

Nested Loops with Python Turtles

Textbook

Nested Loops with Python Turtles



Let's explore what happens when you put a loop inside of another loop! This is called [nested loops](#).

Outer Loop

Let's start by creating a regular loop. This loop will move the turtle across the screen 3 times as it loops 3 times. It will lift the pen each time it moves.

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(2):
5     turtle.penup()
6     turtle.forward(50)
```

```
7 | turtle.pendown()
```

This loop will become our outer loop.

Inner Loop

Now we will add an inner loop. This loop will run its complete cycle at the end of each single time the outer loop runs.

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(2):
5     turtle.penup()
6     turtle.forward(50)
7     turtle.pendown()
8     for second_counter in range(3):
9         turtle.forward(30)
10        turtle.left(120)
```

The outer loop:

```
for my_counter in range(2):
```

The inner loop:

```
for second_counter in range(3):
```

Explanation

Now, each time the outer loop runs, the turtle will move forward. Then, the inner loop will run 3 times, causing the turtle to make a triangle.

After the inner loop completely run the first time, the outer loop will kick in again, moving the turtle forward.

Now the inner loop is ready to run again.

Indentation

Notice how the entire inner loop is indented inside the outer loop. Careful indentation will keep your code clean and running correctly.

Thought Question: In the program above, how many times TOTAL does the inner loop run?

[Show answer/example](#)

In the program above, how many times does the outer loop run?

[Show answer/example](#)

Why Nested Loops?

Techniques like nesting loops can accomplish a lot of programming with only a few lines of code. Our examples here are simple, but imagine a program that will run thousands of times with hundreds of pages of data? These kinds of techniques help speed up and simplify processing data.

Think about how quickly videos stream from the internet? Or how fast you can share photos on a smart phone? Often, the faster a program can accomplish large tasks, the more useful it is.

Decomposing a Problem

When you face a really big or complicated task, whether it's building a complex Lego set, planning a group project, or writing a long computer program, it can feel overwhelming at first. This is where **decomposing a problem** comes in handy! Decomposing means breaking down a complex problem into smaller, more manageable parts.

Instead of trying to solve everything at once, you analyze the big problem, ask questions to understand each piece, and then separate it into individual, smaller problems or subtasks. By focusing on and solving each of these smaller parts one by one, the entire complex problem becomes much easier to tackle, helping you find a complete solution more effectively.

Your Turn: Breaking Down a Problem

Now that you understand how to decompose a problem, it's your turn to practice! Think about a big task you might face, either at school or at home. Maybe it's organizing a class event, building something complicated, or even planning a trip.

Your Challenge: Pick one of these complex tasks and actively work to **break it down into smaller, more manageable parts**. What are the individual steps or smaller problems you would need to solve to complete the main task? You might even want to write them down or draw a diagram. How does seeing the problem in smaller pieces make it feel less overwhelming?

Program Tracing: Predicting What Your Code Will Do

Imagine you're a detective, and your mission is to figure out exactly what a computer program will do, step by step, without actually running it. This skill is called **program tracing**. It helps you understand the logic of your code, find mistakes before they happen, and predict the output.

When you trace a program, you act like the computer, keeping track of every variable's value and every action the program takes. Let's try it with a Python program that creates a simple multiplication table.

```
1 print("---- Multiplication Table ----")
2
3 # The outer loop controls the first number (rows)
4 for i in range(1, 4):
5     # The inner loop controls the second number (columns)
6     for j in range(1, 3):
7         product = i * j
8         print(f"{i} * {j} = {product}")
9     print("---- Next Row ----") # This prints after each inner loop completes
10 print("---- End of Table ----")
11
```

Try it!

Checkpoint

Nested Loops with Python Turtles

Create a program that has a nested loop! The outer loop will move the turtle forward with the pen up. The inner loop will create a square.

1. Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
2. Create an outer for loop that will run twice. Use `my_counter`.

