Sebastian Quesada

11/30/19

Foundations of Python

Assignment 08

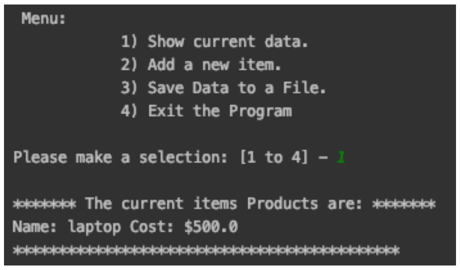
Objects and Classes

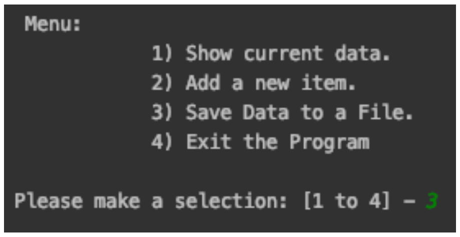
Introduction
This assignment demonstrates the usage and benefits of objects and classes. We will modify code to be able to show current data, add a new item, save data to the file and exit the program The program will ask for a user input and execute the prospective functions, classes and objects.

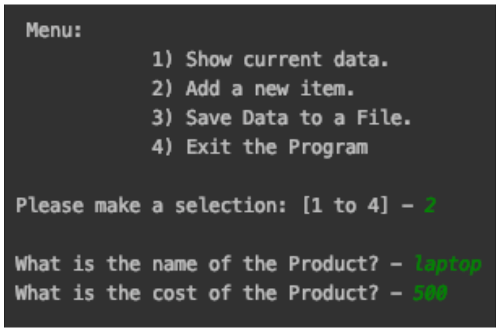
Knowledge Lessons Learned
Q1: What is the difference between a class and the objects made from a class?
A1: In Python, an object is a collection of Data (Variables) and methods (Functions) that act on those Data. The class is the blueprint for the object. The class can be thought of the blueprint of the house and details (attributes, properties) would be the floors, doors, windows, etc.
Q2:When do you use the keyword "self?"
A2:You use self when defining an instance method. It is passed automatically as the first parameter when you call a method on an instance. You also you use it when referencing a class or instance attribute from inside an instance method.
Q3: When do you use the keyword "@staticmethod?"
A3: @staticmethod is a function decorator. It can be called either on the class or on an instance. The instance is ignored except for its class. Self or cls is implicitly passed as the first argument.
Q4: How are fields and attributes and property functions related?
A4: Attributes are described by data variables for example like name age and hight. Property functions are a kind of attribute which have getter, setter and delete methods.
Q5: Why do you include a docstring in a class?
A5: They provide a convenient way of associating documentation with python modules, functions, classes and methods.
Q6: What is the difference between Git and GitHub?
A6:The key difference between Git and GitHub is that Git is an open-source tool developers install locally to manage source code, while GitHub is an online service to which developers who use Git can connect and upload or download resources.
Q7: What is GitHub Desktop?
A7: an open source desktop version for GitHub.

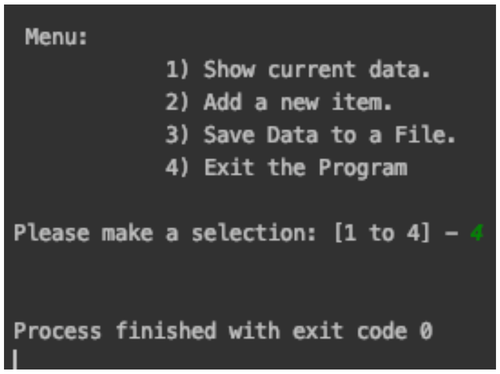
Application User Interface
The user is prompted with a menu with several options. In this figure on the top right, the user enters a laptop with a cost of $500. This information is stored in the computer's memory, which is displayed when needed, on the top left.

