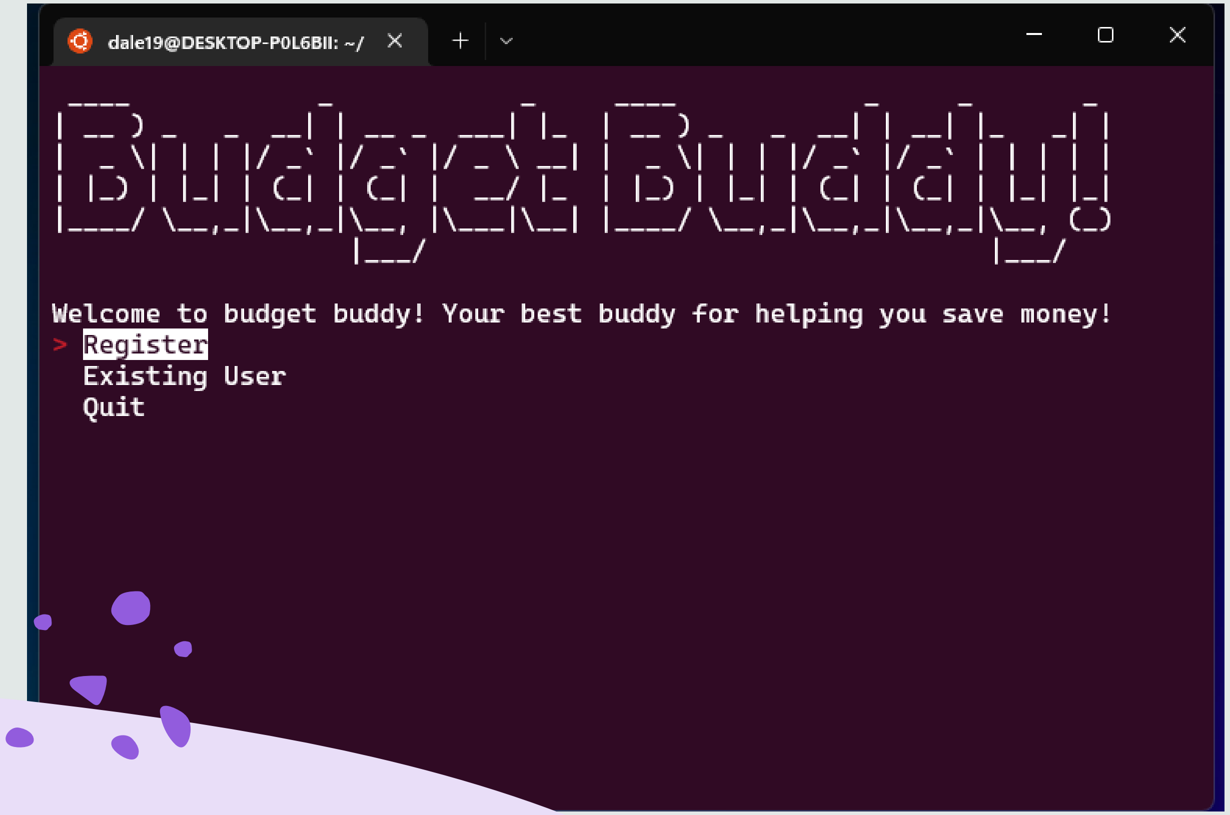
Your Budget Buddy!