3. Inside the outer loop, lift the pen up, move the turtle forward 50, then put the pen down (in that order.)
4. Inside the outer loop, create another loop. This loop will be your inner loop. Use `second_counter` and repeat the inner loop 4 times.
5. Inside the inner loop, move the turtle forward 20 and rotate right 90 degrees.

Note: Be very careful about indentation!

Requirements:

- Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
- Create an outer for loop that will run twice. Use `my_counter` .
- Inside the outer loop, lift the pen up, move the turtle forward 50, then put the pen down (in that order.)
- Inside the outer loop, create another loop. This loop will be your inner loop. Use `second_counter` and repeat the inner loop 4 times.
- Inside the inner loop, move the turtle forward 20 and rotate right 90 degrees.

Questions (8)

1. What is the primary benefit of nesting loops?

MULTIPLE CHOICE

Choose the correct answer:

- A. They make code look better
- B. They make code shorter
- C. Slowing down data processing
- D. Eliminating the need for indentation

2. What is the significance of proper indentation in Python code?

MULTIPLE CHOICE

Choose the correct answer:

- A. Aesthetic purposes only
- B. It is optional and does not affect the code
- C. It influences the loop's speed
- D. It's essential for the code to run correctly

3. In this code example, how many times will the outer loop run?

```
import turtle
turtle.getscreen()
for my_counter in range(5):
    turtle.forward(50)
    for second_counter in range(2):
        turtle.forward(30)
        turtle.left(120)
```

Choose the correct answer:

- A. 5
- B. 50
- C. 2
- D. 30

4. In the code example, how many times will the inner loop run TOTAL?

```
import turtle
turtle.getscreen()
for my_counter in range(5):
    turtle.forward(50)
    for second_counter in range(2):
        turtle.forward(30)
        turtle.left(120)
```

Choose the correct answer:

- A. 5
- B. 50
- C. 2
- D. 10

5. Consider the following code example. What happens after the inner loop finishes running for the first time?

```
import turtle
turtle.getscreen()
for my_counter in range(5):
    turtle.forward(50)
    for second_counter in range(2):
        turtle.forward(30)
        turtle.left(120)
```

Choose the correct answer:

- A. The inner loop will run again
- B. The program will end
- C. The program will throw an error
- D. The outer loop will run again

6. Consider the following code example. What happens after the turtle moves forward 50 for the very first time?

MULTIPLE CHOICE

```
import turtle
turtle.getscreen()
for my_counter in range(5):
    turtle.forward(50)
    for second_counter in range(2):
        turtle.forward(30)
        turtle.left(120)
```

Choose the correct answer:

- A. The outer loop will run again
- B. The inner loop will start to run
- C. The code will throw an error
- D. The turtle will immediately turn left

7. Debug the following code:

DEBUG CODE

Code to Debug:

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(3):
5     turtle.forward(40)
6     for second_counter in range(4)
7         turtle.forward(20)
8         turtle.left(90)
```

8. Debug the following code:

DEBUG CODE

Code to Debug:

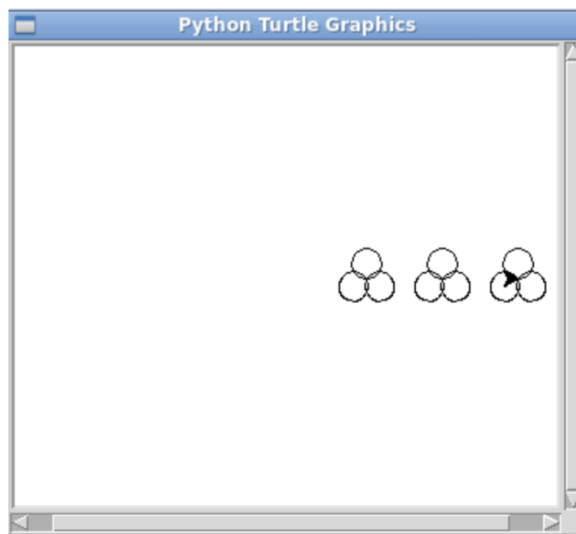
```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(3):
5     turtle.forward(40)
6     for second counter in range(4):
7         turtle.forward(20)
8         turtle.left(90)
```


Challenges (5)

1. 3 Sets of 3

Create a program with an outer loop that moves the turtle forward without drawing and an inner loop that will draw 3 circles.

1. Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
2. Create an outer for loop that will run three times. Use `my_counter` .
3. Inside the outer loop, lift the pen up, move the turtle forward 50, then put the pen down (in that order.)
4. Inside the outer loop, create another loop. This loop will be your inner loop. Use `second_counter` and repeat the inner loop 3 times.
5. Inside the inner loop, draw a circle with a radius of 10 and rotate right 120 degrees.



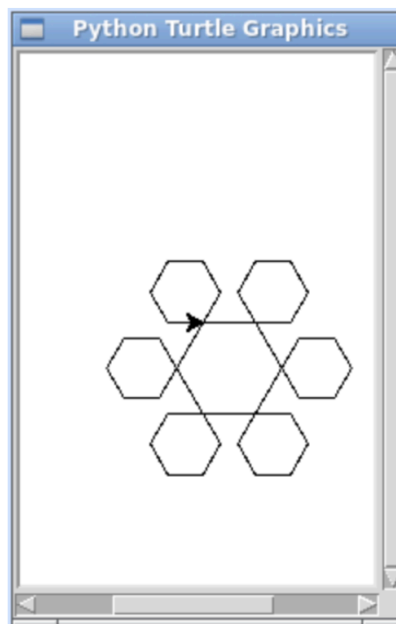
Requirements:

- Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
- Create an outer for loop that will run three times. Use `my_counter` .
- Inside the outer loop, lift the pen up, move the turtle forward 50, then put the pen down (in that order.)
- Inside the outer loop, create another loop. This loop will be your inner loop. Use `second_counter` and repeat the inner loop 3 times.
- Inside the inner loop, draw a circle with a radius of 10 and rotate right 120 degrees.

2. Snowflake

Draw a snowflake using nested loops! The outer loop will move the turtle forward, and the inner loop will draw a hexagon.

1. Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
2. Create an outer for loop that will run six times. Use `my_counter` .
3. Inside the outer loop move the turtle forward 50.
4. Inside the outer loop, create another loop. This loop will be your inner loop. Use `second_counter` and repeat the inner loop 5 times.
5. Inside the inner loop, rotate the turtle to the right 60, then move the turtle forward 20.



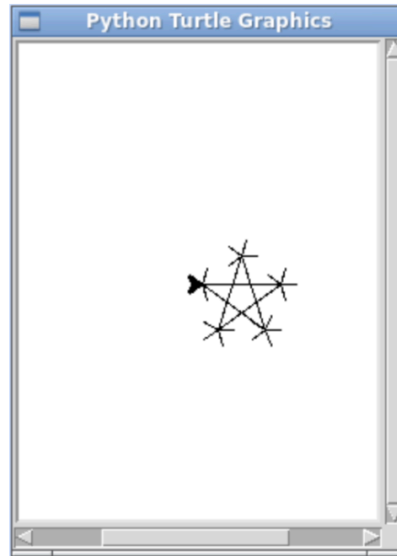
Requirements:

- Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
- Create an outer for loop that will run six times. Use `my_counter` .
- Inside the outer loop move the turtle forward 50.
- Inside the outer loop, create another loop. This loop will be your inner loop. Use `second_counter` and repeat the inner loop 5 times.
- Inside the inner loop, rotate the turtle to the right 60, then move the turtle forward 20.

3. Spiky Star

Create a star that has little stars on the end of it! Use a nested loop.

1. Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
2. Create an outer for loop that will run five times. Use `my_counter` .
3. Inside the outer loop, move the turtle forward 50.
4. Inside the outer loop, create another loop. This loop will be your inner loop. Use `second_counter` and repeat the inner loop 6 times.
