CSI 1401 Week 8 Resources

by Allen Yan

This week, CSI 1401 will introduce user-defined functions in chapter 6. Many of you have already witnessed the convenience offered by Python's vast range of standard library functions. User-defined functions will allow you to give Python custom-made functions not included in the standard library.

As a reminder, I will be leading a group tutoring session on CSI 1401 every Wednesday from 7:00 pm to 8:00 pm (central time). The session will be conducted online via Microsoft Meetings, where I will be available to provide interactive help to students. If the above time window does not work for you, or if you need additional help, Baylor's Tutoring Center also provides 1-on-1 online tutoring appointments. For more information on signing up for group tutoring or individual tutoring, please visit https://www.baylor.edu/tutoring.

Week 8 Important Topics

Chapter 6 – Functions

- Function basics
- Reason for defining functions

1. Function basics

A function is a named block of code. In Python, a function definition consists of a *header* and a *body*. The header of the function includes the name of the function and its parameters, and the body of the function contains all of the code that would be run when the function is called. See below for an example of a function definition:

```
def function_name(parameter_1,parameter_2):  # function header
    return parameter_1+parameter_2  # function body
```

Function *parameters* are the variables defined in a function header that will come from outside of the function when the function is called. When a function is called, the user will provide the function parameters with values called *arguments* to work with. All defined parameters must be provided with an argument during function call in order for the function to execute properly. A function can have one or multiple parameters.

Functions have the option to return some value at the end of its execution. The returned value can then be used for other computations outside the function. For example, Python's standard `len()` function returns the length of the argument passed to the function. To return a value in user-defined function, simply use the `return` keyword followed by the desired variable to return. A function can have any number of return statements, which allows for the option to return different things depending on how the function runs. But keep in mind that during any particular function

call, only one return statement will be executed. Because **the function will end as soon as a return statement is reached**. For this reason, always make sure when defining a function that all of the work in the function is finished before calling the return statement. See the example below for a demonstration:

```
def my_func(my_list):  # bad function
    return sum(my_list)  # returns here
    for elem in my_list:  # this printing loop will never be
reached
        print(elem)


def my_func2(my_list):  # good function
    for elem in my_list:  # printing before returning fixes the
problem
        print(elem)
    return sum(my_list)  # returns here


nums = [1, 2, 3, 4, 5]
print(my_func(nums))  # prints only the sum of nums
print(my_func2(nums))  # prints the contents of nums, followed by
the sum of nums
```

Lastly, keep in mind that each function call can only return one variable. If you need your function to return multiple variables, you can package them into a container, such as a list or a dictionary, then return the container as a single variable. We will discuss more about returning multiple variables next week.

2. Reasons for defining functions

Functions are essentially just blocks of code with names. Many people, especially in the early parts of this class when the programs are relatively simple, would ask: "Why bother with defining functions if I can just write them in my main program and get the same thing?" While this is mostly true: in most beginner programs, writing code inside the main program instead of defining functions indeed has no impact on what the program does. (If you are curious, one example where user-defined function cannot be easily replicated by main program code is recursion, an advanced programming technique involving a function recursively calling itself.) However, there are still some very strong motivations for defining functions.

The first reason is that defining functions *increases code readability*. By separating out sections of code with a distinct purpose into a function, the code becomes much easier to read for both the programmer him/herself and other people, which is highly advantageous for debugging or additional code development. A second reason is that the use of functions allow programmers to do *modular development* on their code. Modular development allows the programmer to divide large, complicated programs into separate modules that can be developed and tested individually, before integrating them into one whole program. Modular development is very important for advanced programming because usually the entire program will be far too large to debug efficiently at that point. Additionally, user-defined functions could potentially save a lot of space by reducing

redundant code, especially if the program features high amounts of repeated code with small variations. See below for an example where using functions saves significant space.

```
pi_val = 3.14159265

diameter_1 = 12.0
circle_radius_1 = diameter_1 / 2.0
circle_area_1 = pi_val * circle_radius_1 * circle_radius_1

diameter_2 = 14.0
circle_radius_2 = diameter_2 / 2.0
circle_area_2 = pi_val * circle_radius_2 * circle_radius_2

diameter_3 = 8.0
circle_radius_3 = diameter_3 / 2.0
circle_area_3 = pi_val * circle_radius_3 * circle_radius_3

diameter_4 = 9.0
circle_radius_4 = diameter_4 / 2.0
circle_area_4 = pi_val * circle_radius_4 * circle_radius_4

total_area = circle_area_1 + circle_area_2 + circle_area_3 +
circle_area_4

print('Total area is', end=' ')
print('{:.2f}'.format(total_area), end=' ')
print('inches squared combined.')
```

The original code repeatedly tries to calculate the area of different circles, generating lots of redundant code. Using user-defined function greatly shortens it:

```
def find_area(diameter):
    pi_val = 3.14159265
    circle_radius = diameter / 2.0
    circle_area = pi_val * circle_radius * circle_radius
    return circle_area


diameter_1 = 12.0
diameter_2 = 14.0
diameter_3 = 8.0
diameter_4 = 9.0

total_area = find_area(diameter_1) + find_area(diameter_2) +
find_area(diameter_3) + find_area(diameter_4)

print('Total area is', end=' ')
print('{:.2f}'.format(total_area), end=' ')
print('inches squared combined.')
```

Useful Resources

- Basic Python Tutorial on GeeksforGeeks:
  https://www.geeksforgeeks.org/python-programming-language/
    - This page provides links to detailed explanations to many entry-level Python concepts along with examples. I encourage taking a look at it if the examples in your textbook were not clear enough.
- Official Python Documentation:
  https://docs.python.org/3/
    - This may be a bit heavy-handed for a beginner-level programmer since the official Python documentation is very thorough and technical. However, learning how to read official documentations is crucial to becoming a good programmer, because the official documentation contains information on everything you can find about Python elsewhere and more. Therefore, I encourage slowly easing yourself into learning how to read the official documentation.