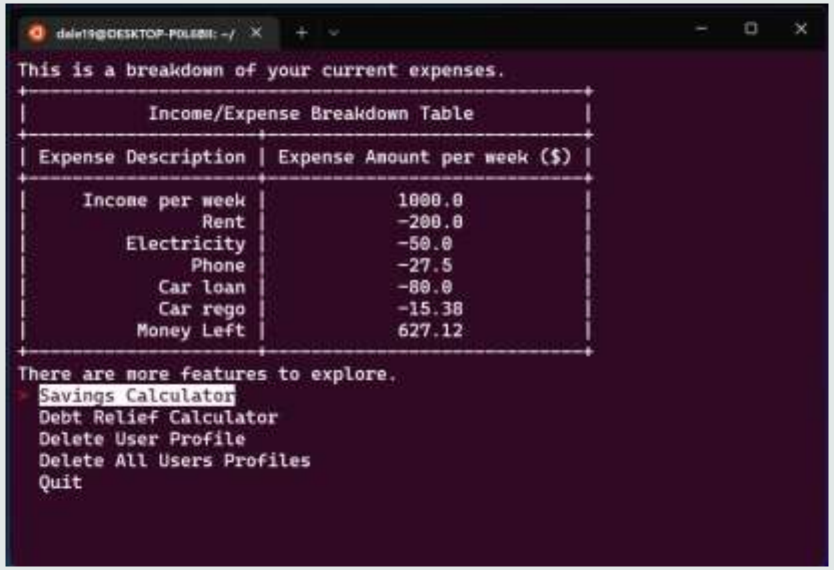
Who’s ready to start saving money

Walkthrough



```
 ____            _            _     ____            _     _       _ 
| __ ) _   _  __| | __ _  ___| |_  | __ ) _   _  __| | __| |_   _| |
|  _ \| | | |/ _` |/ _` |/ _ \ __| |  _ \| | | |/ _` |/ _` | | | | |
| |_) | |_| | (_| | (_| |  __/ |_  | |_) | |_| | (_| | (_| | |_| |_|
|____/ \__,_|\__,_|\__, |\___|\__| |____/ \__,_|\__,_|\__,_|\__, (_)
                   |___/                                    |___/   

Welcome to budget buddy! Your best buddy for helping you save money!
> Register
  Existing User
  Quit
```

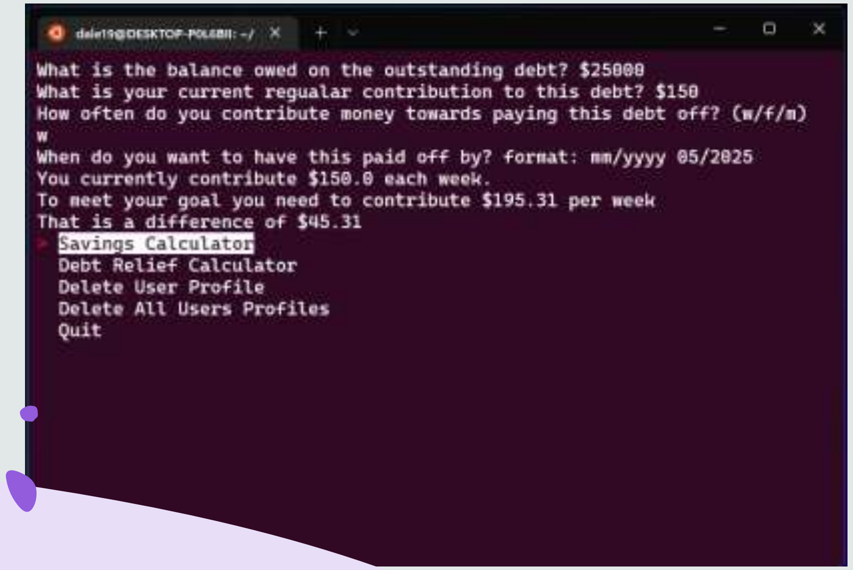
- The main menu provides the option for users to Register, Use an existing user or quit
- If there are no existing users it will prompt you to Create a new one



```
This is a breakdown of your current expenses.
+---------------------------------------------------------+
|             Income/Expense Breakdown Table              |
+---------------------------------------------------------+
| Expense Description | Expense Amount per week ($) |
+---------------------------------------------------------+
|     Income per week |           1000.0               |
|                Rent |           -200.0               |
|         Electricity |           -50.0                |
|               Phone |           -27.5                |
|            Car loan |           -80.0                |
|            Car rego |           -15.38               |
|          Money Left |           627.12               |
+---------------------------------------------------------+
There are more features to explore.
  Savings Calculator
  Debt Relief Calculator
  Delete User Profile
  Delete All Users Profiles
  Quit
```

