Name: Andrew Wysocki
Date: 02/21/24
Course: Introduction to Programming - Python
https://github.com/jaspercasidino/IntroToProg-Python-Mod06

Assignment 06 - Functions

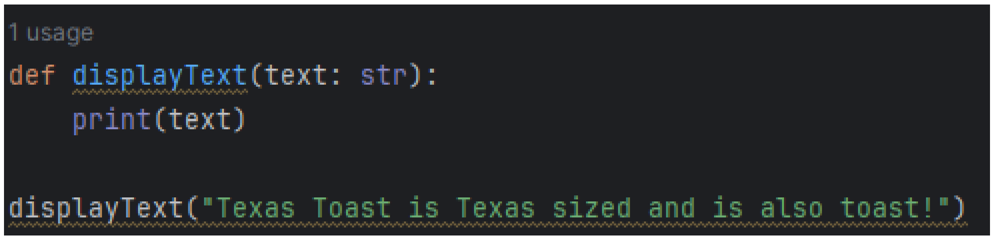
Introduction

This week's lesson covers Functions and Classes.

Topic - Functions

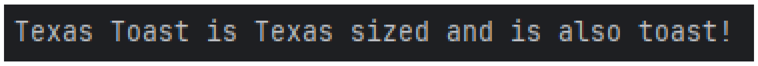
Functions are blocks of code that can be reused by simply calling out the name of the function within the main script. Functions are defined within the script itself and have a couple key elements. First you must define the function using the *def* identifier. Then you must give the function a name as this is how you will call out the function within the main script. After the name is a set of parentheses that may or may not contain parameters which can be passed as arguments when calling the function. You can use global parameters when using functions, but I prefer not to use global variables if I can help it. A function can also return a value which can be useful in data manipulation. Functions in Python operate very similarly to functions in C++.

Simple function example:



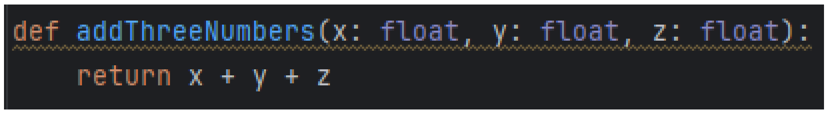
```
1 usage
def displayText(text: str):
    print(text)

displayText("Texas Toast is Texas sized and is also toast!")
```

Result:

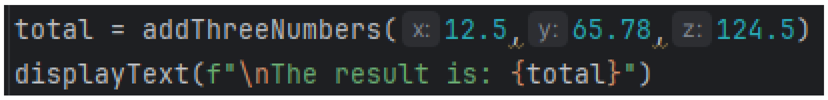


```
Texas Toast is Texas sized and is also toast!
```

Example of a simple function which returns a value:

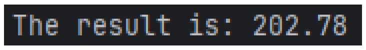


```
def addThreeNumbers(x: float, y: float, z: float):
    return x + y + z
```



```
total = addThreeNumbers( x: 12.5, y: 65.78, z: 124.5)
displayText(f"\nThe result is: {total}")
```

