Networking for Software Developers

# Lab 2 – Shopper Class.

This is an individual lab. Using a jupyter notebook (best done on google colab) write the python statement to define a Shopper class. The test harness is provided to you by the instructor as well as the resulting output given.

## Description of the Shopper Class

### Class variable

There are six class variables i.e. variables that are shared by all objects of this class

**\_\_prices**

This is a python dict that contains the grocery items as well as their unit price. This is defined as follows:

\_\_prices = { 'apple': 1.8, 'bread': 2.2, 'milk': 4.9, 'pepper': 1.2 }

**\_\_sale\_items**

This is a list that contains all the items that are on sale in this period. A discount of 15% is applied to the price of each item in this collection. This is defined as follows:

\_\_sale\_items = 'apple banana'.split()

Four other class variable that are define as follows:

\_\_discount\_threshold = 6

When the total cost of a purchase exceeds this amount then a discount is applied to the cost

\_\_default\_price = 2.50

If an item is not found in the **\_\_prices** dictionary, then this will be the default price

\_\_volume\_discount = 0.9

This is the rate of discount when the total cost exceeds the **discount\_threshold**

\_\_sales\_discount = 0.85

This is the rate of discount on an item if this item is in the **\_\_sales\_time list**.

### Constructor

The constructor takes two argument that represents the name of the shopper and the amount of money she has. It assigns the argument to appropriate instance variable. It also creates an empty list to store the purchases.

Each purchase is a single tuple comprising of the name of the item and the price actually paid for the item.

### Instance property

There is a single instance property that returns the name of this shopper object

### Class method

There are three class methods:

1. One that returns the price list.
2. One that returns the list of sale items.
3. One that adds a new item to the sale items list.

### Instance method

There are two instance method:

**purchase**

This method takes a list of items to be purchase. It calculates the total price of the purchase by processing each item in the argument as show below:

The price of an item is obtained from the price list. If it is not found in the price list then the default price is used. *[How to determine if an item in not in the dict]*

If the item is in the sales list then a discount is applied to the price

The item as well as the result price is added to the list as a tuple. *[How to create a tuple with name and price and how to add the tuple to the list]*

When all the items are processed, if the final cost is over the credit threshold, then a discount is applied to this amount and then subtracted from the amount of money this object has left over.

**\_\_str\_\_**

This method returns a string representing this object i.e. the name, cash and all its purchases pertaining to this shopper.

### How to do this assignment.

From the above description and test harness and the result output below, try to deduce the definition of the Shopper class. Code this class in a jupyter notebook and copy the test harness to a cell below. Execute the notebook and ensure that the output matches EXACTLY with the output on the following page.

You must use python f-strings for your output.

## Documentation.

Because the code is so simple no code documentation is required, however you must put your name and the current date somewhere at the top of your code.

## How to submit this assignment.

Make the notebook shareable and submit the link to the course dropbox.

See the course documentation on deadlines.

# Test Harness

You may not change the test harness.

print(f'Price dict: {Shopper.price\_list()}')

print(f'Sales list: {Shopper.sale\_items()}')

nar = Shopper('Narendra', 20) #create a shopper object

print(f'\n{nar}') #display the object

items = 'bread milk bread'.split() #list of items to buy

print(f'\n{nar.name} is purchasing: {items}')

nar.purchase(items) #volume discount

print(f'{nar}') #display the object

ilia = Shopper('Ilia', 25) #new shopper

items = 'banana milk'.split() #one sale item

print(f'\n{ilia.name} is purchasing: {items}')

ilia.purchase(items)

print(f'{ilia}') #display the object

Shopper.add\_sale\_item('pepper') #add pepper to the sales items list

arben = Shopper('Arben', 15) #new shopper

items = 'apple pepper cauliflower pepper'.split() #cauliflower is not on the price list

print(f'\n{arben.name} is purchasing: {items}')

arben.purchase(items)

print(f'{arben}') #display the object

#you don't need to understand the code below

#it is for verification purposes

members = [member for member in dir(Shopper) if not member.startswith('\_')]

print(f'\nPublic members of the class: {members}')

properties = [member for member in members if not callable(getattr(Shopper, member))]

print(f'Public properties: {properties}')

methods = [member for member in members if callable(getattr(Shopper, member))]

print(f'Public methods: {methods}')

# Program Output

Your output must be identical to the below.

Price dict: {'apple': 1.8, 'bread': 2.2, 'milk': 4.9, 'pepper': 1.2}

Sales list: ['apple', 'banana', 'pepper']

Narendra cash in hand $20.00

items:

[]

Narendra is purchasing: ['bread', 'milk', 'bread']

Narendra cash in hand $11.63

items:

[('bread', 2.2), ('milk', 4.9), ('bread', 2.2)]

Ilia is purchasing: ['banana', 'milk']

Ilia cash in hand $18.68

items:

[('banana', 2.125), ('milk', 4.9)]

Arben is purchasing: ['apple', 'pepper', 'cauliflower', 'pepper']

Arben cash in hand $9.54

items:

[('apple', 1.53), ('pepper', 1.02), ('cauliflower', 2.5), ('pepper', 1.02)]

Public members of the class: ['add\_sale\_item', 'name', 'price\_list', 'purchase', 'sale\_items']

Public properties: ['name']

Public methods: ['add\_sale\_item', 'price\_list', 'purchase', 'sale\_items']