Budget Breakdown

- After Creating a user or logging in to existing user, the application asks for inputs of income and any expenses.
- After gathering this information it will ask if you would like to have it displayed in a table as shown
- The menu after shows the next features that the user can access

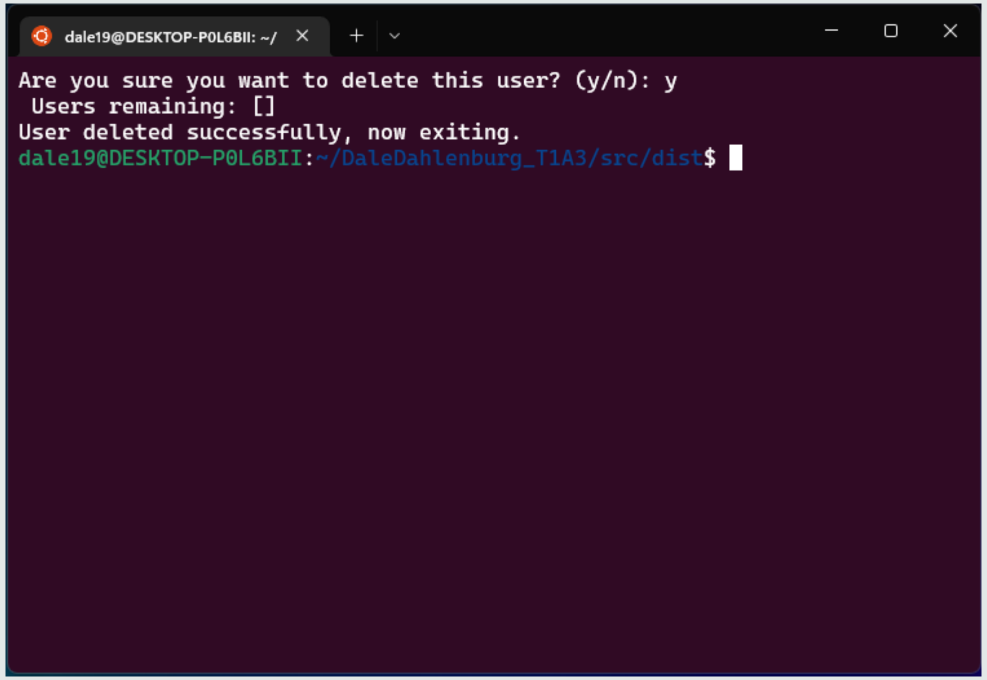


```
What is the balance owed on the outstanding debt? $25000
What is your current regualar contribution to this debt? $150
How often do you contribute money towards paying this debt off? (w/f/m)
w
When do you want to have this paid off by? format: mm/yyyy 05/2025
You currently contribute $150.0 each week.
To meet your goal you need to contribute $195.31 per week
That is a difference of $45.31
  Savings Calculator
  Debt Relief Calculator
  Delete User Profile
  Delete All Users Profiles
  Quit
```

Calculators Breakdown

- The debt and savings calculators are similar as they take inputs from the user to calculate a final message to inform the user
- In this example it shows how much more the user needs to contribute per week to reach there goal of paying this debt off by 05/2025

User Cleanup



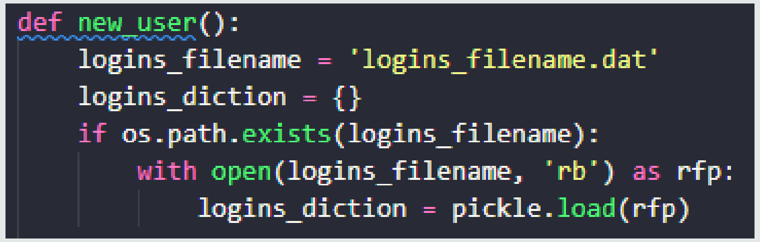
```
Are you sure you want to delete this user? (y/n): y
 Users remaining: []
User deleted successfully, now exiting.
dale19@DESKTOP-P0L6BII:~/DaleDahlenburg_T1A3/src/dist$ 
```

