**VILNIUS UNIVERSITY**

**KAUNAS FACULTY**

****

Information Systems And Cyber Security

Python Programming

Practice Task 2 Report

Created By: Code Junkies

Akeem Ajibare Damola

Neba Forbacha Cheoh-Suh

Kaunas, 2023

Table of content

Task 1:..............................................................................................................3

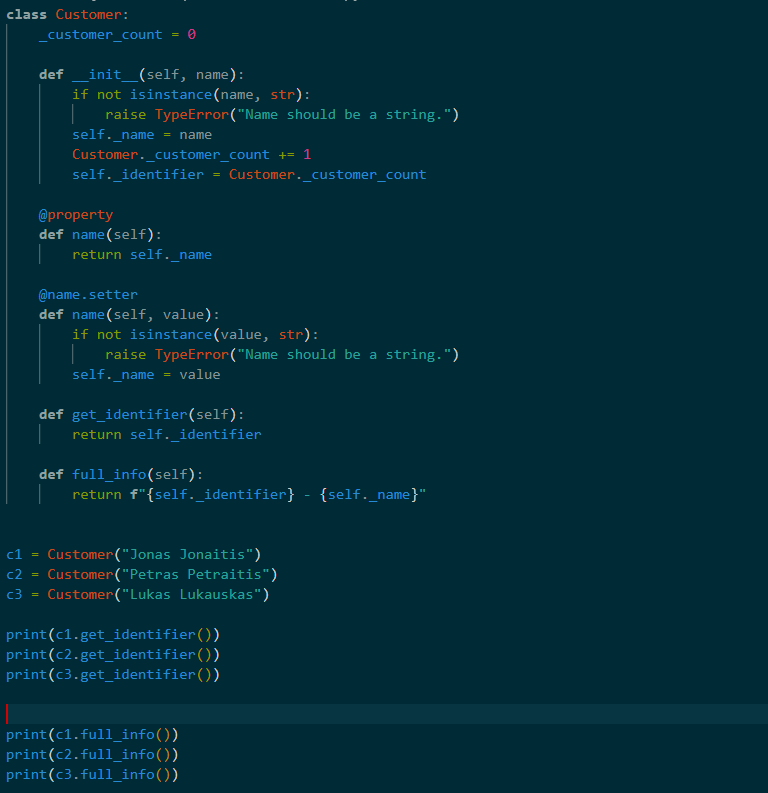
Task 2:..............................................................................................................4

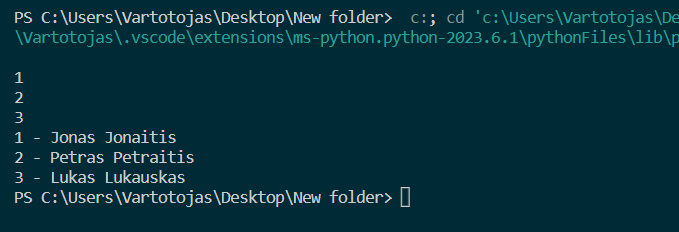
Task 3:..............................................................................................................5

Task 4:..............................................................................................................6

Task 5:..............................................................................................................7

#Task 1





The code defines a Python class named "Customer" that contains instance methods and protected class variables "\_customer\_count" for customers count. This variable is initialized to zero.

The "**init**" method is the constructor for the "Customer" class. It takes two arguments, "name" and "identifier". If the "identifier" argument is not provided, it assigns a unique identifier to the customer by incrementing the "\_customer\_count" variable and assigning its value to the "identifier" instance variable.

The "full\_info" method is a property method that returns a tuple of the customer's name and identifier. It is used to get the full information of the customer.

The "full\_info" method also has a "setter" method that allows changing the customer's name and identifier. The new name and identifier should be passed as a tuple of length 2. If the input tuple is not of length 2, it raises a ValueError.

