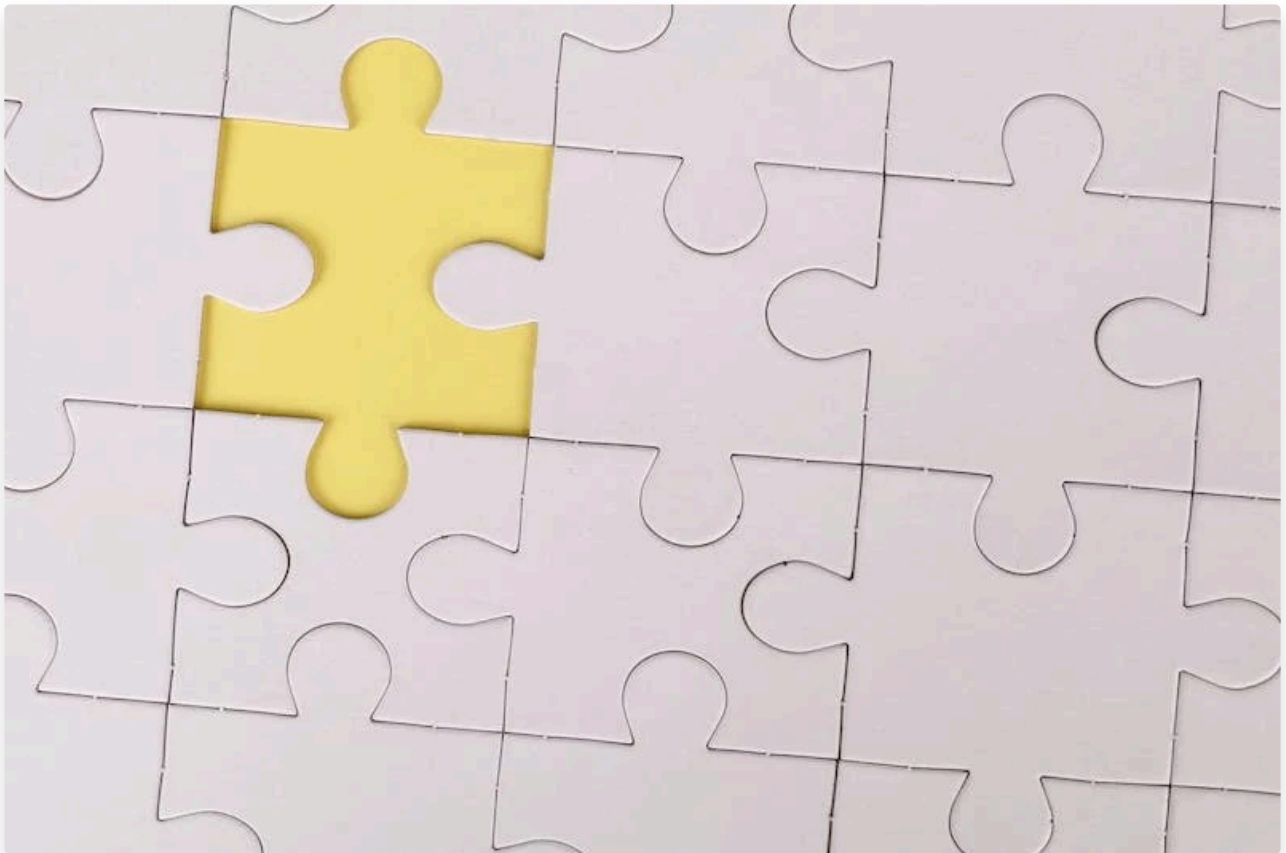


Arguments and Parameters in Turtle Functions

Textbook

Arguments and Parameters in Turtle Functions



Remember how to create a basic Python function?

```
1 def my_function():  
2     turtle.forward(50)  
3     turtle.right(90)  
4     turtle.forward(50)  
5  
6  
7 my_function()
```

Parameters

Now we will learn what goes inside the function parenthesis. [Function parameters](#) are what go inside the parentheses of the function declaration. These parameters can then be used inside the function.

```
1 def my_function(side):
2     turtle.forward(side)
3     turtle.right(90)
4     turtle.forward(side)
5
6
7 my_function()
```

In this function, the parameter is `side`. The parameter is just a way to signal **where in the function the value is supposed to go**.

