

For Loops with Python Turtles

Textbook

For Loops with Python Turtles



Loops are chunks of code that repeat. Often, a loop will run until a certain condition is met.

Let's practice adding some loops to our turtle programs.

For Loop

Let's say we want to draw a square. You might have noticed that to do this, you need to move the turtle forward and then turn 90 degrees, repeated 4 times., that's a pattern.

```
1 import turtle
2 turtle.getscreen()
3
4 turtle.forward(50)
5 turtle.left(90)
6 turtle.forward(50)
7 turtle.left(90)
```

```
8 turtle.forward(50)
9 turtle.left(90)
10 turtle.forward(50)
11 turtle.left(90)
```

Instead of typing out all that code, let's use a [for loop](#). It looks like this:

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(4):
5     turtle.forward(100)
6     turtle.left(90)
```

Let's break down this code a bit.

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

This is the code to set up the for loop. The `my_counter` represents a kind of counter that starts at 0. Each time the loop runs, the counter increases by 1 until it reaches the value in the parentheses. In this case, the loop will run 4 times because there is a 4 inside the parentheses of `range()`.

Indentation

Notice how in the above example, the code you want to repeat is all indented one tab space. [Indentation](#) is **VERY** important in Python. Everything you want to repeat (in other words, everything inside the for loop) should be indented.

Try changing the number in the parentheses of the range! Try changing the angle your turtle turns to see what you can make.

Another Example

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(10):
5     turtle.forward(50)
6     turtle.left(40)
```

Patterns and Structure

Imagine you have a secret code to crack, or a complex puzzle to solve. Often, the key isn't brute force, but noticing what repeats, what fits together, and how the pieces are organized. In math and computing, this is called using patterns and structure to understand and connect concepts. It's like having a superpower that lets you see the hidden blueprint of a problem!

Your Challenge: The next time you face a new math problem or this coding task, don't just jump in. First, take a moment to focus on the relevant details. Can you spot any repeating elements? Does the problem look similar to something you've solved before? Then, try to create a plan or procedure to logically order the steps you'll need to take. If it's a big problem, remember to decompose it into manageable parts. Look for

similarities between what you're learning now and what you've already mastered. By recognizing these patterns and understanding the structure, you'll not only solve the current problem but also connect its solution to more complicated, large-scale situations, making you a much stronger problem-solver!

Assessing the Reasonableness of Solutions

When writing code with for loops—especially when using tools like Python Turtles—it's important to pause and ask: *Does this solution make sense?* Before running your program, think through what each part of the loop will do. How many times will the turtle move? Will it turn enough to complete the shape you're expecting? Predicting the outcome and checking if the logic is sound helps catch mistakes early and encourages clearer thinking.

For example, if you're using a for loop to draw a square and your turn angle is set to 100 degrees instead of 90, the shape won't close properly. By estimating and visualizing the result ahead of time, you can identify these kinds of errors and make corrections before running the code. This kind of reasoning isn't just helpful—it's a key habit of successful programmers.

Checkpoint

For Loops with Python Turtles

Practice creating a for loop in Python Turtles!

1. Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
2. Add a `for loop` with a `range` of your choice. Use `my_counter`.
3. Inside the for loop, include at least 2 commands for your turtle to repeat.

Requirements:

- Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
- Add a `for loop` with a `range` of your choice. Use `my_counter`.
- Inside the for loop, include at least 2 commands for your turtle to repeat.

Questions (10)

1. What is the purpose of a loop in programming?

MULTIPLE CHOICE

Choose the correct answer:

- A. To make code easier to read
- B. To repeat a set of instructions
- C. To speed up the program
- D. To slow down the program

2. In the given For Loop example, how many times will the turtle move forward and turn left?

MULTIPLE CHOICE

```
import turtle
turtle.getscreen()
for my_counter in range(4):
    turtle.forward(50)
    turtle.left(90)
```

Choose the correct answer:

- A. 4
- B. 5
- C. 8
- D. 2

3. Why is indentation important in Python, especially within a loop?

MULTIPLE CHOICE

Choose the correct answer:

- A. It makes the code look neat and organized
- B. It helps the program run faster
- C. It indicates what code is inside the For Loop
- D. It requires less code overall

4. How many times will the following code run? (Notice that the code is NOT indented)

MULTIPLE CHOICE

```
import turtle
turtle.getscreen()
for my_counter in range(3):
    turtle.forward(20)
    turtle.left(45)
```

Choose the correct answer:

- A. 1
- B. It will throw an error
- C. 2
- D. 3

5. What is the purpose of the "range()" function in the For Loop?

MULTIPLE CHOICE

Choose the correct answer:

- A. It defines the size of the turtle screen
- B. It sets the speed of the turtle
- C. It determines the number of iterations for the loop
- D. It specifies the turtle's color

6. What is the role of the variable "my_counter" in the For Loop?**Choose the correct answer:**

- A. It defines the number of sides in the square
- B. It controls the speed of the turtle
- C. It represents a value that increases by 1 each time the loop runs
- D. It specifies the color of the turtle

7. Debug the following code:

DEBUG CODE

Code to Debug:

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(4)
5     turtle.forward(100)
6     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(4):
5     turtle.forward(100)
6     turtle.left(90)
```

9. Debug the following code:

DEBUG CODE

Code to Debug:

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in count(8):
5     turtle.right(30)
6     turtle.forward(20)
```

10. Debug the following code:

[DEBUG CODE](#)

Code to Debug:

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(8):
5     turtleright(30)
6     turtle.forward(20)
```

Challenges (5)

1. Star Continued

Back in Python Turtles 1, we worked on a project where we drew a 5 point star. Here is the code from that project.

```
import turtle

turtle.getscreen()

turtle.forward(100)

turtle.left(144)

turtle.forward(100)

turtle.left(144)

turtle.forward(100)

turtle.left(144)

turtle.forward(100)

turtle.left(144)

turtle.forward(100)

