# CS177

# Project 2: Writing a Budget Program

In this project you will write a budget program. The project has been divided in multiple steps to make it easier to implement.

## Setting up your Environment

Go to your working directory in “data.cs.purdue.edu” and create a directory “cs177/project02”. Refer to the lab1 if you need to remember the steps to do so. Then start the IDLE Python Interpreter.

## Input Files

The program will use two input files: budget.txt and expense.txt.

budget.txt – Budget File

It is a file that contains the expense type and the maximum expense allowed for this type. The Budget Type and the MaxAmount are separated by coma characters and are between quotes. The Budget file contains the following fields:

"ExpenseType" - This is the expense type, such as MEALS, SCHOOL etc

"MaxAmount" - This is the maximum amount that has been allocated to this expense.

|  |
| --- |
| **budget.txt:**  "ExpenseType", "MaxAmount"  "SCHOOL","$100.00"  "UTILITIES","$200.00"  "AUTO", "$100.00"  "RENT", "$600.00"  "MEALS", "$300.00"  "RECREATION", "$100.00" |

expenses.txt – Expenses File

It is a file that contains the expense entries, one expense for each line. This is a file that very likely you can obtain from your bank account web page. The format of each line is the following:

* **"Date" - Date of the expense**
* **"Description" - Description of the expense**
* **"Type" - Type of the expense**
* **"Check Number" – The check number if expense was paid by check.**
* **"Amount" - The amount of the expense.**
* **"Balance" - The balance in the account at the time the expense was made.**

Here is an example of the expenses.txt file.

|  |
| --- |
| **expenses.txt**:  "Date","Description","Type","Check Number","Amount","Balance"  "02/03/2014","INDIANA - AMERIC - 99999999999 - PAYMENT","UTILITIES","","($30.00)",""  "02/12/2014","COMCAST NW - 9999999 - LAFAYETTE","UTILITIES","","($101.53)",""  "02/23/2014","SHARE DRAFT # 346 VECTREN GAS","UTILITIES","346","($60.00)",""  "02/24/2014","DUKE ENERGY IN - 9999999999 - WEB\_PAY","UTILITIES","","($52.45)","4"  "02/24/2014","STAR SIMPSON BP 3827 WEST LAFAYETTINUS","AUTO","","($51.17)",""  "02/24/2014","SHARE DRAFT # 345 ","RENT","345","($600.00)",""  "02/25/2014","PURDUE PMU RETAIL OPS WEST LAFAYETT IN","MEALS","","($5.10)",""  "02/25/2014","STARBUCKS #02856 WEST LAF West Lafayett IN","MEALS","","($2.05)",""  "02/26/2014","AMAZON MKTPLACE PMTS AMZN.COM/BILL WA","SCHOOL","","($30.00)",""  "02/26/2014","STARBUCKS #02856 WEST LAF West Lafayett IN","MEALS","","($3.05)",""  "02/27/2014","APL\*APPLE ITUNES STORE 866-712-7753 CA","RECREATION","","($1.38)","$"  "02/28/2014","PURDUE PMU RETAI ","MEALS","","($4.10)","" |

The goal of this project is to read the budget.txt and expense.txt files, determine how much money has been spent in a specific expense type and tell you if there is an over budget.

Download the initial files from https://www.cs.purdue.edu/homes/cs177/Projects/project2/project02.zip

## Step 1. How to read the budget.txt file

For reading the budget.txt file we give you the following example code of the program budget.py. It is important that you understand well this code since it will help you write the other parts of the program. The function readBudget reads all the lines of the file and puts them in a list called lines. It uses the function lines[i].split(",")to split the line into multiple fields and stores it in list. It then extracts the expense type exptype and max amount maxamount from the line and strips them from any quotes and $ sign. It then creates a dictionary entry with the fields ‘exptype’ and ‘maxamount’ that is added to the list budget that contains all the budget entries.

