Expense Tracker

October 11, 2024

1 Libraries and Dictionary

[2]:
```
import csv
from datetime import datetime
import os
from IPython.display import FileLink, display

# Dictionary to store expenses, categorized by year and month
expenses = {}
# Dictionary to store monthly budgets
budgets = {}
expense_id = 1  # Initial ID for the first expense
```

2 Function to Set Budget:

[3]:
```
def set_budget():
    year = int(input("Enter the year (e.g., 2024): "))
    month = int(input("Enter the month (1-12): "))

    while True:
        try:
            budget = float(input(f"Enter your budget for {year}-{month}: "))
            if budget < 0:
                raise ValueError("Budget cannot be negative!")  # Handle␣
 ↪negative budgets
            break  # Break the loop once a valid budget is entered
        except ValueError as e:
            print(f"Invalid budget! {e} Please enter a valid positive number.")

    # Store the budget for the specified month
    budgets[(year, month)] = budget
    print(f"Monthly budget for {year}-{month} set to ${budget:.2f}\n")
```

3 Function to Add an expense

[5]:
```
def add_expense():
    global expense_id

    # Handle the date input with error handling
    while True:
        try:
            date = input("Enter the date (YYYY-MM-DD): ")
            # Ensure the correct date format
            date_obj = datetime.strptime(date, '%Y-%m-%d')
            year = date_obj.year
            month = date_obj.month
            break  # Break the loop once a valid date is entered
        except ValueError:
            print("Invalid date format! Please enter the date in YYYY-MM-DD␣
↪format.")

    # Check if a budget is set for the month before adding expenses
    if (year, month) not in budgets:
        print(f"No budget set for {year}-{month}. Please set a budget first to␣
↪track expenses.\n")
        return  # Exit the function if there's no budget

    category = input("Enter the category (e.g., Food, Travel): ")

    # Handle the amount input with error handling
    while True:
        try:
            amount = float(input("Enter the amount: "))
            if amount < 0:
                raise ValueError("Amount cannot be negative!")  # Handle␣
↪negative amounts
            break  # Break the loop once a valid amount is entered
        except ValueError as e:
            print(f"Invalid amount! {e} Please enter a valid positive number.")

    description = input("Enter a brief description: ")

    # Store the expense in a dictionary with an ID
    expense = {
        'id': expense_id,
        'date': date,
        'category': category,
        'amount': amount,
        'description': description
    }
```

```
    # Store expenses by year and month
    if (year, month) not in expenses:
        expenses[(year, month)] = []
    expenses[(year, month)].append(expense)
    print(f"Expense added successfully with ID: {expense_id}\n")

    expense_id += 1  # Increment the ID for the next expense

    # Check if the new total for the month exceeds the budget
    total_spent = sum(exp['amount'] for exp in expenses[(year, month)])
    budget = budgets[(year, month)]
    if total_spent > budget:
        print(f"Warning: You have exceeded the budget for {year}-{month}!\n"
              f"Total spent: ${total_spent:.2f}, Budget: ${budget:.2f}\n")
    else:
        remaining = budget - total_spent
        print(f"Total spent: ${total_spent:.2f}, Remaining budget for␣
↪{year}-{month}: ${remaining:.2f}\n")
```

4 Function to View expenses for a specific month

[19]:
```
def track_budget():
    year = int(input("Enter the year (e.g., 2024): "))
    month = int(input("Enter the month (1-12): "))

    # Calculate total spent for the given month
    total_spent = sum(expense['amount'] for expense in expenses.get((year,␣
↪month), []))

    if (year, month) in budgets:
        budget = budgets[(year, month)]
        print(f"Total spent for {year}-{month}: ${total_spent:.2f}")
        if total_spent > budget:
            print("Warning: You have exceeded your budget!\n")
        else:
            remaining = budget - total_spent
            print(f"You have ${remaining:.2f} left for the month.\n")
    else:
        print(f"No budget set for {year}-{month}. Please set a budget first.\n")
```

