5.4. Object Oriented Concepts

It's been fun drawing things with the turtles. In the process, we've slipped in some new concepts and terms. Let's pull them out and examine them a little more carefully.

5.4.1. User-defined Classes

First, just as Python provides a way to define new functions in your programs, it also provides a way to define new classes of objects. Later in the book you will learn how to define functions, and much later, new classes of objects. For now, you just need to understand how to use them.

5.4.2. Instances

Given a class like `Turtle` or `Screen`, we create a new instance with a syntax that looks like a function call, `Turtle()`. The Python interpreter figures out that Turtle is a class rather than a function, and so it creates a new instance of the class and returns it. Since the Turtle class was defined in a separate module, (confusingly, also named turtle), we had to refer to the class as turtle.Turtle. Thus, in the programs we wrote `turtle.Turtle()` to make a new turtle. We could also write `turtle.Screen()` to make a new window for our turtles to paint in.

5.4.3. Attributes

Each instance can have attributes, sometimes called **instance variables**. These are just like other variables in Python. We use assignment statements, with an =, to assign values to them. Thus, if alex and tess are variables bound to two instances of the class Turtle, we can assign values to an attribute, and we can look up those attributes. For example, the following code would print out 1100.

```
alex.price = 500
tess.price = 600
print(alex.price + tess.price)
```

5.4.4. Methods

Classes have associated **methods**, which are just a special kind of function. Consider the expression `alex.forward(50)` The interpreter first looks up alex and finds that it is an instance of the class Turtle. Then it looks up the attribute forward and finds that it is a method. Since there is a left parenthesis directly following, the interpreter invokes the method, passing 50 as a parameter.

The only difference between a method invocation and other function calls is that the object instance itself is also passed as a parameter. Thus `alex.forward(50)` moves alex, while `tess.forward(50)` moves tess.

Some of the methods of the Turtle class set attributes that affect the actions of other methods. For example, the method pensize changes the width of the drawing pen, and the color method changes the pen's color.

Methods return values, just as functions do. However, none of the methods of the Turtle class that you have used return useful values the way the `len` function does. Thus, it would not make sense to build a complex expression like `tess.forward(50) + 75`. It could make sense, however to put a complex expression inside the parentheses: `tess.forward(x + y)`

You have attempted 1 of 1 activities on this page

5.3. Instances: A Herd of Turtles">
tances: A Herd of Turtles">
5.5. Repetition with a For Loop">
✔ Completed. Well Done!

| Back to top