

Python Functions

Textbook

Python Functions

A [function](#) is a chunk of code that only runs when it is called. Sometimes you will want to write out some code, but aren't ready for it to run yet. This is where functions are useful.



The programming approach where the code is constructed by applying and composing functions is called [Functional Programming](#).

Functions in Python are defined, using the abbreviation `def` followed by the name followed by some parenthesis `()`. Don't worry about putting anything into the parenthesis for now.

```
1 def weather():  
2     print("It's a soggy day outside!")  
3
```

Try it!

Notice that the code below the function named `weather` is indented. This means the indented code is responding to the function named `weather`.

Hit run and see what happens. You'll notice that nothing happens at this point. This is because we have not yet [called the function](#). Functions are like an obedient pet and only come when called.

```
1 def weather():
2     print("It's a soggy day outside!")
3
4
5 weather()
```

Try it!

See how we not only defined the function, but we also called it.

Remember how sometimes you will want to write out some code now but aren't ready for it to run yet? You can write it out any time and then just call it later by using [functions](#). Think of the code that sets an alarm for you to wake up in the morning. The code for the alarm to ring exists, but it doesn't trigger until the time you set it for. Once the time comes, the function calls to ring your alarm. See how functions are useful for programming?

The Pass Statement

If you want to start writing a function, but aren't sure what you want to put inside yet, you can use the `pass` statement. The pass statement works as a placeholder until you add your code.

```
1 def weather():
2     pass
3
4
5 weather()
```

Try it!

Abstraction

Python functions are an example of abstraction in programming. Remember that abstraction is representing something complex with something simple.

Python functions are like recipe instructions. When you use a function, you're using a predefined set of steps to do something specific, like baking a cake. You don't need to know all the details of how the oven works or how ingredients interact; you just follow the recipe.

This idea of functions hiding the details and providing a simpler way to get things done is what we mean by abstraction in programming. It makes coding more organized and easier to manage because you can reuse these "recipes" (functions) whenever you need them without worrying about the nitty-gritty every time

Local and Global Scope Variables

Sometimes you want to create a variable *inside* a function. This is called a [local variable](#). Local variables can only be used inside the function.

Here is an example of a local variable.

```
1 favorite = "I love juice"
2 def myfunction():
3     fruit = "apple"
```

```
4     print(fruit)
5
6 myfunction()
7
8 print(favorite)
```

In this example, `fruit` is the local variable.

Variables that are created outside functions are called [global variables](#). Global variables can be used anywhere in the program.

In the above example, `favorite` is the global variable.

If you want a variable inside a function to be global, you can use the `global` keyword.

```
1 favorite = "I love juice"
2 def myfunction():
3     global fruit
4     fruit = "apple"
5
6 myfunction()
7
8 print(favorite)
9 print(fruit)
```

Try it!

Now the variable named `fruit` can be used in other places in the program. See how it is printed at the end? This is possible because it's a global variable.

Void and Non Void Functions

In Python, a **void function** is a function that performs an action or a series of operations but **does not explicitly return a value**. Think of it like a chef who prepares a meal but doesn't hand you a plate; they just put it on the table.

When a Python function finishes execution without a `return` statement, or with a `return` statement that has nothing after it (like `return` by itself), it implicitly returns `None`. `None` is a special constant in Python that simply means "no value."

For example, a function that prints a message to the console, changes a global variable, or writes data to a file would typically be a void function. Its main purpose is to create a **side effect** (like showing text or saving data) rather than giving back a calculated result. If you try to assign the output of one of these functions to a variable, that variable will just end up holding `None`.

Built in Functions vs User Defined Functions

Sure! Here's a shorter version:

Built-in functions are provided by Python and is available for use without any setup. Examples include `print()`, `len()`, and `int()`. (The string methods learned previously in the String Methods lessons are built in functions.)

User-defined functions are created by the programmer using the `def` keyword to perform custom tasks. For example:

```
1 def greet(name):
```

```
2  
3 print("Hello, " + name)
```

Try it!

- Built-in functions are pre-defined in Python.
- User-defined functions are written by the programmer to meet specific needs.

The Return Statement

Sometimes, a function doesn't just do something; it also gives a value back to the part of the code that called it. This is where the `return` statement comes in. When a `return` statement is used inside a function, the function stops running, and the value specified after `return` is sent back.

Think of it like asking a question and getting an answer. You might ask a function what today's flavor it, and the `return` statement gives you the flavor of the day. If a function doesn't have a `return` statement, it automatically returns `None`.

Functions that don't have a return statement are sometimes referred to as **void functions**.

Functions that do have a return statement are sometimes referred to as **non-void functions**.

```
1 def todaysflavor():  
2  
3     return "cherry"  
4  
5 flavor = todaysflavor()  
6 print(flavor)
```

Try it!

