Day 8

For Loops in Python:

What is a for loop?

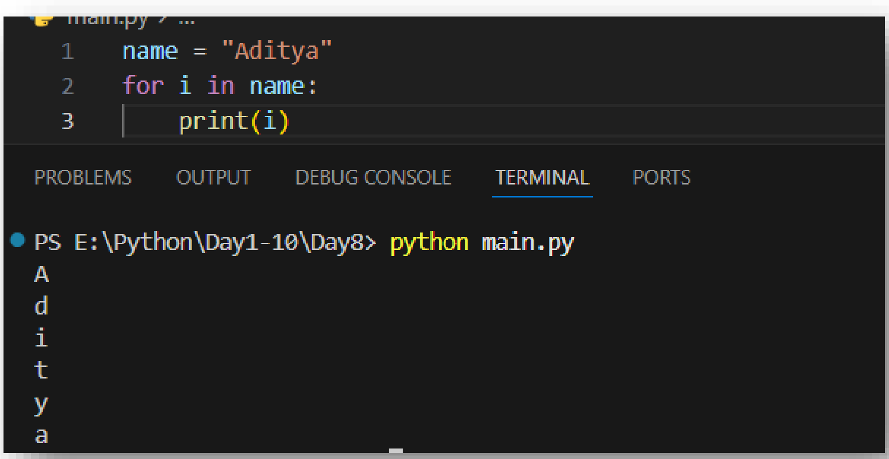
A for loop is used to repeat a block of code multiple times. It is mainly used to loop through sequences like:

- string
- list
- tuple
- range of numbers

Basic Syntax:

```
for variable in sequence:
    # code to repeat
```

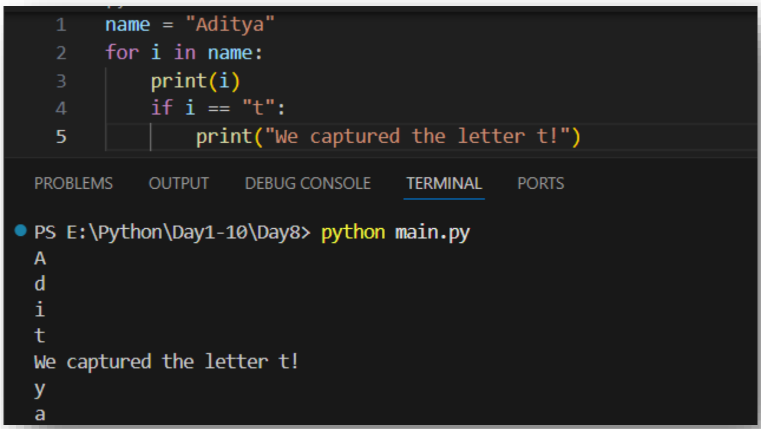
Example: a basic example of iterating the characters of the string using the for loops.



```python
name = "Aditya"
for i in name:
    print(i)
```

```
PS E:\Python\Day1-10\Day8> python main.py
A
d
i
t
y
a
```

Example: using the for loop and if statement together

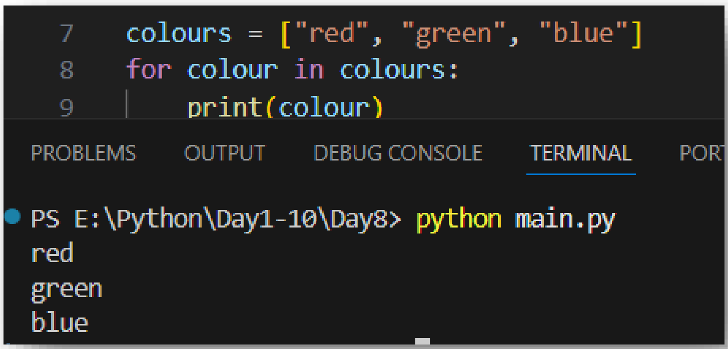


```python
name = "Aditya"
for i in name:
    print(i)
    if i == "t":
        print("We captured the letter t!")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS E:\Python\Day1-10\Day8> python main.py
A
d
i
t
We captured the letter t!
y
a
```

