Universität St.Gallen

School of Management, Economics, Law, Social Sciences,
International Affairs and Computer Science

An Expense Tracking Tool in Python

Skills: Programming with Advanced Computer Languages

Mario Silic

21/12/2023

Authors:	**Student IDs:**
Giuseppe Baccaro	19-615-996
Tianyi Ren	23-600-265
Gyula Szalai	23-600-579
Marco Ritter	19-615-749

1. Python Code for Expense Tracking

This project develops a Python-based expense tracking tool to offer users an easy way to manage their finances. The core features of the code include:

- **Data Input and Validation:** Users can use an already existing dataset. For example, the user can download his monthly spending datasheet from his online banking application. In addition to the dataset, the code allows the user to input additional expense entries including the date, category, and amount. The tool validates the input to ensure that the data conforms to the expected format and that amounts are not negative.
- **Expense Summary:** The first feature of the code is that it allows users to view a summary of expenses grouped by category, providing insights into where their money is going.
- **Visualization:** The tool generates several types of visualizations, including a pie chart of expenses by category, a line chart of the cumulative spending over time, and a day-by-day expense chart. These visualizations help in understanding spending patterns and trends. The user can choose which visualization he wants to see.
- **Budget Comparison:** An additional feature of the tool is that it will ask the user for the budget he has set within each category. It will then compare actual spending against these budgeted amounts. The code then highlights areas where the user is over budget and tells him by how much he should reduce his spending within each category. If he is under budget, the code will congratulate the user on respecting his limits
- **Interactive User Interface:** Through a command-line interface, users can perform various actions, such as adding new expense entries, generating visualizations, setting a budget, or comparing expenses against their budget.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime

#First we create a path to the original dataset and an additional path to a new dataset in case additional expense entries are added from the user
# Define the file paths
#The dataset we use is an example of a expense sheet which can typically be downloaded from Online Banking apps e.g. UBS and CS

ORIGINAL_FILE_PATH = 'Expenses_November_2023.csv'
TEMP_FILE_PATH = 'Temp_Expenses_November_2023.csv'

# First we define all the functions we will need for the tool
# Function to read the original dataset

def read_dataset(file_path):
    try:
        return pd.read_csv(file_path)
```

```
19
20      # Create cases in case the file is not correct
21
22      except FileNotFoundError:
23          print(f"File not found: {file_path}")
24          exit()
25      except pd.errors.ParserError:
26          print(f"Error parsing file: {file_path}")
27          exit()
28
29  # We want the user to be able to see his total expenses per category
30  # Function to display the summary of expenses per category
31
32  def display_expenses_summary(df):
33      summary = df.groupby('Category')['Amount'].sum().reset_index()
34      print("\nSummary of expenses by category:")
35      print(summary)
36      total_spent = summary['Amount'].sum()
37      print(f"\nTotal spent in the past month: {total_spent:.2f}")
38
39
40  # Function to add a new entry to the dataset
41  # Allow the user to enter additional expense entries if he wants
42
43  def add_new_entry(df):
44      while True:
45          try:
46              date = input("Enter the date of the expense (YYYY-MM-DD): "
    )
47              datetime.strptime(date, "%Y-%m-%d")  # Validate date format
48              category = input("Enter expense category: ")
49              amount = float(input("Enter the expense amount: "))
50
51              #create scenarios if the input amount or format is not
    correct
52
53              if amount < 0:
54                  raise ValueError("The amount cannot be negative.")
55              new_entry = pd.DataFrame({'Date': [date], 'Category': [
    category], 'Amount': [amount]})
56              return df.append(new_entry, ignore_index=True)
57          except ValueError as e:
58              print(f"Invalid input: {e}. Please try again.")
59
60  # Function to save the new dataset
61
62  def save_dataset(df, file_path):
63      df.to_csv(file_path, index=False)
64
65  # Function to plot the pie chart
66
67  def plot_pie_chart(df):
68
69      # Create exception case
70
71      if df.empty:
72          print("No data available to plot.")
73          return
```

