

ITSE 1359 – Lab 2 Assignment (Refer to Ch. 1 and Ch. 2 as needed):

NOTE: This lab requires you to work on and submit two programs. Read this document carefully and comply with submission and grading criteria.

In your first problem (Interest on Loan) you are provided a video to help you walk through the steps to build it. See end of this document for the link to the video. For your second program (Total Purchase) focus on the problem definition and pseudocode to help you develop your solution.

The purpose of this lab assignment is to learn how to work with numeric variables, expressions, and f strings. Your output screens should match those provided. Do not deviate from requirements.

Problem #1: Interest on Loan (interest.py)

Problem Definition: Create a Python program that calculates the total interest paid on a loan. The following coding concepts must be exhibited: numeric variables, formula to compute interest on loan, use of escape codes to align data, and currency formatting. Name your program interest.py. Do not forget general and specific comments.

Coding: Create variables for years of loan, interest rate, loan amount, and total interest paid. Initialize years to 15, rate to 6, and loan to 5000. You can initialize interest to 0.0 if you like. Calculate total interest paid using this formula: $\text{loan} * (\text{rate}/100) * \text{years}$.

Output: Use f strings to print out Loan amount, Interest rate, No. of years, and Interest paid. To format money use `${salary:,.2f}}`. Output must match screenshot below and money must be formatted as currency. Be sure to use the tab escape code to create the second column of output.

Screenshot of Output (interest.py):

Your completed work should look like this when you run it:

```
Loan amount:    $5,000.00
Interest rate:   6%
No. of years:    15
Interest paid:   $4,500.00
[Finished in 0.1s]
```

Specific Comments (interest.py):

```
# declare and initialize variables for years, rate, and loan
# calculate total interest ( interest = loan * (rate/100) * years )
# use formatted strings to show output
```

Code Example (interest.py):

Formatting for currency (not in book):

```
print(f"Interest paid: \t${interest:,.2f}")
```

The `:,` adds a comma as a thousand separator, and the `.2f` limits the string to two decimal places (or adds enough zeroes to get to 2 decimal places, as the case may be) at the end. The `$` is added before the variable placeholder.

Problem #2: Total Purchase (total_purchase.py)

Problem Definition: Create a Python program that calculates the total purchase price of four items. These items can be anything you can buy at a department store and you can price the items as you like. However, you may want to use my data, to make sure your program is working right and matches output screen shots. Name your file “total_purchase.py”. Your code should show use of variables, formulas, and f strings.

Coding: Assume a sales tax of 6% which you should store in a constant. You should use variable names to hold the price of each of the four items. You should calculate subtotal, sales tax, and grand total. To calculate subtotal, add the price of the four sales items. To calculate tax, multiply subtotal by the tax rate. To find grand total add subtotal and tax.

Output: Your program should show the name of the program at the top, followed by a listing of the things purchased and their price. Also show subtotal, sales tax, and grand total, all formatted for currency. Single and double lines should be used to demark the sections of the output. The output should simulate two columns, with the right column being right aligned:

```

Program:  Total Purchase

-----
Item #1 - Pants:      $ 175.00
Item #2 - Shirt:      $ 125.00
Item #3 - Shoes:      $ 300.00
Item #4 - Belt:       $ 100.00
-----
Sub Total:            $ 700.00
Sales Tax:            $  42.00
Grand Total:          $ 742.00
=====

[Finished in 0.1s]
```

Pseudocode/Comments (total_purchase.py):

```
# declare constant for sales tax rate and set to 6%
# initialize numeric variables to hold price of four sales items

# calc subtotal
# calc sales tax
# calc grand total

# print name of program
# display single dash line as a separator line
# display items and their cost in f string in two columns

# display single dash line as a separator line
# display subtotal, sales tax, and grand total
# display double dash line as a separator line
# print statement used to add blank line at the end
```

Coding Examples (total_purchase.py):

How to declare a constant?

```
TAX_RATE = 0.06
```

- Note Python really does not have constants, but we can code this way to simulate the idea.
- Weblink: [Does Python have Constants?](#)

Why use separator lines?

- Use a single dash to separate sections of the output and use a double dash line at the very end. This has significance in the field of Accounting.

How do you create the separator lines?

- You can manually do them:

```
# display single dash line
print("-----")
```

- Or you can multiply a string:

```
# display double line
print(28 * "=")
```

How can I learn more about multiplying strings?

- Weblink: [How to use Python to Multiply String](#)

How to right align the right column?