turtle.left(144)
```

Take the code as an example and improve it by using a for loop instead of repeating code over and over again. See how the for loop has less code overall? It's considered best practice in coding to do the most amount of programming with the least amount of code possible.

1. Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
2. Add a `for loop` with a `range` of 5.
3. Add the appropriate code to your for loop improve the above example. Use `turtle.left()`.

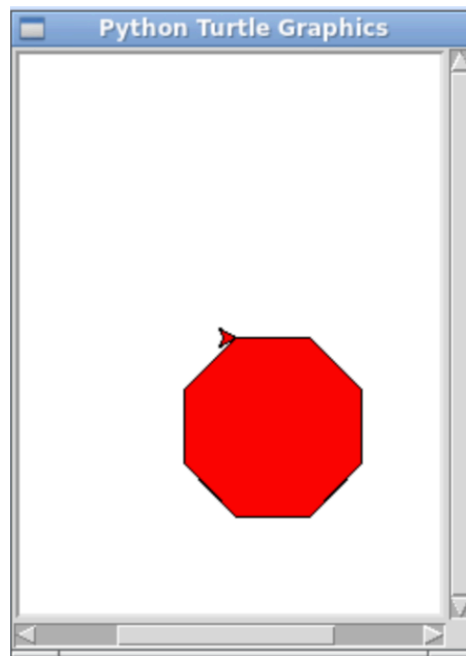
Requirements:

- Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
- Add a `for loop` with a `range` of 5. Use `my_counter`.
- Add the appropriate code to your for loop improve the above example. Use `turtle.left()`.

2. Stop Sign

Draw a stop sign using Python turtles! A stop sign is a red octagon that has 8 sides. Each interior angle in a stop sign is 135 degrees.

1. Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
2. Use `begin_fill()` , `end_fill()` , and `fillcolor()` to make the stop sign `red` .
3. Add a `for-loop` with the `range` needed to make the stop sign. Use `my_counter` .
4. Inside the for loop use the necessary commands to create your stop sign.



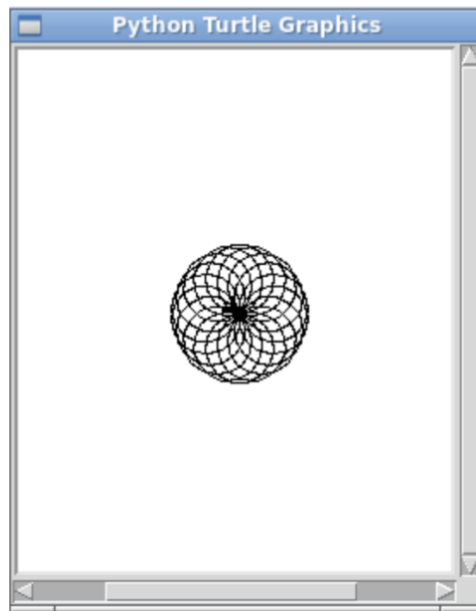
Requirements:

- Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
- Use `begin_fill()` , `end_fill()` , and `fillcolor()` to make the stop sign `red` .
- Add a `for-loop` with the `range` needed to make the stop sign. Use `my_counter` .
- Inside the for-loop, use the necessary commands to create your stop sign.

3. 20 20 20 Flower

Create a program that has 3 values of 20: The range, Circle radius, Rotating to the right.

1. Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
2. Add a `for loop` with a `range` of 20. Use `my_counter` .
3. Inside the for loop, draw a circle with a radius of 20.
4. Inside the for loop, rotate to the right 20 degrees.



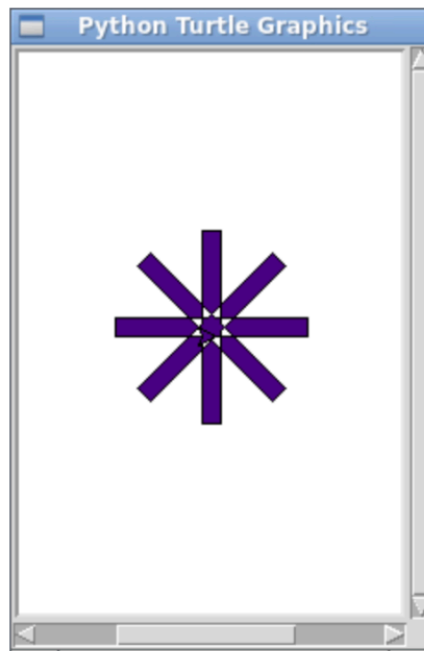
Requirements:

- Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
- Add a `for loop` with a `range` of 20. Use `my_counter` .
- Inside the for loop, draw a circle with a radius of 20.
- Inside the for loop, rotate to the right 20 degrees.

4. Square Petals

Create a flower with square petals!

1. Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
2. Include a `fillcolor` , `begin_fill` , and `end_fill` .
3. Include a for loop that uses range and `my_counter` .
4. Include at least 3 petals. The petals must have corners and have at least 2 turns of 90 degrees. (After creating a petal, rotate your turtle to create another one.)



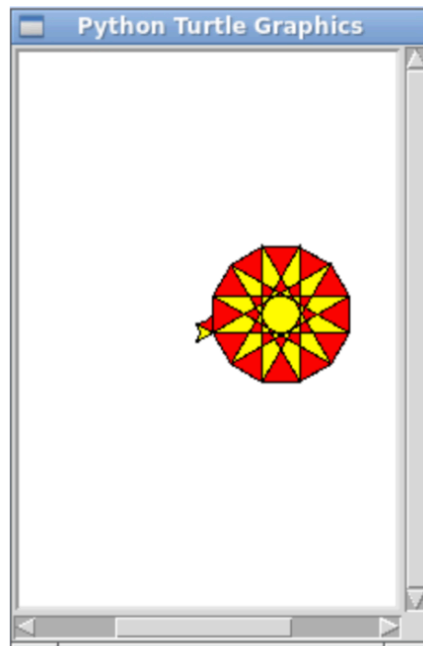
Requirements:

- Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
- Include a `fillcolor` , followed by a `begin_fill` command.
- Include a for loop that uses range and `my_counter` .
- Include at least 3 petals. The petals must have corners and have at least 2 turns of 90 degrees. (After creating a petal, rotate your turtle to create another one.)
- Include an `end_fill` command.

5. Stained Glass

Create a cool piece of stained glass! Do this by adding **two** for loops to your code! See what kind of shapes you can make.

1. Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
2. Create a new turtle variable using `turtle.Turtle()`.
3. Include two instances of a `fillcolor`, `begin_fill`, and `end_fill`.
4. Include two examples of for loops that uses range and `my_counter`.
5. Inside each for loop, have at least two commands for that turtle. Each for loop should include a unique turtle variable.
6. Include at least 2 comments in the code.



Requirements:

- Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
- Create a new turtle variable using `turtle.Turtle()`.
- Include two examples of a `fillcolor`, `begin_fill`, and `end_fill`.
- Include two examples of for loops that uses range and `my_counter`.
- Inside each for loop, have at least two commands for that turtle.
- Include at least 2 comments in the code.

Answer Keys & Solutions

Checkpoint Solutions

For Loops with Python Turtles

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(6):
5     turtle.forward(40)
6     turtle.right(60)
```

Questions

1. What is the purpose of a loop in programming?

MULTIPLE CHOICE

Correct Answer:

- A. To make code easier to read ✗ Incorrect
- B. To repeat a set of instructions ✓ Correct
- C. To speed up the program ✗ Incorrect
- D. To slow down the program ✗ Incorrect

Explanation:

Loops allow a chunk of code to repeat.

