners to excel in their careers.

Through his deep industry knowledge and hands-on approach, Gowtham continues to **bridge the gap between learning and real-world implementation**, empowering individuals to build **scalable, high-performance data solutions**.

Socials

YouTube - https://www.youtube.com/@dataengineeringvideos

Instagram - https://instagram.com/dataengineeringtamil

Instagram - https://instagram.com/thedatatech.in

Connect for 1:1 - https://topmate.io/dataengineering/

LinkedIn - https://www.linkedin.com/in/sbgowtham/

Website - https://codewithgowtham.blogspot.com

GitHub - http://github.com/Gowthamdataengineer

Whats App - https://lnkd.in/g5JrHw8q

Gowtham SB
www.linkedin.com/in/sbgowtham/ Instagram - @dataengineeringtamil

📧 **Email** - atozknowledge.com@gmail.com

📱 **All My Socials** - https://lnkd.in/gf8k3aCH