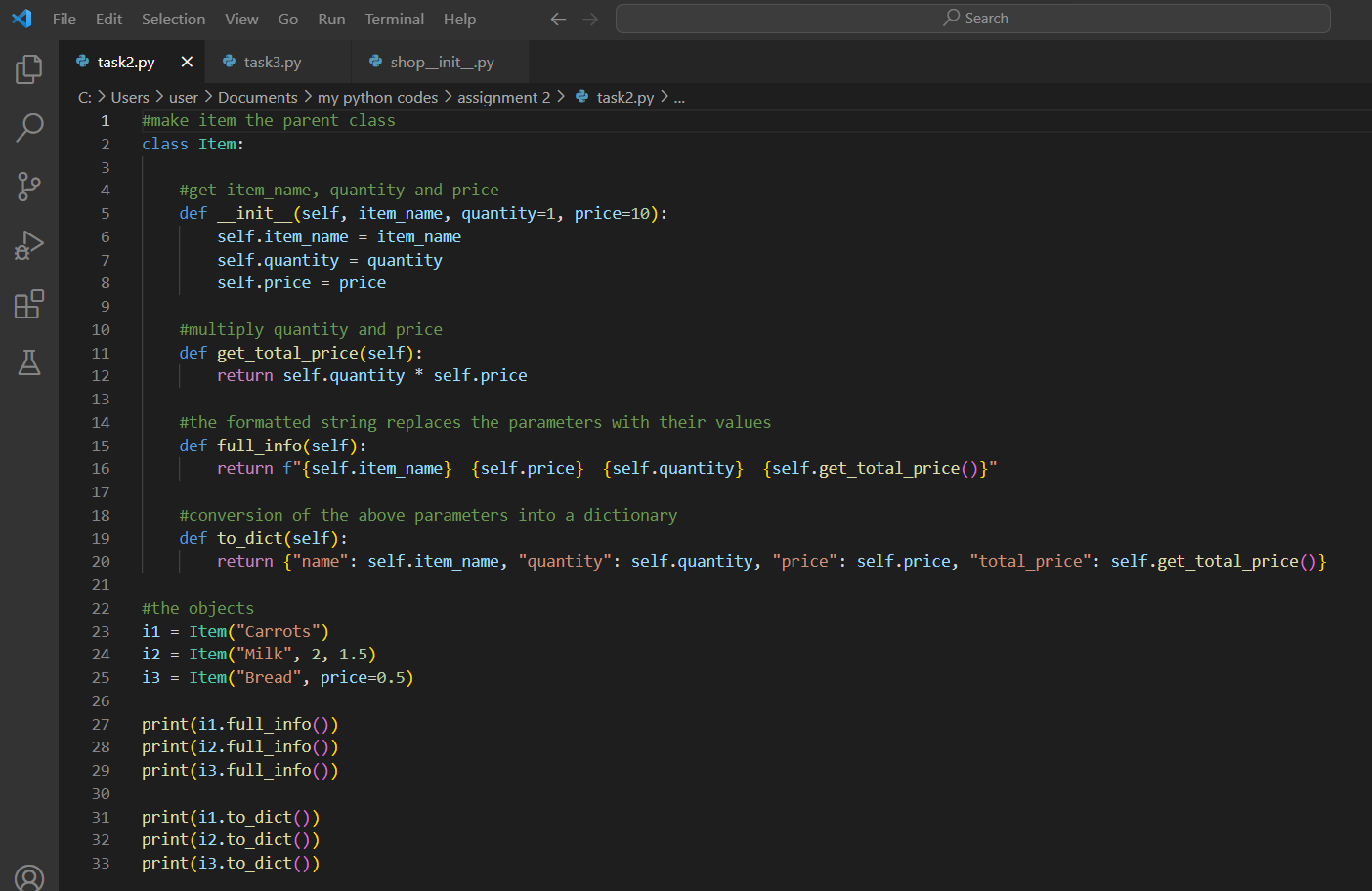
The "full\_info" method also has a "deleter" method that deletes the "name" and "identifier" instance variables.

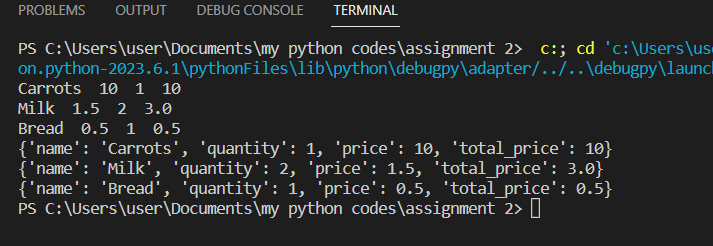
The "show" method is used to print the customer's name and identifier in a specific format.

In the second part of the code, three instances of the "Customer" class are created, and their information is printed using the "full\_info" and "show" methods.

