Practicing Some Old Tricks and Learning Some New Ones

Learning Objectives:

- Learn how to use `eval()`
- Learn how to use `round()` and `divmod()`
- Practice using floor division and modulo

Prerequisites: Before starting this lab, you should be able to:

- Run Python interactively in the IDLE Shell window
- Write and save programs (scripts) using the IDLE Editor window
- Run saved programs from IDLE
- Compose programs using good programming habits, that have headers, and that make use of comments, variables, basic arithmetic, data types, and basic I/O statements.

Task 0: If you haven’t done so already, open an account on Piazza (piazza.com). Be sure to activate your account and choose your Lab group by going to the Resources link at the top and then the Groups link.

Task 1: Parsing A Four-digit Number (pair task)

Using the IDLE Editor window, write a program that prompts a user for a positive, four-digit integer and then prints the digits one line at a time with the most significant digit first. You can use floor division and the modulo functions for this problem, but a more efficient way is to use `divmod()` as shown in class. Experiment with the `divmod()` function in the IDLE Shell window to see how it works. Save this program and all other programs in a new folder under CS111 called Lab2.

Example results are shown below, but try other test cases!

```
Enter a positive, four-digit integer: 1234
1
2
3
4
Enter a positive, four-digit integer: 5023
5
0
2
3
```
When you’ve completed this task, demonstrate it to your TA for credit.

**Task #2: Creating a Simple Calculator (pair task)**

Python provides a very powerful function which should be used with caution because of its power. It’s the `eval()` function; it evaluates any string argument as a Python expression. For example,

```python
>>> string = '5 + 12'
>>> print(string)
5 + 12
>>> type(string)
<class 'str'>
>>> eval(string)
17
>>> result = eval(string)
>>> type(result)
<class 'int'>
>>> print(string, '=', eval(string))
5 + 12 = 17
>>> eval('print('Hello, World!')')
Hello, World!
```

Using the `eval()` function, write a short program that will prompt the user for an arithmetic expression and store it as a string. This expression may consist of any of the standard Python arithmetic operators (`+`, `-`, `*`, `/`, `//`, `%`, and `**`), parentheses, the `divmod()` function, and `int` or `float` literals. Next print this expression together with an equal sign and the value to which the expression evaluates.

Your results should be similar to the examples shown below.

**Enter an arithmetic expression:** `1 + 1`

`1 + 1 = 2`

**Enter an arithmetic expression:** `
`100 ** 3 / 3`

`100 ** 3 / 3 = 333333.333333`

**Enter an arithmetic expression:** `(5 + 3.2 / (943 % 37) - 43.12 ** 3)`

`(5 + 3.2 / (943 % 37) - 43.12 ** 3) = -80169.32155022222`

After you’ve completed this task, demonstrate it to your TA to get credit.

**Task 3: Displaying Dollars (pair task)**

When it comes to dealing with money, numbers are rounded to the nearest cent. For example, suppose we buy something for $8.99 and the sales tax is 8.1%. The total cost should be $9.71819, but because one cent is the smallest denomination of currency we have, we’re charged $9.72. Thus, programs written for any company dealing with U.S. currency have to make sure it is displayed
properly. If our bank statement showed a balance of $125.1 or $125.1349, we would have serious concerns about our bank!

In this task you are to write a program that prompts the user for any value in decimal form and displays it in the standard form for money. There are many ways to accomplish this, but I suggest the following: multiply the number by 100, use the round() function to delete unwanted decimal digits, use the divmod() function twice (first to get dollars and cents and then to get dimes and pennies). Experiment with the round() function in the the Shell window of IDLE to get a feeling for how it works.

Some examples are given below.

1. Enter a decimal number: 93.0102235
2. $93.01
3. Enter a decimal number: 93.0192235
4. $93.02
5. Enter a decimal number: 0.257897
6. $0.26

Demonstrate your program to your TA to get credit.

__________________________

Task #4: Making Change (pair task)

Write a program that prompts a user for the total cost of a purchase and the cash payment made using a U.S. currency bill. Your program should display the amount of change due to the customer as well as the number of dollars, quarters, dimes, nickels, and pennies.

Here’s an example result.

1. Enter the cost of the purchase: 14.32
2. Enter the cash payment (U.S. currency bill): 20
3. Change due: $5.68
4. Dollars: 5
5. Quarters: 2
6. Dimes: 1
7. Nickels: 1
8. Pennies: 3

__________________________

Task #5: Making Your zyLab PA #1 Output Display Prettier! (pair task)

You can now display the output of your first program properly! Modify your Student Loan Repayment Calculator program so the values reported are in the standard form for money using the algorithm you developed in the previous task. As a reminder, the equation for a monthly payment
is given by:

\[
p = \frac{P(i/12)}{1 - (1 + i/12)^{-n}} \tag{1}
\]

where \( p \) is the monthly payment, \( P \) is loan amount, \( i \) is the annual percent interest rate (must be a fraction in the program), and \( n \) is the number of monthly payments.

Here’s an example of my output:

1. Enter the amount you owe [no commas]: 30000
2. Enter the interest rate [%]: 6.8
3. Enter the number of years you want to spend to pay back your loan: 10
4. Your monthly payment is $345.24.
5. The total amount you ended up paying is $41428.92.
6. The total amount of interest you paid is $11428.92.

Demonstrate your program to your TA to get credit.

---

**Task #6: The Doomsday Algorithm**

The Doomsday Algorithm [https://en.wikipedia.org/wiki/Doomsday_rule](https://en.wikipedia.org/wiki/Doomsday_rule) allows you to tell what day of the week it is for any given year, but you must first know the Doomsday for the year, and to figure that out, you have to know the anchor day. The anchor day for the years 2000-2099 is Tuesday which has a numeric value of 2. A year’s Doomsday is given by the equation:

\[
\text{Doomsday} = (((\text{yr} \div 12) + (\text{yr} \mod 12) + (\text{yr} \mod 12) \div 4) \mod 7 + \text{anchor}) \mod 7 \tag{2}
\]

where \( \text{yr} \) is the last two digits of the year and the anchor day is an integer between 0 and 7 with 0 corresponding to Sunday and 7 to Saturday.

Write a program to calculate a year’s Doomsday. Prompt a user for the last two digits of the year and for the anchor day as an integer. Print out the Doomsday.

Example results are shown below.

1. Enter the last two digits of the year: 05
2. Enter the anchor day as an integer [0=Sunday, 7=Saturday]: 2
3. Doomsday = 1
4. Enter the last two digits of the year: 17
5. Enter the anchor day as an integer [0=Sunday, 7=Saturday]: 2
6. Doomsday = 2

In January, Doomsday falls on Jan. 3rd (except during leap years) and in February and March it falls on the last day of February. After that it falls on the day of the month for even months (e.g.,
on the 4th of April and 12th of December), and for odd months you use the mnemonic, 9-to-5 at the 7-11 (an old commercial from the 20th century). Thus, for May Doomsday falls on 5/9, for July on 7/11, for September on 9/5, and for November on 11/7. This means that if your lab is on 9/5/17, it must be Tuesday because Tuesday is this year's Doomsday.

After you’ve completed this task, demonstrate it to your TA to get credit.