Lab 1: Data types and arithmetic operations

Labs are graded for participation rather than correctness. Keep all your lab code in your course GitHub repo to receive credit for your work. We'll be looking to see that you have at least partially completed all problems in each lab.

If you finish the lab assignment early, you may get started on the homework.

Getting started

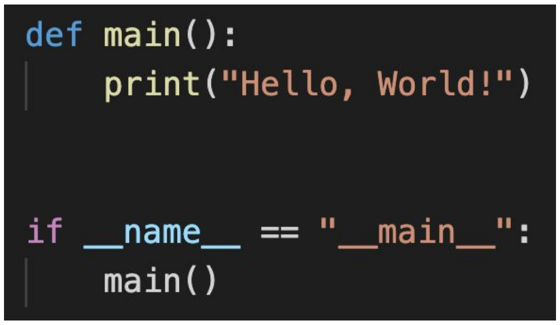
In your local repo*, create a folder called "Labs" if you haven't already done so. In your Labs folder, create a folder for Lab 1. Create a new .py file for each problem and save them in the Lab 1 folder. This will ensure that we can find your work when we're grading.

*Not sure what this means? Follow the GitHub instructions on Canvas. If you're still not sure, check with an instructor or TA.

Documentation and format guidelines

Every file you create for this course should have a file comment.
A file comment goes at the top of the file and tells the reader who wrote the file, when, and *what the code contained in the file is for*. Type three single quotes on the first line of the file, press return/Enter a couple of times, then enter another three single quotes. This creates a multi-line comment block. Inside the comment block (between the two sets of single quotes), enter your name, the course number and semester, and write a sentence or two describing the purpose of the code. Use the problem description to help you write the purpose statement.

Programs should all start with `main()`
In many languages, execution begins with a function called `main()`. Although Python is more flexible than this, writing programs in this manner makes it easy to understand where your program begins. Therefore, we require that all programs begin with a `main()` function:



```python
def main():
    print("Hello, World!")


if __name__ == "__main__":
    main()
```

Replace the print statement with your code as you work. Be sure to preserve the indentation! We'll explain what this syntax is doing in the coming weeks. For, now just make sure your files follow this format.

General tips

Test every program you write by running it several times, providing different inputs each time and verifying the output.

Don't worry about printing dollar amounts properly (i.e. to 2 decimal places). You also don't need to worry about validating user input e.g. checking that the tip amount is valid in problem 1. We haven't covered how to address these issues yet.

Problem 1: Eating out with a group

Write a program to figure out how to split a restaurant bill.
Your program should prompt the user for the following information:

- The total amount of the bill, a float.
- The percentage everyone is willing to tip, a float between 0 and 1. So 0.2 would be entered for a 20% tip.
- The number of ways to split the bill (i.e. the number of people in the group), an integer.

The program should use the information supplied by the user to calculate how much each person should contribute. For example, if the total bill was $60.00, everyone is willing to tip 20% (0.2), and there are 3 people willing to split the bill, then each person should contribute $24. The program should then print that amount for the user.

Suggested steps:

1. Work out the math on paper before trying to write code. Determine how you will break the problem down into discrete steps.
2. Write code to get the three pieces of information from the user. Save each input to a variable e.g.
   ```
   amount = float(input("How much was the bill?"))
   ```
   Remember that `input()` always returns a string so you will need to convert the values entered by the user to the appropriate type, as you can see in the above line of code.
3. Translate the steps you came up with in 1 above into code.
4. Print the result of the calculation for the user.

Problem 2: Buying a house

This time, write a program to figure out how long until a user can buy a house given important factors like the cost of the house and how much money they make.

You will need the following inputs from the user:

- Cost of the house
- Annual salary
- Percent the user can save from their monthly salary (a float between 0 and 1)

Calculate the following:

- How much the user needs for the down payment (assume the down payment 25% of the cost of the house)
- How much money the user saves per month
- How many months it will take to save enough for the down payment
- How many years + months that is

...and then print a message that explains how much can be saved per month and how long it will take to save up enough for the down payment.

For example, if ...

- the house is $90,000
- the annual salary is $50,000
- 25% of the salary can be saved per month

...the amount to be saved per month is $1041.67 and the down payment amount is $22,500. Therefore, it will take 1 year and 10 months to save. In this case, your program would print a string along the lines of, "If you save $1041.67 per month, it will take 1 year and 10 months to save enough for the down payment!"