Finally, the "full\_info" attribute of two instances is deleted using the "del" keyword, which raises an AttributeError because the "name" and "identifier" attributes are deleted along with it.

**task 2**

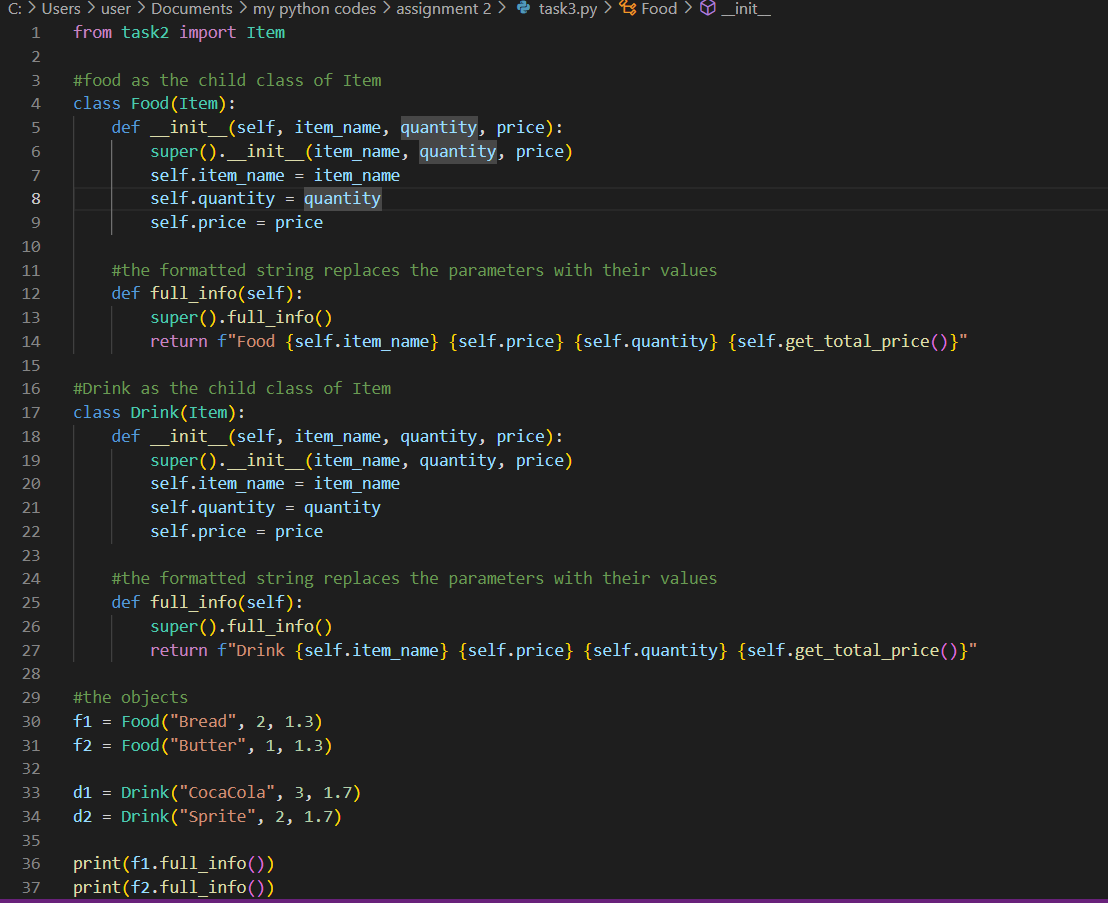


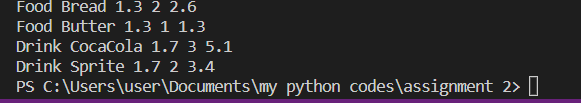
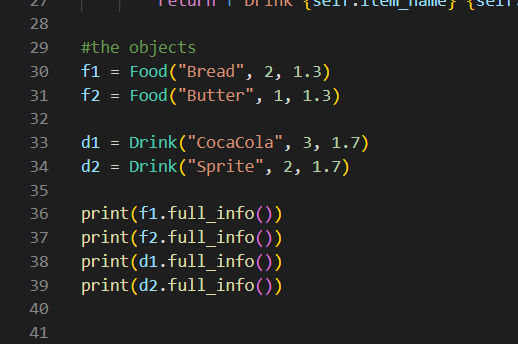


The above code defines a class called ‘Item’ which has three instance variables: ‘item\_name’, ‘quantity’, and ‘price’. The class has a constructor method (‘\_\_init\_\_’) that initializes these instance variables. The class also has two methods: ‘get\_total\_price’ which calculates the total price by multiplying the quantity and the price, ‘and full\_info’ which returns a formatted string of the item's name, price, quantity, and total price.