On the bottom left the user selects option 3 which is to save the selected data. To exit the application, the user selects option 4, shown on the bottom right.


```
 Menu:
           1) Show current data.
           2) Add a new item.
           3) Save Data to a File.
           4) Exit the Program

Please make a selection: [1 to 4] - 1

******* The current items Products are: *******
Name: laptop Cost: $500.0
**********************************************
```

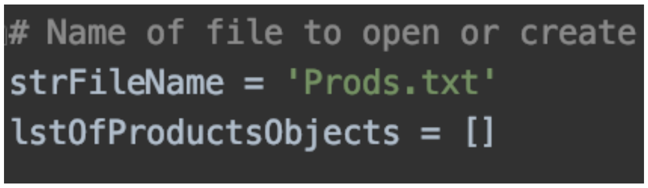

```
 Menu:
           1) Show current data.
           2) Add a new item.
           3) Save Data to a File.
           4) Exit the Program

Please make a selection: [1 to 4] - 3
```


```
 Menu:
           1) Show current data.
           2) Add a new item.
           3) Save Data to a File.
           4) Exit the Program

Please make a selection: [1 to 4] - 2

What is the name of the Product? - laptop
What is the cost of the Product? - 500
```

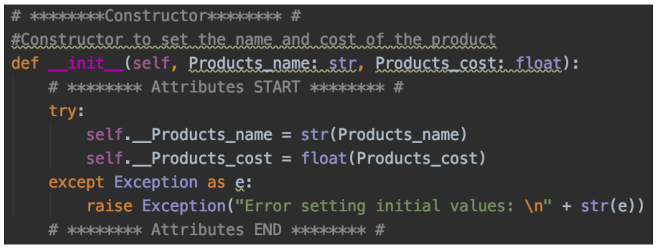

```
 Menu:
           1) Show current data.
           2) Add a new item.
           3) Save Data to a File.
           4) Exit the Program

Please make a selection: [1 to 4] - 4


Process finished with exit code 0
```

Application Code

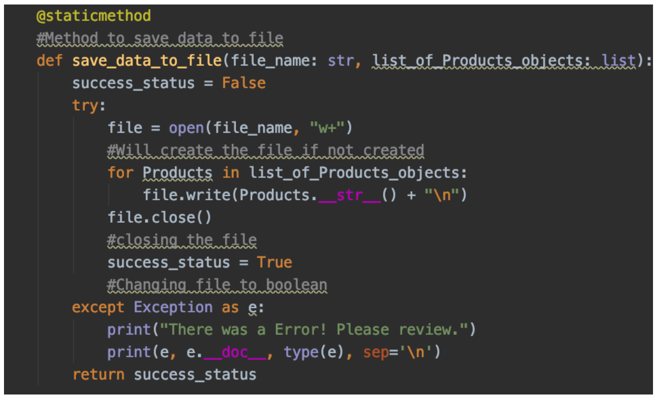
The objective in this application is to learn how to use objects and classes. The application opens a text file named Prods.txt. If the file does not exist it is created. The image below uses a constructor initializer named init that creates attributes. The properties have getters and setters. There are two static methods that save data to a file and read data to a. file.



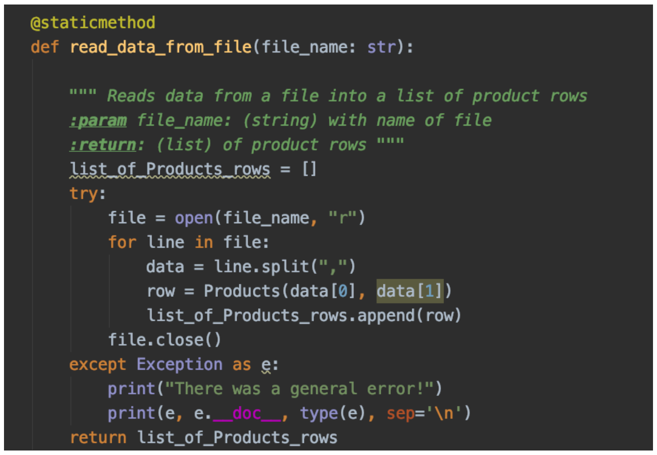
```python
# Name of file to open or create
strFileName = 'Prods.txt'
lstOfProductsObjects = []
```