Checkpoint

Python Functions Checkpoint

Create a function named `race`.

Inside the function, create a `print` statement that says who won the race.

Call the function named `race`.

Requirements:

- Create a function named `race`.
- Inside the function, create a print statement that says who won the race.
- Call the function named `race`.

Questions (8)

1. When will a Python function actually run?

MULTIPLE CHOICE

Choose the correct answer:

- A. When it is created.
- B. When it is called.
- C. At the beginning of the program.
- D. When it is defined.

2. Which of the following is an example of a function call? Choose one.

MULTIPLE CHOICE

Choose the correct answer:

- A. `def dinner()`
- B. `dinner()`
- C. `run dinner`
- D. `dinner`
- E. `call dinner()`

3. Edit the text box below to debug (fix) the code:

DEBUG CODE

Code to Debug:

```
1 def dinner();
2     print("Here is my function!")
```

4. True or False: Things "inside the function" need to be indented.

MULTIPLE CHOICE

Choose the correct answer:

- A. True
- B. False

MULTIPLE CHOICE

5. What will the following code print out?

```
def alarm(): print("It's time to wake up!") alarm()
```

Choose the correct answer:

- A. alarm
- B. def alarm
- C. It's time to wake up!
- D. alarm()

MULTIPLE CHOICE

6. What will the following code print out?

```
def alarm(): print("It's time to wake up!")
```

Choose the correct answer:

- A. It's time to wake up!
- B. It won't print anything.
- C. alarm
- D. def alarm

DEBUG CODE

7. Edit the text box below to debug (fix) the code:

Code to Debug:

```
1 def alarm:
2     print("It's time to wake up!")
3
4
5 alarm()
```

DEBUG CODE

8. Edit the text box below to debug (fix) the code:

Code to Debug:

```
1 alarm():
2     print("It's time to wake up!")
3
4 alarm()
```

Challenges (6)

1. Charlie the Dog

You have the best dog in the world named Charlie. You like Charlie so much, you created a function in his honor.

1. Create a function named `charlie` .
2. Inside the function create a **print statement** that talks about how great Charlie is.
3. **Call the function** named `charlie` .

Requirements:

- Create a function named `charlie` .
- Inside the function, create a print statement that says how great Charlie is.
- Call the function named `charlie` .

2. Start the Game

You were hired by a game developing company. You are super excited to be helping create cutting edge games that you know your friends are going to love. Currently, you are working on the start sequence for the game.

1. Write a function that asks the user if they want to start the game. If they answer yes, "initialize" the game.
2. **Create an input** that asks the user if they want to start the game.
3. **Create a function.**
4. Inside the function, check to see if the user answer is yes.

For example:

Input: `yes`

Output: `Initialization Complete`

Another example:

Input: `no`

Output: `Initialization Failed`

Call the function.

3. Chickens and Eggs

You are excited to get some chickens for your yard! You are wondering how many eggs total you can expect to get in a week. This depends on how many eggs each chicken can lay in a week. Create a function to calculate it for you.

1. Create a **variable** named `chickens` and assign it to an integer.
2. Create a variable named `eggs_per_week` and assign it to an integer. This number is how many eggs one chicken will lay per week.
3. **Create a function** named `farmer` .
4. Inside the function named `farmer` , include a **print** statement.
5. Inside the print statement, **calculate** how many eggs you can get a week if each chicken laid the same amount of eggs per week.
6. **Call the function** named `farmer` .

Requirements:

- Create a variable named `chickens` and assign it to an integer
- Create a variable named `eggs_per_week` and assign it to an integer.
- Create a function named `farmer`
- Inside the function, include a print statement that multiplies chickens by `eggs_per_week`
- Call the function named farmer.

4. Hello Generator

1. Create a program that duplicates the word `Hello` as many times as the user inputs and prints it all on one line.
2. Create an input that asks the user for an integer.
3. **Create a function.**
4. Inside the function, create a print statement that prints out the word `Hello` as many times as the input.
5. **Call the function.**

For example:

Input: `4`

Output: `HelloHelloHelloHello`

Another example:

Input: `2`

Output: `HelloHello`

5. Hungry for Apples

1. Define a function. Inside that function include an input statement that asks the hungry user for the name of a fruit.
2. Inside the function, use this fruit variable, then print it back for the hungry user three times in the same print statement before they get hangry!
3. Keep your eye out for Hungry for Apples Challenges 2 and 3!
4. Be sure to **call the function** afterwards!

Example:

Input: `Apple`

Output: `AppleAppleApple`

Another example:

Input: `Orange`

Output: `OrangeOrangeOrange`

6. Cube Calculations

You have a cube of solid gold. You are sure it is worth a lot of money. Before you can take it somewhere to sell, you want to measure the size of the cube.

