DS2001 - Computer Science Practicum #2
Fall 2021
September 15 - 16, 2021
Deadline: September 17, 2021 at 12pm noon Boston time via Gradescope

What we're practicing today:

- Getting comfortable with data types
- variables
- Type conversion & return values
- Dealing with user input
- Printing and string formatting

Handouts:

- variables & input (from DS 2000)
- formatting output & numbers (from DS 2001)

What to do if you miss practicum this week:

- Fill out the "I'm missing class" form ASAP
- Follow up in office hours with Felix (find office hours links on the course website)
- Make sure to come to practicum next week if you are well again. **Do not come to class if you are not feeling well.**

This practicum *will* be included in your final score.

Create one file for the whole practicum called **p2.py**, and we'll add to it as we go. We have a few problems to tackle today; get through as many as you can, and submit whatever you have done at the end of your section. Each question in this quiz will describe a problem to work on, then will ask you questions that you'll answer together as a group.

At the end, everyone should make sure to turn in your code to gradescope **with your group's answers to the embedded questions in comments in your file!**

Task 0: Who's in your group?

First, write down your name in a comment in at the top of your file and a small description of what it is:

```
# Felix Muzny
# DS 2001 - CS
# Jan 27, 2021 (Jan 28 for Thursday folks)
# Practicum 2 - Puzzles & Variable manipulation
# Group: Arushi, Arian, Smit
```

Write down your group members names in comments below this.

Task 1: Help Waldo out!

Waldo is having trouble keeping track of his variables (maybe this is why he's so hard to find)!

Take each code snippet below and type it into your file exactly as written. Then run your file and do the following two things:

1) Write down what error(s) you get (and why those errors happen)
2) Fix the code so that it runs with no errors! You may need to do any combinations of the following three things:
    a) you may need to *change the order* of the existing lines of code
    b) you may need to add a variable declaration to a line to save unsaved return value
    c) you won't need to add any new lines of code

Note! you should **not** delete any lines, nor add any completely new lines!

1.1
```
print(x)
x = 7
```

1.2
```
num1 = 0
num2 = 1
print(num3)
num3 = num1 + num2 + num3
num3 = num2 + num2
```

1.3
```
text1 = "Harriet the spy's favorite number is"
text2 = input("Harriet's favorite number: ")
text2 = int(text2)
print(text1, text2)
print(text1 + text2)
```

1.4
```
input("How many feet? ")
int(feet)
inches = feet * 12
str(inches)
print("That's " + inches + " inches!")
```

When you've finished with your code, answer the following question in comments in your file:

1) With your group, describe the differences that you've found between using print(value1, value2) and print(value1 + value2)

Make sure to check your answer with a TA! (Raise your hand! Feel free to continue working with your group while you wait.)

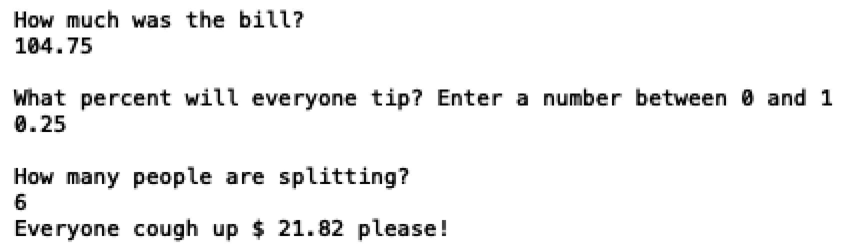
Task 2: Eating out with a group
This early in the semester and we're already solving one of the most challenging problems in the world—eating out with a group (outdoors, of course!) and splitting the bill. We can't help you pick a restaurant everyone will like, but we'll work on splitting the bill and ensuring that everyone leaves a good size tip for our wait staff.

Add on to your Python program code that that collects the following information from the user:

- How many people are there?

- What was the total amount of the bill?
- What is everyone willing to tip (hopefully at least 20%)?

Then, print the total amount each person should pony up. Here's a quick example:



```
How much was the bill?
104.75

What percent will everyone tip? Enter a number between 0 and 1
0.25

How many people are splitting?
6
Everyone cough up $ 21.82 please!
```

Your output should be similar to what we've put here but you need not reproduce it exactly.

When you've finished with your code, answer the following question:

2) With your group, describe one error that you encountered while writing this program, what caused it, and how you fixed it.

Make sure to check your answer with a TA! (Raise your hand! Feel free to continue working with your group while you wait.)

Challenge problem:
Notice how we print two numbers after the decimal point? That's because it's a currency and we wouldn't expect to see a number like 21.8229… or even 25.0 here. To make this work, look up the info on rounding and *formatting* in today's handout on output and formatting (linked at the top of this handout or from the course website).

Task 3: How long will it take to walk home?
Next, write a program that tells the user how long it will probably take them to walk home based on 1) their walking speed 2) the number of miles they are from home and 3) the number of obstacles (such as angry cats, devious dragons, or annoying construction projects) in their way. The nature and effect of obstacles is up to you!

Example program output, where we have chosen the obstacle of an angry cat:

```
How fast do you walk? 3.4
How far are you from home? 2
With no obstacles, it will take you 35.294117647058826 minutes to get home.
How many angry cats? 2
If you encounter 2 angry cats, it will take you 45.294117647058826 minutes to get home!
```

Hint: To convert miles per hour to a number of minutes per mile, multiply the number the user entered (e.g. "3.4") by 60 (since there are 60 minutes in an hour.

When you've finished with your code, answer the following question:

3) Did you have to type cast the information that the user entered? If yes, what type did you change it to and why?

Make sure to check your answer with a TA! (Raise your hand or use the "ask for help" button and we'll send someone by. Feel free to continue working with your group while you wait.)

Task 4 (Bonus Problem): Be a better ATM
You are a vending machine. Everything you sell costs the same amount of money, $2.37. The user will give you some money and you need to give back some change. Being a very efficient vending machine, you want to give back as few coins as possible.

Prompt the user for a whole dollar amount. You can assume they'll enter a whole number **no smaller than 3**. Report the change that goes back to the user in terms of:

- Number of dollars
- Number of quarters
- Number of dimes
- Number of nickels
- Number of pennies

Here's an example:

```
How much are you putting in?
5
Your change is $2.63
I'm giving you back:
2 dollars
2 quarters
1 dimes
0 nickels
3 pennies
```

This is called an **optimization problem**. If I ask you for change for a dollar, you could give me 100 pennies, and that's technically correct, but probably not what I had in mind. Instead, we want a correct solution, with the fewest coins/dollars possible.

Tips & Tricks:

- It'll be easier to work with whole numbers instead of floats. Once you figure out how much change to give back, multiply by 100 so you're working with a whole number of cents..
- Save values you need (like 25, 10, etc.) as constants.
- In Python, the mathematical operator `//` does integer division. I like to call it *Price is Right division*! **`x // y`** gives you an integer, the number of times *y* goes into *x* without going over. For example, **`15 // 6`** would give you **`2`**.

Task 5 (Bonus Problem): Isolating Digits
As you might recall if you dig way back into your memory stores from third grade or so, we can talk about integers as having a ones place, a tens place, a hundreds place, etc.

For this part, we want to isolate the digits in all of these places. Prompt the user for a whole (non-negative) number, and then report back the ones digit, tens digit, and hundreds digit. Like this:

```
Enter a non-negative integer
167
hundreds = 1
tens = 6
ones = 7
```