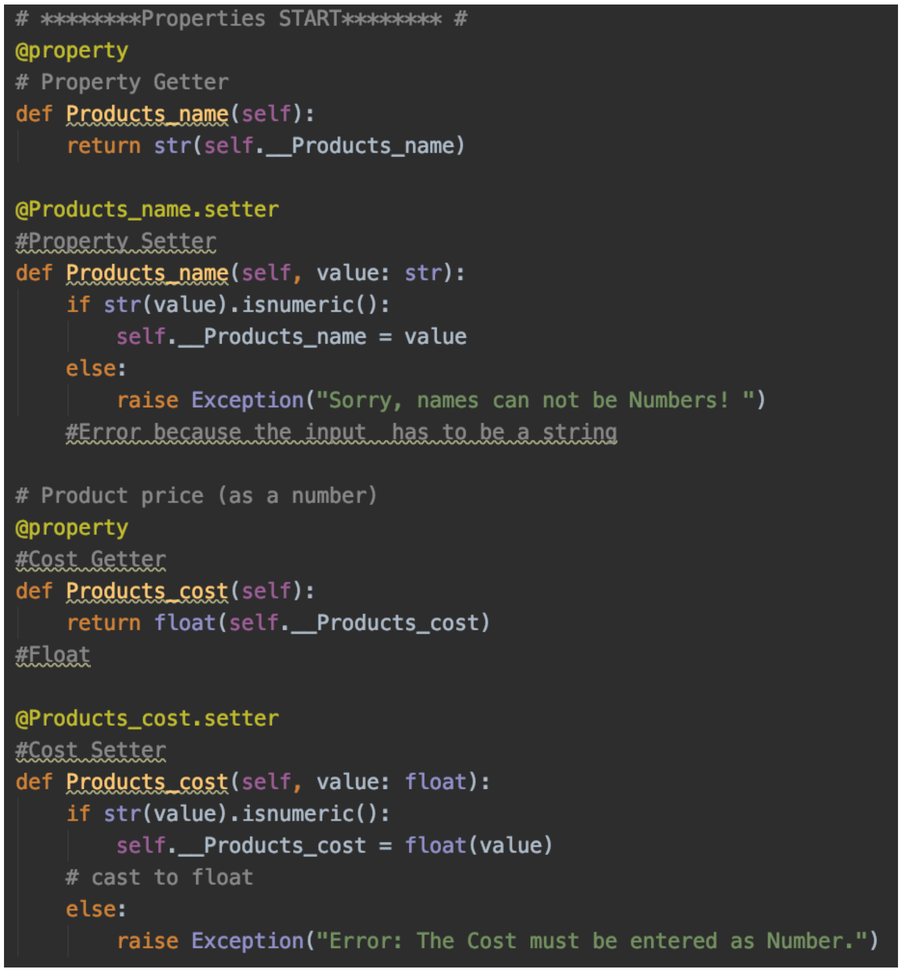
```python
# ********Constructor******** #
#Constructor to set the name and cost of the product
def __init__(self, Products_name: str, Products_cost: float):
    # ******** Attributes START ******** #
    try:
        self.__Products_name = str(Products_name)
        self.__Products_cost = float(Products_cost)
    except Exception as e:
        raise Exception("Error setting initial values: \n" + str(e))
    # ******** Attributes END ******** #
```



```python
@staticmethod
#Method to save data to file
def save_data_to_file(file_name: str, list_of_Products_objects: list):
    success_status = False
    try:
        file = open(file_name, "w+")
        #Will create the file if not created
        for Products in list_of_Products_objects:
            file.write(Products.__str__() + "\n")
        file.close()
        #closing the file
        success_status = True
        #Changing file to boolean
    except Exception as e:
        print("There was a Error! Please review.")
        print(e, e.__doc__, type(e), sep='\n')
    return success_status
```



```python
@staticmethod
def read_data_from_file(file_name: str):

    """ Reads data from a file into a list of product rows
    :param file_name: (string) with name of file
    :return: (list) of product rows """
    list_of_Products_rows = []
    try:
        file = open(file_name, "r")
        for line in file:
            data = line.split(",")
            row = Products(data[0], data[1])
            list_of_Products_rows.append(row)
        file.close()
    except Exception as e:
        print("There was a general error!")
        print(e, e.__doc__, type(e), sep='\n')
    return list_of_Products_rows
```



```python
# ********Properties START******** #
@property
# Property Getter
def Products_name(self):
    return str(self.__Products_name)

@Products_name.setter
#Property Setter
def Products_name(self, value: str):
    if str(value).isnumeric():
        self.__Products_name = value
    else:
        raise Exception("Sorry, names can not be Numbers! ")
    #Error because the input  has to be a string

# Product price (as a number)
@property
#Cost Getter
def Products_cost(self):
    return float(self.__Products_cost)
#Float

@Products_cost.setter
#Cost Setter
def Products_cost(self, value: float):
    if str(value).isnumeric():
        self.__Products_cost = float(value)
    # cast to float
    else:
        raise Exception("Error: The Cost must be entered as Number.")
```

Summary

This assignment demonstrated how objects are created in classes and how they can be used to open a file and save data. This could have endless possibilities. One thing to keep in mind is the natural objects that python has such as init and double underscore variables.