Lab 1: Introduction to Python

1 Objectives

- Be comfortable with the command line
- Download and install Python
- Be comfortable using Python to implement simple algorithms
- Create a Python program

2 Resources

- Python Basics Playlist (videos 1-7): `https://www.youtube.com/watch?v=D48iCw3WWpI&list=PL82YdDfxhWsDJTq5f0Ae7M7yGcA26wevJ`
- Python Commands Cheat Sheet: `https://ehmatthes.github.io/pcc/cheatsheets/README.html`
- Python Basics (all topics - dry read...) `https://www.techbeamers.com/python-tutorial-step-by-step/`
- Introduction to Python: `https://docs.python.org/3/tutorial/index.html`
- Installing Sublime (Lightweight and Aesthetic Text Editor): `https://www.sublimetext.com/download`
- Cmd Prompt for Windows users: `https://riptutorial.com/cmd`
- Terminal for Mac/Linux users: `https://www.learnenough.com/command-line-tutorial/basics`

3 Installing Python

You can download Python version 3 in various ways; you can get it from the official Python website `www.python.org`, or through any third-party programs that offer the Python language as well as extra packages pre-installed. See below for further details.

3.1 Mac

If you are on a Mac, Python 2.7 comes pre-installed. You may or may not already have Python 3 installed as well. To check, open up the Terminal application, and type in *python - -version* to see which versions are there. If Python 3 is not installed, download the latest version from the Python website. Your default version will be 2.7, so go here to see how to change the default to 3: `https://opensource.com/article/19/5/python-3-default-mac`.

3.2 Linux

There's a very good chance your Linux distribution has Python installed already. To find out which version you have, open a terminal window and try the following commands:

- python - -version
- python2 - -version
- python3 - -version

One or more of the commands should respond with a version number, e.g., *Python 3.8.1*. If the version shown is not 3.x.x, then you will want to install that version. The procedure for doing so will depend on the distribution you are running. See instructions here: `https://realpython.com/installing-python/#linux`

3.3 Windows

For installation instructions, go here: `https://www.howtogeek.com/197947/how-to-install-python-on-windows/`. Running python involves using the command prompt, which you can read about here: `https://www.lifewire.com/command-prompt-2625840`

4 Installing a Code Editor or IDE

You have two options when writing/running Python: a code editor, or an IDE (Integrated Development Environment).

Code editors are simply text editors (e.g., Notepad or TextEdit). However, they have some nicer features like syntax highlighting, automatic indenting, etc. If you choose to go this route, then you write/edit your python files in the code editor, and then you run the Python program using the command prompt or terminal. We recommend Sublime (link in the resources section above), but any of the following are good alternatives:

- Notepad++ (Windows only)
- Atom (browser based)
- Vim
- Visual Studio Code

IDEs are software packs that consist of features like coding, compiling, debugging, executing, autocomplete, libraries, etc. All of these are in one place for the developer, making tasks simpler. If you choose to go with an IDE, you have the following options:

- PyCharm
- Spyder
- PyDev
- Idle
- Wing

5 Your first program

Time for a sanity check to make sure everything is working. Open up your code editor or IDE, and create a new file called *hello.py*. Save this to either your desktop or to a known place on your computer. In your editor/IDE, write some code that will print out the words Hello World!. Then run your program by going into your terminal/command prompt, or through your IDE.*

*If you run into any issues, make sure that you are in the correct path. If the file is saved on your desktop, then in the terminal, make sure you are in the correct directory. If this sentence doesn't make any sense to you at all, ask your TF for help.

6 Homework Questions

You may not use any additional libraries beyond textstat.

6.1 Average Joe (5 pts)

- **Create a program that asks the user for two integer inputs and outputs the average of the two inputs as a float. Assume the user will only put in integer values (i.e., the user is cooperative and will give you inputs that make sense).**
- **Implement this algorithm within average() in lab1.py**

6.2 Mo Money Mo Problems (10 pts)

- **Suppose also that you are in a foreign country where they only give out coins in 15, 4, and 1 cent increments. Create a program that asks the user for an amount in change. Given this input, output the minimum number of coins it would take. Hint: Using a greedy strategy approach will give you an optimal solution (aka a minimum number). In a greedy strategy, at each step you maximize the amount you can remove. So for the amount 80 cents, you want to first take away as many 15 cent coins as possible. Then as many 4 cent coins, then 1 cent coins.**
- **Implement this algorithm within money() in lab1.py.**

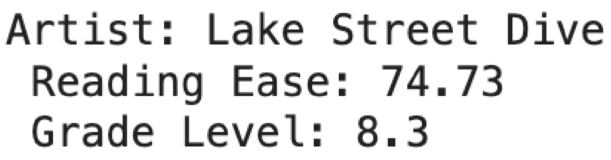
6.3 Better Than (20 pts)

This problem utilizes dictionaries, strings, and loops. In addition, you will download and install a python library called Textstat.

