Kasia Rozanska

Dec. 15, 2020

Foundations of Programming: Python

Assignment08

https://github.com/KasiaPaloma/IntroToProg-Python-Mod08

Python Classes

# Introduction

*Object-oriented programming* (OOP) is programming with objects, which contain data and code. Python is an object oriented programming language and uses Classes as blueprints for creating objects. In this demonstration we will walk through the basics of creating a class.

## Class

As described by [GeeksforGeeks](https://www.geeksforgeeks.org/python-classes-and-objects/): “Class creates a user-defined data structure, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.” A class allows for one structure to be created, that is then reused by objects however many times you need.

A class and an object from a class will be stored in different parts of memory. The object is created using the class, but the class is loaded into memory only once. This way you have one class but multiple instances of the class, i.e. objects. In general you will use class directly to process data and indirectly to store data.

Below are the components that make up the standard pattern of a class:

|  |  |
| --- | --- |
| Classification (class) | This is the structure of the class and what you will use to refer to it. |
| Fields | The components, or data, that you can create a limitless amount of. Once you begin creating properties these will not be necessary. |
| Constructor | Sets the initial values of the class. The purpose of a constructor is to run automatically when you create an object from the class. |
| Attributes | These are like fields and can be flagged “private.” |
| Properties | Ways to manage fields and attributes. There are typically two, the “getter” and the “setter.” These are methods that will be used to complete any tasks. |

Sometimes data in a class is called a variable or attribute, but in general the term you want to learn is “field.” A function in class is a method, also known as a procedure.

## Program A Class

For our demonstration we will create the class “Product,” and just like with functions we will include a docstring. This will be helpful for future use and other developers.

class Product:  
 *"""Stores data about a product:  
  
 properties:  
 product\_name: (string) with the product's name  
 product\_price: (float) with the product's standard price  
 methods:  
 \_\_str\_\_: -> (string with product objects)  
 changelog: (When,Who,What)  
 RRoot,1.1.2030,Created Class  
 KRozanska,12.13.2020,Modified code to complete assignment 8  
 """*

Now for the constructor. This piece of code uses a method that already exists in Python, “\_\_init\_\_,” which will assign values to the data members of the class. You will also notice the use of “self.” You use "self" when referring to data or functions found in an object. Other languages use the keyword "this."

# -- Constructor --  
def \_\_init\_\_(self, product\_name, product\_price):  
 # -- Attributes --  
 self.\_\_product\_name = product\_name  
 self.\_\_product\_price = product\_price

Below you will get a clearer idea of how the Property components of a class act as the “getter” and “setter.” The @property gets the data (product\_name for this demo) and assigns how it will be returned. The @product\_name.setter interacts with the data and confirms that all the rules comply. You will find the product\_price properties in the full script on GitHub.

# -- Properties --  
# product\_name  
@property  
def product\_name(self):  
 return str(self.\_\_product\_name).title()  
  
@product\_name.setter  
def product\_name(self, value):  
 if str(value).isnumeric() == False:  
 self.\_\_product\_name = value  
 else:  
 raise Exception("Product name cannot have numbers in it.")

Lastly, we have one method. Methods are functions within the class that allow you to organize processes into named groups. Below we have an example of the most common method, the “\_\_str\_\_()” Method, which returns some of all of the class’s data as a string.

# -- Methods --  
def \_\_str\_\_(self):  
 return self.product\_name + ',' + self.product\_price

## Use A Class

Now we can use our class Product to complete tasks that we have programmed in past demonstrations. Below is an example of a function that we used to capture a user’s input. Notice that we are telling “p” to call our class and assign the input values to product\_name and product\_price. The reason for using a class is that now these two objects will reference the properties we already scripted and we don’t need to worry about exceptions and data types. We can still code the data types to make our work clear but the bulk of the work is happening in the class.

@staticmethod  
def input\_new\_product\_and\_price():  
 *""" Gets the product and price from user  
  
 :return: object  
 """* product = str(input("Which product would you like to add? - ")).strip().title()  
 price = float(input("What is the price? - "))  
 p = Product(product\_name=product, product\_price=price)  
 return p

# Summary

Follow the GitHub webpage to find the full code and see for yourself how all the pieces come together. There is still a lot about classes that I’m wrapping my head around so don’t stop here! OOP is incredibly important and this is where the magic of Python truly begins.