Final Project

for the course:

30517 Python Programming for Economics, Management and Finance.

Group: :

Jerry Fanelli (3032456) and Filippo Borzatta (3024373)

Name of the project:

# YourPay

Brief description:

“YourPay” is an application which allows users create a scheduler to manage daily transactions. It’s given the chance to create a multitude of accounts for different reasons. Operations insert are characterized by four attributes:

* Account where to save the transaction - Category
* Transaction Date
* Amount of the operation (+ for expenses and – for income)
* Description/comment on the operation

Users can register on the platform and save the bank accounts they will use to make transactions. For bank account we refer to: "a bank account or any other type of account to which the user wants to distinguish from the other ones. For example transactions carried out through cash, through Unicredit bank account, or any other bank account."

*In this way it is possible to keep under control, in a simple and fast way,   
all the balances of the user's accounts.*

Once the user has been registered in the platform, all his data are encrypted, via AES encryption, by using the password used during registration. Only the user has the availability of his data, which can be downloaded at any time from the dashboard.

In addition to managing transactions, the user could be grateful to check their statistics: - Grouping by category, or by comment:

* Category in which the most was spent in the last month/year
* Category from which the largest entry was received
* Amount spent per month
* Amount earned per month
* Pie chart of the categories in which you have spent the most

## Method:

Libraries used:

json, base64, haslib, Crypto, bottle, AESCipher, os, pprint, random, ﻿numpy, ﻿matplotlib

We created two classes, one for managing the action of the users (“Account”) and one for managing the account statistics (“Stats”).

The “Account” class is able to:

1. Create new accounts
2. Enter transactions
3. Checking the balances
4. Get the amounts sorted by category 5. Get the total amount spent by month 6. Adding account.
5. Convert data to JSON
6. Loading data from a JSON

The “Stats” class is able to:

1. Reorder datas in dictionaries and lists
2. Create charts

## Documentation:

|  |  |  |
| --- | --- | --- |
| Function name | Parameters | Function description |
| create\_account() | Self;  Lists: list used to create a new account: [account\_name, account\_balance] | **Create a new** record in the **account** list |
| new\_row() | ﻿Self;  Date: date of the transaction *(int)*  amount: of the transaction, ‘+’ for expenses, ‘-‘ for income*(Int)*  account\_name: name of the account where to insert the transaction *(str)*  category: name of the category *(str)*  comment: notes on the transaction *(str)*  change\_balance: if True update the balance of the Account | **Updating the balance of the account**, thanks to this function is possible to change the amount per account. |
| get\_balance() | Self;  Account\_name: name of the account the user asks the balance *(str)* | View **account balance** |
| get\_accounts() | Self; | Function which returns a **list of the accounts** with the balance |
| ﻿get\_amount\_by\_category\_date() | Self;  Category: category users need to filter *(str)*  initial\_date: starting date where to start collecting amounts (int) | Function useful to **filter the amounts by category and date**. |
| get\_total\_spent() | Self | Function which returns the **total expense** from all the accounts |
| get\_total\_balance() | Self | Function which returns the **total balance** of all the accounts |
| get\_amount\_by\_month\_year() | Self;  Month: month under investigation (int)  Year: year under investigation (int) | Function which returns the **amount of the period** filtered |
| get\_category\_list() | Self | Function which returns **the list of the categories** of expenses |
| get\_first\_date() | self | Function which returns the **date of the first operation** |
| get\_rows() | Self;  date: starting time from **where collecting data** (int) | Function which returns **the list of operations made** in a **certain date** |
| get\_all\_rows() | Self | Function which returns **all the operation** of a certain account |
| remove\_row() | Self;  Row\_id: number of the operation the user wants to delete *(int)* | Function which **removes** a certain operation |
| reorder\_datas() | Self | Function useful **for sorting data** in decreasing order |
| toJSON() | Self | Function which **converts Class in JSON** string *(str)* |
| loadJSON() | Self;  Dictionary: Account dictionary in JSON  change\_balance: ??? | Function which **uploads the data from JSON file.** 1) creating a list of accounts from the dictionary  2) Reordering the rows by date  3) creating the rows from reordered data |
| Statistics | | |
| chart\_total\_amount\_spent\_perday() | Self; | Function which builds a graph on the **total amount spent per day** and builds a graph with blue values for the positive ones and green for the negatives |
| chart\_total\_amount\_spent\_permonth() | Self; | Function which builds a **graph on total amount spent per month and builds** a graph with blue values for the positive ones and green for the negatives |
| chart\_amount\_spent\_permonth\_account() | Self;  Account: account id | Function which returns **the total amount spent per month** given the account number, and builds a graph |
| chart\_amount\_spent\_permonth\_category() | Self;  Category: category description | Function which returns the total **amount spent per month** given the category field, and builds a graph |
| chart\_amount\_spent\_perday\_account() | Self;  Account: account id | Function which returns the **total amount spent** per day given the account id, and builds a graph |
| chart\_amount\_spent\_perday\_category() | Self;  Category: category description | Function which returns the **total amount spent per day** given the category field, and builds a graph |
| chart\_balance\_permonth\_account() | Self;  Account: account id | Function which **builds the graph for the balance trend** of an account in a certain month |
| chart\_balance\_perday\_account() | Self;  Account: account id | Function which **builds the graph for the balance** trend of an account in a certain day |
| chart\_balance\_permonth() | Self; | Function which **builds the graph for the general balance** trend of all the accounts an user in a certain month |
| chart\_balance\_perday() | Self; | Function which builds the graph for the general **balance trend of all the accounts of an user in a certain day** |

Then through the "bottle" library, we have created a website base with a minimum of HTML and CSS. In this way we are able to design a registration and login system, and a simple dashboard consisting of a form to add transactions and a table to display them.

Each time the user adds new transactions, his personal Account class is updated and converted to a JSON file. This JSON file is encrypted using AES encryption through the user password. Then, each time the user logs in, his data will be made visible through his password.

**Intended field of application:** we have loaded the program on a VPS where is hosted the website linked to a DNS: https://yourpay.club/ . Afterwards, we started to use the program in first person, to test it and in case add features or improve existing ones. The site is public, so anyone can test and try the application.