5 Function to Track spending against the budget

[7]:
```
def view_expenses():
    while True:
        try:
            year = int(input("Enter the year (e.g., 2024): "))
            if year < 0:
                raise ValueError("Year cannot be negative!")  # Handle negative␣
↪years
            month = int(input("Enter the month (1-12): "))
            if month < 1 or month > 12:
                raise ValueError("Month must be between 1 and 12!")  # Handle␣
↪invalid months
            break  # Break the loop once valid year and month are entered
        except ValueError as e:
            print(f"Invalid input! {e} Please enter valid numbers.")

    # Check if a budget is set for the specified year and month
    if (year, month) not in budgets:
        print(f"No budget set for {year}-{month}. Cannot view expenses without␣
↪a budget.\n")
        return  # Exit the function if there's no budget

    # Check if there are expenses recorded for the specified year and month
    if (year, month) not in expenses or not expenses[(year, month)]:
        print(f"No expenses recorded for {year}-{month}.\n")
        return

    print(f"\nExpenses for {year}-{month}:")
    for expense in expenses[(year, month)]:
        print(f"ID: {expense['id']}, Date: {expense['date']}, Category:␣
↪{expense['category']}, "
              f"Amount: ${expense['amount']:.2f}, Description:␣
↪{expense['description']}")
    print()
```

6 Function to Save and Load expenses

[11]:
```
def save_expenses():
    # Check if there are any expenses or budgets before saving
    if not expenses and not budgets:
        print("No records found to save. Please add budgets or expenses first.
↪\n")
        return  # Exit if no records exist

    file_path = 'expenses.csv'  # Save the file in the current working directory
```

```
    with open(file_path, mode='w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(['year', 'month', 'id', 'date', 'category', 'amount',␣
 ↪'description', 'budget', 'remaining_budget'])
        for (year, month), monthly_expenses in expenses.items():
            total_spent = sum(exp['amount'] for exp in monthly_expenses)
            budget = budgets.get((year, month), 0)
            remaining_budget = budget - total_spent
            for expense in monthly_expenses:
                writer.writerow([year, month, expense['id'], expense['date'],␣
 ↪expense['category'], expense['amount'], expense['description'], budget,␣
 ↪remaining_budget])
    print(f"Expenses saved to {file_path}\n")
    return file_path

# Function to download the CSV file
def download_csv():
    file_path = save_expenses()
    if file_path:  # Only create a link if the file was successfully saved
        return FileLink(file_path)
    return None  # No link if the file wasn't saved

# Function to load expenses from a CSV file
def load_expenses():
    global expense_id
    try:
        with open('expenses.csv', mode='r') as file:
            reader = csv.reader(file)
            next(reader)  # Skip the header
            for row in reader:
                year, month = int(row[0]), int(row[1])
                expense = {
                    'id': int(row[2]),
                    'date': row[3],
                    'category': row[4],
                    'amount': float(row[5]),
                    'description': row[6]
                }
                if (year, month) not in expenses:
                    expenses[(year, month)] = []
                expenses[(year, month)].append(expense)
                expense_id = max(expense_id, expense['id'] + 1)
        print("Expenses loaded from file.\n")
    except FileNotFoundError:
        print("No saved expenses found.\n")
```

7 Function to Clear History

[9]:
```python
# Function to clear history
def reset_records():
    global expenses, budgets, expense_id
    expenses.clear()  # Clear the expenses dictionary
    budgets.clear()   # Clear the budgets dictionary
    expense_id = 1    # Reset the expense ID
    print("All records cleared.\n")
```