```
74
75      # Add correct formatting and color palette
76
77      category_expenses = df.groupby('Category')['Amount'].sum()
78      sorted_expenses = category_expenses.sort_values()
79      colors = sns.color_palette("Blues", n_colors=len(sorted_expenses))
80
81      plt.figure(figsize=(10, 8))
82      plt.pie(sorted_expenses, labels=sorted_expenses.index, autopct='
        %1.1f%%', colors=colors, startangle=140)
83      plt.title('Expenses by Category')
84      plt.axis('equal')
85      plt.show()
86
87  # Function to plot cumulative spending over time
88
89  def plot_cumulative_spending(df):
90      if df.empty:
91          print("No data available to plot.")
92          return
93
94      df['Date'] = pd.to_datetime(df['Date'])
95      df_sorted = df.sort_values(by='Date')
96      df_sorted['Cumulative Expense'] = df_sorted['Amount'].cumsum()
97      colors = sns.color_palette("Blues", as_cmap=True)
98
99      plt.figure(figsize=(12, 6))
100     plt.plot(df_sorted['Date'], df_sorted['Cumulative Expense'], color=
        'blue', marker='o')
101     plt.title('Cumulative Spending Over Time')
102     plt.xlabel('Date')
103     plt.ylabel('Cumulative Expense')
104     plt.grid(True)
105     plt.xticks(rotation=45)
106     plt.show()
107
108 # Function to plot day-by-day expenses
109
110 def plot_day_by_day_expenses(df):
111     if df.empty:
112         print("No data available to plot.")
113         return
114
115     df['Date'] = pd.to_datetime(df['Date'])
116     df_sorted = df.sort_values(by='Date')
117     colors = sns.color_palette("Blues", as_cmap=True)
118
119     plt.figure(figsize=(12, 6))
120     plt.plot(df_sorted['Date'], df_sorted['Amount'], color='blue',
        marker='o')
121     plt.title('Day-by-Day Expenses')
122     plt.xlabel('Date')
123     plt.ylabel('Expense')
124     plt.grid(True)
125     plt.xticks(rotation=45)
126     plt.show()
127
128 # Function to collect budget data
```

```
129
130 def collect_budget_data(categories):
131     budgets = {}
132     print("\nEnter budget for next month for each category:")
133     for category in categories:
134         while True:
135             try:
136                 budget = float(input(f"Budget for {category} next month
    : "))
137                 if budget < 0:
138                     raise ValueError("The budget cannot be negative.")
139                 budgets[category] = budget
140                 break
141             except ValueError:
142                 print("Invalid input. Please enter a valid number.")
143     return budgets
144
145 # We want to create a function that compares the total budget to actual
      expenses and gives recommendations on how to save money if over
    budget or congratulate the user  if under budget
146
147 def compare_budget_expenses(expenses, budgets):
148
149     # Plot the Budget vs Actual bar chart
150
151     expenses_summary = expenses.groupby('Category')['Amount'].sum()
152     budget_df = pd.DataFrame.from_dict(budgets, orient='index', columns
    =['Budget'])
153     budget_df['Expenses'] = expenses_summary
154     budget_df['Difference'] = budget_df['Budget'] - budget_df['Expenses
    ']
155
156
157     budget_df[['Budget', 'Expenses']].plot(kind='bar', color=['skyblue'
    , 'salmon'])
158     plt.title("Expenses vs Budget")
159     plt.ylabel("Amount")
160     plt.xlabel("Category")
161     plt.show()
162
163     # Calculate the total difference between actual spending and budget
      and provide feedback to the user on his spending habits
164
165     total_difference = budget_df['Difference'].sum()
166     over_budget = budget_df[budget_df['Difference'] < 0]
167
168     if total_difference < 0:
169         print("\nRecommendations to stay on budget in December:")
170         for category, row in over_budget.iterrows():
171             print(f"- Reduce spending in {category}: Over by {abs(row['
    Difference']):.2f}")
172     elif not over_budget.empty:
173         print("\nCongratulations! You are under the total budget.")
174         print("However, you have exceeded the budget in the following
    categories:")
175         for category, row in over_budget.iterrows():
176             print(f"- {category}: Budget was {row['Budget']}, Expenses
    were {row['Expenses']}, Over by {abs(row['Difference']):.2f}")
```

```
177     else:
178         print(f"\nCongratulations! You are under the total budget by {
    total_difference:.2f}.")
179 
180 #This function is the one for the user interaction!
181 
182 def main():
183     # Load the original dataset from a CSV file
184 
185     df = read_dataset(ORIGINAL_FILE_PATH)
186 
187     #The code automatically showed the option "Other" as the first one.
     We did not want it so we change in order to have the category "Food
    " as the first one
188     # Ensure 'Food' is the first category in the list for display
    purposes
189 
190     category_order = sorted(df['Category'].unique(), key=lambda x: (x
    != 'Food'))
191     df['Category'] = pd.Categorical(df['Category'], categories=
    category_order, ordered=True)
192     df = df.sort_values('Category')
193 
194     # Show the user a summary of last month's expenses by category
195 
196     display_expenses_summary(df)
197 
198     # Ask the user if they want to add more entries to the dataset
199 
200     while True:
201         print("\nIs your dataset complete? (yes/no)")
202         complete = input().lower()
203         if complete == 'no':
204 
205             # If the dataset is not complete, allow the user to add new
     entries
206 
207             df = add_new_entry(df)
208 
209             # Save the updated dataset to a temporary file
210 
211             save_dataset(df, TEMP_FILE_PATH)
212         elif complete == 'yes':
213 
214             # If the dataset is complete, exit the loop and move on
215 
216             break
217 
218     # Collect budget information from the user for the upcoming month
219 
220     categories = df['Category'].unique()
221     budgets = collect_budget_data(categories)
222 
223     # Main interactive loop for the user to choose an action
224 
225     while True:
226         print("\nChoose an option:")
227         print("1: Pie Chart")
```