- **Create a dictionary where the keys are the following bands/musicians, and the values are the song lyrics to the following songs:**
  - **Lake Street Dive, Twenty-Five**
  - **Bob Dylan, Mr. Tambourine Man**
  - **Taylor Swift, Invisible String**
  - **George Gershwin, Someone to Watch Over Me**

  Note that copying/pasting text into python, any apostrophes and quotation marks will need to be specially marked so that you do not get an end of line error. You do not need to input the entire song, but you should use at least a few verses to get an accurate reading. You may also need to edit to remove extra spaces and other formatting which will impact the scores.
  https://www.digitalocean.com/community/tutorials/how-to-format-text-in-python-3.
- **Next, install the Textstat library. https://pypi.org/project/textstat/**
- **The Textstat library has two functions we will be looking at: Flesch Reading Ease and Flesch Kincaid Grade. For each key in your dictionary, print out the band/musician name, reading ease, and Kincaid grade.**

  Your final output should look like this:
- **Implement this algorithm within lyrics() in lab1.py. The output should also include the title of the song right below "Artist" and above "Reading Ease."**

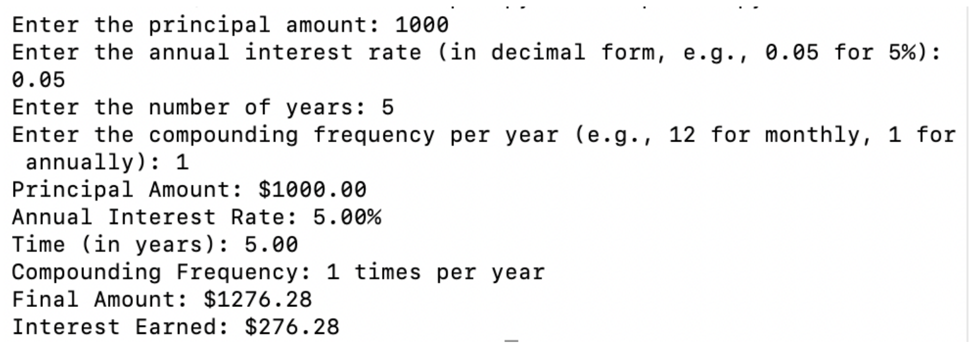

```
Artist: Lake Street Dive
 Reading Ease: 74.73
 Grade Level: 8.3
```

6.4 Money in the Bank (10 pts)

Compound interest is the interest you earn on interest. This can be illustrated by using basic math: if you have $100 and it earns 5% interest each year, you'll have $105 at the end of the first year. At the end of the second year, you'll have $110.25.

You can find the compounding equation here:
`https://www.wallstreetprep.com/knowledge/compound-interest/`

Create a program that calculates the amount of compound interest you would receive from a given amount. Your program will ask the user for a principal amount, an interest rate, a number of years to invest, and a compound frequency (once a year would be 1, once per month would be 12). The output of the program would return the total amount of money, and the total interest. Here is an example of what the user input and the user output would look like. The first four prompts (the lines that start with "Enter") are for the user to input, and the last six lines are the program output.


```
Enter the principal amount: 1000
Enter the annual interest rate (in decimal form, e.g., 0.05 for 5%):
0.05
Enter the number of years: 5
Enter the compounding frequency per year (e.g., 12 for monthly, 1 for
 annually): 1
Principal Amount: $1000.00
Annual Interest Rate: 5.00%
Time (in years): 5.00
Compounding Frequency: 1 times per year
Final Amount: $1276.28
Interest Earned: $276.28
```

Implement this program in interest() in lab2.py.

6.5 I Like My Sugar With Coffee and Cream (30 pts)

In this problem, you are going to translate the coffee graphic (on the next page) into Python. Your program will prompt the user and provide options for answers. Based on responses, you will output a response for what you think the user wants as a coffee order. At the end, you will ask the user if this guess is correct. If the guess is correct, then that ends the program. If the guess is no, then restart the questions.