Arguments

Notice that the same word inside the parenthesis can be found inside the turtle commands, which are inside the function. Now, to determine what the parameter named `forecast` actually IS, you say so when you call the function.

The value within the function call is an [argument](#).

In this example, the argument is the value `45`.

```
1 def my_function(side):
2     turtle.forward(side)
3     turtle.right(90)
4     turtle.forward(side)
5
6
7 my_function(45)
```

Notice that when we call the function named `my_function`, we put in the value `45`. The function then takes that value and puts it inside the function to determine how far the turtle will go. The value of `45` is the argument.

Call the Function Multiple Times

See what happens when you call the function multiple times with different arguments.

```
1 def my_function(side):
2     turtle.forward(side)
3     turtle.right(90)
4     turtle.forward(side)
5
6
7 my_function(45)
8 turtle.left(20)
9
10 my_function(100)
11 turtle.left(20)
12
13 my_function(10)
```

Why Arguments and Parameters are Useful

Functions are useful because you can use the same chunk of code over and over again without typing it all out. Arguments and parameters allow us to customize how the code behaves, while still using it over and over again.

In the previous example, we were able to still use the function to make a corner, and we can determine the size when we call the function.

Multiple Arguments/Parameters

Functions can also have multiple parameters and arguments.

```
1 def my_function(side, fill):
2     turtle.fillcolor(fill)
3     turtle.begin_fill()
4
5     turtle.forward(side)
6     turtle.right(90)
7     turtle.forward(side)
8
9     turtle.end_fill()
10
11
12 my_function(45, "blue")
```

Experiment to see what can happen with 3 or more parameters!

Input and Output in Functions

When we talk about functions, it's also helpful to think about **input** and **output**. In the world of programming, "input" refers to the information or data that a function receives to do its job. For our Turtle functions, the arguments we put inside the parentheses (like the `45` in `my_function(45)`) are the inputs. These inputs tell the function exactly how it should behave – for example, how far the turtle should move.

The "output" of a function can be what the function *produces* or *does* as a result of that input. For our Turtle commands, the output is often a visual result: the turtle drawing lines, turning, or changing color on the screen. So, you give the function an input (an argument like a number for `side`), and the function uses that input to produce an output (the turtle moving a specific distance and drawing a line).

Checkpoint

Arguments and Parameters in Turtle Functions

Practice creating a function that has a parameter! Then use an argument when you call the function.

1. Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
2. Create a function named `my_function` that has a parameter of `side`.
3. Inside the function, move the turtle `forward(side)`, `right(90)`, and `forward(side)`. (Indent each command.)
4. Call the function named `my_function` with an argument of 45.
5. Rotate the turtle to the left 20.
6. Call the function named `my_function` again with an argument of 100.

Requirements:

- Include the necessary code to start up a Python screen. (Import the library and generate a screen.)

- Create a function named `my_function` that has a parameter of `side` .
- Inside the function, move the turtle `forward(side)` , `right(90)` , and `forward(side)` .(Indent each command in.)
- Call the function named `my_function` with an argument of 45.
- Rotate the turtle to the left 20.
- Call the function named `my_function` again with an argument of 100.

Questions (10)

1. What is an argument in the context of Python functions?

MULTIPLE CHOICE

Choose the correct answer:

- A. A value inside the parentheses of a function call
- B. A value inside the parentheses of a function declaration
- C. The result of the function
- D. The name of the function

2. What is the purpose of function parameters in Python?

MULTIPLE CHOICE

Choose the correct answer:

- A. To signal where in the function the value is supposed to go
- B. To store values outside the function
- C. To define the function name
- D. To pause the program temporarily

3. In the following code, what is "side"?

MULTIPLE CHOICE

```
def my_function(here): turtle.forward(here) my_function(side)
```

Choose the correct answer:

- A. argument
- B. parameter
- C. function name
- D. printed word

4. In the following code, what is "here"?

```
def my_function(here): turtle.forward(here) my_function(side)
```

Choose the correct answer:

- A. argument
- B. parameter
- C. function name
- D. printed string

5. What is an argument in Python functions?**Choose the correct answer:**

- A. The same as a parameter
- B. The value within the function call
- C. A turtle command
- D. A variable outside the function

