W02 Learning Activity (2 of 2): Function Details

During this lesson, you will learn additional details about writing and calling functions. These details include variable scope, default parameter values, and optional arguments and will help you understand functions better and write them more effectively.

Variable Scope

The *scope* of a variable determines how long that variable exists and where it can be used. Within a Python program, there are two categories of scope: local and global. A variable has *local scope* when it is defined (assigned a value) inside a function. A variable has *global scope* when it is defined outside of all functions. Here is a small Python program that has two variables: *g* and *x*. *g* is defined outside of all functions and therefore has global scope. *x* is defined inside the **`main`** function and therefore has local scope.

```python
# g is a global variable because it
# is defined outside of all functions.
g = 25
def main():
    # x is a local variable because
    # it is defined inside a function.
    x = 1
```

As shown in the following table, a local variable (a variable with local scope) is defined inside a function, exists for as long as its containing function is executing, and can be used within its containing function but nowhere else. A global variable (a variable with global scope) is defined outside all functions, exists for as long as its containing Python program is executing, and can be used within all functions in its containing Python program.

	Python Variable Scope	
	Local	Global
Where to Define	Inside a function	Outside all functions
Owner	The function where the variable is defined	The Python file where the variable is defined
Lifetime	Only as long as its containing function is executing	As long as its containing program is executing
Where Usable	Only inside the function where it is defined	In all functions of the Python program

The following Python code example contains parameters and variables. Parameters have local scope because they are defined within a function, specifically within a function's header and exist for as long as their containing function is executing. The variable *nShapes* is global because it is defined outside of all functions. Because it is a global variable, the code in the body of all functions may use the variable *nShapes*. Within the **square_area** function, the parameter named *length* and the variable named *area* both have local scope. Within the **rectangle_area** function, the parameters named *width* and *length* and the variable named *area* have local scope.

```python
nShapes = 0
def square_area(length):
  area = length * length
  return area
def rectangle_area(width, length):
  area = width * length
  return area
```

Because local variables are visible only within the function where they are defined, a programmer can define two variables with the same name as long as he defines them in different functions. In the previous example, both the **square_area** and **rectangle_area** functions contain a parameter named *length* and a variable named *area*. All four of these variables are entirely separate and do not conflict with each other in any way because the scope of each variable is local to the function where it is defined.

When you write a program, you should use local variables whenever possible and use global variables only when absolutely necessary. Global variables (not global constants) make a program hard to understand and easy to write mistakes in the code. If you must use a global variable in your program, it is important to know that all functions can *access* or *read* the value of a global variable directly. However, in order for a function to *modify* the value of a global variable, the global variable must be declared as **`global`** inside the function. W3Schools has examples of using global variables in Python.

Common Mistake

A common mistake that many programmers make is to assume that a local variable can be used inside other functions. For example, the Python program in example 3 includes two functions named **`main`** and **`circle_area`**. The function **`main`** defines a variable named *radius*. Some programmers assume that the variable *radius* that is defined in **`main`** (and is therefore local to **`main`** only) can be used in the **`circle_area`** function. However, local variables from one function cannot be used inside another function. The local variables from **`main`** cannot be used inside **`circle_area`**.

```python
# Example 3
import math
def main():
    radius = float(input("Enter the radius of a circle: "))
    area = circle_area()
    print(f"area: {area:.1f}")
def circle_area():
    # Mistake! There is no variable named radius
    # defined inside this function, so the variable
    # radius cannot be used in this function.
    area = math.pi * radius * radius
    return area
main()
```

```
> python example_3.py
Enter the radius of a circle: 4.17
Traceback (most recent call last):
  File "c:\Users\cse111\example_3.py", line 17, in <module>
    main()
  File "c:\Users\cse111\example_3.py", line 7, in main
    area = circle_area()
  File "c:\Users\cse111\example_3.py", line 14, in circle_area
    area = math.pi * radius * radius
```

```
                        ^^^^^^
NameError: name 'radius' is not defined
```

The correct way to fix the mistake in example 3 is to add a parameter to the **`circle_area`** function and pass the radius from the **`main`** function to the **`circle_area`** function as shown in example 4.

```python
# Example 4
import math
def main():
  radius = float(input("Enter the radius of a circle: "))
  area = circle_area(radius)
print(f"area: {area:.1f}")
  def circle_area(radius):
  area = math.pi * radius * radius
  return area
main()
```

```
> python example_4.py
Enter the radius of a circle: 4.17
area: 54.6
```

Default Parameter Values and Optional Arguments

Python allows function parameters to have default values. If a parameter has a default value, then its corresponding argument is optional. If a function is called without an argument, the corresponding parameter gets its default value.