|  |
| --- |
| **budget.py:**  def readBudget(budgetFile):  # Read the file into list lines  f = open(budgetFile)  lines = f.readlines()  f.close()  budget = []    # Parse the lines  for i in range(len(lines)):  list = lines[i].split(",")    exptype = list[0].strip('" \n')  if exptype == "Type":  continue    maxamount = list[1].strip('$" \n')  entry = {'exptype':exptype, 'maxamnt':float(maxamount)}  budget.append(entry)  #print(budget)    return budget  def printBudget(budget):  print()  print("================= BUDGET ==================")  print("Type".ljust(12), "Max Amount".ljust(12))    total = 0  for b in budget:  print(b['exptype'].ljust(12), str("$%0.2f" %b['maxamnt']).ljust(50))  total = total + b['maxamnt']    print("Total: ", "$%0.2f" % total)    def Main():  budget = readBudget("budget.txt")  printBudget(budget)    if \_\_name\_\_ == '\_\_main\_\_':  Main() |

The function printBudget iterates over all the budget entries and prints them in a nice format. It uses **"$%0.2f" % num** for formatting of the decimals and it uses ljust and rjust form alignment. This is an example of the output:

|  |
| --- |
| >>>  ================= BUDGET ==================  Type Max Amount  SCHOOL $100.00  UTILITIES $200.00  AUTO $100.00  RENT $600.00  MEALS $300.00  RECREATION $100.00  Total: $1400.00  >>> |

## Step 2

Write a program readExpense.py that it will read the expenses.txt file and it will print the expenses in a table similar to what was printed in step 1. We suggest you to write a function that reads the expenses and a function that prints the expenses. Use as base the code in budget.py. The Main() program is called only when readExpense.py is run directly and not when it is imported from another file.

|  |
| --- |
| #  # This file has the same format as the one you download from a bank  # like purduefed.com  #  # You have to put in the Comments section the expense type.  #  # It returns a list of dictionaries with: date, desc, exptype, checknum, amnt  #  def readExpenses(file):  # Write your code here  # Read file into list of lines  # Split lines into fields  # For each list create a dictionary  # Add dictionary to expense list.  #It returns expenses is a list of dictionaries with fields  # date, desc, exptype, checknum, amnt  return expenses  def printExpenses(expenses):  # Print expenses  print()  # Write your code here  def Main():  expenses = readExpenses("expenses.txt")  printExpenses(expenses)    if \_\_name\_\_ == '\_\_main\_\_':  Main() |

Here is the output expected of readExpense.py:

|  |
| --- |
| >>>  ================================== EXPENSES ===================================  Date Description Type Check Amount  02/03/2014 INDIANA - AMERIC - 99999999999 - PAYMENT UTILITIES $30.00  02/12/2014 COMCAST NW - 9999999 - LAFAYETTE UTILITIES $101.53  02/23/2014 SHARE DRAFT # 346 VECTREN GAS UTILITIES 346 $60.00  02/24/2014 DUKE ENERGY IN - 9999999999 - WEB\_PAY UTILITIES $52.45  02/24/2014 STAR SIMPSON BP 3827 WEST LAFAYETTINUS AUTO $51.17  02/24/2014 SHARE DRAFT # 345 RENT 345 $600.00  02/25/2014 PURDUE PMU RETAIL OPS WEST LAFAYETT IN MEALS $5.10  02/25/2014 STARBUCKS #02856 WEST LAF West Lafayett IN MEALS $2.05  02/26/2014 AMAZON MKTPLACE PMTS AMZN.COM/BILL WA SCHOOL $30.00  02/26/2014 STARBUCKS #02856 WEST LAF West Lafayett IN MEALS $3.05  02/27/2014 APL\*APPLE ITUNES STORE 866-712-7753 CA RECREATION $1.38  02/28/2014 PURDUE PMU RETAI MEALS $4.10  Total: $940.83  >>> |

## Step 3

Write a program called expenseByType.py that it will print the total of the expenses by type. It will read both the expense.txt file and the budget.txt file and it will print the total for each expense type. The

|  |
| --- |
| from readExpense import \*  from budget import \*  # Returns the expenses by expense type  def expensesByType(expenses, budget):  # Create a list of expenses by expense type  # The list contains dictionaries for each  # expense type with the fields exptype, amnt, and maxamt    # Write your code here    return expByType;  def printExpenseByType(expByType):  print()  # Prints the expense type and adds overbudget if total  # exceeds amount.  # Write your code here  def Main():  budget = readBudget("budget.txt")  #printBudget(budget)  expenses = readExpenses("expenses.txt")  #printExpenses(expenses)  expByType = expensesByType(expenses, budget)  printExpenseByType(expByType)    if \_\_name\_\_ == '\_\_main\_\_':  Main() |

Here is an example of the output:

|  |
| --- |
| ================= EXPENSE BY TYPE ==================  Type Max Amount Current Condition  SCHOOL $100.00 $30.00  UTILITIES $200.00 $243.98 Over budget  AUTO $100.00 $51.17  RENT $600.00 $600.00  MEALS $300.00 $14.30  RECREATION $100.00 $1.38  Total: $940.83 |

## Turnin in Steps 1,2,3

You will have to turn in step 1, 2, 3 before 11:59pm March 14th.