8 Main function to run the program

[22]:
```python
# Function to display the interactive menu
def display_menu():
    print("1. Set Monthly Budget")
    print("2. Add Expense")
    print("3. View Expenses for a Specific Month")
    print("4. Track Budget for a Specific Month")
    print("5. Save and Download Expenses")
    print("6. Clear History")
    print("7. Exit")

# Main function to run the program
def main():
    load_expenses()  # Load expenses when program starts
    while True:
        display_menu()
        choice = input("Choose an option: ")

        if choice == '1':
            set_budget()
        elif choice == '2':
            add_expense()
        elif choice == '3':
            view_expenses()
        elif choice == '4':
            track_budget()
        elif choice == '5':
            download_link = download_csv()
            if download_link:  # Check if the download link is valid
                print("Click the link below to download the CSV file:")
                display(download_link)
            else:
                print("No records found to download.\n")
        elif choice == '6':
```

```
            reset_records()  # Clear all records
        elif choice == '7':
            save_expenses()  # Save expenses before exiting
            print("Exiting the program. Goodbye!")
            break
        else:
            print("Invalid option. Please try again.\n")

# Run the main program
if __name__ == "__main__":
    main()
```

```
Expenses loaded from file.

1. Set Monthly Budget
2. Add Expense
3. View Expenses for a Specific Month
4. Track Budget for a Specific Month
5. Save and Download Expenses
6. Clear History
7. Exit

Choose an option:  6

All records cleared.

1. Set Monthly Budget
2. Add Expense
3. View Expenses for a Specific Month
4. Track Budget for a Specific Month
5. Save and Download Expenses
6. Clear History
7. Exit

Choose an option:  2
Enter the date (YYYY-MM-DD):  2012465265

Invalid date format! Please enter the date in YYYY-MM-DD format.

Enter the date (YYYY-MM-DD):  2024-09-11

No budget set for 2024-9. Please set a budget first to track expenses.

1. Set Monthly Budget
2. Add Expense
3. View Expenses for a Specific Month
4. Track Budget for a Specific Month
5. Save and Download Expenses
6. Clear History
7. Exit
```

```
Choose an option:  1
Enter the year (e.g., 2024):  2024
Enter the month (1-12):  09
Enter your budget for 2024-9:  20000.00

Monthly budget for 2024-9 set to $20000.00

1. Set Monthly Budget
2. Add Expense
3. View Expenses for a Specific Month
4. Track Budget for a Specific Month
5. Save and Download Expenses
6. Clear History
7. Exit

Choose an option:  2
Enter the date (YYYY-MM-DD):  2024-09-11
Enter the category (e.g., Food, Travel):  Food
Enter the amount:  14000
Enter a brief description:  Bday party

Expense added successfully with ID: 1

Total spent: $14000.00, Remaining budget for 2024-9: $6000.00

1. Set Monthly Budget
2. Add Expense
3. View Expenses for a Specific Month
4. Track Budget for a Specific Month
5. Save and Download Expenses
6. Clear History
7. Exit

Choose an option:  2
Enter the date (YYYY-MM-DD):  2024-09-14
Enter the category (e.g., Food, Travel):  Travel
Enter the amount:  4800.58
Enter a brief description:  Tour travel

Expense added successfully with ID: 2

Total spent: $18800.58, Remaining budget for 2024-9: $1199.42

1. Set Monthly Budget
2. Add Expense
3. View Expenses for a Specific Month
4. Track Budget for a Specific Month
5. Save and Download Expenses
6. Clear History
7. Exit
```

```
Choose an option:  3
Enter the year (e.g., 2024):  2024
Enter the month (1-12):  09


Expenses for 2024-9:
ID: 1, Date: 2024-09-11, Category: Food, Amount: $14000.00, Description: Bday
party
ID: 2, Date: 2024-09-14, Category: Travel, Amount: $4800.58, Description: Tour
travel

1. Set Monthly Budget
2. Add Expense
3. View Expenses for a Specific Month
4. Track Budget for a Specific Month
5. Save and Download Expenses
6. Clear History
7. Exit

Choose an option:  4
Enter the year (e.g., 2024):  2024
Enter the month (1-12):  08

No budget set for 2024-8. Please set a budget first.

1. Set Monthly Budget
2. Add Expense
3. View Expenses for a Specific Month
4. Track Budget for a Specific Month
5. Save and Download Expenses
6. Clear History
7. Exit

Choose an option:  4
Enter the year (e.g., 2024):  2024
Enter the month (1-12):  09

Total spent for 2024-9: $18800.58
You have $1199.42 left for the month.

1. Set Monthly Budget
2. Add Expense
3. View Expenses for a Specific Month
4. Track Budget for a Specific Month
5. Save and Download Expenses
6. Clear History
7. Exit

Choose an option:  2
Enter the date (YYYY-MM-DD):  2024
```

```
Invalid date format! Please enter the date in YYYY-MM-DD format.

Enter the date (YYYY-MM-DD):  2024-09-18
Enter the category (e.g., Food, Travel):  Food
Enter the amount:  5000
Enter a brief description:  Party

Expense added successfully with ID: 3


Warning: You have exceeded the budget for 2024-9!
Total spent: $23800.58, Budget: $20000.00

1. Set Monthly Budget
2. Add Expense
3. View Expenses for a Specific Month
4. Track Budget for a Specific Month
5. Save and Download Expenses
6. Clear History
7. Exit

Choose an option:  4
Enter the year (e.g., 2024):  2024
Enter the month (1-12):  09

Total spent for 2024-9: $23800.58
Warning: You have exceeded your budget!

1. Set Monthly Budget
2. Add Expense
3. View Expenses for a Specific Month
4. Track Budget for a Specific Month
5. Save and Download Expenses
6. Clear History
7. Exit

Choose an option:  5

Expenses saved to expenses.csv


Click the link below to download the CSV file:

/voc/work/expenses.csv

1. Set Monthly Budget
2. Add Expense
3. View Expenses for a Specific Month
4. Track Budget for a Specific Month
5. Save and Download Expenses
6. Clear History
7. Exit

Choose an option:  7
```

```
Expenses saved to expenses.csv

Exiting the program. Goodbye!
```