Going Beyond an Initial Python Script

Nick DeRobertis

March 3, 2021

1 Structuring a Complex Python Model

- There are two layers of organization we should have in our Python models
- The Python code itself should be well organized by using functions, which are self-contained logical units with inputs and outputs and optionally a description of what it does as well as the inputs and outputs
- We can use Jupyter features to provide additional structure on top of this. The sub-problems or sub-models can be divided into sections using Jupyter markdown. A table of contents can be provided to easily navigate through the sections and to give an overview of the structure of the model.
- The main inputs should be at the top of the notebook and the main outputs should be at the bottom of the notebook

2 Branching Logic with Python Conditionals

- Conditionals let you choose which logic to run based on some logical condition
- This is just like using Excel =IF, but more flexible as we can run any arbitrary operation rather than just returning a single value
- If/else if/else pattern becomes a lot more clear in Python as they are written as separate blocks rather than a nested =IF statement in Excel
- Logical conditions are always evaluated first to True or False. If True, goes into if part, if False, goes into else part (if included)
- Be careful about single = vs double ==, it is an easy mistake to make and you will get a SyntaxError applying the incorrect one
- Elif is never strictly necessary but can help simplify the code substantially
- The lab exercises test knowledge of conditionals but also build on our prior knowledge of for loops and show how they can be combined

3 Grouping Objects with Python Lists

- Lists are one of the basic container data types in Python. They hold other objects so we can work with them as a group
- Lists hold objects one by one in order and objects can be looked up from the list using the numeric index of the object
- It is a very common pattern to create an empty list, then go through some logic in a loop to create the object you want in the list and add it in each run of the loop

- This numeric index is zero-based, so look up the first object by 0, the second object by 1, and so on.
- We can pass a slice to get a group of objects out of the list as a new list or a single integer to get a single object out of the list
- Negative numbers means count from the end of the list, -1 is last object, -2 is second to last object, and so on
- Objects can be added to the list when it is created, but also later on using .append and .insert. Objects can be removed using .pop
- Lab exercise 1 tests list building for loop pattern, 2 tests adding objects to lists, and 3 tests list indexing and slicing

4 Grouping Logic with Python Functions

- Functions are the logical building blocks for any Python program, including our models
- Functions contain a piece of logic which then can be reused flexibly with different inputs
- We have functions in Excel, e.g. =SUM, =IF, =VLOOKUP, but as an Excel user you typically do not define functions. Python has built-in functions and also it is very easy to define your own.
- Ultimately, you should be able to run your entire model by running a single function
- The sub-models should also be functions, and the steps within the sub-models should also be functions. This can be broken into as many layers as is necessary. Your higher-level custom functions will be calling your lower-level custom functions.
- Not only will you have better readability and organization of your model, but also you will be able to reuse logic more often which lowers maintenance costs
- Functions scope the variables defined within. You cannot use those variables outside the function. This lets you use more general variable names which increases readabilityDefault arguments are a great way to keep your function flexible but also easy to use
- For these lab exercise, I have provided cells to check your work. After adding your solution and running the exercise cell, then run the check cell which will throw an AssertionError or NameError if incorrect and nothing if correct.
- The exercises test creating functions with and without default arguments and adding docstrings to functions

5 Python Basic Data Types

- Everything in Python is an object and every object has a type. The type defines the data structure and functions for that object
- A class defines a type (see next section)
- Strings are used to represent text and have a lot of convenience functions attached to them.
- We can make a "format" or "f" string by putting f in front of it. This allows us to place variables in the string and apply formatting to them.
- Most of the time you don't need to care whether a number is an integer (int) or floating point number (float). Probably the only place we will care about it in this class is looping a number of times requires an integer.
- range allows us to loop a certain number of times
- tuple: think list that you cannot change after creating. Better to use these when you know the objects you want in the group at the outset and you know they will not change.

- Dictionaries associate one object (value) with another object (key) and allow you to look up the value by the key
- Because everything in Python is an object, you can have any arbitrary nesting of objects. Lists of lists of strings, dictionaries where keys are integers and values are lists of strings, and so on, whatever is appropriate to solve your problem
- The lab exercises test working with tuples, dicts, and nested data structures

6 Creating Python Data Types with Classes

- We've talked about strings, floats, ints, lists, tuples, and dicts as basic data types. It is possible to define your own custom data types as well using classes
- I will not be expecting you to construct your own data types in this class. This section is mainly so you can understand how Python works in general. We will start using custom data types defined in external packages so it is good to have an understanding of this and how to use custom objects.
- The difference between objects of the same data type (same class definition) is the data contained within. They will all have the same functions attached to them (called methods when it is functions attached to objects)
- Feel free to study the car example in more detail to get a greater understanding of how to define classes, but we will not spend time on this
- Dataclasses are simplified versions of classes meant to group data together. We will use them to organize our model inputs.
- The lab exercises test working with externally-defined custom data types and creating and using dataclasses for our model inputs.

7 Handling Errors in Python

- Error handling is generally a more advanced pattern and so we won't put a lot of focus on it in the class. But sometimes it is the best solution to a problem.
- Combining lists of different lengths is one possible application of error handling, but there are many others