

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.

The screenshot shows a terminal window with a dark background. At the top, there's a file navigation bar with 'main.py' and an ellipsis. Below it is a code editor pane containing the following Python script:

```
1 name = "Aditya"  
2 for i in name:  
3     print(i)
```

Below the code editor are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is currently selected. In the terminal pane, the command 'python main.py' is run, followed by the output of the for loop, which prints each character of the string 'Aditya' on a new line:

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

Example: using the for loop and if statement together

```
1 name = "Aditya"
2 for i in name:
3     print(i)
4     if i == "t":
5         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

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

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● PS E:\Python\Day1-10\Day8> **python main.py**

```
red
green
blue
```

Example: printing the characters of the colours.

```
7 colours = ["red", "green", "blue"]
8 for colour in colours:
9     print(colour)
10    for letter in colour:
11        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()

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

```
0
1
2
3
4
```

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

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

```
1
2
3
4
5
```

Example: range(start, stop)

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

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

```
1
2
3
4
```

Example: range(start, stop, step)

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

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PO
```

● PS E:\Python\Day1-10\Day8> python main.py

```
1
3
5
7
9
```

Example: break statement-stops the loop

```
19     for i in range(1, 6):
20         if i == 3:
21             break
22         print(i)

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PO
```

● PS E:\Python\Day1-10\Day8> python main.py

```
1
2
```

Example: continue statement- skip current iteration

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

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PO
```

● PS E:\Python\Day1-10\Day8> python main.py

```
1
2
4
5
```

Example: nested for loop

```
30     for i in range(1, 4):
31         for j in range(1, 3):
32             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

```
1   for i in range(3):
2       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

```
4 i = 0
5 while i < 3:
6     print(i)
7     i += 1
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● 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

```
4 i = 5
5 while i < 3:
6     print(i)
7     i += 1
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

Example: using the increment line before the print()

```
4 i = 0
5 while i < 3:
6     i += 1
7     print(i)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● 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.

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
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

```
10    i = 1
11    while i <= 5:
12        print(i)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
1
1
```

Example: iterating in the reverse order

```
23    i = 5
24    while i > 0:
25        print(i)
26        i -= 1
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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

Example:

The screenshot shows a code editor interface with a dark theme. At the top, there is a code block containing four lines of Python code:

```
23 i = 5
24 while i > 0:
25     i -= 1
26     print(i)
```

Below the code block are five tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is currently selected, indicated by an underline. In the terminal window, the command `python .\while_loop.py` is run, followed by the output of the loop iterations:

```
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--