6. Why are arguments and parameters useful in functions?**Choose the correct answer:**

- A. To make the code shorter
- B. To customize how the function behaves
- C. To eliminate the need for functions
- D. To define turtle commands

7. What happens when you call a function multiple times with different arguments?**Choose the correct answer:**

- A. The function will produce an error
- B. The function will ignore the arguments
- C. The function will use the same arguments every time
- D. The function can produce different outcomes based on the arguments

8. How can you define multiple parameters in a Python function?

MULTIPLE CHOICE

Choose the correct answer:

- A. Separate them with commas inside the parenthesis
- B. Use different function names
- C. Place them outside the function
- D. Define them after the function call

9. In the following code, what is "blue"?

MULTIPLE CHOICE

```
def my_function(first, second): turtle.circle(first) turtle.fillcolor(second) my_function(45, "blue")
```

Choose the correct answer:

- A. argument
- B. parameter
- C. function name
- D. printed string

10. Debug the following code:

DEBUG CODE

Code to Debug:

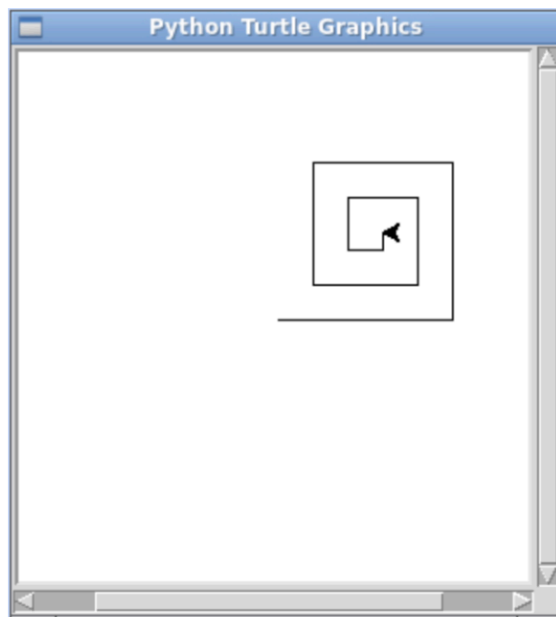
```
1 def my_function(first, second):  
2     turtle.circle(first)  
3     turtle.fillcolor(second)  
4  
5 my_function(45; "blue")
```

Challenges (5)

1. Spiral

Draw a spiral using functions with parameters!

1. Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
2. Create a function named `spiral` that has a parameter of `go`.
3. Inside the function move the turtle `forward()` and use the parameter of `go` in the parentheses.
4. Inside the function, rotate the turtle to the left 90 degrees.
5. Call the function named `spiral` 10 times. Start with an argument of `100` and decrease by 10 each time you call the function.