Delete User

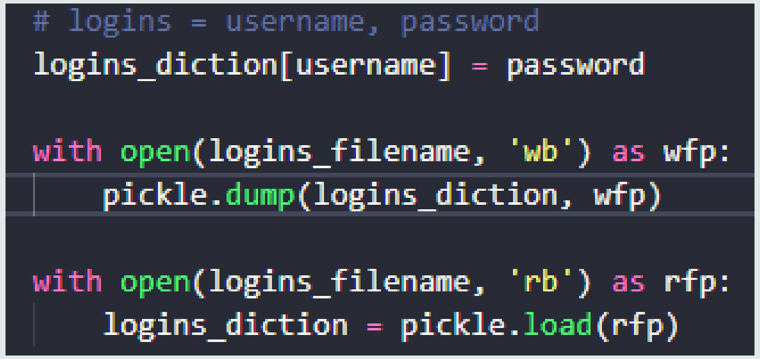
- The user is also offered the chance to delete their saved profile
- Also an option to delete all users, however do need a master password to accomplish this
- Delete all option as the code creates new files for new users, offers way to clean up folders

Login logic

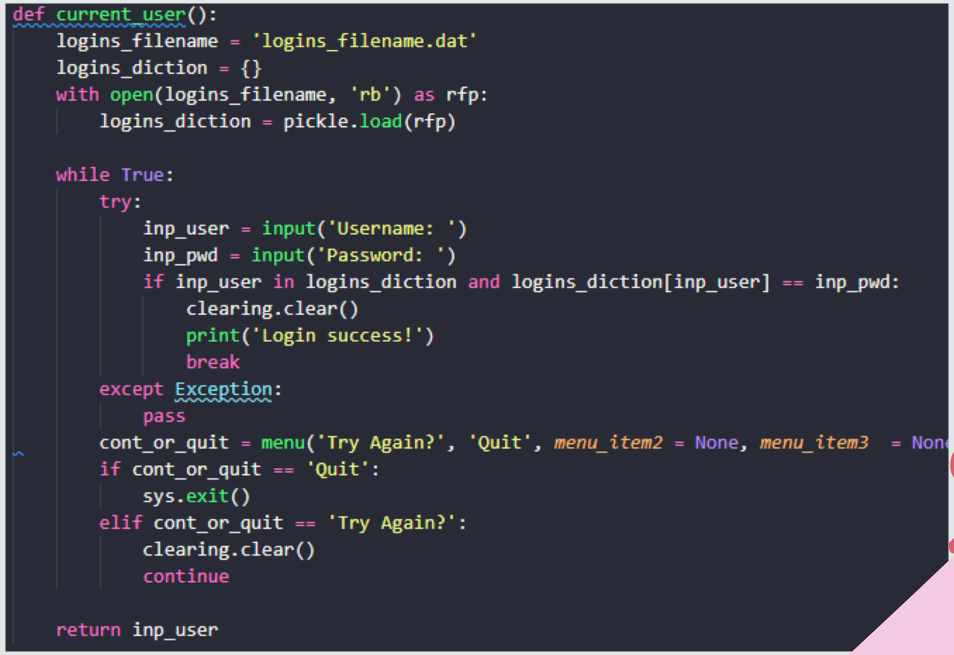
- This logic was crucial for repeat users
- Runs through how pickle creates new users
- Also shows how pickle opens existing files



```python
def new_user():
    logins_filename = 'logins_filename.dat'
    logins_diction = {}
    if os.path.exists(logins_filename):
        with open(logins_filename, 'rb') as rfp:
            logins_diction = pickle.load(rfp)
```