- You can use escape codes (\t):

```
# display items and their cost in f string in two columns
print(f"Item #1 - Pants:\t${price_pants:>7,.2f}")
```

- You might find it easier to simply use blank space:

```
# display dividing line and then subtotal, sales tax, and grand total
print(f"Sub Total:          ${sub_total:>7,.2f}")
```

- You just got to make sure things are aligned right and this might change depending on how big or small your numbers are, so you may have to experiment.

What is that >7 after the : in the formatting of the variables?

- In the placeholder you find a variable and a colon (:). The colon means that the variable will be formatted as follows.
- The > means right align and the 7 is the number of digits in the field (the field size). You may have to experiment with this field size depending on how small or large your data is.

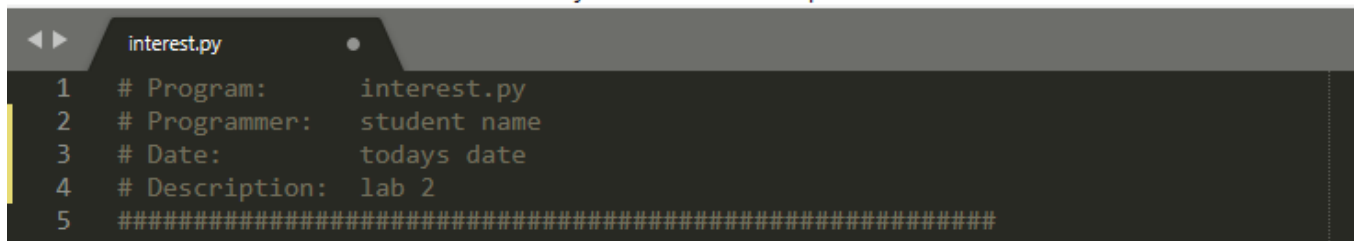
How can I learn more about formatting strings and numbers?

- That is beyond this class. I have already gone beyond the book.
- The internet has tons of stuff on this topic. Google it.
- Weblink: [Python f-string tutorial](#)

Comments in your Programs:

You should include general and specific comments in each program that you submit. General comments should appear at the top and include program name, programmer's name, date, and description. Specific comments should appear along with your code and briefly document the logic of your program. You can use the comments provided in this lab assignment or condense them. Points off for non-compliance.

Example of General Comments:



```

1 # Program:      interest.py
2 # Programmer:   student name
3 # Date:        todays date
4 # Description:  lab 2
5 #####

```

Submit your lab assignment:

Using the Canvas assignment tool, upload your completed work (**2 files**) to the **lab 2 assignment**. Attach the first file (interest.py) and then the second (total_purchase.py) and then submit.

All lab assignments must be submitted using the CANVAS assignments tool. **Lab Assignments will not be accepted any other way.** Make sure you submit your work to the right lab assignment number otherwise you will not get credit.

Food for Thought:

Using float variables in this class is OK but professionals will need something more precise when dealing with large numbers and money. Type decimal is used for precise monetary values. We will not cover this and other advanced topics but stick to basic programming concepts. You are invited to explore this topic independently if you like.

Grading Criteria:

- ✓ You must use the provided video to help you do your lab.
- ✓ Don't forget general comments.
- ✓ Specific comments are optional.
- ✓ Use white space to make your code easy to read.
- ✓ Comply with PEP-8 conventions for variable names, file names, etc.
- ✓ Satisfy the problem definition and other grading standards.
- ✓ Your work should not have syntax errors.
- ✓ Your work must be your own.
- ✓ Match your output screen to screenshot provided.
- ✓ Any deviations from lab specifications will result in points off.
- ✓ If you want to vary, do so on your own.

VIP VIDEOS:

How to do Interest on Loan (step-by-step): [Lab 2 Interest on Loan Video](#)

Lectures and other videos: [VIEW MY PYTHON PLAYLIST](#)

Closing:

If you have questions about this lab send me a message using canvas inbox or attend zoom office hours. See Unit 0 for link for my office hours. Programming tutors are also available and listed in your Canvas class as announcements.

By-the-way, a great way to get ready for your lab assignments (and exams) is do the Try It Yourself problems in your book. Most of the [solutions](#) are on the authors website and I also discuss them in my YouTube video lectures. Another way to get ready for your lab assignments is to review my lecture notes for each chapter. In Unit 0, find the link for my chapter notes.

Warning: Labs will increase in complexity with each lab assignment. A beginner may need 5 to 15 hours to complete each lab. Please start early because there will be no extensions.

Consider doing this lab over a three-day period:

- ✓ Day 1 – Get confused and run out of time.
- ✓ Day 2 – Research and debug errors.
- ✓ Day 3 – Polish, double-check everything, and submit.

Have an exception free day!

Prof. Benavides