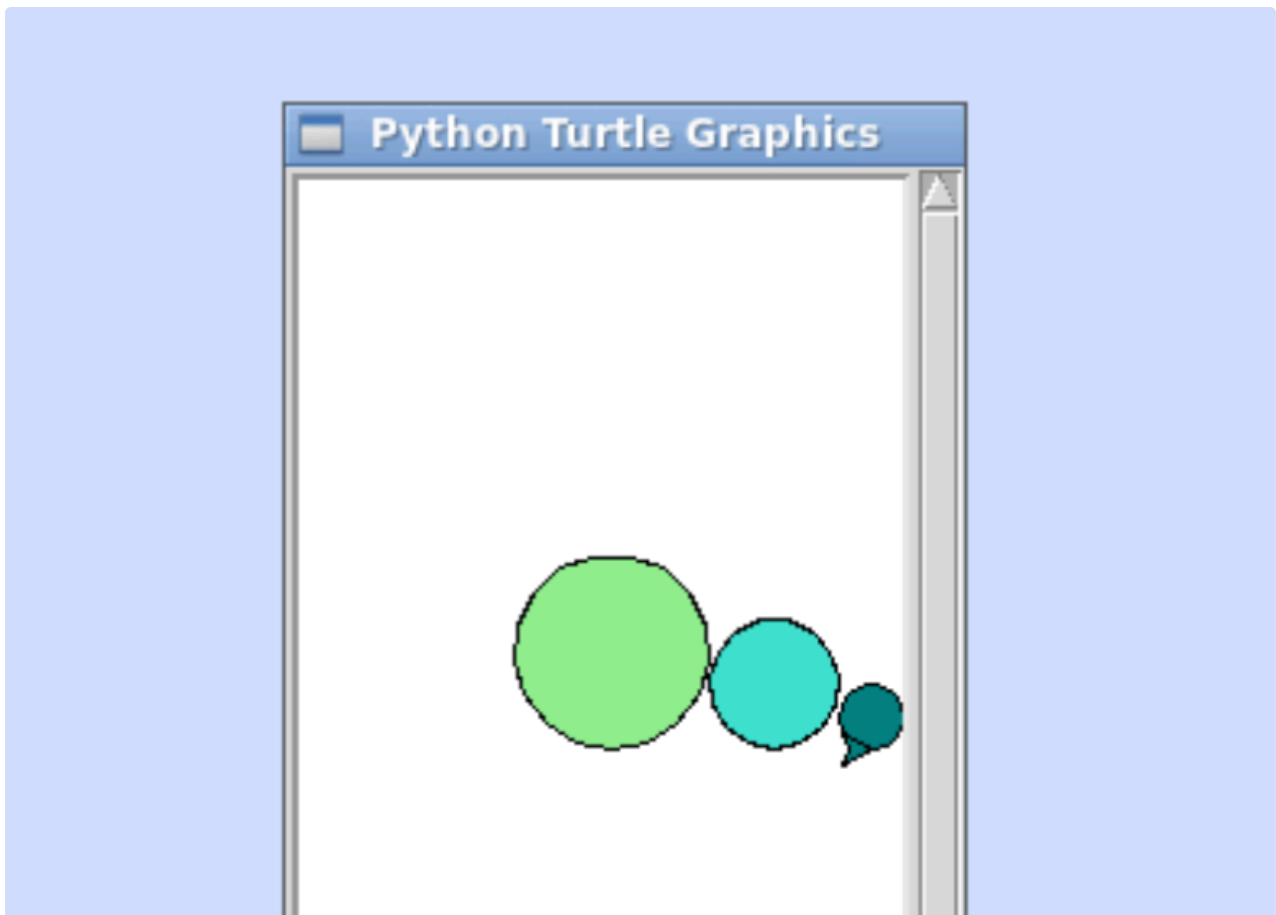
Requirements:

- Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
- Create a function named `spiral` that has a parameter of `go`.
- Inside the function move the turtle `forward()` and use the parameter of `go` in the parentheses.
- Inside the function, rotate the turtle to the left 90 degrees.
- Call the function named `spiral` 10 times. Start with an argument of `100` and decrease by 10 each time you call the function.

2. Polka Dots

Now let's try creating a program that has two functions! The first function will have 2 parameters. The second function will have 1 parameter.

1. Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
2. Create a function named `polka_dot` that has the parameters of `radius` and `color` .
3. Inside the function named `polka_dot` , draw a circle with the parameter of `radius` .
4. Inside the function named `polka_dot` , add the `fillcolor()` , `begin_fill()` , and `end_fill()` . Use the parameter named `color` in the `fillcolor()` command.
5. Create a function named `move` that has a parameter of `distance` .
6. Inside the function named `move` , add the commands of `penup()` , `turtle.forward()` , and `pendown()` . Place the parameter in the forward command.
7. Call the function named `polka_dot` 3 times using different arguments. Don't forget to put an integer and color value for your arguments.
8. Between the `polka_dot` function calls, call the function named `move()` a total of 2 times. Use an integer argument in the function.



Requirements:

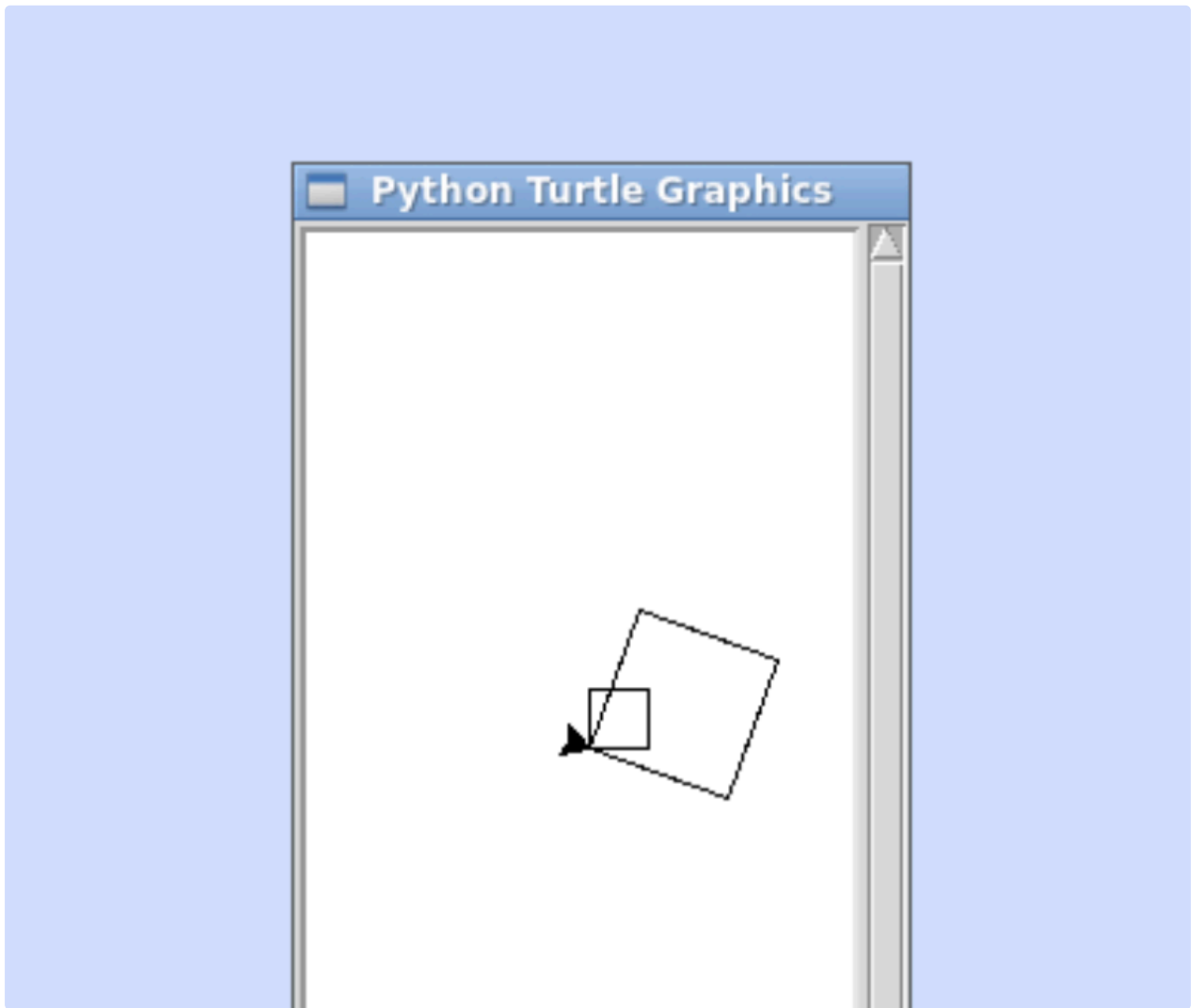
- Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
- Create a function named `polka_dot` that has the parameters of `radius` and `color` .
- Inside the function named `polka_dot` , draw a circle with the parameter of `radius` .
- Inside the function named `polka_dot` , add the `fillcolor()` , `begin_fill()` , and `end_fill()` . Use the parameter named `color` in the `fillcolor()` command.
- Create a function named `move` that has a parameter of `distance` .

- Inside the function named `move` , add the commands of `penup()` , `forward()` , and `pendown()` . Place the parameter in the forward command.
- Call the function named `polka_dot` 3 times using different arguments. Don't forget to put an integer and color value for your arguments.
- Between the `polka_dot` function calls, call the function named `move()` a total of 2 times. Use an integer argument in the function.

3. Square Function Continued

In the previous lesson, we made a function that draws a square. Now update that code so that we can determine the size of the square when we call the function.

1. Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
2. Create a function named `square` that has a parameter of `length`.
3. Inside the function, draw a square using `forward(length)` and `left(90)`. (Indent each command in.)
4. Call the function named `square` with an argument of 20.
5. Rotate the turtle to the right 20.
6. Call the function named `square` `again` with an argument of 50. See how it makes a different square size?