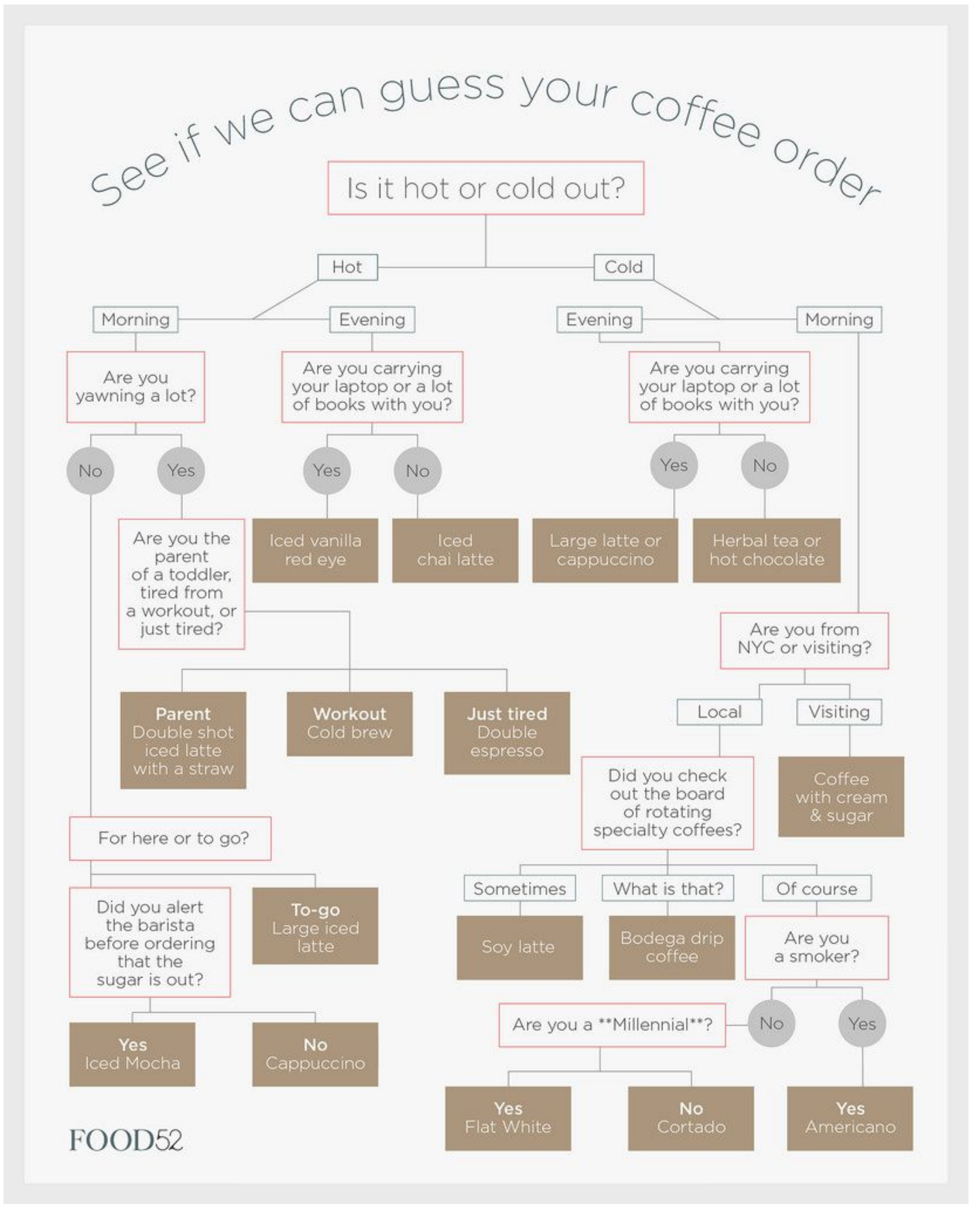
Implement this program in coffee() in lab1.py.

7 Submitting Homework to ELMS

In homework, make sure to list all assumptions, and use comments to notate your code whenever possible. Clarity and style will matter. Your TF should be able to read your code and understand everything. You will be submitting one file.

1. A python lab1.py file that contains all comments and pseudocode

Do not change the name of the files. Add your name and section number to the top of each file as a comment.


See if we can guess your coffee order
Is it hot or cold out?
Hot
Cold
Morning
Evening
Evening
Morning
Are you yawning a lot?
Are you carrying your laptop or a lot of books with you?
Are you carrying your laptop or a lot of books with you?
No
Yes
Yes
No
Yes
No
Are you the parent of a toddler, tired from a workout, or just tired?
Iced vanilla red eye
Iced chai latte
Large latte or cappuccino
Herbal tea or hot chocolate
Are you from NYC or visiting?
Parent Double shot iced latte with a straw
Workout Cold brew
Just tired Double espresso
Local
Visiting
Did you check out the board of rotating specialty coffees?
Coffee with cream & sugar
For here or to go?
Sometimes
What is that?
Of course
Did you alert the barista before ordering that the sugar is out?
To-go Large iced latte
Soy latte
Bodega drip coffee
Are you a smoker?
Are you a **Millennial**?
No
Yes
Yes Iced Mocha
No Cappuccino
Yes Flat White
No Cortado
Yes Americano
FOOD52