ICS Spring 2017

Problem Set 6

Exercise 1 – Bank Account(OOP)

Jeff is planning to start up a local bank. He plans to offer simple checking and savings account services, and needs some help designing a simple system to manage a customer's account. If the account type is "checking", the annual interest rate is 0.08; if the type is "saving", the rate is 0.04, respectively.

Customer Class have:

- A Name
- An Account Class

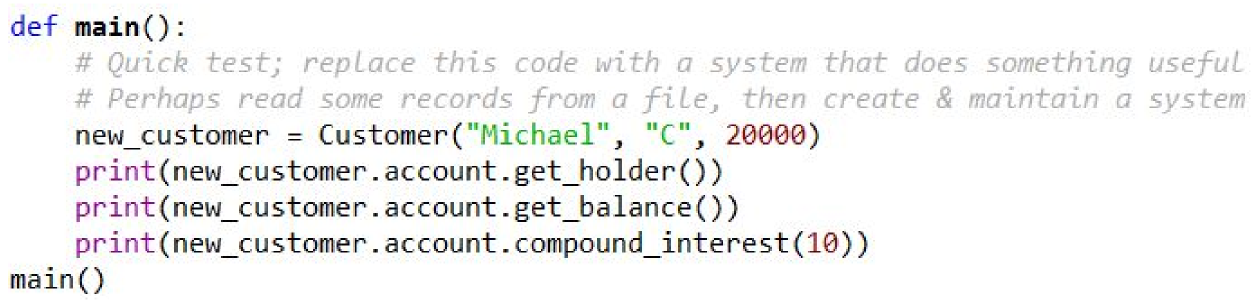
Account Class have:

- An account holder
- An account number
- An account type [Checkings / Saving]
- Balance

Methods you need to create:

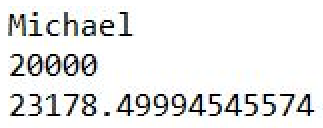
- Getters and setters for account holder & balance
- Calculator for simple interest after x years
- Calculator for compound interest after x year

Example:



```python
def main():
    # Quick test; replace this code with a system that does something useful
    # Perhaps read some records from a file, then create & maintain a system
    new_customer = Customer("Michael", "C", 20000)
    print(new_customer.account.get_holder())
    print(new_customer.account.get_balance())
    print(new_customer.account.compound_interest(10))
main()
```

And the output will be:



```
Michael
20000
23178.49994545574
```

Exercise 2 – `RetailItem` Class(OOP)

Write a class named `RetailItem` that holds data about an item in a retail store. The class should store the following data in attributes: item description, units in inventory, and price.

Once you have written the class, write a program that creates three `RetailItem` objects and stores the following data in them:

	Description	Units in Inventory	Price
Item #1	Jacket	12	59.95
Item #2	Designer Jeans	40	34.95
Item #3	Shirt	20	24.95

Exercise 3 – Cash Register(OOP)

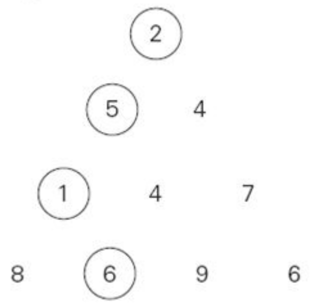
This exercise assumes that you have created the `RetailItem` class in Exercise 2. Create a `CashRegister` class that can be used with the `RetailItem` class. The CashRegister class should be able to internally keep a list of `RetailItem` objects. The class should have the following methods:

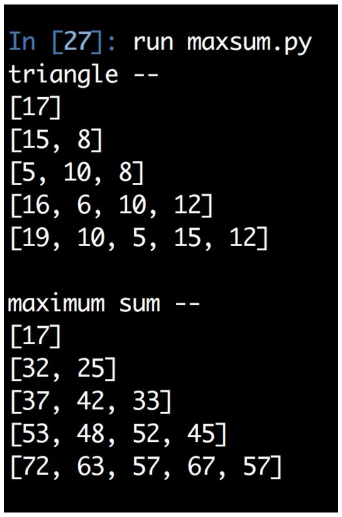
- A method named `purchase_item` that accepts a `RetailItem` object as an argument. Each time the `purchase_item` method is called, the `RetailItem` object that is passed as an argument should be added to the list.
- A method named `get_total` that returns the total price of all the `RetailItem` objects stored in the `CashRegister` object's internal list.
- A method named `show_items` that displays data about the `RetailItem` objects stored in the `CashRegister` object's internal list.
- A method named `clear` that should clear the `CashRegister` object's internal list.

Demonstrate the `CashRegister` class in a program that allows the user to select several items for purchase. When the user is ready to check out, the program should display a list of all the items he or she has selected for purchase, as well as the total price

Exercise 4 – Maximum Sum Descent in OOP Style

- Positive integers in a triangle
- **Goal**: a descent from the root to the base, with the largest sum.





```
In [27]: run maxsum.py
triangle --
[17]
[15, 8]
[5, 10, 8]
[16, 6, 10, 12]
[19, 10, 5, 15, 12]

maximum sum --
[17]
[32, 25]
[37, 42, 33]
[53, 48, 52, 45]
[72, 63, 57, 67, 57]
```

Bonus: Alternative Implementation of Pancake Sorting

Approach:

- move down level by level
    - if the current level has smaller size:
        - flip above
        - and then flip this pancake up to the top
        - then flip to its proper location

Feel free to use the starting code and modify it from the in-class pancake sorting.