horizontale Linie

Authors: Tra My Sitz, Oleksandr Kotkov, Ovidio Herrera, Robinson Quiroz

Group 5: Personal Finance

**6th February 2023**

# Required libraries:

install seaborne

install PyQt5

Install pyqt5-tools

install Splitwise

install pandas

install numpy

install matplotlib

install scikit-learn

# Used packages:

import sqlite3

import random

import json

import datetime

import requests

# Task 1: SPLITWISE SYNC

Presented by: Oleksandr Kotkov

The code used in this task retrieves all expense data from the Splitwise account and inserts it to the local .sqlite database

# Used packages:

|  |  |
| --- | --- |
| import sqlite3  from sqlite3 import Error | Connection with the database |
| from splitwise import Splitwise | Connection with the Splitwise server |

# Functionalities:

access\_to\_splitwise() - getting the instance of a Splitwise account

splitwise\_sync() - synchronizing local .sqlite database with the Splitwise database

crate\_tables(), fill\_tables() - creation and filling of tables in a local .sqlite database

# Task 2: INCOME INPUT

Presented by: Oleksandr Kotkov

The code used in this task provides functionality of adding new categories and transactions.

# Used packages:

|  |  |
| --- | --- |
| import sqlite3  from sqlite3 import Error | Connection with the database |
| from splitwise import Splitwise | Connection with the Splitwise server, retrieving data of a main user of the account |

# Functionalities:

insert\_category(),insert\_subcategory(),insert\_transaction(), insert\_transaction\_item() - queries for inserting rows into the corresponding tables of the database

# Task 3: DEFAULT CURRENCY

Presented by: Robinson Quiroz

Developers create a mechanism that calculates an equivalent of amount in Euro for each transaction. API https://fixer.io/ is used to get the historical currency exchange data.

# sql\_currency.py

Converts SQL expense records (Splitwise) from foreign currency amount to default Euros (€).

# Used packages:

def currency(s\_obj: Splitwise, settings: dict):

df\_sql = symbol\_date\_sqldf(s\_obj)

status = fixer\_api\_latest(settings)

# Functionalities:

fixer\_api\_latest(settings)

Requests today's live currency-rates from an API service and saves them to a local json-file. In case of API call errors, it takes past currency rates from our local json-file. This prevents interruptions for the application processes and tasks integration.

symbol\_date\_sqldf(s\_obj)

Retrieves a pandas dataframe from Splitwise transactions stored in our local sql database

currency(s\_obj: Splitwise, settings: dict)

For each transactions in our dataframe, it converts amounts from foreign currency to euro and fills base\_amount with the respective values

# Task 4: UNRECORDED TRANSACTIONS

Presented by: Robinson Quiroz

Developers create a mechanism that calculates the amount that the user forgot to record in Splitwise and inserts this amount as an income or an expense into the DB.

From splitwise retrieves user’s friends debts in any currency and converts them to euros

Accountability definition

unrec\_trans can be less, equal or larger than income - expense + owes - owed

if unrect\_trans = 0 then income + owes = expenses + owed

if unrect\_trans < [or >] 0 then PseudoIncome < [or >] PseudoExpenses

To balance total transactions we use fact balances “fact\_bal” as follows

unrec\_trans = income - expense + owes - owed -(-fact\_bal)

that balances to zero when

unrect\_trans = -(-fact\_bal) = fact\_bal

Fact balance will be recorded as income transaction:

if fact balance > 0 then positive transaction

if fact balance < 0 then negative transaction

# unrect\_transac.py

# Used Packages and Functionalities:

* def friends\_balance\_euros(s\_obj: Splitwise, settings:dict):

friends\_balance\_currency(s\_obj)

These functions retrieve from Splitwise API all records for user’s friends *owes*, *owed* and convert them to euros (€), as splitwise fails to report them in euros.

Here it is used the local json-file for live or past currency rates created and used in Task 3.

* unrt.unrecorded\_transaction\_no\_write(sObj)

return unrecorded\_trans, income, expense, net\_debt, owes\_base, owed\_base

It displays current values, without modifying database

* unrt.unrecorded\_transaction\_write(sObj)

return unrecorded\_trans, income, expense, net\_debt, owes\_base, owed\_base

It adds an income transaction, in the database, the amount entered as fact balance

# Task 5: PREDICTION

Presented by: Tra My Sitz

Developers create a mechanism to predict user’s balance for the next year based on the (repeating) expenses, income for the recent months as well as current balance. Prediction is saved as the plot in .pdf.

# Used packages & Functionalities:

|  |  |
| --- | --- |
| **Packages** | **Functionalities** |
| from sklearn import linear\_model | to create linear regression model |
| import sqlite3 | * connect with sql database * did queries for filtering the last months, differentiate from expenses and income and sum up base amount in € |
| import pandas as pd | * convert all queries in dataframe * sum all base amounts in the last month * calculate the balances for each month |
| import matplotlib.pyplot as plt | * to create the regression model in a diagramme and show the tendency of balance in the next month * to save it as jpg and pdf file |

# Task 6: REPORTING

Presented by: Tra My Sitz

Developers implement a routine of aggregating the data from the database, visualizing it and saving it as the [date\_time].pdf file. These figures should be included into the pdf: pie charts of expenses per subcategory for the last 3 months (one chart per month); column chart showing income vs expenses for the last 3 months.

# Used packages & functionalities

|  |  |
| --- | --- |
| **Packages** | **Functionalities** |
| import sqlite3 | * connect with sql database * did queries for filtering the last months, differentiate from expenses and income and sum up base amount in € |
| import pandas as pd | * convert all queries in dataframe * sum all base amounts in the last month * calculate the balances for each month |
| import matplotlib.pyplot as plt | * to create the pie chart and column chart to save it as jpg and pdf * to save it as jpg and pdf file |

# 

# Task 7: USER INTERFACE

Presented by: Ovido Herrera

The code for the user interface is located in two files: console.py and personal\_finance.py  
-console.py contains the commands for creating the user interface, this includes: creating widgets, buttons, labels and setting up their visual aspects.  
-personal\_finance.py contains the commands for initializing the user interface as well as for the functionality of the interface objects (signal and events of each button and collecting input for the user). All the tasks are incorporated in this script.

# Used packages:

For Console:

|  |  |
| --- | --- |
| from PyQt5 import QtCore, QtGui, QtWidgets | Importing the different types of widgets and objects used in the user interface. |

For Personal Finance:

|  |  |
| --- | --- |
| from PyQt5 import QtCore, QtWidgets, QtGui  from PyQt5.QtWidgets import QApplication, QMainWindow, QMessageBox  import sys | Importing the different types of widgets and objects used in the user interface. |
| from console import Ui\_MainWindow | Importing the specification and visual set up of the user interface |
| from sql\_queries\_methods import \* | Importing the functions that are used to execute the sql queries |
| from integrated\_tasks import \* | Importing the function where Task1 and Task 3 have been integrated |
| import unrect\_transac as unrt | Importing the function to execute Unrecorded Transactions |
| from Prediction\_TraMy import \* | Importing functions to execute prediction |
| from reporting\_TraMy import report, save\_pie | Importing functions to execute reporting |
| from Balance\_Plots import plot\_bal\_cummsum | Importing functions to execute Daily balances |

# Functionalities:

run\_app()

Executes Task 1 and 3 and initiates the user interface.