```
228         print("2: Cumulative Spending Chart")
229         print("3: Day-by-Day Expenses Line Chart")
230         print("4: Compare Expenses with Budget")
231         print("5: Exit")
232         choice = input()
233 
234         # Based on the user's choice, perform an action or exit the
    program
235 
236         if choice == '1':
237             plot_pie_chart(df)
238         elif choice == '2':
239             plot_cumulative_spending(df)
240         elif choice == '3':
241             plot_day_by_day_expenses(df)
242         elif choice == '4':
243             compare_budget_expenses(df, budgets)
244         elif choice == '5':
245             break
246         else:
247 
248             # If the user makes an invalid choice, prompt them again
249 
250             print("Invalid choice. Please try again.")
251 
252 # Run the program
253 
254 main()
```

2. Example Run

The following section shows an example run in which the user will be able to add his dataset. The dataset we use in the example run is an example of expenses from the month of November 2023

- **Show the summary of expenses per category:** The user can visualise his expenses in November 2023.
- **The user can add additional expense entries:** The user is asked whether his dataset is complete or whether he wishes to add entries. If so, he can add the date, category and amount of his expense.
- **The user can add additional expense entries:** The user is asked whether his dataset is complete or whether he wishes to add entries. If so, he can add the date, category and amount of his expense.
- **The user is asked to input his predicted budget per category for the following month:** before visualizing the expenses, the user can input his expected expenses within each category. This will allow him to compare his past expenses with his budget and understand where he has over- or under-spent.
- **Visualisations:** The user can visualize four charts. He can see a pie chart of his expenses, a line chart of his cumulative expenses within the time frame, the daily expenses fluctuation and the difference between his budgeted and actual expenses.
- **Recommendations:** The user will receive recommendations on where to decrease his spending if he is over budget in certain categories. The code distinguishes between multiple scenarios. In this case, the user has spent less than his total budget but is over budget in specific categories. The code will tell him where he is over budget and by how much he should reduce.

2.1. Trial Run 1: Expenses are under the total budget

`Program:` Summary of expenses by category:

Category	Amount
Food	28.3
Other	151.8
Health	224.8
Food & non-alcoholic drinks	95.8
Housing(net), fuel & power	281.5
Miscellaneous goods & services	110.3
Communication	389.1
Education	255.3
Clothing & footwear	137.4
Household goods & services	254.6
Transport	77.9
Recreation & culture	271.2
Restaurants & hotels	138.4
Alcoholic drinks, tobacco & narcotics	83.4

`Program:` Total spent in the past month: 2499.80
`Program:` Is your dataset complete? (yes/no)
`User:` no
`Program:` Enter the date of the expense (YYYY-MM-DD):
`User:` 2023-11-20
`Program:` Enter expense category:
`User:` Food
`Program:` Enter the expense amount:
`User:` 250
`Program:` Is your dataset complete? (yes/no)
`User:` yes

`Program:` Enter budget for December for each category:
`Program:`Budget for Food in December:
`User:`150
`Program:`Budget for Other in December:
`User:` 150
`Program:`Budget for Health in December:
`User:`200
`Program:`Budget for Food & non-alcoholic drinks in December:
`User:`200
`Program:`Budget for Housing(net), fuel & power in December:
`User:`200
`Program:`Budget for Miscellaneous goods & services in December:
`User:`500
`Program:`Budget for Communication in December:
`User:`300
`Program:`Budget for Education in December:
`User:`100
`Program:`Budget for Clothing & footwear in December:

User:400
Program:Budget for Household goods & services in December:
User:300
Program:Budget for Transport in December:
User:200
Program:Budget for Recreation & culture in December:
User:200
Program:Budget for Restaurants & hotels in December:
User:250
Program:Budget for Alcoholic drinks, tobacco & narcotics in December:
User:10

Program: Choose an option:
1: Pie Chart
2: Cumulative Spending Chart
3: Day-by-Day Expenses Line Chart
4: Compare Expenses with Budget
5: Exit

User: 1

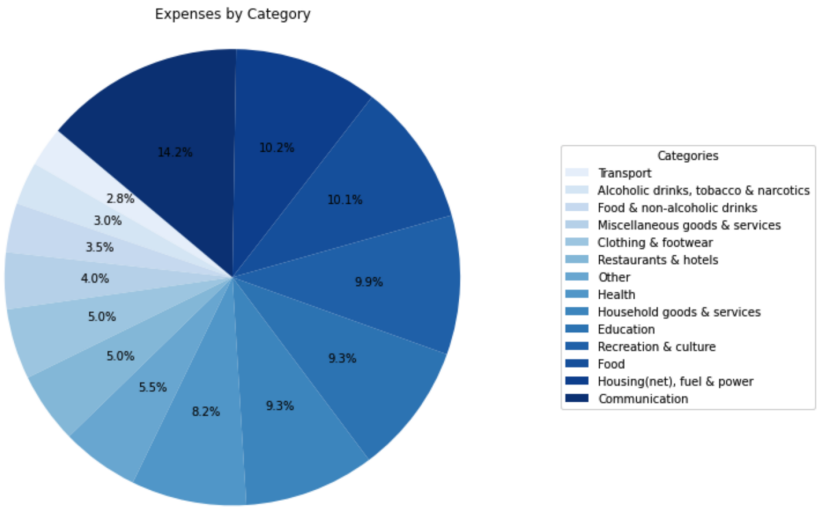

Figure 1: Pie chart of expenses. It includes the additional expense entry of Food, 250 CHF