Consider the program in example 5. Notice at line 14 in the header for the **`arc_length`** function, that the parameter *radius* does not have a default value but the parameter *degrees* has a default value of 360. This means that when a programmer writes code to call the **`arc_length`** function, the programmer must pass a value for *radius* but is not required to pass a value for *degrees*. At line 6, the programmer wrote code to call the **`arc_length`** function and passed 4.7 for the *radius* parameter but did not pass a value for the *degrees*, so during that call to **`arc_length`**, the value of *degrees* will be the default value from line 14, which is 360. At line 10, the programmer wrote code to call the **`arc_length`** function again and passed two arguments: 4.7 and 270, so during that call to **`arc_length`**, the value of *degrees* will be 270.

```python
1 # Example 5
```

```python
import math
def main():
    # Call the arc_length function with only one argument
    # even though the arc_length function has two parameters.
    len1 = arc_length(4.7)
    print(f"len1: {len1:.1f}")
    # Call the arc_length function again but
    # this time with two arguments.
    len2 = arc_length(4.7, 270)
    print(f"len2: {len2:.1f}")
# Define a function with two parameters. The
# second parameter has a default value of 360.
def arc_length(radius, degrees=360):
    """Compute and return the length of an arc of a circle"""
    circumference = 2 * math.pi * radius
    length = circumference * degrees / 360
    return length
main()
```

```
> python example_5.py
len1: 29.5
len2: 22.1
```

Function Design

What are the properties of a good function?

There are many things to consider when writing a function, and many authors have written about design concepts that make functions easier to understand and less error prone. In future courses at BYU-Idaho, you will study some of these design concepts. For CSE 111, the following list contains a few properties that you should incorporate into your functions.

- A good function is understandable by other programmers. One way to make a function understandable is to write a documentation string at the top of the function that describes the function, its parameters, and its return value. Another way to make a function understandable is to write comments in the body of the function as needed.
- A good function performs a single task that the programmer can describe, and the function’s name matches its task.
- A good function is relatively short, perhaps fewer than 20 lines of code.

- A good function has as few decision points (`if` statements and loops) as possible. Too many decision points in a function make a function error prone and difficult to test.
- A good function is as reusable as possible. Functions that use parameters and return a result are more reusable than functions that get input from a user and print results to a terminal window.

As you read the sample code in CSE 111, observe how the sample functions fit these good properties, and as you write programs for CSE 111, do your best to write functions that have these good properties.

Summary

During this lesson, you are studying variable scope, default parameter values, and optional arguments. A variable that is defined (assigned a value) inside a function has local scope, and it can be used inside only the function where it is defined. Parameters always have local scope.

A parameter may have a default value like the parameter *degrees* in this function header:

```
def arc_length(radius, degrees=360):
    ⋮
```

When a function’s parameter has a default value, you can write a call to that function without passing an argument for the parameter like this:

```
length = arc_length(3.5)
```

In other words, the argument for a parameter that has a default value is optional.

04 Checkpoint: Variable Scope

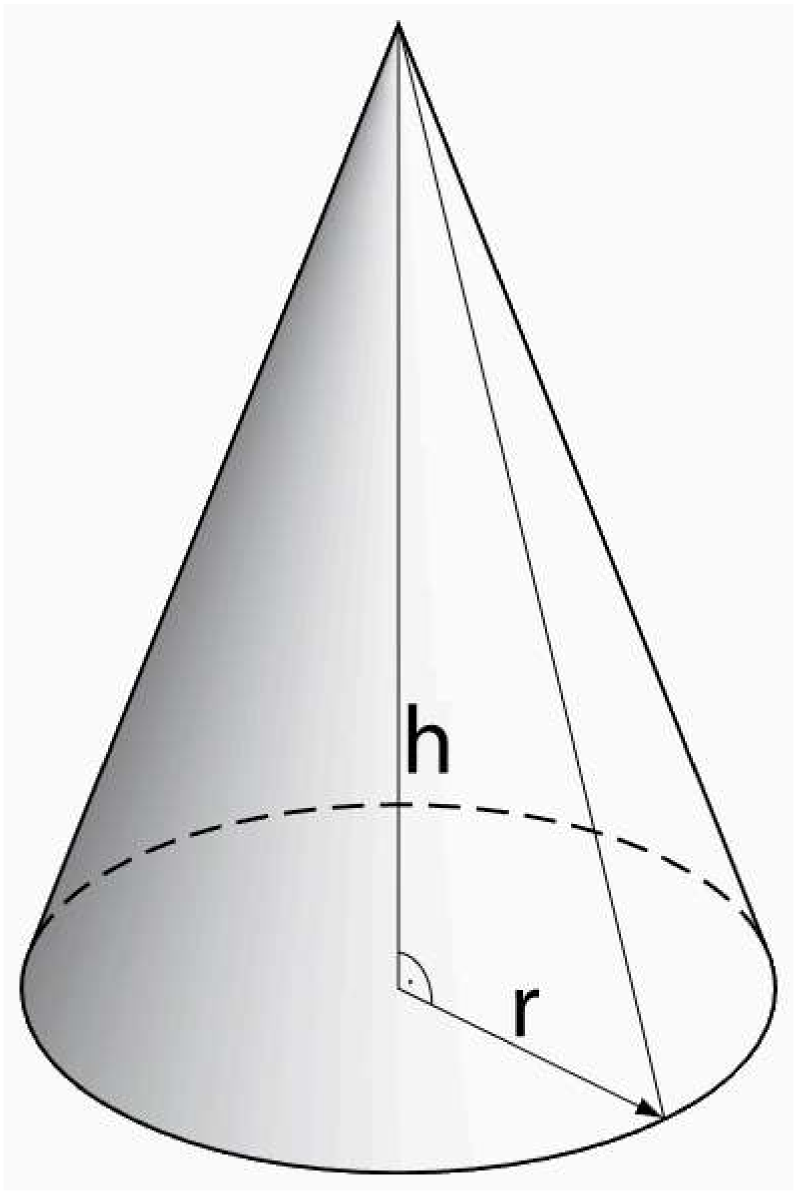
Purpose

Check your understanding of parameters, arguments, and local variable scope by fixing a program that won’t run because the programmer tried to use variables that were defined in a different function.

