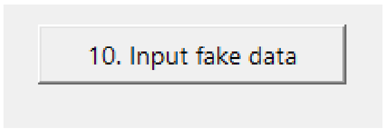
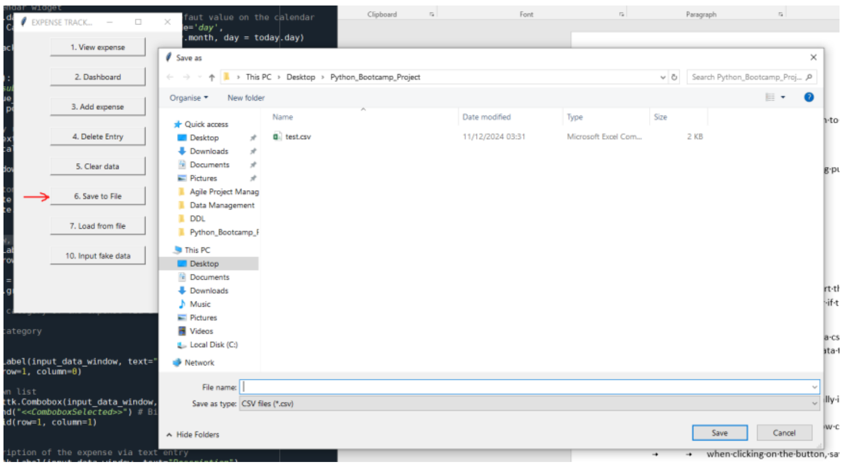
def convert_columns(df)
Take the data frame and convert the column to the right format based on the dictionary df_dtype {col:dtype}

def input_fake_data()
generate fake data for testing and debugging purporse
{date, category, description, amount}

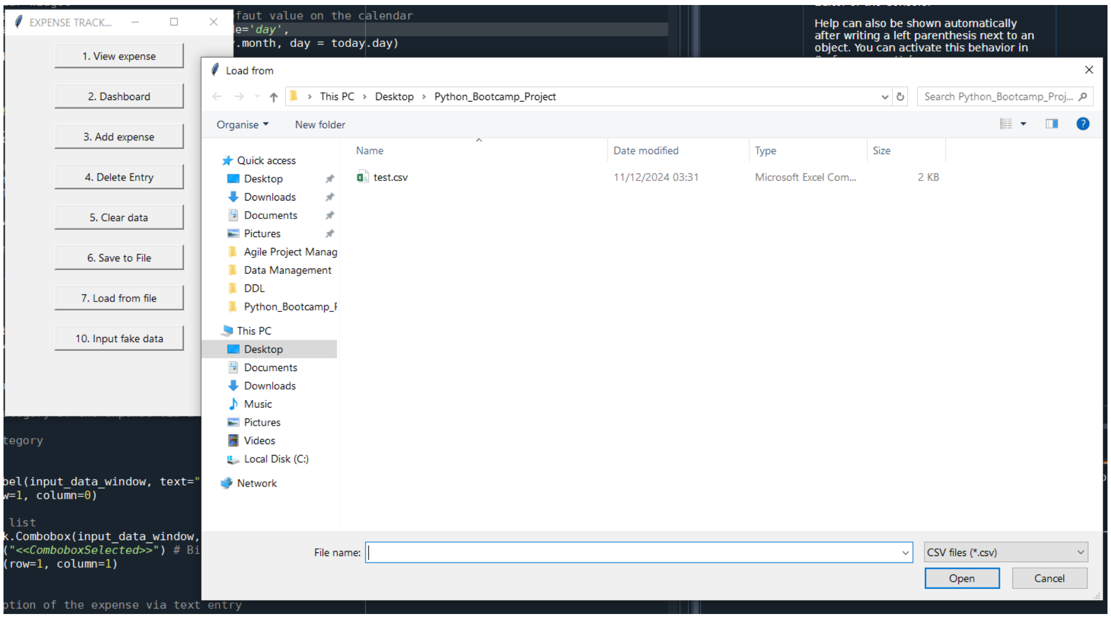


def save_to_file()
Open a dialog box allowing the user to export the dataframe into a csv file
Have a handling error if the operation fail or if the user close the dialog
Functionality: the user can save its data by exporting a csv file



```
faut value on the calendar
e='day',
.month, day = today.day)
ategory
abel(input_data_window, text="
ow=1, column=0)
n list
tk.Combobox(input_data_window,
nd("<<ComboboxSelected>>") # Bi
d(row=1, column=1)
iption of the expense via text entry
```

def load_from_file()
Open a dialog box allowing the user to load a csv file into the dataframe.
The csv file is expected to follow the right data type structure
Functionality: the user can upload an existing csv file to input some data



```
faut value on the calendar
e='day',
.month, day = today.day)
```