Requirements:

- Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
- Create a function named `square` that has a parameter of `length`.
- Inside the function, draw a square using `forward(length)` and `left(90)`. (Indent each command in.)
- Call the function named `square` with an argument of 20.
- Rotate the turtle to the right 20.
- Call the function named `square` `again` with an argument of 50. See how it makes a different square size?

4. Blue Triangle Continued

Last lesson we created a program that would make blue triangles. Update the code so that you can determine the color of the triangle when you call the function by using arguments and parameters.

1. Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
2. Create a function named `triangle` with a parameter of `mycolor` .
3. Inside the function, draw a triangle using `forward(50)` and `left(120)` . (Indent each command in so they are inside the function.)
4. Inside the function, include `begin_fill()` and `end_fill()` commands.
5. Inside the function, include a `fillcolor` with the parameter of `mycolor` .
6. Call the function named `triangle` with an argument of `"red"` .
7. Rotate to the left 45.
8. Call the function named `triangle` again with an argument of `"orange"` .

Requirements:

- Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
- Create a function named `triangle` with a parameter of `mycolor` .
- Inside the function, draw a triangle using `forward(50)` and `left(120)` . (Indent each command in so they are inside the function.)
- Inside the function, include `begin_fill()` and `end_fill()` commands.
- Inside the function, include a `fillcolor` with the parameter of `mycolor` .
- Call the function named `triangle` with an argument of `"red"` .
- Rotate to the left 45.
- Call the function named `triangle` again with an argument of `"orange"` .

5. Star Stamp Continued

In the previous lesson, we made a function that draws a star. Now update that code so that we can determine the size of the star when we call the function.

1. Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
2. Create a function named `star` that has a parameter of `distance` .
3. Inside the function, draw a star. Use 144 degree turns. Use the parameter named `distance` .
4. Call the star function a total of 3 times. Use an argument of `10` , `20` , and `30` .
5. Move the turtle between your stars, using `penup()` and `pendown()` .

Requirements:

- Include the necessary code to start up a Python screen. (Import the library and generate a screen.)
- Create a function named `star` that has a parameter of `distance` .
- Inside the function, draw a star. Use 144 degree turns. Use the parameter named `distance` .
- Call the star function a total of 3 times. Use an argument of `10` , `20` , and `30` .
- Move the turtle between your stars, using `penup()` and `pendown()` .

Answer Keys & Solutions

Checkpoint Solutions

Arguments and Parameters in Turtle Functions

```
1 import turtle
2 turtle.getscreen()
3
4 def my_function(side):
5     turtle.forward(side)
6     turtle.right(90)
7     turtle.forward(side)
8
9
10 my_function(45)
11 turtle.left(20)
12
13 my_function(100)
```

Questions

1. What is an argument in the context of Python functions?

MULTIPLE CHOICE

Correct Answer:

- A. A value inside the parentheses of a function call ✓ Correct
- B. A value inside the parentheses of a function declaration ✗ Incorrect
- C. The result of the function ✗ Incorrect
- D. The name of the function ✗ Incorrect

Explanation:

A parameter is in the parentheses of a function declaration.

2. What is the purpose of function parameters in Python?

MULTIPLE CHOICE

Correct Answer:

- A. To signal where in the function the value is supposed to go ✓ Correct

B. To store values outside the function

✗ Incorrect

C. To define the function name

✗ Incorrect

D. To pause the program temporarily

✗ Incorrect

Explanation:

The parameter is just a placeholder that tells where the argument is supposed to go.

3. In the following code, what is "side"?

MULTIPLE CHOICE

Correct Answer:

A. argument

✓ Correct

B. parameter

✗ Incorrect

C. function name

✗ Incorrect

D. printed word

✗ Incorrect

Explanation:

The parameters go in the parentheses of the function declaration.

4. In the following code, what is "here"?

MULTIPLE CHOICE

Correct Answer:

A. argument

✗ Incorrect

B. parameter

✓ Correct

C. function name

✗ Incorrect

D. printed string

✗ Incorrect

Explanation:

Arguments go inside the function call

5. What is an argument in Python functions?

MULTIPLE CHOICE

Correct Answer:

- A. The same as a parameter ✗ Incorrect
- B. The value within the function call ✓ Correct
- C. A turtle command ✗ Incorrect
- D. A variable outside the function ✗ Incorrect

Explanation:

Arguments go inside the function call

6. Why are arguments and parameters useful in functions?

MULTIPLE CHOICE

Correct Answer:

- A. To make the code shorter ✗ Incorrect
- B. To customize how the function behaves ✓ Correct
- C. To eliminate the need for functions ✗ Incorrect
- D. To define turtle commands ✗ Incorrect

Explanation:

Arguments and parameters make functions even more useful.