Example: iterating over a list

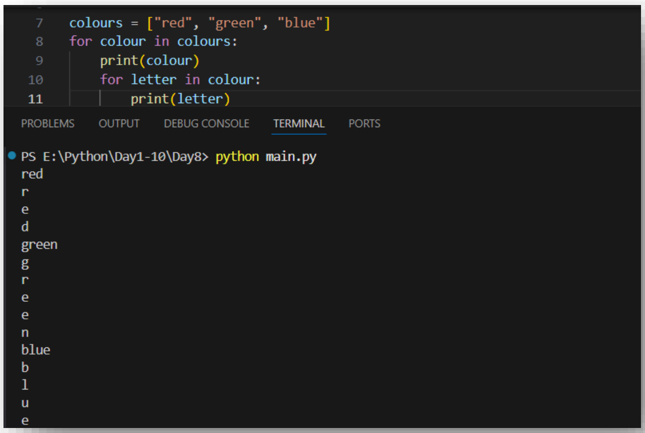


```python
colours = ["red", "green", "blue"]
for colour in colours:
    print(colour)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORT

```
PS E:\Python\Day1-10\Day8> python main.py
red
green
blue
```

Example: printing the characters of the colours.

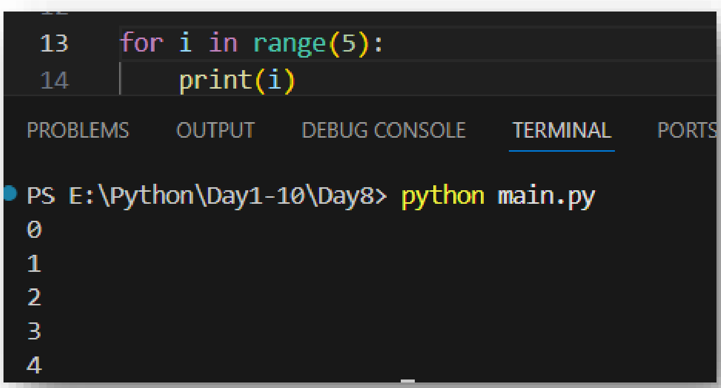


```python
colours = ["red", "green", "blue"]
for colour in colours:
    print(colour)
    for letter in colour:
        print(letter)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

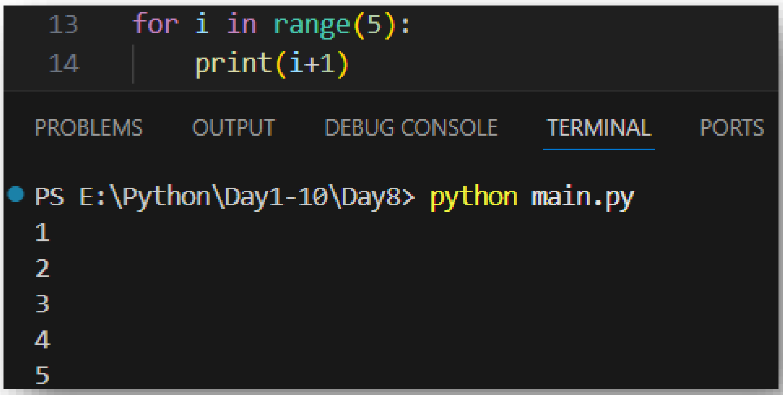
```
PS E:\Python\Day1-10\Day8> python main.py
red
r
e
d
green
g
r
e
e
n
blue
b
l
u
e
```

Example: basic use of range()


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

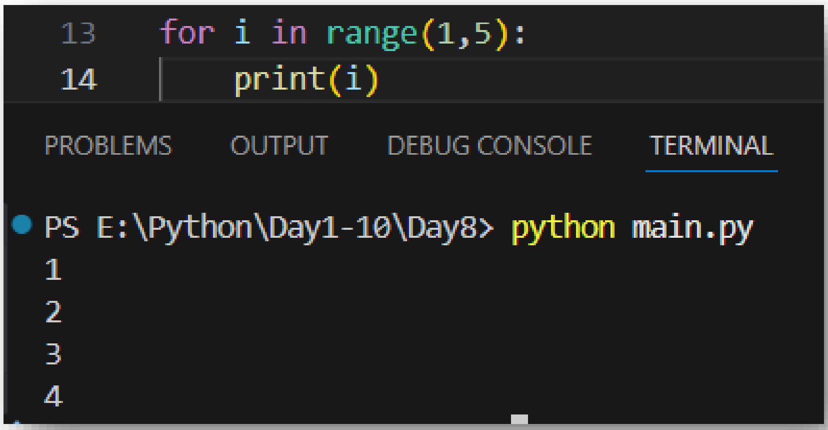
```
PS E:\Python\Day1-10\Day8> python main.py
0
1
2
3
4
```