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in

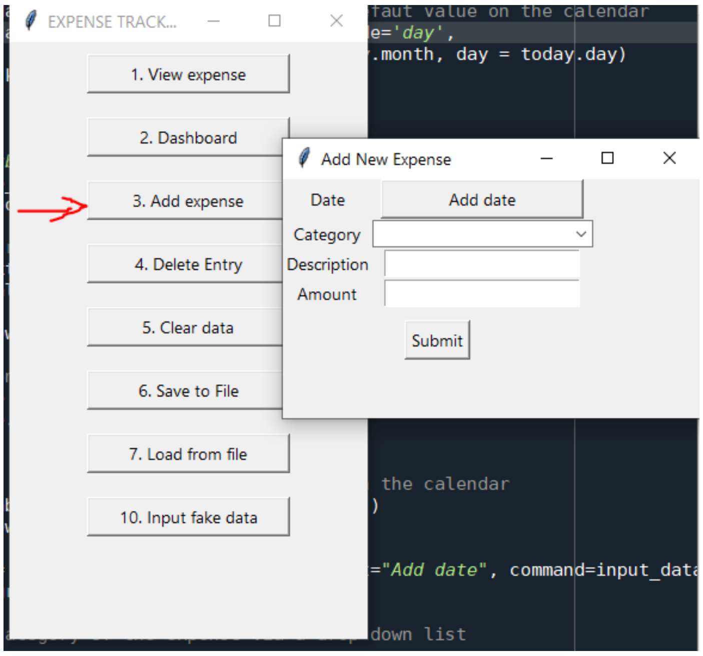
Name	Date modified	Type	Size
test.csv	11/12/2024 03:31	Microsoft Excel Com...	2 KB

```
tegory

el(input_data_window, text="
w=1, column=0)

 list
k.Combobox(input_data_window,
("<<ComboboxSelected>>") # Bi
(row=1, column=1)

ption of the expense via text entry
```



```
faut value on the calendar
e='day',
.month, day = today.day)

 the calendar
)

="Add date", command=input_data

down list
```

def input_data()

Create a window where the user can manually input an entry: {date, category, description, amount}

def input_data_calendar()

(1) open a sub window, the user can now choose the date via a calendar

def submit_date()

(2) when clicking on the button, save the date into a variable value_date and close the calendar