```
Program: Choose an option:
1:  Pie Chart
2:  Cumulative Spending Chart
3:  Day-by-Day Expenses Line Chart
4:  Compare Expenses with Budget
5:  Exit
User: 2
```

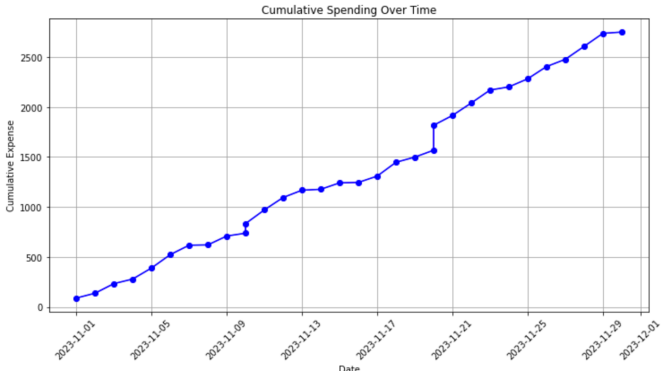

Figure 2: Cumulative line chart of expenses. It includes the additional expense entry of Food, 250 CHF

```
Program: Choose an option:
1:  Pie Chart
2:  Cumulative Spending Chart
3:  Day-by-Day Expenses Line Chart
4:  Compare Expenses with Budget
5:  Exit
User: 3
```

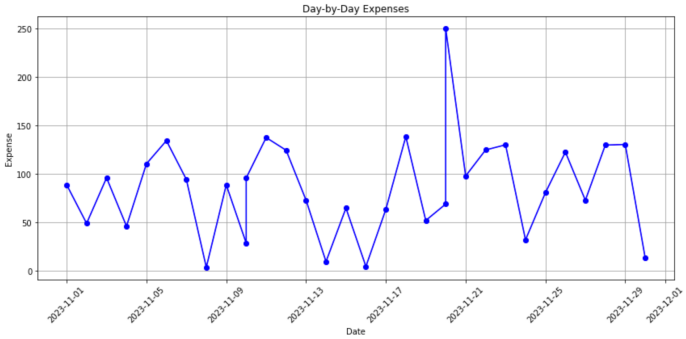

Figure 3: Line chart of expenses day-to-day. It includes the additional expense entry of Food, 250 CHF

`Program:` Choose an option:
`1: Pie Chart`
`2: Cumulative Spending Chart`
`3: Day-by-Day Expenses Line Chart`
`4: Compare Expenses with Budget`
`5: Exit`
`User:` 4

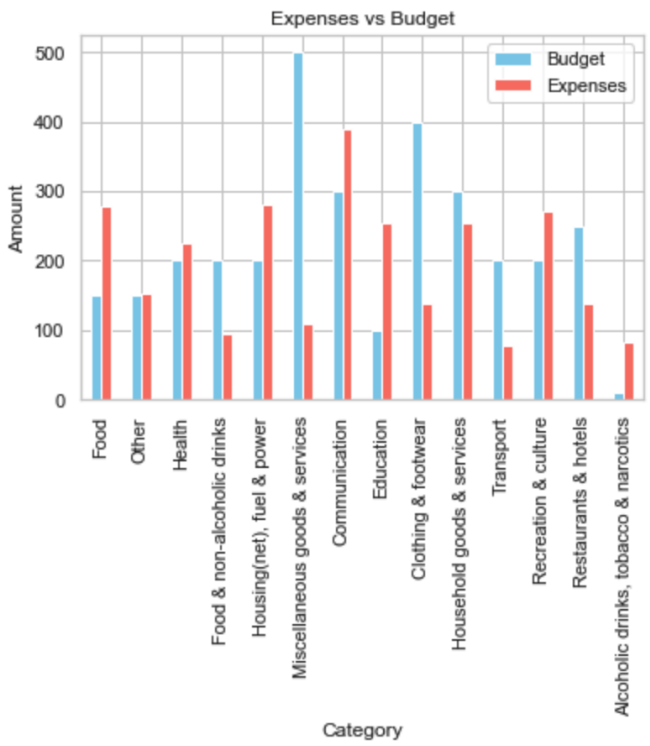

Figure 4: Comparison of expenses to budget. It includes the additional expense entry of Food, 250 CHF

`Program:` Congratulations! You are under the total budget. However, you have exceeded the budget in the following categories:

- Food: Budget was 150.0, Expenses were 278.3, Over by 128.30
- Other: Budget was 150.0, Expenses were 151.8, Over by 1.80
- Health: Budget was 200.0, Expenses were 224.8, Over by 24.80