Example: printing from 1 to 5 using the range()


```
for i in range(5):
    print(i+1)
```

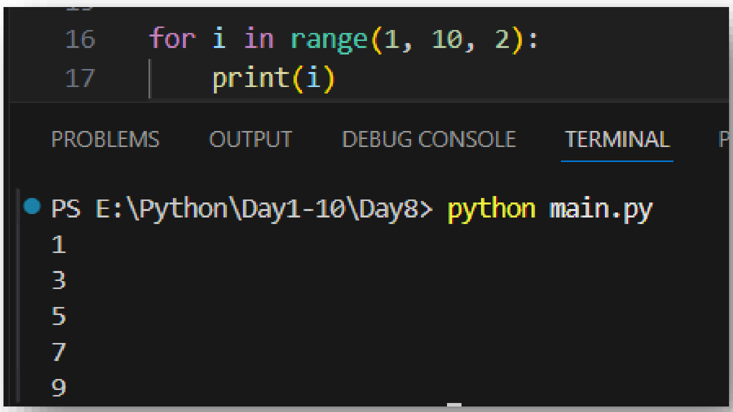
```
PS E:\Python\Day1-10\Day8> python main.py
1
2
3
4
5
```

Example: range(start, stop)


```
for i in range(1,5):
    print(i)
```

```
PS E:\Python\Day1-10\Day8> python main.py
1
2
3
4
```

Example: range(start, stop, step)

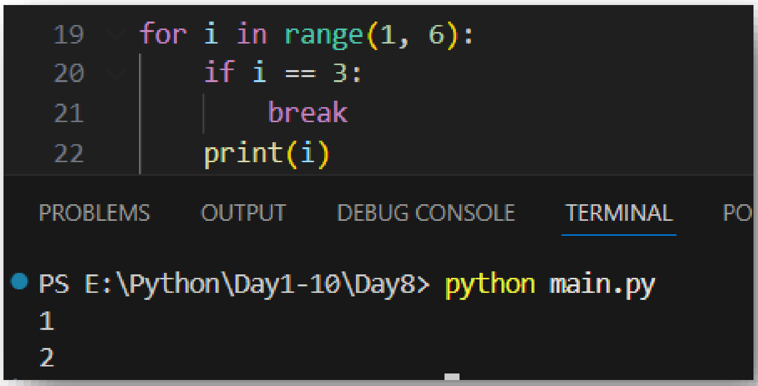


```python
for i in range(1, 10, 2):
    print(i)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS E:\Python\Day1-10\Day8> python main.py
1
3
5
7
9
```

Example: break statement-stops the loop

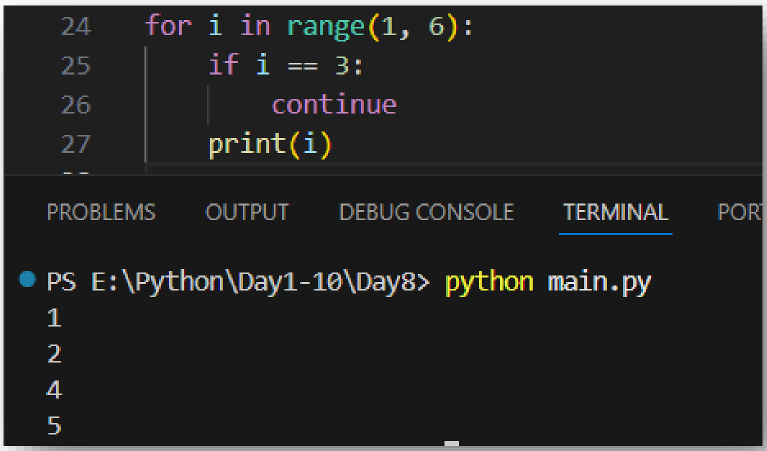


```python
for i in range(1, 6):
    if i == 3:
        break
    print(i)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS E:\Python\Day1-10\Day8> python main.py
1
2
```