label for Date + button for to open the calendar

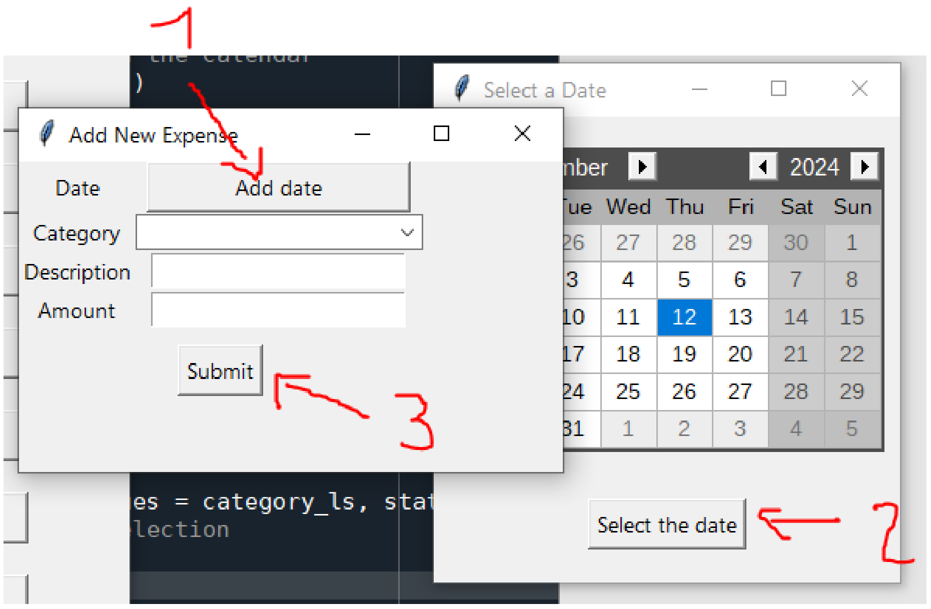
label + drop-down list for the Category

label + text entry for Description

label + test entry for Amount

def submit_data()

(3) Save the data into the dataframe and perform data check to ensure the user has enter t the right type of data (int for the amout, have selected a date or a category)



Functionality: the user can record some expense. He can select the date (1) (via a calendar), select the category via a (2) drop-down list and add the amount and a decription. All field are mandatory except the description.

def delete_entry():

    def try_delete()

    Search for the corresponding expense to delete and manage records with the same date

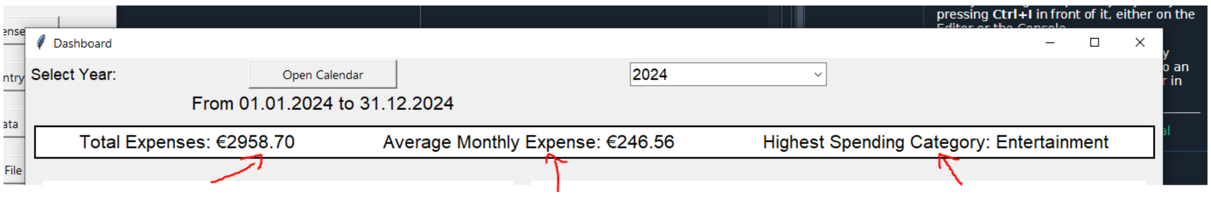
        def confirm_selected()

        Delete the entry, in case of multiples records on the same date, ask for which one to delete

Functionality: the user can delete a record (a spending) by searching the date (mandatory search) and can select which expense to delete in case of same day expense

def show_expense()
Show the dataframe to the user.
The user can sort the dataframe by clicking on the corresponding column
Functionality: the user can look at the current data

def update_kpi_frame(kpi_frame, filtered_df):
In the dashboard, call the function to refresh the numeric KPI (delete and recreate)
As an input, it takes in consideration the filtered dataframe (period of time selected on the dashboard)
KPIs are total amount of expenses, average monthly spending and highest category of spending

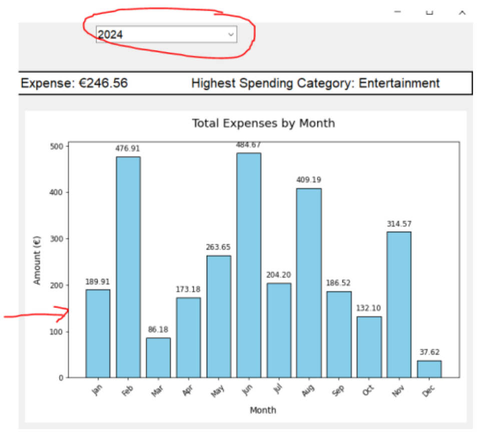


From 01.01.2024 to 31.12.2024

Total Expenses: €2958.70   Average Monthly Expense: €246.56   Highest Spending Category: Entertainment