5. Inside the inner loop, move the turtle forward 10, backward 10, then rotate right 144 degrees.



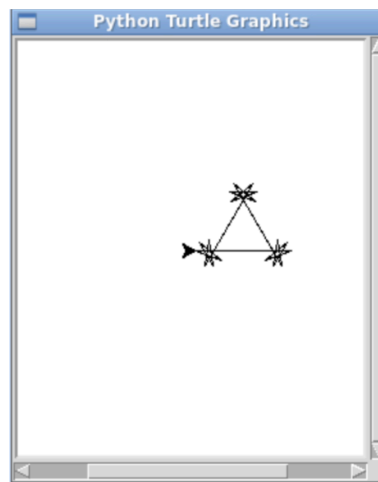
Requirements:

- Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
- Create an outer for loop that will run five times. Use `my_counter` .
- Inside the outer loop, move the turtle forward 50.
- Inside the outer loop, create another loop. This loop will be your inner loop. Use `second_counter` and repeat the inner loop 6 times.
- Inside the inner loop, move the turtle forward 10, backward 10, then rotate right 144 degrees.

4. Spiky Triangle

Draw a triangle with spiky stars on the corners of it. Use a nested loop to do this.

1. Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
2. Create an outer for loop that will run three times. Use `my_counter` .
3. Inside the outer loop, move the turtle forward 50.
4. Inside the outer loop, create another loop. This loop will be your inner loop. Use `second_counter` and repeat the inner loop 6 times.
5. Inside the inner loop, move the turtle forward 20, then rotate right 160 degrees.



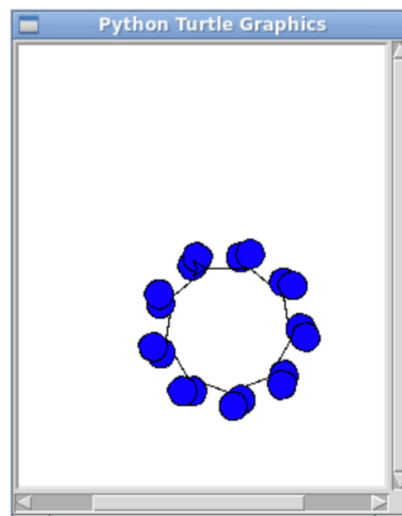
Requirements:

- Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
- Create an outer for loop that will run three times. Use `my_counter` .
- Inside the outer loop, move the turtle forward 50.
- Inside the outer loop, create another loop. This loop will be your inner loop. Use `second_counter` and repeat the inner loop 6 times.
- Inside the inner loop, move the turtle forward 20, then rotate right 160 degrees.

5. Anemone

Draw a kind of anemone with blue edges. Use nested loops.

1. Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
2. Create an outer for loop that will run nine times. Use `my_counter` .
3. Inside the outer loop, move the turtle forward 30, and rotate to the left 40 degrees.
4. Inside the outer loop, create another loop. This loop will be your inner loop. Use `second_counter` and repeat the inner loop 2 times.
5. Inside the inner loop, start to fill with `"blue"` using `begin_fill()` , `fillcolor()` and `end_fill()` .
6. Inside the inner loop, draw blue circles with a radius of 10 then rotate to the right 40 degrees.



Requirements:

- Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
- Create an outer for loop that will run nine times. Use `my_counter` .
- Inside the outer loop, move the turtle forward 30, and rotate to the left 40 degrees.
- Inside the outer loop, create another loop. This loop will be your inner loop. Use `second_counter` and repeat the inner loop 2 times.
- Inside the inner loop, start to fill with `"blue"` using `begin_fill()` , `fillcolor()` and `end_fill()` .
- Inside the inner loop, draw blue circles with a radius of 10 then rotate to the right 40 degrees.

Answer Keys & Solutions

Checkpoint Solutions

Nested Loops with Python Turtles

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(2):
5     turtle.penup()
6     turtle.forward(50)
7     turtle.pendown()
8     for second_counter in range(4):
9         turtle.forward(20)
10        turtle.right(90)
```