Additionally, the class has a method called ‘to\_dict’ which converts the item's attributes into a dictionary. Three objects of the ‘Item’ class are created: ‘i1’, ‘i2’, and ‘i3’ with different parameters passed in. Finally, the ‘full\_info’ and ‘to\_dict’ methods are called on each object to display their respective outputs.

#Task 3



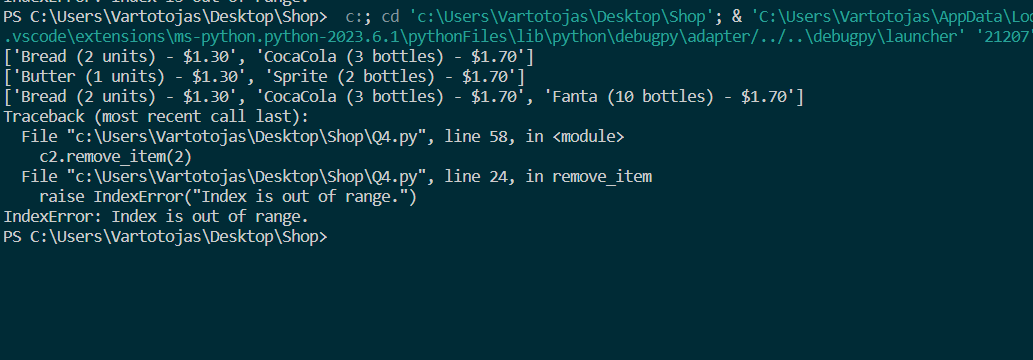
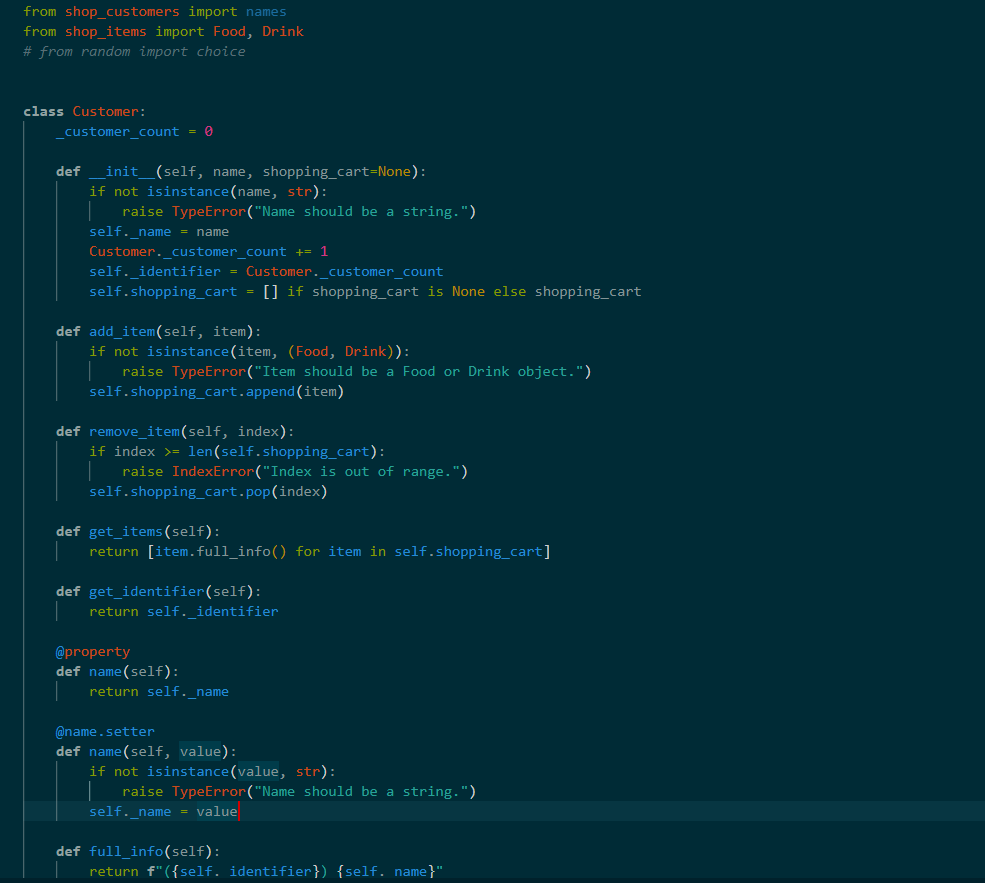