1. Write a program that will calculate the surface area and the volume of the cube.
2. Create a **variable** named `length` and assign it to an integer.
3. **Create a function named** `surface_area` .
4. Inside the function include a **print** statement that will calculate the surface area according to the length.
5. **Call the function named** `surface_area` . Hint: Surface area is length times length times 6.
6. **Create a function named** `volume` . Inside the function, include a **print** statement that will calculate the volume according to the length. Hint: Volume is length times length times length.
7. **Call the function named** `volume` .

Note: The volume of a cube is length times length times length.

The surface area of a cube is length times length times 6.

Requirements:

- Create a variable named `length` and assign it to a variable
- Create a function named `surface_area`
- Inside the function named `surface_area` , include a print statement that calculates the surface area.
- Call the function named `surface_area` .
- Create a function named `volume`
- Inside the function named `volume` , include a print statement that calculates the volume.
- Call the function named `volume` .

Answer Keys & Solutions

Checkpoint Solutions

Python Functions Checkpoint

```
1 def race():  
2     print("Sally won the race!")  
3  
4  
5 race()
```

Questions

1. When will a Python function actually run?

MULTIPLE CHOICE

Correct Answer:

- A. When it is created. ✗ Incorrect
- B. When it is called. ✓ Correct
- C. At the beginning of the program. ✗ Incorrect
- D. When it is defined. ✗ Incorrect

Explanation:

Functions wait to run until the programmer says for them to run.

2. Which of the following is an example of a function call? Choose one.

MULTIPLE CHOICE

Correct Answer:

- A. `def dinner()` ✗ Incorrect
- B. `dinner()` ✓ Correct
- C. `run dinner` ✗ Incorrect
- D. `dinner` ✗ Incorrect
- E. `call dinner()` ✗ Incorrect

Explanation:

Function calls are the name of the function with empty parentheses.

3. Edit the text box below to debug (fix) the code:

DEBUG CODE

Incorrect Code:

```
1 def dinner();  
2     print("Here is my function!")
```

Correct Solution:

```
1 def dinner():  
2     print("Here is my function!")
```

Explanation:

The semicolon needs to be something else.

4. True or False: Things "inside the function" need to be indented.

MULTIPLE CHOICE

Correct Answer:

A. True

✓ Correct

B. False

✗ Incorrect

Explanation:

Indentation matters in Python

5. What will the following code print out?

MULTIPLE CHOICE

Correct Answer:

A. alarm

✗ Incorrect

B. def alarm

✗ Incorrect

C. It's time to wake up!

✓ Correct

D. alarm()

✗ Incorrect

Explanation:

The function named alarm was called, running the indented code.

6. What will the following code print out?

MULTIPLE CHOICE

Correct Answer:

- A. It's time to wake up! ✗ Incorrect
- B. It won't print anything. ✓ Correct
- C. alarm ✗ Incorrect
- D. def alarm ✗ Incorrect

Explanation:

Functions wait to run until they are called.

7. Edit the text box below to debug (fix) the code:

DEBUG CODE

Incorrect Code:

```
1 def alarm:
2     print("It's time to wake up!")
3
4
5 alarm()
```

Correct Solution:

```
1 def alarm():
2     print("It's time to wake up!")
3
4
5 alarm()
```

Explanation:

The function is missing parentheses

8. Edit the text box below to debug (fix) the code:

DEBUG CODE

Incorrect Code:

```
1 alarm():
2     print("It's time to wake up!")
3
4 alarm()
```

Correct Solution:

```
1 def alarm():
2     print("It's time to wake up!")
3
4 alarm()
```

Explanation:

The function needs the word def

Challenges

1. Charlie the Dog

Solution:

```
1 def charlie():
2     print("Charlie is so fun and smart!")
3
4
5 charlie()
```

2. Start the Game

Solution:

```
1 start = input("Would you like to launch the game? enter yes or no")
2
3 def start_game():
4     if start == "yes":
5         print("Initialization Complete")
6     else:
7         print("Initialization Failed")
8
9 start_game()
```

3. Chickens and Eggs

Solution:

```
1 chickens = 5
2 eggs_per_week = 9
3
4 def farmer():
5     print(chickens * eggs_per_week)
```

```
6
7
8 farmer()
```

4. Hello Generator

Solution:

```
1 number = int(input("Choose a number"))
2
3 def numBot():
4     print("Hello" * number)
5
6 numBot()
```

5. Hungry for Apples

Solution:

```
1 def hungryForApples():
2     fruit = input("Please enter the name of a fruit: ")
3     print(fruit + fruit + fruit)
4
5
6 hungryForApples()
```

6. Cube Calculations

Solution:

```
1 length = 5
2
3 def surface_area():
4     print(length*length*6)
5
6
7 surface_area()
8
9 def volume():
10    print(length*length*length)
11
12
13 volume()
```