Day20_Loops

June 9, 2025

Today I explored how to use **loops** in Python — both **while** and **for** loops. Loops are essential for writing efficient code that repeats actions without manual repetition. Instead of copying and pasting the same line many times, loops help automate tasks like printing, calculating, or processing data. I learned how to run code multiple times using while (based on conditions) and for (over sequences like ranges, lists, or strings).

I also explored loop controls like **break, continue, and pass**, which help manage how and when a loop runs or stops. With these tools, I can now make my code smarter, cleaner, and much more powerful!

Python Loops: While & For Loops

- Loops help us execute a block of code repeatedly without writing it manually.
- For example: printing "Data Science" 1000 times would be impossible manually!

Two types of loops in Python: 1. while loop (condition-based) 2. for loop (sequence-based)

- **while:** condition-based loop, good for unknown repetitions
- **for:** sequence-based loop, good for known number of iterations
- **break, continue, pass:** control loop flow

Loops make automation possible — from printing 1000 values to building tables, or even machine learning training cycles. Mastering them builds coding confidence!

[1]:
```
# We can print for limited time, but what for 10,000 times
# We can not copy past for 10,000 times then run
# It does not make any sence and not good way
print('Data Science')
print('Data Science')
print('Data Science')
print('Data Science')
print('Data Science')
# Insted use loop
```

```
Data Science
Data Science
Data Science
Data Science
Data Science
```

1 While Loop

[2]:
```
# Basic while loop example: Print "Data Science" 5 times

i = 1  # initialization
while i <= 5:  # condition
    print('Data Science')
    i = i + 1  # increment
```

```
Data Science
Data Science
Data Science
Data Science
Data Science
```

[3]:
```
# Countdown using while loop

i = 5
while i >= 1:
    print('Data Science')
    i = i - 1
```

```
Data Science
Data Science
Data Science
Data Science
Data Science
```

shows the loop count (i) for better understanding.

[4]:
```
# While loop with index tracking

i = 1
while i <= 5:
    print('Data Science', i)
    i += 1
```

```
Data Science 1
Data Science 2
Data Science 3
Data Science 4
Data Science 5
```

[5]:
```
# Reverse with index tracking

i = 5
while i >= 1:
    print('Data Science', i)
    i -= 1
```

```
Data Science 5
Data Science 4
Data Science 3
Data Science 2
Data Science 1
```

1.1 Nested While Loop

It always starts with the outer loop, completes the inner loop, then comes back to the next iteration of outer loop.

In nested loops, the outer loop (i) runs once, and for each value of i, the inner loop (j) runs fully.

[6]:
```python
# Simple nested while loop

i = 1
while i <= 5:
    print('Data Science')  # outer loop
    j = 1
    while j <= 4:
        print('tech')  # inner loop
        j += 1
    i += 1
    print()  # newline between blocks
```

```
Data Science
tech
tech
tech
tech

Data Science
tech
tech
tech
tech

Data Science
tech
tech
tech
tech

Data Science
tech
tech
tech
tech
```

```
Data Science
tech
tech
tech
tech
```

end=' ' → used in print() to stay on the same line (no newline).

[7]:
```python
# Nested while loop with end=" " to print in one line

i = 1
while i <= 5:
    print('Data Science', end=" ")  # stays on same line
    j = 1
    while j <= 4:
        print('tech', end=" ")
        j += 1
    i += 1
    print()  # move to next line after inner loop
```

```
Data Science tech tech tech tech
Data Science tech tech tech tech
Data Science tech tech tech tech
Data Science tech tech tech tech
Data Science tech tech tech tech
```

[8]:
```python
# Nested while with indices (i and j) for clarity

i = 1
while i <= 5:
    print('Data Science', i)
    j = 1
    while j <= 4:
        print('tech', j)
        j += 1
    i += 1
    print()
```

```
Data Science 1
tech 1
tech 2
tech 3
tech 4

Data Science 2
tech 1
tech 2
tech 3
```

```
tech 4

Data Science 3
tech 1
tech 2
tech 3
tech 4

Data Science 4
tech 1
tech 2
tech 3
tech 4

Data Science 5
tech 1
tech 2
tech 3
tech 4
```

[9]:
```python
# Multiplication example using nested while loop

i = 1
while i <= 2:
    j = 0
    while j <= 2:
        print(i * j, end=" ")
        j += 1
    print()
    i += 1
```

```
0 1 2
0 2 4
```

[10]:
```python
# Another multiplication table pattern

i = 1
while i <= 4:
    j = 0
    while j <= 3:
        print(i * j, end=" ")
        j += 1
    print()
    i += 1
```

```
0 1 2 3
0 2 4 6
0 3 6 9
```

```
0 4 8 12
```

