Project 1 – Budget Tracker

Waqas Ali

Instructions: (How to use and test the project)

- Run the following commands in Python terminal and respond to the prompts. (Page 2 has detailed instructions).

```
>>> from Budget_Tracker import budget_manager
>>> Deloite = budget_manager()
>>> Deloite.get_accounts()
>>> Deloite.fill_income()
>>> Deliote.fill_expenses()
>>> Deliote.fill_investments()
>>> Deloite.get_accounts()
>>> Deloite.change_an_entry()
>>> Deloite.accounts_snapshots()
```

Reflections:

- I am new to the programing world (let alone Python programing) and it was first time I wrote that many lines of code. It was an overwhelming experience in terms of how OOP works, how different classes and objects can interact each other and how to manage a large code base. It was a great learning experience and I thoroughly enjoyed it.

- Drills, class material/discussions and finally the homeworks really helped me a lot to understand OOP concepts. One thing that took me a while to wrap my head around was the manager class/object. But now I understand the importance of manager class and how to use this to make different classes interact with each other (thanks to Gunnar's Youtube video on manager class).

- I completed what I initially planned but I have few ideas to further improve Budget-Tracker. One thing I realize is pretty cumbersome about Budget-Tracker is that you need to manually enter all the income, expense, and investment data. I think it will be really nice to have this program extract income data directly from the payslips, expense data directly from receipts and investment data from the investment account statement. These are some of the features that can be added to the Budget-Tracker.

Detailed Instructions: (How to use and test the project)

- First you need to import budget_manager from Budget_Tracker

  >>> from Budget_Tracker import budget_manager

- Create an object of budget_manager class. Think of this object as a manager in a finance company who has to manage budget of his several clients. Let say "Deloite has to manager budget of his clients "Maverick" and "Max" for the year 2020.

  >>> Deloite = budget_manager()

  It will give following prompts.

  How many accounts you will be managing: 2
  For which year you will be managing the budget: 2020
  Enter the account holder name: Maverick
  Enter the account holder name: Maxx

- Deloite wants to see the Income, expense and investment data of his clients. But at this point data will be empty since so data has been entered so far.

  >>> Deloite.get_accounts()

- Deloite wants to enter income data for his clients.

  >>> Deloite.fill_income()

  It will give following prompts.

  What is the salary of Maverick for the month of January: 20000
  What is the bonus of Maverick for the month of January: 0
  What is the salary of Maverick for the month of February: 20000
  What is the bonus of Maverick for the month of February: 0
  .
  .
  .

  What is the salary of Maverick for the month of December: 23000
  What is the bonus of Maverick for the month of December: 10000
  What is the salary of Maxx for the month of January: 6000
  What is the bonus of Maxx for the month of January: 1000
  .
  .
  .
  What is the salary of Maxx for the month of December: 6000
  What is the bonus of Maxx for the month of December: 1000

- Now Deliote wants to enter the expense and investment data for his clients.

>>> Deliote.fill_expenses()

It will give you prompts to enter rent, grocery, auto, medical and other expenses for both clients for each month (similar to income)

>>> Deloite.fill_investments()

It will give you prompts to enter stocks, crypto, 401K and real-state investment data for both clients for each month (similar to income)

- Deloite wants to double check all the data entered for both of his clients.
  >>> Deloite.get_accounts()

- Deloite see some mistakes in data and wants to correct salary for Maverick for January.
  >>> Deloite.change_an_entry()

  It will give following prompts.

  Which account you want to update: Maverick
  Which month you want to update: January
  Which parameter you want to update: salary
  What is the new value: 23000

- Deloite also wants to correct stocks investment for Maxx for May.
  >>> Deloite.change_an_entry()

  It will give following prompts.

  Which account you want to update: Maxx
  Which month you want to update: May
  Which parameter you want to update: stocks
  What is the new value: 10000

- Deloite now wants to see the cash flow of both of his clients for each month of 2020 and wants to see how the net worth of his clients changing month over month in 2020.
  >>> Deloite.accounts_snapshots()

Note: I have shared the jupyter notebook of the project along with the outputs I got for the above inputs.

Framework

Class: Income (year, month, salary, bonus)

Methods:

monthly_income (month),

add_income (month, salary, bonus),

update_salary (month, salary),

update_bonus (month, bonus),

__str__(),

__init__()

Class: Expense (year, month, rent, grocery, medical, travel, other)

Methods:

monthly_expenses (month),

add_expenses (month, rent, grocery, medical, travel, other),

update_rent (month, rent),

update_grocery (month, grocery),

update_auto_expenses (month, auto),

update_medical_expenses (month, medical),

update_other_expenses (month, other),

__str__(),

__init__()

Class: Investment (year, month, stocks, crypto, retirement_fund, realstate)

Methods:

monthly_investment (month),

add_investment (month, stocks, crypto, retirement_fund, realstate),

update_stocks (month, stocks),

update_crypto (month, crypto),

update_retirement_fund (month, retirement_fund),

update_realstate (month, realstate),

__str__(),

__init__()

Class: Budget (year)

Methods:

cash_flow(), net_worth(), __str__(), __init__()

Class: budget_manager (year, # of accounts, account_names)

Methods:

get_accounts(),

fill_income(),

fill_expenses(),

fill_investments(),

accounts_snapshot(),

change_an_entry()

- budget_manager class instantiates an object of Budget class for each account.
- There can be as many accounts under a single instance of budget_manager as defined by the user.
- These accounts can be updated using budget_manager's methods (i.e. fill_income() etc.)
- Useful insights can be displayed for each of the account using methods get_accounts() and accounts_snapshot().