Result:



```
The result is: 202.78
```

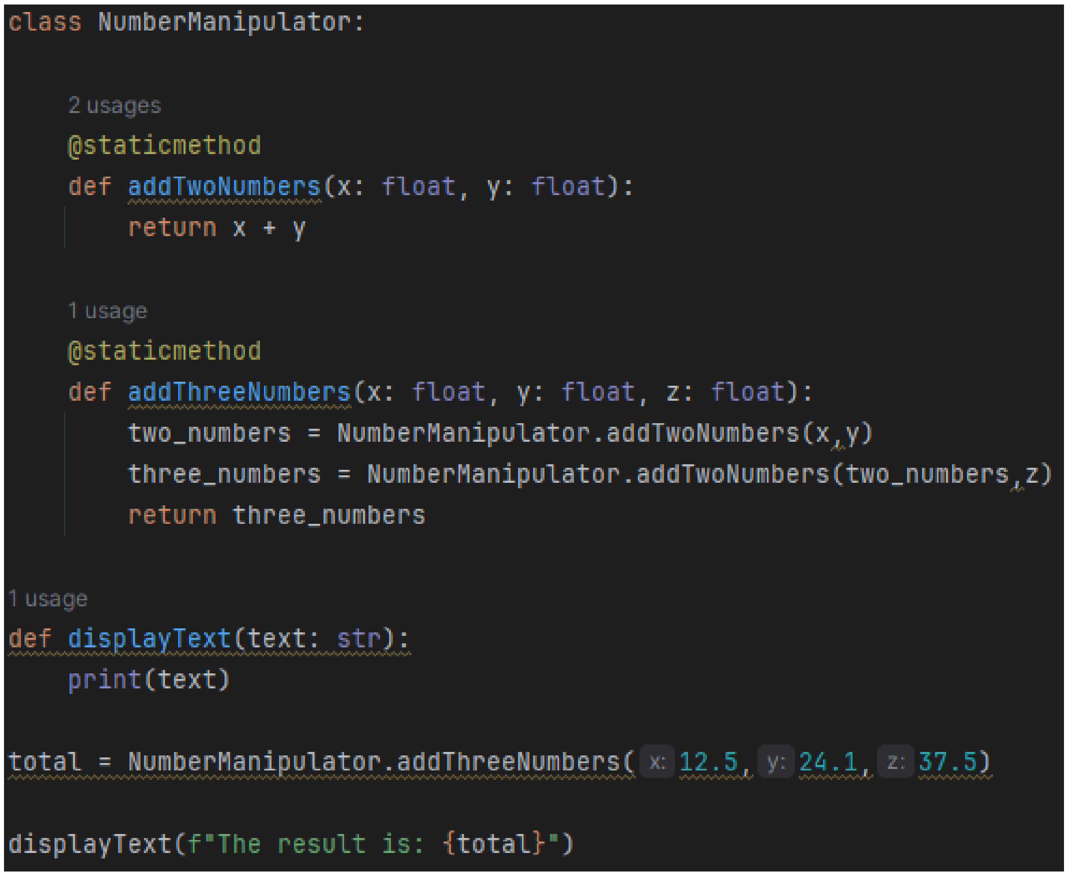
Topic - Classes

Classes are a way of grouping variables, constants and functions by collecting them under a common name and utilizing that name when trying to use these variables, constants and functions. Classes are a cornerstone of Object Oriented Program and can be powerful tools that help simplify reading and creating code through the power of grouping. In the module this week, we defined 2 types of classes: one for input/output manipulation of files and one for processing data.

To create a class, you simply write *class* and the name that you want for the class. Then under the class name, you can create functions or variables that will be utilized when working with the class. To define a function, you type *@staticmethod* then return followed by the function that you would like to define.

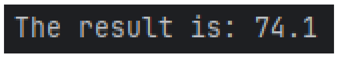
To call a function that is part of a class, you call out the class name followed by a dot followed by the function name (complete with arguments if the function requires it).

Example of a class called NumberManipulator with a couple simple addition functions:



```
class NumberManipulator:

    2 usages
    @staticmethod
    def addTwoNumbers(x: float, y: float):
        return x + y

    1 usage
    @staticmethod
    def addThreeNumbers(x: float, y: float, z: float):
        two_numbers = NumberManipulator.addTwoNumbers(x,y)
        three_numbers = NumberManipulator.addTwoNumbers(two_numbers,z)
        return three_numbers

1 usage
def displayText(text: str):
    print(text)

total = NumberManipulator.addThreeNumbers( x: 12.5, y: 24.1, z: 37.5)

displayText(f"The result is: {total}")
```

Result:



```
The result is: 74.1
```

Summary

This module covered the basic foundations of functions and classes. I am excited to see how we can expand the use of functions and classes, in particular when a class contains other classes in it such as a Student class that can contain some strings like first_name and last_name but also perhaps a list of classes that student has take which could contain a course_name and a letter_grade or semester. This is something that I learned in C++ programming that I found useful and powerful.