Example: continue statement- skip current iteration

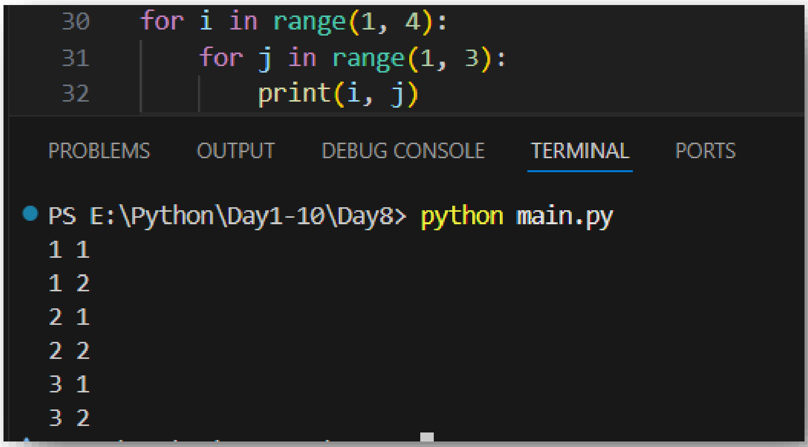


```python
for i in range(1, 6):
    if i == 3:
        continue
    print(i)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS E:\Python\Day1-10\Day8> python main.py
1
2
4
5
```

Example: nested for loop



```
for i in range(1, 4):
    for j in range(1, 3):
        print(i, j)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS E:\Python\Day1-10\Day8> python main.py
1 1
1 2
2 1
2 2
3 1
3 2
```

Summary:

- for loop repeats code multiple times
- Used with sequences and range()
- Syntax: for var in sequence:
- break stops loop
- continue skips iteration
- Can use else with loop
- Nested loops are allowed

While Loops in Python:

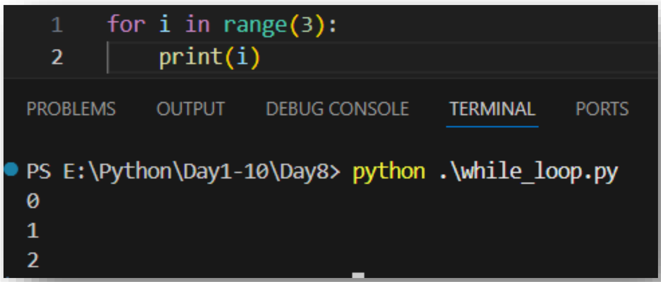
What is a while loop?

A while loop is used to repeat a block of code as long as a condition is True. When the condition becomes False, the loop stops.

Basic Syntax

```
while condition:
    # code to repeat
```

Example: printing using the for loops

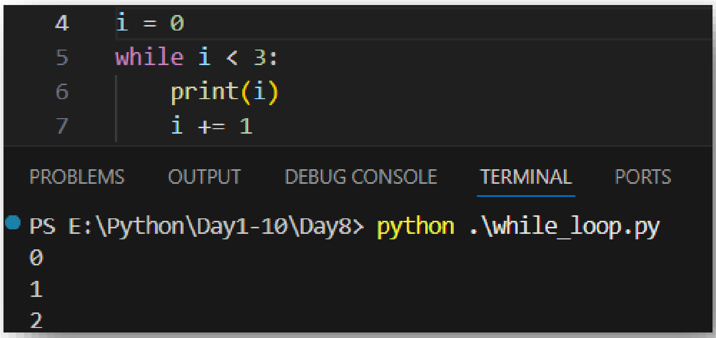


```
for i in range(3):
    print(i)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS E:\Python\Day1-10\Day8> python .\while_loop.py
0
1
2
```