2. In the given For Loop example, how many times will the turtle move forward and turn left?

MULTIPLE CHOICE

Correct Answer:

- A. 4 ✓ Correct
- B. 5 ✗ Incorrect
- C. 8 ✗ Incorrect
- D. 2 ✗ Incorrect

Explanation:

What number is inside the range()?

3. Why is indentation important in Python, especially within a loop?

MULTIPLE CHOICE

Correct Answer:

- A. It makes the code look neat and organized ✗ Incorrect
- B. It helps the program run faster ✗ Incorrect
- C. It indicates what code is inside the For Loop ✓ Correct
- D. It requires less code overall ✗ Incorrect

Explanation:

Indentation allows you to know what code is to be repeated.

4. How many times will the following code run? (Notice that the code is NOT indented)

MULTIPLE CHOICE

Correct Answer:

- A. 1 ✗ Incorrect
- B. It will throw an error ✓ Correct
- C. 2 ✗ Incorrect
- D. 3 ✗ Incorrect

Explanation:

This code won't work because the indentation is incorrect.

5. What is the purpose of the "range()" function in the For Loop?

MULTIPLE CHOICE

Correct Answer:

A. It defines the size of the turtle screen

✗ Incorrect

B. It sets the speed of the turtle

✗ Incorrect

C. It determines the number of iterations for the loop

✓ Correct

D. It specifies the turtle's color

✗ Incorrect

Explanation:

The number in the parentheses of `range()` determines how many times the loop runs

6. What is the role of the variable "my_counter" in the For Loop?

MULTIPLE CHOICE

Correct Answer:

A. It defines the number of sides in the square

✗ Incorrect

B. It controls the speed of the turtle

✗ Incorrect

C. It represents a value that increases by 1 each time the loop runs

✓ Correct

D. It specifies the color of the turtle

✗ Incorrect

Explanation:

The counter is counting how many times the loop runs

7. Debug the following code:

DEBUG CODE

Incorrect Code:

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(4)
5     turtle.forward(100)
6     turtle.left(90)
```

Correct Solution:

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(4):
5     turtle.forward(100)
6     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(4):
5     turtle.forward(100)
6     turtle.left(90)
```

Correct Solution:

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(4):
5     turtle.forward(100)
6     turtle.left(90)
```

Explanation:

This code is missing an underscore.

9. Debug the following code:[DEBUG CODE](#)**Incorrect Code:**

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in count(8):
5     turtle.right(30)
6     turtle.forward(20)
```

Correct Solution:

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(8):
5     turtle.right(30)
6     turtle.forward(20)
```

Explanation:

The word count needs to be something else

10. Debug the following code:

DEBUG CODE

Incorrect Code:

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(8):
5     turtleright(30)
6     turtle.forward(20)
```

Correct Solution:

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(8):
5     turtle.right(30)
6     turtle.forward(20)
```

Challenges

1. Star Continued

Solution:

```
1 import turtle
2 turtle.getscreen()
3
4 for my_counter in range(5):
5     turtle.forward(80)
6     turtle.left(144)
```

2. Stop Sign

Solution:

```
1 import turtle
2 turtle.getscreen()
3
4 turtle.fillcolor("red")
5 turtle.begin_fill()
6
7 for my_counter in range(8):
8     turtle.forward(40)
9     turtle.right(45)
10
```

```
11 turtle.end_fill()
```

3. 20 20 20 Flower

Solution:

```
1 import turtle
2 turtle.getscreen()
3
4
5 for my_counter in range(20):
6     turtle.circle(20)
7     turtle.right(20)
```

4. Square Petals

Solution:

```
1 import turtle
2 turtle.getscreen()
3
4 turtle.fillcolor("indigo")
5 turtle.begin_fill()
6
7 for my_counter in range(8):
8     turtle.forward(50)
9     turtle.left(90)
10    turtle.forward(10)
11    turtle.left(90)
12    turtle.forward(50)
13    turtle.right(45)
14
15 turtle.end_fill()
```

5. Stained Glass

Solution:

```
1 import turtle
2 turtle.getscreen()
3
4 shane = turtle.Turtle()
5 shane.right(60)
6
7 #red background
8 shane.fillcolor("red")
9 shane.begin_fill()
10
11 for my_counter in range(12):
12     shane.forward(20)
13     shane.left(30)
14
15 shane.end_fill()
```



```
16
17 #yellow star
18 turtle.fillcolor("yellow")
19 turtle.begin_fill()
20
21 for my_counter in range(12):
22     turtle.forward(75)
23     turtle.left(150)
24
25 turtle.end_fill()
```