1. Create a directory cs177/project2

2. Copy all the files budget.py, readExpense.py, expenseByType.py, (if any). Include also other files you use.

3. Login to data.cs.purdue.edu and type:

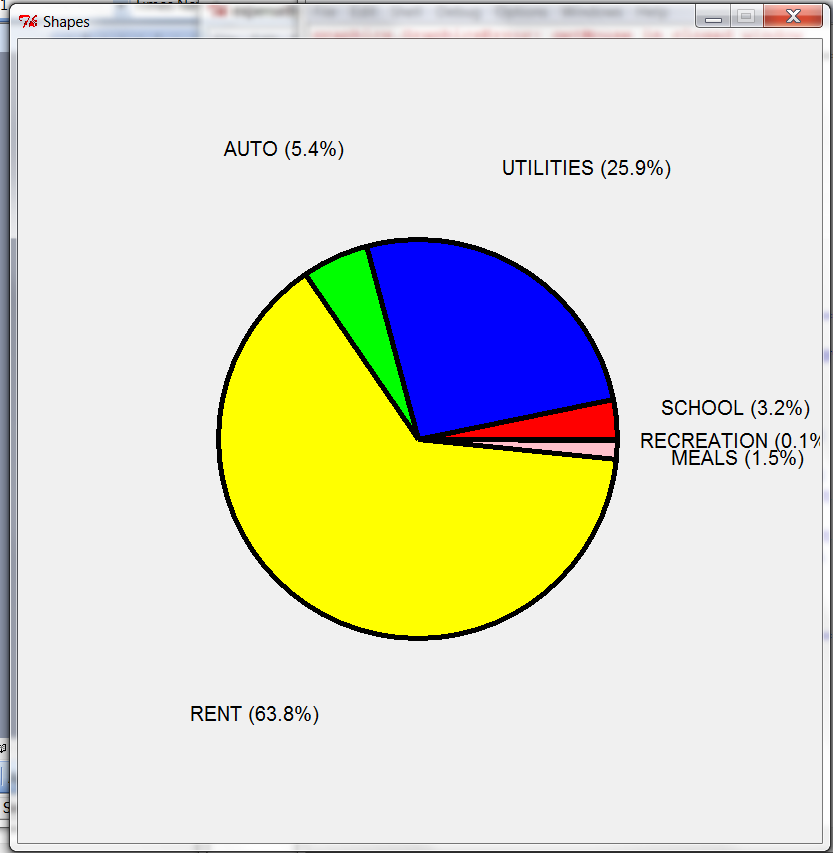
cd cs177

turnin –c cs177 –p project2-1 project2

This submission will count for 10% of the grading of the project. Submit your files even if they don’t work as expected to get a partial grade. You may continue working on these files after your submission.

## Step 4. Draw an Expense Paychart

Draw an expense pie chart like the following one that shows the expenses as a percentage of the total. Only use the graphics.py library. Hint: Draw lines of different colors from the center of the screen (0,0) to the edges of the circle (R\*cos(2\*pi\*i/n),R(sin(2\*pi\*i/n)) where n is the total number of lines and I goes from 0 to n. Change the color of the lie according to the percentage displayed.



## Turnin

This program is due 11:59pm March 28th.

1. Create a directory cs177/project2

2. Copy all the files budget.py, readExpense.py, expenseByType.py, and piechart.py (if any). Include also other files you use.

3. Login to data.cs.purdue.edu and type:

cd cs177

turnin –c cs177 –p project2-2 project2

Start early!

# Grading Form

# Project 2

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Login:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| Feature | Max Grade | Current Grade |
| Step 2  Correct Values (20p)  Aligned (10p)  Formatting (10p) | 30 |  |
| Step 3  Correct Values (20p)  Aligned (10p)  Formatting (10p) | 40 |  |
| Step 4  Pie draw correctly (15p)  Slices are proportioned (5p)  Text (5p)  Looks nice (5p) | 30 |  |
|  | Total: |  |