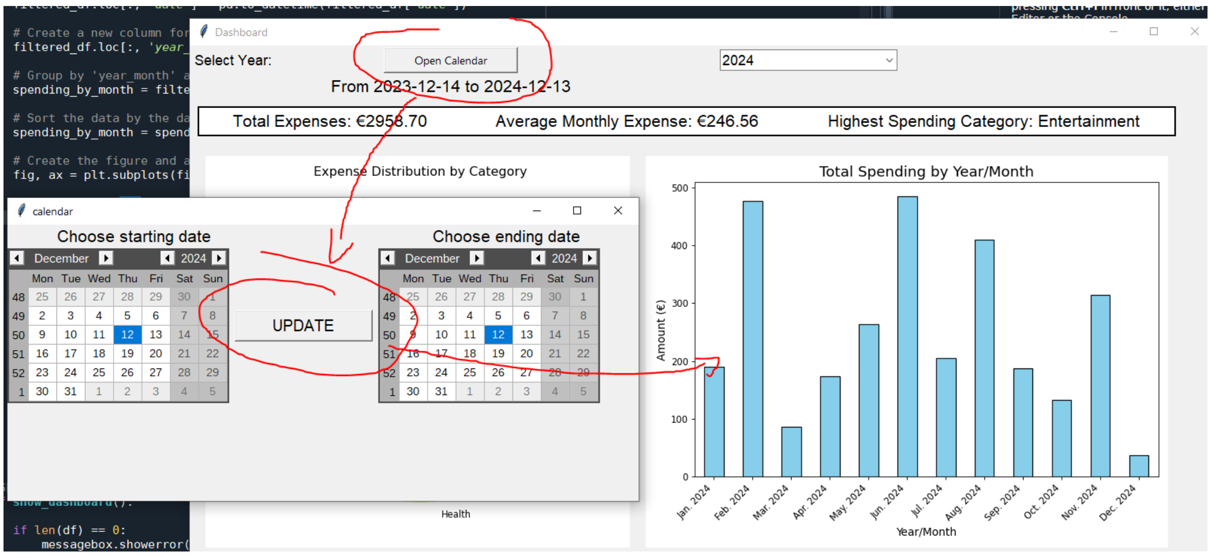
def create_pie_chart(chart_frame, filtered_df):
Create a pie chart showing the expense distribution by category given the filtered date period set by user

def create_bar_chart(chart_frame, filtered_df, selected_year)
Create a bar chart showing monthly expenses. The bar chart has the specificity to be set to a 1 year period with 12 bars showing each months.
It is the default barchart that appear in the dashboard without pre-selection
It is only called when the user select the period with the drop down list of dashboard


2024

Expense: €246.56   Highest Spending Category: Entertainment

def create_bar_chart_2(chart_frame, filtered_df):
Create the same bar chart but this one is not set on a 1 year period and is more flexible
It is only called when the time period is selected via the calendar in the dashboard


```
# Create a new column for
filtered_df.loc[:, 'year_

# Group by 'year_month' a
spending_by_month = filte

# Sort the data by the da
spending_by_month = spend

# Create the figure and a
fig, ax = plt.subplots(fi

if len(df) == 0:
    messagebox.showerror(
```

Dashboard

Select Year: Open Calendar   2024

From 2023-12-14 to 2024-12-13

Total Expenses: €2958.70   Average Monthly Expense: €246.56   Highest Spending Category: Entertainment

calendar

Choose starting date: December 2024 (12 selected)   UPDATE   Choose ending date: December 2024 (12 selected)

def show_dashboard():
The dashboard with some metrics, a pie chart, a barchart, and some tools to select the date
It will call some function from above code to create the chart & the metrics

  def open_calendar():
  (1) Open a window with 2 calendar for selecting a period with a start date & endate

    def update_graph():
    (2) When click on the UPDATE button, update the filtered data and recreate all KPI +

charts (call pie chart and barchart_2)

Retrieve the start and end date value from the two calendar + filter the dataframe

Make sure that the period are valid

Call the KPI, pie_chart and barchart_2 to recreate them

def on_year_selected(selected_year):

(3) Update when dropdrown for Year Filter is changed

retrive the year + filter the dataframe

Call the KPI, pie_chart and barchart_2 to recreate them

dynamically update the graph (call pie_chart and barchart) when a year is selected from the drop-down list