2 For Loop

[11]:
```
# For loop works on sequence datatypes (like list, string, tuple, etc.)

name = 'nit'
name1 = [1, 3.5, 'hallo']

for i in name1:
    print(i)
```

```
1
3.5
hallo
```

[12]:
```
for i in name:
    print(i)
```

```
n
i
t
```

[13]:
```
# Using range() with for loop

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

```
0
1
2
3
4
```

[14]:
```
# Using range(start, stop, step)

for i in range(1, 10, 3):
    print(i)
```

```
1
4
7
```

[15]:
```
# Print 5's table

for i in range(5, 51, 5):
    print(i)
```

```
5
```

```
10
15
20
25
30
35
40
45
50
```

[16]:
```python
# Or Print numbers divisible by 5
for i in range(1,51):
    if i %5==0:
        print(i)
```

```
5
10
15
20
25
30
35
40
45
50
```

[17]:
```python
# Print numbers NOT divisible by 5

for i in range(1, 51):
    if i % 5 != 0:
        print(i)
```

```
1
2
3
4
6
7
8
9
11
12
13
14
16
17
18
19
21
```

22
23
24
26
27
28
29
31
32
33
34
36
37
38
39
41
42
43
44
46
47
48
49

3 Special Keywords in Loops

3.1 break – Stops the loop immediately

[18]:
```
for i in range(1, 11):
    if i == 6:
        break
    print(i)  # stops printing when i reaches 6
```

1
2
3
4
5

[19]:
```
for i in range(1, 11):
    if i == 8:
        break
    print(i)
```

1
2
3
4

```
5
6
7
```

3.2 continue – Skips the current iteration

[20]:
```python
for i in range(1, 11):
    if i == 8:
        continue
    print(i)  # 8 will be skipped
```

```
1
2
3
4
5
6
7
9
10
```

3.3 pass – Placeholder, does nothing

[21]:
```python
for i in range(1, 11):
```

```
  Cell In[21], line 1
    for i in range(1, 11):
                          ^
SyntaxError: incomplete input
```

[22]:
```python
for i in range(1, 11):
    pass  # no error, loop runs but nothing happens
```

4 Interesting and real-life inspired examples

4.1 While Loop – Drinking Water Reminder

[26]:
```python
# Drink water every 30 minutes until you've had 5 glasses

glass = 1

while glass <= 5:
    print(f"Glass {glass}: Drink water!")
    glass += 1
```

```
Glass 1: Drink water!
Glass 2: Drink water!
Glass 3: Drink water!
Glass 4: Drink water!
Glass 5: Drink water!
```

4.2 Nested While Loop – Birthday Party

[25]:
```python
# At a birthday party, each of 3 kids gets 2 balloons

kid = 1
while kid <= 3:
    print(f"Kid {kid} gets:", end=" ")

    balloon = 1
    while balloon <= 2:
        print("Ballon", end=" ")
        balloon += 1

    print()  # new line for next kid
    kid += 1
```

```
Kid 1 gets: Ballon Ballon
Kid 2 gets: Ballon Ballon
Kid 3 gets: Ballon Ballon
```

4.3 For Loop – Pizza Slice Tracker

[28]:
```python
# There are 8 slices of pizza. Show each person getting one.

for slice in range(1, 9):
    print(f"Slice {slice} given!")
```

```
Slice 1 given!
Slice 2 given!
Slice 3 given!
Slice 4 given!
Slice 5 given!
Slice 6 given!
Slice 7 given!
Slice 8 given!
```

4.4 Break Statement – ATM Cash Limit

[30]:
```python
# Stop giving  500 notes after reaching  2000 total

total = 0
```

```
for i in range(1, 10):
    if total == 2000:
        print(" ATM Limit Reached!")
        break
    print(f" 500 dispensed. Total:  {total + 500}")
    total += 500
```

```
500 dispensed. Total:  500
500 dispensed. Total:  1000
500 dispensed. Total:  1500
500 dispensed. Total:  2000
 ATM Limit Reached!
```

4.5 Continue Statement – Skipping Homework

[32]:
```
# Skip subject 3, rest do homework

for subject in range(1, 6):
    if subject == 3:
        print("  Skipped subject 3")
        continue
    print(f"Doing homework for subject {subject}")
```

```
Doing homework for subject 1
Doing homework for subject 2
  Skipped subject 3
Doing homework for subject 4
Doing homework for subject 5
```

4.6 Pass Statement – Reserved but not implemented

[33]:
```
# Reserved for feature in future

for i in range(1, 4):
    if i == 2:
        pass  # to be done later
    else:
        print(f"Processing item {i}")
```

```
Processing item 1
Processing item 3
```