The above code extends the ‘Item’ class defined in ‘task2.py’ to create two child classes, ‘Food’ and ‘Drink’. Both child classes inherit the properties of the ‘Item’ class and have their own unique method called ‘full\_info’.

The ‘Food’ and 'Drink’ classes have a constructor method that calls the parent class constructor using ‘super()’, which initializes the instance variables ‘item\_name’, ‘quantity’, and ‘price’. The ‘full\_info’ method is overridden to include a prefix string before the output of the ‘Item’ class's ‘full\_info’ method.

Finally, four objects of the ‘Food’ and 'Drink’ classes are created, with different parameters passed in, and their ‘full\_info’ method is called to display their respective outputs. The output displays a formatted string that includes the prefix string, followed by the name, price, quantity, and total price of the item.

#Task 4



Now, the class accepts a shopping\_cart parameter in the constructor, which can be used to pass in a list of Food or Drink objects.

The code begins by importing the names list from the shop\_customers module, and the Food and Drink classes from the shop\_items module. Then it defines the Customer class.

The add\_item method adds a Food or Drink object to the customer's shopping cart. If the item is not a Food or Drink object, a TypeError is raised.

The remove\_item method removes an item from the shopping cart at the specified index. If the index is out of range, an IndexError is raised.

The get\_items method returns a list of strings representing the information of each item in the shopping cart. Each string is obtained by calling the full\_info method of the corresponding Food or Drink object.

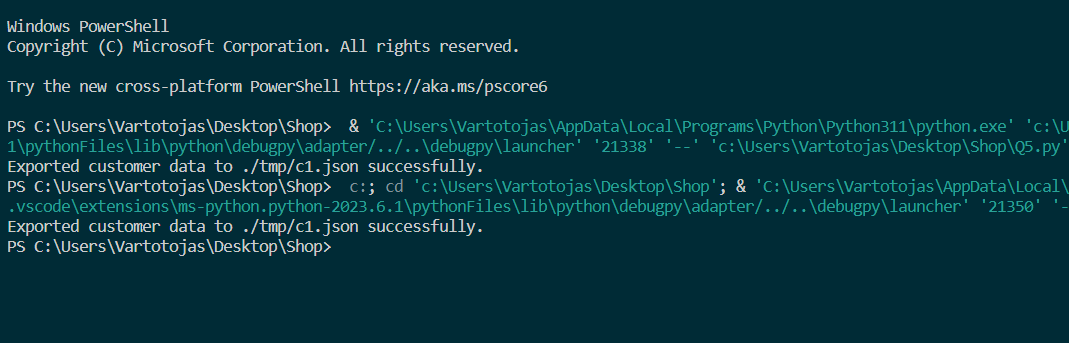
The get\_identifier method returns the unique identifier of the customer.

The name attribute is defined as a property, with a getter and a setter. If a non-string value is passed to the setter, a TypeError is raised.

The full\_info method returns a string representation of the customer object, including the unique identifier and the name.

The get\_identifier, name, and full\_info methods are unchanged from the previous version of the class.

#Task 5



Now, the "json" module is used to export data from the customer object "c1" into a JSON file. The method "export\_to\_json" of the "Customer" class does the conversion from the customer object to a JSON-formatted string and writes it to a file at the given file path.

The "json.dump()" method takes two arguments: the data to be written to the file, and the file object to which the data should be written. In this case, the data to be written is a dictionary that contains the customer's name, identifier, and a list of items. The items list is created by iterating over the customer's items and converting each item to a dictionary using the "**dict**" attribute. This attribute returns a dictionary containing all the attributes and their values for the given object.

The "indent" parameter is set to 4, which adds four spaces for each level of indentation in the output JSON file to make it more human-readable.

Finally, the "print()" statement confirms that the customer data has been successfully exported to the specified file path.