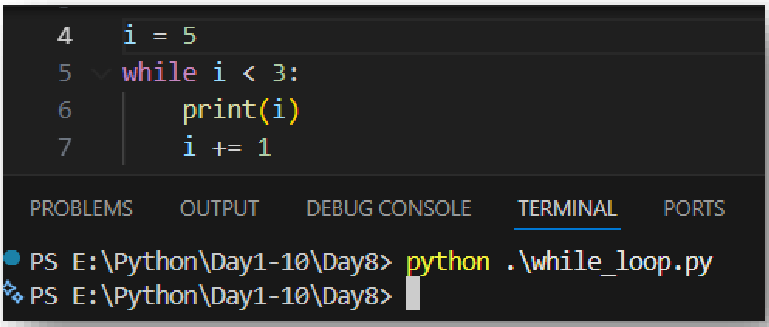
Example: basic code using the while loops



```python
i = 0
while i < 3:
    print(i)
    i += 1
```

```
PS E:\Python\Day1-10\Day8> python .\while_loop.py
0
1
2
```

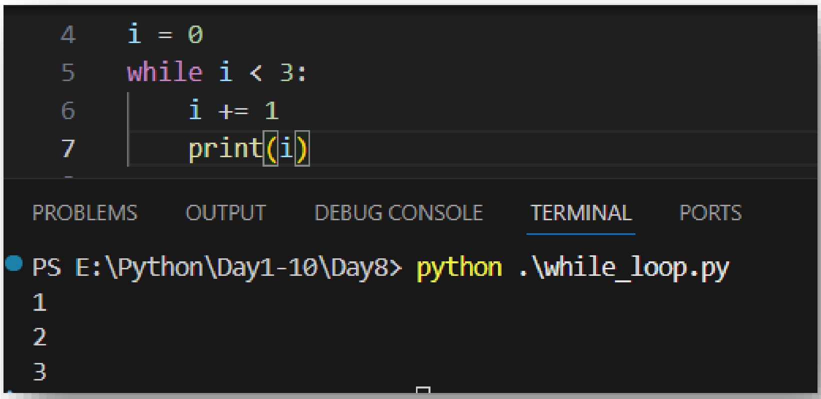
Example: condition never met in the below code and that’s why nothing gets printed



```python
i = 5
while i < 3:
    print(i)
    i += 1
```

```
PS E:\Python\Day1-10\Day8> python .\while_loop.py
PS E:\Python\Day1-10\Day8>
```

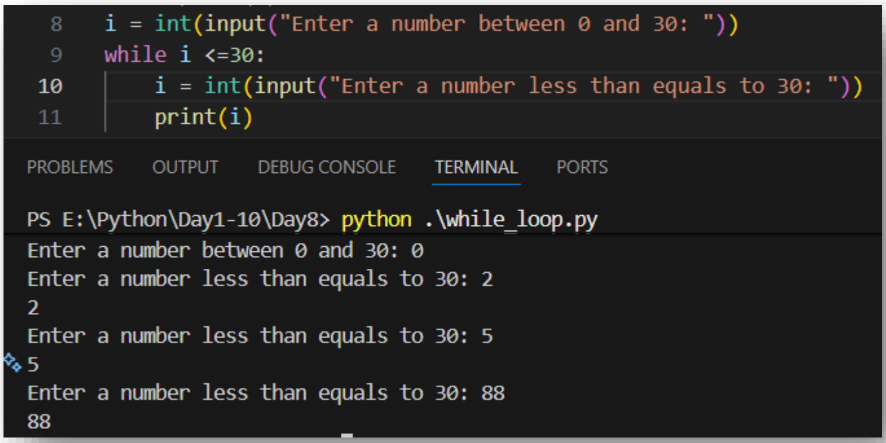
Example: using the increment line before the print()



```python
i = 0
while i < 3:
    i += 1
    print(i)
```

```
PS E:\Python\Day1-10\Day8> python .\while_loop.py
1
2
3
```