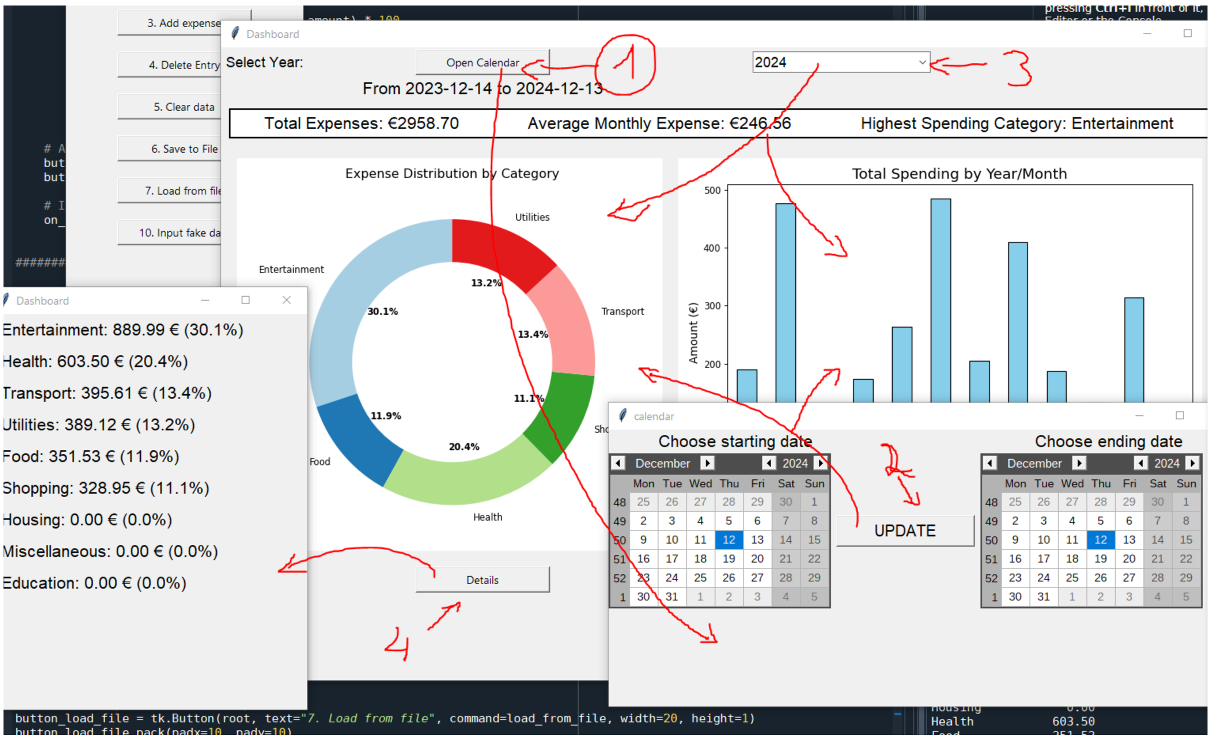
def open_detail():

(4) Open a window showing the spending by category, orderer by highest spending

Functionality: the user get a dashboard with a pie chart showing the total expense by category and a barchart showing the expense by time (month and year).

He can filter the data either by selection a given year on the (3) drop-down filter or Open the calendar button (1) to get two calendar to input a given period of time and (2) update the dashboard.

Finally, in the (4) detail button, he can see the amount spent by category ordered by most spent category.



From 2023-12-14 to 2024-12-13

Total Expenses: €2958.70 Average Monthly Expense: €246.56 Highest Spending Category: Entertainment

Entertainment: 889.99 € (30.1%)
Health: 603.50 € (20.4%)
Transport: 395.61 € (13.4%)
Utilities: 389.12 € (13.2%)
Food: 351.53 € (11.9%)
Shopping: 328.95 € (11.1%)
Housing: 0.00 € (0.0%)
Miscellaneous: 0.00 € (0.0%)
Education: 0.00 € (0.0%)

Choose starting date

December 2024

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
48	25	26	27	28	29	30	1
49	2	3	4	5	6	7	8
50	9	10	11	12	13	14	15
51	16	17	18	19	20	21	22
52	23	24	25	26	27	28	29
1	30	31	1	2	3	4	5

Choose ending date

December 2024

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
48	25	26	27	28	29	30	1
49	2	3	4	5	6	7	8
50	9	10	11	12	13	14	15
51	16	17	18	19	20	21	22
52	23	24	25	26	27	28	29
1	30	31	1	2	3	4	5

```
button_load_file = tk.Button(root, text="7. Load from file", command=load_from_file, width=20, height=1)
```

```
Health          603.50
```