index

February 28, 2022

1 Object-Oriented Programming - Introduction

1.1 Introduction

In this section, you'll be introduced to the concept of Object-oriented programming (OOP) in Python. OOP has become a foundational practice in much of software development and programming, allowing developers to build upon each other's code in a fluent manner.

1.2 Object-oriented programming (OOP)

When programmers started writing code, originally they took a procedural approach. They'd write a series of sequential steps, using conditional statements and even the dreaded GOTO statements to branch their logic and their program flow. There would be a set of global variables (variables with values that anyone could change anywhere within the application) for keeping track of the "state" of the application.

Unfortunately, as they got bigger such programs were really hard to manage. Firstly, if they ever wanted to repeat the same business logic, they either had to copy it or loop back to where it was in the program, and secondly, it became really hard to keep track of where those variables got changed, making it really hard to reason about the program and to avoid small changes to the program breaking huge pieces of the application.

One solution was to break the common subroutines into separate "functions". This was a huge step forward. Now you could write a short script, and by calling it multiple times, passing various parameters you could reuse it safely. You could also write small automated tests (unit tests) to verify the behavior of those functions so you'd know how they would behave with various types of inputs. There is a whole branch of software development devoted just to functional programming and it can be a very effective way to write and reason about complex code bases. Languages such as Haskell and Clojure are primarily "Functional Programming languages". They are extremely powerful but have a reputation for being a little harder for developers to learn and use and because of that are less popular than languages that are primarily object-oriented.

Languages like Ruby and Python are often considered to be primarily "Object-oriented" (OO) programming languages. In part, OO programming came from the question "Where do we put our functions and our data?". One way to organize functions is in libraries. This is still done even in OO programming languages, so you might well have a library like `statsmodels` and use `import statsmodels.formula.api as smf` to get access to a series of functions related to formulae.

However, in OO programming languages, you also get the ability to create objects. Objects are a logical bundle of functions (they're called methods if they are associated to an object) and variables (often called properties when they're associated to an object), and for many classes of programming, once you get used to them, they provide a really useful abstraction.

So, you might have a `Person` class with its code saved in a file called `person.py` which describes both the properties of a person (height, weight, date of birth) and their behaviors (everything from their `full_name()` to their `current_age()`).

Python comes with a few basic built-in objects to get us started, things like `int` for integer, `str` for string, `list` for list, etc. We call these base types of objects "Primitives." Primitives already have methods we can call on them, for example: `.title()` for a string. But what if we wanted to create a new type of object in our programming universe, a new kind of object for our code? With Object-oriented programming, we can do just that by using the `class` keyword!

1.3 Classes and methods

In this section, you'll learn a lot about classes. A Python class can be thought of as the blueprint for creating a code object. These objects are known as an instance objects. Since nearly everything in Python is an object, understanding how to work with objects is critical to developing strong Python skills. You'll also learn about instance methods which are analogous to functions but are "bound" to instance objects. Don't worry if this sounds confusing right now, you'll have plenty of opportunities to explore classes and methods in this section.

1.4 Summary

Object-oriented programming (OOP) is a way of organizing your code that can make many types of applications easier to write by combining related variables/properties and functions/methods into objects containing both behavior and state.