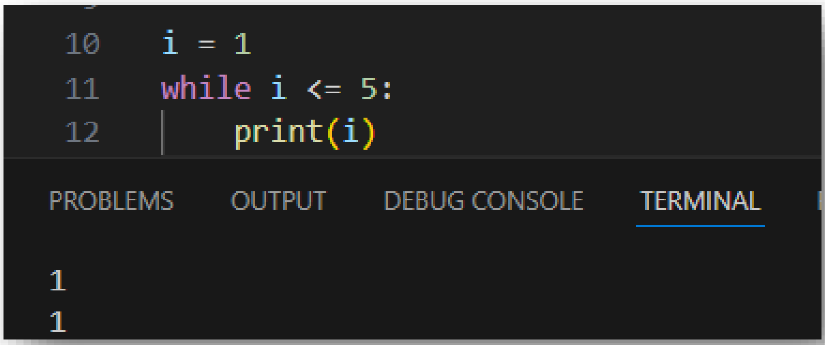
Example: taking the user input until the user enters less than 30.



```python
i = int(input("Enter a number between 0 and 30: "))
while i <=30:
    i = int(input("Enter a number less than equals to 30: "))
    print(i)
```

```
PS E:\Python\Day1-10\Day8> python .\while_loop.py
Enter a number between 0 and 30: 0
Enter a number less than equals to 30: 2
2
Enter a number less than equals to 30: 5
5
Enter a number less than equals to 30: 88
88
```

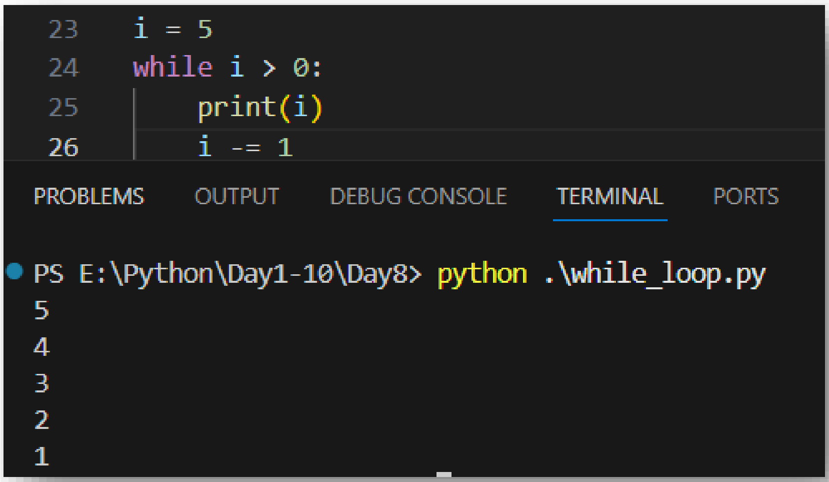
Example: infinite loop



```python
i = 1
while i <= 5:
    print(i)
```

```
1
1
```

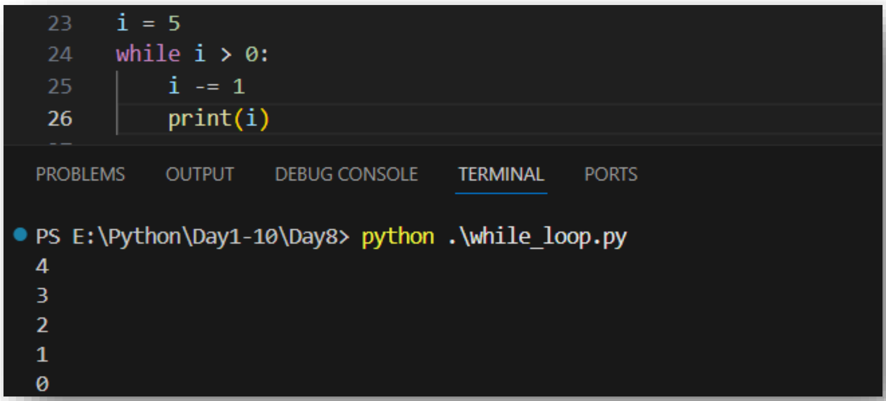
Example: iterating in the reverse order



```python
i = 5
while i > 0:
    print(i)
    i -= 1
```

```
PS E:\Python\Day1-10\Day8> python .\while_loop.py
5
4
3
2
1
```

Example:



```python
i = 5
while i > 0:
    i -= 1
    print(i)
```

```
PS E:\Python\Day1-10\Day8> python .\while_loop.py
4
3
2
1
0
```

Summary,

- while loop runs while condition is True
- Best when number of iterations is not known
- Always update loop variable
- break stops loop
- continue skips iteration
- else runs after loop ends normally

--The End--