7. What happens when you call a function multiple times with different arguments?

MULTIPLE CHOICE

Correct Answer:

- A. The function will produce an error ✗ Incorrect
- B. The function will ignore the arguments ✗ Incorrect
- C. The function will use the same arguments every time ✗ Incorrect
- D. The function can produce different outcomes based on the arguments ✓ Correct

Explanation:

Functions are highly useful because you can use different arguments.

8. How can you define multiple parameters in a Python function?

MULTIPLE CHOICE

Correct Answer:

- A. Separate them with commas inside the parenthesis ✓ Correct
- B. Use different function names ✗ Incorrect
- C. Place them outside the function ✗ Incorrect
- D. Define them after the function call ✗ Incorrect

Explanation:

Here's an example `def my_function(first, second, third):`

9. In the following code, what is "blue"?

MULTIPLE CHOICE

Correct Answer:

- A. argument ✓ Correct
- B. parameter ✗ Incorrect
- C. function name ✗ Incorrect
- D. printed string ✓ Correct

Explanation:

Arguments go inside the function call.

10. Debug the following code:

DEBUG CODE

Incorrect Code:

```
1 def my_function(first, second):
2     turtle.circle(first)
3     turtle.fillcolor(second)
4
5 my_function(45; "blue")
```

Correct Solution:

```
1 def my_function(first, second):
2     turtle.circle(first)
3     turtle.fillcolor(second)
4
5 my_function(45, "blue")
```

Explanation:

The semicolon needs to be something else.

Challenges

1. Spiral

Solution:

```
1 import turtle
2 turtle.getscreen()
3
4 def spiral(go):
5     turtle.forward(go)
6     turtle.left(90)
7
8 spiral(100)
9 spiral(90)
10 spiral(80)
11 spiral(70)
12 spiral(60)
13 spiral(50)
14 spiral(40)
15 spiral(30)
16 spiral(20)
17 spiral(10)
```

2. Polka Dots

Solution:

```
1 import turtle
2 turtle.getscreen()
3
4 def polka_dot(radius, color):
5     turtle.fillcolor(color)
6     turtle.begin_fill()
7     turtle.circle(radius)
8     turtle.end_fill()
9
10 def move(distance):
11     turtle.penup()
12     turtle.forward(distance)
13     turtle.pendown()
14
15
16 polka_dot(30, "lightgreen")
```



```
17
18 move(50)
19
20 polka_dot(20, "turquoise")
21
22 move(30)
23
24 polka_dot(10, "teal")
```

3. Square Function Continued

Solution:

```
1 import turtle
2 turtle.getscreen()
3
4 def square(length):
5     turtle.forward(length)
6     turtle.left(90)
7     turtle.forward(length)
8     turtle.left(90)
9     turtle.forward(length)
10    turtle.left(90)
11    turtle.forward(length)
12    turtle.left(90)
13
14 square(20)
15 turtle.right(20)
16 square(50)
```

4. Blue Triangle Continued

Solution:

```
1 import turtle
2 turtle.getscreen()
3
4 def triangle(mycolor):
5     turtle.fillcolor(mycolor)
6     turtle.begin_fill()
7     turtle.forward(50)
8     turtle.left(120)
9     turtle.forward(50)
10    turtle.left(120)
11    turtle.forward(50)
12    turtle.end_fill()
13
14 triangle("red")
15
16 turtle.left(45)
```

```
17 triangle("orange")
```

5. Star Stamp Continued

Solution:

```
1 import turtle
2 turtle.getscreen()
3
4 def star(distance):
5     for my_counter in range(5):
6         turtle.forward(distance)
7         turtle.left(144)
8
9 star(10)
10
11 turtle.penup()
12 turtle.forward(40)
13 turtle.pendown()
14
15 star(20)
16
17 turtle.penup()
18 turtle.forward(40)
19 turtle.pendown()
20
21 star(30)
```