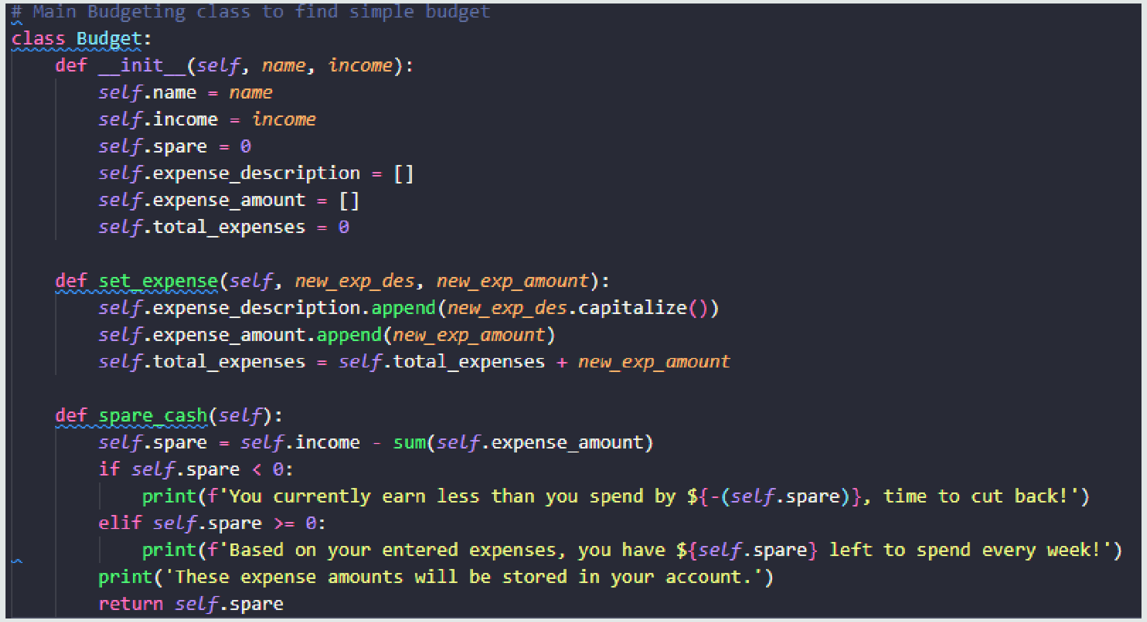
```python
# logins = username, password
logins_diction[username] = password

with open(logins_filename, 'wb') as wfp:
    pickle.dump(logins_diction, wfp)

with open(logins_filename, 'rb') as rfp:
    logins_diction = pickle.load(rfp)
```



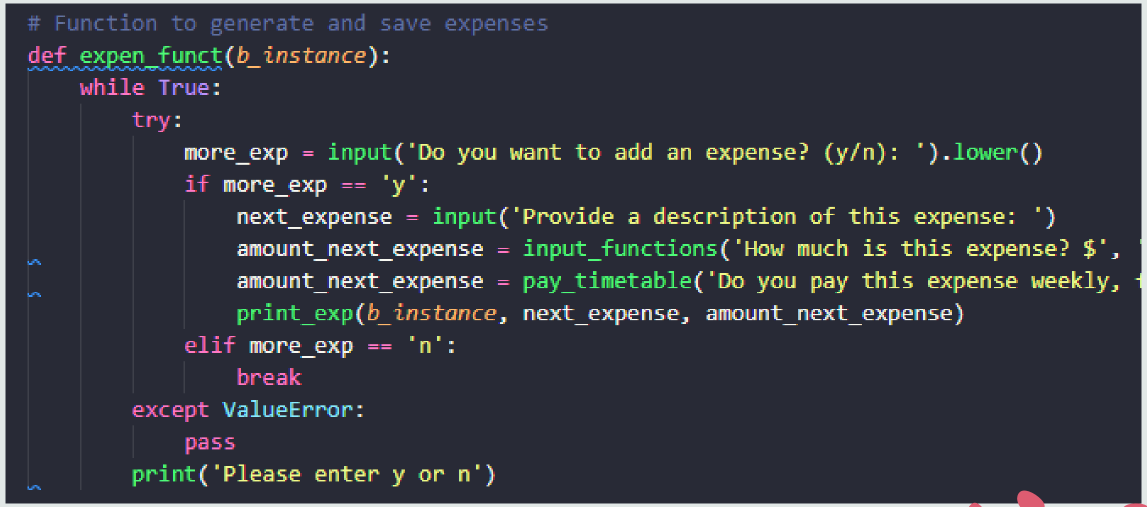
```python
def current_user():
    logins_filename = 'logins_filename.dat'
    logins_diction = {}
    with open(logins_filename, 'rb') as rfp:
        logins_diction = pickle.load(rfp)

    while True:
        try:
            inp_user = input('Username: ')
            inp_pwd = input('Password: ')
            if inp_user in logins_diction and logins_diction[inp_user] == inp_pwd:
                clearing.clear()
                print('Login success!')
                break
        except Exception:
            pass
        cont_or_quit = menu('Try Again?', 'Quit', menu_item2 = None, menu_item3  = Non
        if cont_or_quit == 'Quit':
            sys.exit()
        elif cont_or_quit == 'Try Again?':
            clearing.clear()
            continue

    return inp_user
```