Questions

1. What is the primary benefit of nesting loops?

MULTIPLE CHOICE

Correct Answer:

- A. They make code look better ✗ Incorrect
- B. They make code shorter ✓ Correct
- C. Slowing down data processing ✗ Incorrect
- D. Eliminating the need for indentation ✗ Incorrect

Explanation:

Loops allow you to accomplish a lot of programming with just a few lines of code

2. What is the significance of proper indentation in Python code?

MULTIPLE CHOICE

Correct Answer:

- A. Aesthetic purposes only ✗ Incorrect
- B. It is optional and does not affect the code ✗ Incorrect
- C. It influences the loop's speed ✗ Incorrect

D. It's essential for the code to run correctly

✓ Correct

Explanation:

Indentation matters

3. In this code example, how many times will the outer loop run?

MULTIPLE CHOICE

Correct Answer:

A. 5

✓ Correct

B. 50

✗ Incorrect

C. 2

✗ Incorrect

D. 30

✗ Incorrect

Explanation:

The outer loop is the first loop

4. In the code example, how many times will the inner loop run TOTAL?

MULTIPLE CHOICE

Correct Answer:

A. 5

✗ Incorrect

B. 50

✗ Incorrect

C. 2

✗ Incorrect

D. 10

✓ Correct

Explanation:

It will run 2 times for every time the outer loop runs.

5. Consider the following code example. What happens after the inner loop finishes running for the first time?

MULTIPLE CHOICE

Correct Answer:

- A. The inner loop will run again ✓ Correct
- B. The program will end ✗ Incorrect
- C. The program will throw an error ✗ Incorrect
- D. The outer loop will run again ✗ Incorrect

Explanation:

Once the inner loop is finished, it will run again for the second time.

6. Consider the following code example. What happens after the turtle moves forward 50 for the very first time?

MULTIPLE CHOICE

Correct Answer:

- A. The outer loop will run again ✗ Incorrect
- B. The inner loop will start to run ✓ Correct
- C. The code will throw an error ✗ Incorrect
- D. The turtle will immediately turn left ✗ Incorrect

Explanation:

After the turtle moves forward, what is the very next line of code?

7. Debug the following code:

DEBUG CODE

Incorrect Code:

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(3):
5     turtle.forward(40)
6     for second_counter in range(4)
7         turtle.forward(20)
8         turtle.left(90)
```

Correct Solution:

```
1 import turtle
2 turtle.getscreen()
3
```

```
4 for my_counter in range(3):
5     turtle.forward(40)
6     for second_counter in range(4):
7         turtle.forward(20)
8         turtle.left(90)
```

Explanation:

This code is missing a colon

8. Debug the following code:

DEBUG CODE

Incorrect Code:

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(3):
5     turtle.forward(40)
6     for second counter in range(4):
7         turtle.forward(20)
8         turtle.left(90)
```

Correct Solution:

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(3):
5     turtle.forward(40)
6     for second_counter in range(4):
7         turtle.forward(20)
8         turtle.left(90)
```

Explanation:

This code is missing an underscore

Challenges

1. 3 Sets of 3

Solution:

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(3):
5     turtle.penup()
6     turtle.forward(50)
7     turtle.pendown()
8     for second_counter in range(3):
```



```
9 turtle.circle(10)
10 turtle.right(120)
```

2. Snowflake

Solution:

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(6):
5     turtle.forward(50)
6     for second_counter in range(5):
7         turtle.right(60)
8         turtle.forward(20)
```

3. Spiky Star

Solution:

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(5):
5     turtle.forward(50)
6     for second_counter in range(6):
7         turtle.forward(10)
8         turtle.backward(10)
9         turtle.right(144)
```

4. Spiky Triangle

Solution:

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(3):
5     turtle.forward(50)
6     for second_counter in range(6):
7         turtle.forward(20)
8         turtle.right(160)
```

5. Anemone

Solution:

```
1 import turtle
2 turtle.getscreen()
```

```
3
4 for my_counter in range(9):
5     turtle.forward(30)
6     turtle.left(40)
7     for second_counter in range(2):
8         turtle.begin_fill()
9         turtle.fillcolor("blue")
10        turtle.circle(10)
11        turtle.right(40)
12        turtle.end_fill()
```