- Housing(net), fuel & power: Budget was 200.0, Expenses were 281.5, Over by 81.50
- Communication: Budget was 300.0, Expenses were 389.1, Over by 89.10
- Education: Budget was 100.0, Expenses were 255.3, Over by 155.30
- Recreation & culture: Budget was 200.0, Expenses were 271.2, Over by 71.20
- Alcoholic drinks, tobacco & narcotics: Budget was 10.0, Expenses were 83.4, Over by 73.40

2.1. Trial Run 2: Expenses are over the total budget

This scenario is similar to before except we imagine that the expenses are much higher than the budget. The recommendations will be the following:

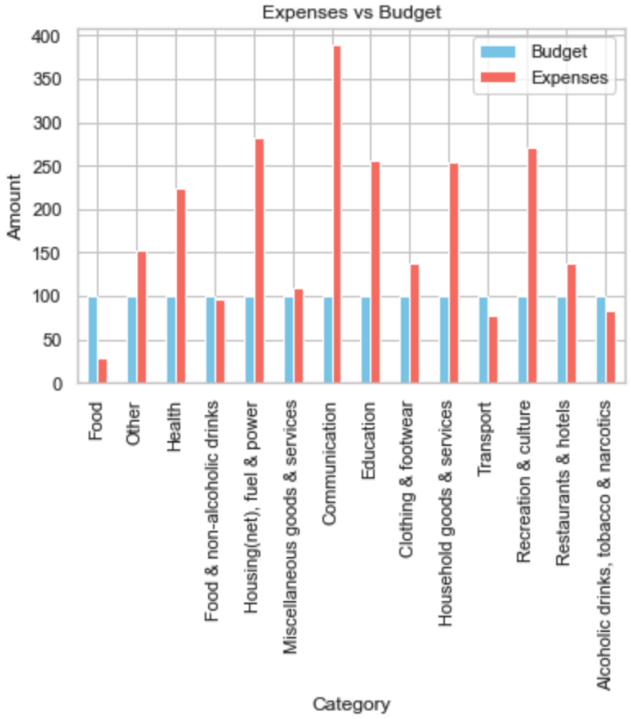

Figure 5: Comparison of expenses to budget

`Program:` Recommendations to stay on budget in December:

- Reduce spending in Other: Over by 51.80
- Reduce spending in Health: Over by 124.80
- Reduce spending in Housing(net), fuel & power: Over by 181.50
- Reduce spending in Miscellaneous goods & services: Over by 10.30
- Reduce spending in Communication: Over by 289.10
- Reduce spending in Education: Over by 155.30
- Reduce spending in Clothing & footwear: Over by 37.40
- Reduce spending in Household goods & services: Over by 154.60
- Reduce spending in Recreation & culture: Over by 171.20
- Reduce spending in Restaurants & hotels: Over by 38.40