Budget Logic

- Uses Budget Class to be able to pass in user input data to create lists of expenses
- These lists are then appended with user input data that has been correctly formatted
- Functions also incorporate other previously stated functions to create DRY code

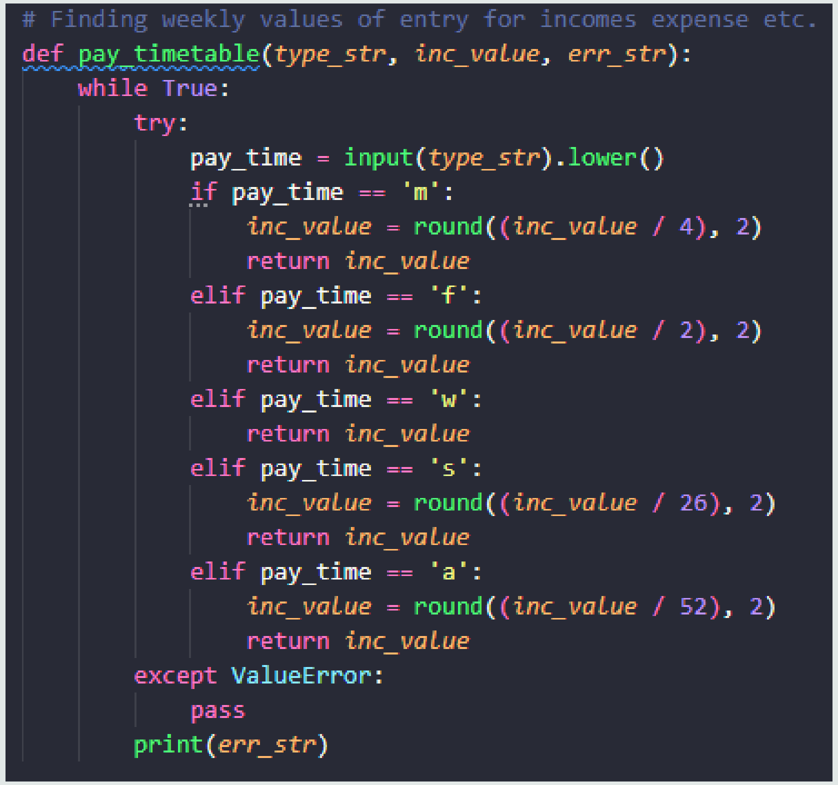


```python
# Main Budgeting class to find simple budget
class Budget:
    def __init__(self, name, income):
        self.name = name
        self.income = income
        self.spare = 0
        self.expense_description = []
        self.expense_amount = []
        self.total_expenses = 0

    def set_expense(self, new_exp_des, new_exp_amount):
        self.expense_description.append(new_exp_des.capitalize())
        self.expense_amount.append(new_exp_amount)
        self.total_expenses = self.total_expenses + new_exp_amount

    def spare_cash(self):
        self.spare = self.income - sum(self.expense_amount)
        if self.spare < 0:
            print(f'You currently earn less than you spend by ${-(self.spare)}, time to cut back!')
        elif self.spare >= 0:
            print(f'Based on your entered expenses, you have ${self.spare} left to spend every week!')
        print('These expense amounts will be stored in your account.')
        return self.spare
```