Helpful Documentation

- The preparation content this week's first prepare exercise explains what a parameter is.
- The preparation content for lesson 2 explains what an argument is.
- The preparation content for this lesson contains an example program with a mistake similar to the mistake that is in the example program below.

Problem Statement



A right circular cone with radius *r* and height *h*

The following example Python program is supposed to ask the user for the radius and height of a right circular cone and then compute and print the volume of that cone. The code is almost correct, but it contains a common mistake. The programmer who wrote it assumed that the **`cone_volume`** function could use the *radius* and *height* variables that are defined in the **`main`** function. Of course, this assumption is incorrect because as the preparation content for this lesson explains, variables defined inside a function have local scope and can be used only inside their containing function.

```python
"""Compute and print the volume of a right circular cone."""
# Import the standard math module so that
# math.pi can be used in this program.
import math
def main():
    # Call the cone_volume function to compute
    # the volume of an example cone.
    ex_radius = 2.8
    ex_height = 3.2
    ex_vol = cone_volume()
    # Print several lines that describe this program.
    print("This program computes the volume of a right")
    print("circular cone. For example, if the radius of a")
    print(f"cone is {ex_radius} and the height is {ex_height}")
    print(f"then the volume is {ex_vol:.1f}")
    print()
    # Get the radius and height of the cone from the user.
    radius = float(input("Please enter the radius of the cone: "))
    height = float(input("Please enter the height of the cone: "))
    # Call the cone_volume function to compute the volume
    # for the radius and height that came from the user.
    vol = cone_volume()
    # Print the radius, height, and
    # volume for the user to see.
    print(f"Radius: {radius}")
    print(f"Height: {height}")
    print(f"Volume: {vol:.1f}")
def cone_volume():
    """Compute and return the volume of a right circular cone."""
    volume = math.pi * radius**2 * height / 3
    return volume
# Start this program by
# calling the main function.
main()
```

Assignment

Do the following:

1. Using VS Code, open a new file. Copy and paste all of the example program above into the new file. Save the new file as **`cone_volume.py`**

2. Run your **`cone_volume.py`** program. What error message do you see in the terminal frame?
3. Fix the **`cone_volume`** function which begins on line 28 by adding two parameters to its header. What parameter names should you use?

   Hint: the parameter names at line 28 and the variable names at line 30 must be the same.
4. Fix the call to the **`cone_volume`** function that is in **`main`** at line 10 by adding two arguments. Because the **`cone_volume`** function has two parameters (you added the two parameters in the previous step), each call to the **`cone_volume`** function must have two arguments. What variable names should you use for the arguments?

   Hint: the only variables that you can use must be defined inside **`main`** above line 10.
5. Fix the call to the **`cone_volume`** function that is in **`main`** at line 28 by adding two arguments. What variable names should you use for the arguments?

   Hint: the only variables that you can use must be defined inside **`main`** above line 22.
6. Save your program and run it again. Did it run correctly? If not, fix the mistakes until it runs correctly.

Testing Procedure

Verify that your program works correctly by following each step in this testing procedure:

1. After you finish the steps of this assignmnet, run your program and enter the inputs shown below. Ensure that your program's output matches the output below.

```
> python cone_volume.py
    This program computes the volume of a right circular cone.
    For example, if the radius of a cone is 2.8 and
    the height is 3.2, then the volume is 26.3
    Please enter the radius of the cone: 5
    Please enter the height of the cone: 8.2
    Radius: 5.0
    Height: 8.2
    Volume: 214.7
```

Sample Solution

When your program is finished, view the sample solution for this assignment to compare your solution to that one. Before looking at the sample solution, you should work to complete this checkpoint program. However, if you have worked on it for at least an hour and are still having problems, feel free to use the sample solution to help you finish your program.

Ponder

The original program that you copied and pasted to start this assignment didn't work because the programmer tried to use the *radius* and *height* variables inside the **`cone_volume`** function. However, the *radius* and *height* variables are defined inside the **`main`** function and therefore have local scope and cannot be used outside of **`main`**. To fix the program, you added two parameters to the **`cone_volume`** function and then added two arguments to each call of the **`cone_volume`** function.

Why was local variable scope invented by computer scientists? How does local variable scope make a program easier to write and understand?

Submission

When complete, report your progress in the associated Canvas quiz.

Useful Links:

- Return to: Week Overview | Course Home