Introduction to Data Science and Programming

Assignment 1

Due September 15, 2020, 23:59

Description

In this week's assignment, you will implement some functionalities for a hypothetical bank, that build and compute products of the transactions of a bank account. Each function specification is described in a listing below. Write all your code into the provided *single* library file `bank.py`. We've provided the signatures of functions, together with examples of input and output, below each exercise.

Submission

- Submit the python file `bank.py` containing your solutions.
- Your function calls must reside inside the `main` function.
- Make sure that we are able to run the python file by running the following command: *python bank.py*.
- Also, make sure that we can import your bank library without automatically running the `main` function.
- Use only the resources (i.e. imports) you learned during the classes!

Assignment

1. Create a function called `calculate_balance` that takes a list of transactions, sums the transactions and returns a balance.

   ```
   >>> transactions = [100, 5000, -30, -1200]
   >>> calculate_balance(transactions)
   3870
   ```

2. Implement a function called `calculate_compound` that takes an account balance, rate, and number of years and returns the balance after a compounding period of years with the specified rate.

   ```
   >>> calculate_compound(500, 0.05, 25)
   1693.177470449694
   ```

   Note: There are only two decimal places (do not use string formatting).

3. The bank wants a way to list only transactions that are negative, called *withdrawals*. Implement a function called `filter_withdrawals` that takes a list of transactions and returns all withdrawals.

   ```
   >>> filter_withdrawals(transactions)
   [-30, -1200]
   ```

4. The risk department at the bank must automatically examine millions of transactions each day for potential threats to clients and to the bank as soon as possible. For example, debit card theft can cause sizeable damage within a short amount of time. For this bank, they are only interested in one type of potential threat: very large withdrawals. Write a function called `risk_analysis` that finds and returns the largest withdrawal among the transaction.

```
>>> transactions = [-5000, 200, 120, -99999]
>>> risk_analysis(transactions)
-99999
```

5. The bank has provided you with a list of transaction records that contain the names of those who either withdrew or deposited the transaction. Implement a function called `join_records` that takes a list of names transactions and returns a list of tuples.

```
>>> names = ["Muhammed", "emma", "Emma", "julie"]
>>> join_records(names, transactions)
[('Muhammed', -5000), ('emma', 200), ('Emma', 120), ('julie', -99999)]
```

6. The bank has agreed to help out Skat, the danish tax agency, by providing them with a list of all those who are making deposits (positive transactions) to their accounts. Write a function named `unique_deposit_names` that takes the list of transaction tuples that we just built, and returns all unique lower-cased names.

```
joined_transactions = [('Muhammed', -5000), ('emma', 200), ('Emma', 120), ('julie', -99999)]
>>> unique_deposit_names(joined_transactions)
['emma']
```