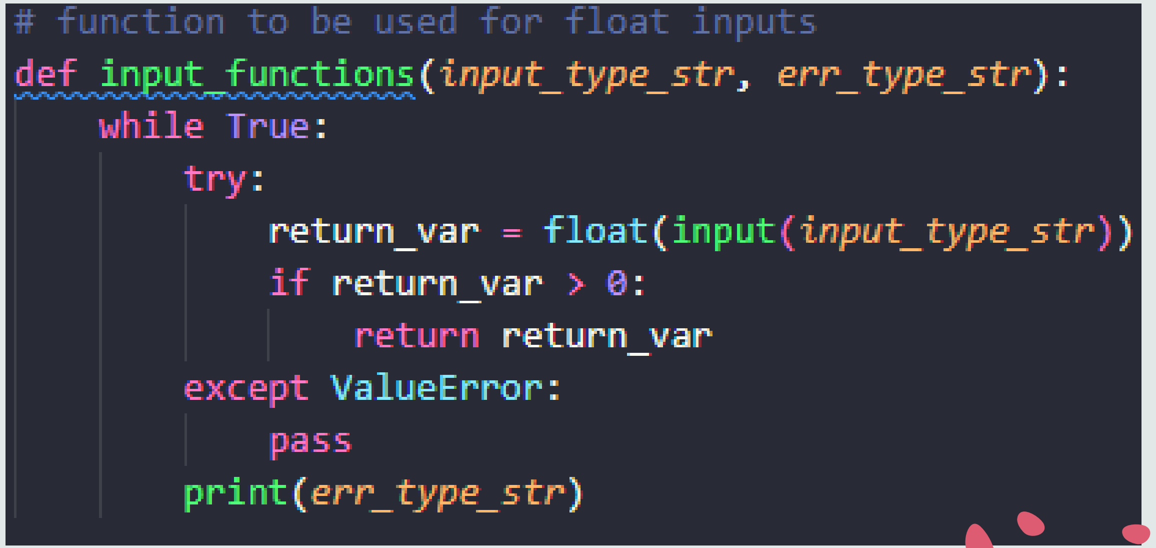
```python
# Function to generate and save expenses
def expen_funct(b_instance):
    while True:
        try:
            more_exp = input('Do you want to add an expense? (y/n): ').lower()
            if more_exp == 'y':
                next_expense = input('Provide a description of this expense: ')
                amount_next_expense = input_functions('How much is this expense? $',
                amount_next_expense = pay_timetable('Do you pay this expense weekly,
                print_exp(b_instance, next_expense, amount_next_expense)
            elif more_exp == 'n':
                break
        except ValueError:
            pass
        print('Please enter y or n')
```

Error Handling



```
# Finding weekly values of entry for incomes expense etc.
def pay_timetable(type_str, inc_value, err_str):
    while True:
        try:
            pay_time = input(type_str).lower()
            if pay_time == 'm':
                inc_value = round((inc_value / 4), 2)
                return inc_value
            elif pay_time == 'f':
                inc_value = round((inc_value / 2), 2)
                return inc_value
            elif pay_time == 'w':
                return inc_value
            elif pay_time == 's':
                inc_value = round((inc_value / 26), 2)
                return inc_value
            elif pay_time == 'a':
                inc_value = round((inc_value / 52), 2)
                return inc_value
        except ValueError:
            pass
        print(err_str)
```

- As the script accepts user input frequently, error handling for user input was developed using while-True loops
- Each of the different input functions had the error handling directly in the function



```
# function to be used for float inputs
def input_functions(input_type_str, err_type_str):
    while True:
        try:
            return_var = float(input(input_type_str))
            if return_var > 0:
                return return_var
        except ValueError:
            pass
        print(err_type_str)
```