Day 21 - Loops in Python

Loops are used in Python to **execute a block of code repeatedly**. Instead of writing the same code again and again, we use loops to run logic multiple times.

Types of Loops in Python

1. `while` **Loop**
   Repeats a block of code as long as a condition is `True`.
2. `for` **Loop**
   Used to iterate over sequences like lists, tuples, strings, or ranges.

In [1]:
```
print('data science')
print('data science')
print('data science')
print('data science')
print('data science')
```

```
data science
data science
data science
data science
data science
```

While Loop

A `while` loop runs **as long as the condition remains** `True`.

Useful when we don't know how many times we need to repeat something.

In [2]:
```
#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
```

In [3]:
```
i = 5          # initializing

while i>=1:    # condition
    print('data science')
    i = i - 1  # decrement
```

```
data science
data science
data science
data science
data science
```

In [5]:
```
# While loop with index tracking

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

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

In [6]:
```python
# Reverse with index tracking

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

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

Nested While Loop

You can place one `while` loop inside another — useful for patterns, grids, or matrix operations.

In [7]:
```python
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
```

In [8]:
```python
# Nested while loop with end=" " to print in one line

i = 1
```

```
while i <= 5:
    print('Data Science', end=" ")  # when we mention end then new line will not create
    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
```

In [9]:
```
# while loop using some numbers
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
```

In [10]:
```
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
```

For Loop

`for` loops are best used when you want to **iterate over a known sequence** such as a list, tuple, string, or range.

In [31]:
```
name = 'loop'

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

```
l
o
o
p
```

In [14]:
```
name1 = [1,3.5,'hello']

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

```
1
3.5
hello
```

In [15]:
```
# Using range()
```

```
range(5)
```

Out[15]: range(0, 5)

In [16]:
```
for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

In [17]:
```
# Using range(start, stop, step)

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

```
1
4
7
```

In [18]:
```
for i in range(2,5):
    print(i)
```

```
2
3
4
```

In [21]:
```
# print the 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
```

In [22]:
```
# Print numbers which are 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

Special Keywords in Loops

break – Stops the loop immediately

In [23]:
```
for i in range(1,11):
    print(i)
```

1
2
3
4
5
6
7
8
9
10

In [26]:
```
for i in range(1,11):
    if i == 5:
        break
    print(i)
```

1
2
3
4

In [27]:
```
for i in range(1,11):
    if i == 8:
        break
    print(i)
```

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

continue – Skips the current iteration

In [28]:
```
for i in range(1, 11):
    if i == 6:
        continue
    print(i)
```

```
1
2
3
4
5
7
8
9
10
```

pass – do nothing (placeholder)

In [29]:
```
for i in range(1,11):
```

```
  Cell In[29], line 1
    for i in range(1,11):
                         ^
_IncompleteInputError: incomplete input
```

In [30]:
```
for i in range(1,11):
    pass
```

More Examples

1. Print all even numbers from 1 to 20

In [32]:
```
for i in range(1,21):
    if i % 2 == 0:
        print(i, end = ' ')
```

```
2 4 6 8 10 12 14 16 18 20
```

2. Ask the user to enter a password until it's correct

In [36]:
```
password = ""
while password != "secret":
    password = input('Enter password: ')
    if password == 'secret':
        print('Access Granted')
    else:
        print('Wrong password. Try again.')
```

```
Wrong password. Try again.
Wrong password. Try again.
```

```
Access Granted
```

3. Count how many times a specific letter appears in a sentence

In [39]:
```
sentence = 'Data Science is amazing'
count = 0
for char in sentence:
    if char == 'a':
        count += 1
print('The letter a appears', count, 'times')
```

```
The letter a appears 4 times
```

4. Sum of first 10 natural numbers

In [44]:
```
total = 0
for i in range(1,11):
    total += i
print('Sum =', total)
```

```
Sum = 55
```

5. Simple multiplication table (1 to 5)

In [50]:
```
for i in range(1,6):
    print(f"Table of {i}:")
    for j in range(1,11):
        print(f"{i} x {j} = {i*j}")
    print('------------')
```

```
Table of 1:
1 x 1 = 1
1 x 2 = 2
1 x 3 = 3
1 x 4 = 4
1 x 5 = 5
1 x 6 = 6
1 x 7 = 7
1 x 8 = 8
1 x 9 = 9
1 x 10 = 10
------------
Table of 2:
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
------------
Table of 3:
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30
------------
Table of 4:
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40
------------
Table of 5:
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
------------
```