Physics Without Frontiers

Practical Programming
in Python

Inspired by 'Practical Programming' by Paul Gries, Jennifer Campbell, Jason Montojo

Lecture 3: Summary & Exercises
Using & Designing Functions

What are functions?, Python Built-in Functions, Local Variables, Designing Functions

> *"No amount of genius can overcome obsession with detail."*
>
> – Traditional

Lecture 3: Summary

In this lecture you learned the following:

- A function definition introduces a new variable that refers to a function object. The `return` statement describes the value that will be produced as a result of the function when this function is done being executed.
- A parameter is a variable that appears between the parentheses of a function header.
- A local variable is a variable that is used in a function definition to store an intermediate result in order to make code easier to write and read.
- A function call tells Python to execute a function.
- An argument is an expression that appears between the parentheses of a function call. The value that is produced when Python evaluates the expression is assigned to the corresponding parameter.
- If you made assumptions about the values of parameters or you know that your function won't work with particular values, write a precondition to warn other programmers.

Lecture 3: Exercises

When writing code, only use Python concepts that have been introduced in the lectures already.

Exercise 1:

Two of Python's built-in functions are `min` and `max`. In the Python shell, execute the following function calls:

a. `min(2, 3, 4)`

b. `max(2, -3, 4, 7, -5)`

c. `max(2, -3, min(4, 7), -5)`

Exercise 2:

For the following function calls, in what order are the subexpressions evaluated?

a. `min(max(3, 4), abs(-5a))`

b. `abs(min(4, 6, max(2, 8)))`

c. `round(max(5.572, 3.258), abs(-2))`

Exercise 3:

Following the function design recipe, define a function that has one parameter, a number, and returns that number tripled.

Exercise 4:

Following the function design recipe, define a function that has two parameters, both of which are numbers, and returns the absolute value of the difference of the two. Hint: Call built-in function `abs`.

Exercise 5:

Following the function design recipe, define a function that has one parameter, a distance in kilometers, and returns the distance in miles. (There are 1.60934 kilometers per mile.)

Exercise 6:

Following the function design recipe, define a function that has three parameters, grades between 0 and 100 inclusive, and returns the average of those grades.

Exercise 7:

Following the function design recipe, define a function that has four parameters, all of them grades between 0 and 100 inclusive, and returns the average of the *best 3* of those grades. Hint: Call the function that you defined in the previous exercise.

Exercise 8:

Complete the examples in the docstring and then write the body of the following function:

```
def weeks_elapsed(day1, day2):
    """
    Return the number of full weeks between two days.

    Day1 and day2 are days in the same year.

    Examples:

        >>> weeks_elapsed(3, 20)
        2
        >>> weeks_elapsed(20, 3)
        2
        >>> weeks_elapsed(8, 5)

        >>> weeks_elapsed(40, 61)

    """
```

Exercise 9:

Consider this code:

```
def square(n):
    """
    Return the square of n.

    Examples:

        >>> square(3)
        9
    """
```

In the table below, fill in the Example column by writing `square`, `num`, `square(3)`, and `3` next to the appropriate description.

Description	Example
Parameter	
Argument	
Function name	
Function call	

Exercise 10:

Write the body of the `square` function from the previous exercise.