Northeastern University DS 2000 — Programming with Data, Fall 2022, Profs. Muzny and Rachlin
Fall 2022
Homework 2
Released: September 16, 2022
Due: September 23, 2022 @ 9pm Boston time

Homework 2: Donuts Galore!

Introduction
One of our primary skills in this course and as data scientists is going to be reading data in and dealing with different formats of data files. As new programmers, we'll start practicing those skills with small data.

Expectations
Write a separate program to answer each question. Be sure to document your code with a descriptive header and concise comments. It isn't necessary to comment every line, particularly those lines whose purpose is obvious. Instead, use comments to outline the main steps of your program. Use variable names that are descriptive, concise, and not excessively verbose. Practice good style, for instance: x = 5, not x=5 and separate sections of your program by blank lines.

See the style guide for more information: https://course.ccs.neu.edu/ds2000/ds2000_fall2022_styleguide.pdf

For your homework submission, submit these files:

- `applecider.py`
- `donutshop.py`
- `wages.py`

- Each .py file should have one (and only one!) `main()`, as you saw in lecture on 9/16!

- Note that we are not asking for an output.pdf file this time. This is because we'll be using an autograder for this homework that will automatically run your code for you with different combinations of inputs! It is **extremely important** that you name your python files **exactly** as we have asked you to do!
    - We'll post a video demonstration of what you should see when you submit to gradescope on Piazza, similarly to HW 1.

You are welcome to upload a file multiple times, but only your last submission will be graded. Verify that all files have been submitted.

See the grading rubric for more information:
https://course.ccs.neu.edu/ds2000/ds2000_fall2022_gradingpolicy.pdf

Question 1 (`applecider.py`)

Last Fall, Prof. Felix went on an adventure to taste test and rate a number of the apple cider donuts available in the greater Boston area. In all, they tested 4 different donut locations.

We've provided you with two `donut_ratings` files, each containing the following information, with a line specifying the donut shop name followed by a line specifying the rating of the apple cider donuts at that establishment:

`muzny_donut_ratings.txt`	`rachlin_donut_ratings.txt`

`Union Square`
`11.1`
`Calareso's Farm Stand`
`13.35`
`Mahoney's Garden Center`
`6.3`
`Wilson Farm`
`12.3`
	`Union Square`
	`17.1`
	`Calareso's Farm Stand`
	`15.01`
	`Mahoney's Garden Center`
	`18.9`
	`Wilson Farm`
	`7.46`

Each file contains a total of 8 lines of data.

For this program, you will:

- Read the data from one `donut_ratings` text file (ask the user which one to read)
- Print out the maximum donut rating in this file
- Print out the minimum donut rating in this file

Python has two built-in functions, `max()` and `min()` to help you with this. They can each take any number of parameters and will return to you either the maximum value from those parameters or the minimum value.

Examples:

```
max_num = max(97, -17, 89, 117, 0, 2)
print(max_num)  # 117
min_word("bat", "dog", "albatros")
print(min_word)  # albatros
```

For example, the output of our program might be (user input in **bold underlined**):

```
Which donut rating file? muzny_donut_ratings.txt
Maximum (best) donut rating: 13.35
Minimum (worst) donut rating: 6.3
```

Question 2 (`donutshop.py`)

Next, you are starting your very own donut shop that sells glazed, chocolate, and sprinkled donuts. Each time you run your program represents one customer entering your donut shop. Each customer may buy as many donuts as they want, but they can only buy *one kind* of donut.

Your program should:

- Ask the user what kind of donut they'd like.
- Ask the user how many donuts they'd like.
- Tell the user how much their donuts cost.
- Report how much profit your donut shop made from this sale.

We'll give you the starting files, which contain the sale price (on the first line) and the ingredient cost (on the second line) of each kind of donut.

`glazed.txt`	`chocolate.txt`	`sprinkled.txt`
`1.5` `1.0`	`2.0` `1.25`	`2.25` `1.30`

For example, the shop sells glazed donuts for $1.50 and the ingredients cost $1.00, so each glazed donut sold makes a profit of $0.50.

Example (user input is in **bold underlined**):

```
What kind of donut would you like [glazed, chocolate, sprinkled]? glazed
How many donuts would you like? 7
Please pay: $10.5
This sale made us $3.5!
Enjoy your donuts!
```

The next time the program is run, the customer might ask for chocolate donuts:

```
What kind of donut would you like [glazed, chocolate, sprinkled]? chocolate
How many donuts would you like? 2
Please pay: $4.0
This sale made us $1.5!
Enjoy your donuts!
```

The next time the program is run, the customer might ask for sprinkled donuts:

```
What kind of donut would you like [glazed, chocolate, sprinkled]? sprinkled
How many donuts would you like? 23
Please pay: $51.75
This sale made us $21.85!
Enjoy your donuts!
```

Question 3 (wages.py)

When your shop has closed for the day, you want to run a summary analysis of the business at your shop that day to determine the maximum amount you can afford to pay your employees (yes, we are giving 100% of our profits to our workers). We're giving you some new files that have the numbers from a few different days. Note that we expect there to be variation from day to day! If this was how we calculated wages in the real world, no one would ever want to work at our donut shop.

We're giving you one file that lists the total profits made from each kind of donut (top line is glazed, middle is chocolate, last is sprinkled) on this day and one file that lists the number of employees that worked that day and how many hours they worked.

profits_sep16.txt	employees_sep16.txt
3.50 -1.50 21.85	2 3

Write a program that:

- Asks the user which day to read the data for.
- Reads in the data about both the profits and the employees for that day.
- Reports the total profit made that day from selling donuts.
- Reads in the data about how many employees worked on that day.
- Reports the hourly wage for the donut shop employees.

Example (user input is in **bold underlined**):

```
What day should we analyze? sep16
On sep16, you made $23.85.
2 employees worked for 3 hours each, totaling 6 hours.
You should pay your employees $3.98 per hour.
```

Here's an example with the files for September 17th (a much more lucrative day)

profits_sep17.txt	employees_sep17.txt
51.50 9.00 43.70	1 4

And the corresponding program output:

```
What day should we analyze? sep17
On sep17, you made $104.2.
1 employees worked for 4 hours each, totaling 4 hours.
You should pay your employees $26.05 per hour.
```