# ECE 364 Prelab Assignment 02

Basics of Python

**Passing this lab will satisfy course objectives CO2, CO3**

## Instructions

- You must meet all *base requirements* in the syllabus to receive any credit.
- Work in your Prelab02 directory, and copy all files from ~ee364/DataFolder/Prelab02 into your working directory:

  ```
  cp -r ~ee364/DataFolder/Prelab02/* ./
  ```

- Remember to add and commit all **requird** files to SVN. **We will grade the version of the file that is in SVN!**
- Do *not* add any file that is *not* required. **You will lose points if your repository contains more files than required!**
- Make sure you file compiles. **You will not receive any credit if your file does not compile.**
- Name and spell the file, and the functions, exactly as instructed. Your scripts will be graded by an automated process. **You will lose some points, per our discretion, for any function that does not match the expected name.**
- Make sure your output from all functions match the given examples. **You will not receive points for a question whose output mismatches the expected result.**
- Unless otherwise specified, you cannot use any external library, but you can use any module in the **Python Standard Library** to solve this lab, i.e. anything under:

  ```
  https://docs.python.org/3.7/library/index.html
  ```

- Make sure you are using Python 3.7 for your Prelab.

# Basics of Python

**Required File** `ioTasks.py`

Create a Python file named `ioTasks.py`, and do all of your work in that file. This is the only file you need to submit. Refer to the **Python File Structure** section below for a reminder on the requirements. Note that the use of type annotation is *not* required, but it is recommended for better compile-time static analysis.

You are given a set of files that contain stock data for some companies where each file name has the format `'<Company_Symbol>.dat'`. All of the files cover the same time period and have an identical structure, where each line contains the following values for a single date: the value of the stock at the opening and closing of the trading, the highest and lowest values of the stock on that date, and the exchange volume. All of the stock values are float values, while the volume is a positive integer (The volume is shown as a float in the file, but it should be regarded as an integer.) Examine these files and use them to create the functions below.

Notes:

- Do not hardcode the file names. We will use different files for testing.
- Round all floats results, from mathematical operations as well as returned values, to 4 decimal digits.
- The dates used in the questions will be strings that match the format used in the files.
- In all of the functions that require a symbol as an argument, return `None` if that symbol does not exist among the files provided.
- It is unlikely that you will obtain duplicate results in those functions, but if you do, return the first match.

## Implementation Requirements

1. **[15 pts]** Write a function called `getMaxDifference(symbol)` that takes in a company symbol and returns the date, as a string, where the difference between the high and low values is the largest for that company.

2. **[15 pts]** Write a function called `getGainPercent(symbol)` that takes in a company symbol and returns the percentage of days where the company's closing price is higher than its opening price.

   Note: The return value should be a float out of 100, e.g. 23.5419.

3. **[15 pts]** Write a function called `getVolumeSum(symbol, date1, date2)` that takes in a company symbol as well as two dates, where `date2` > `date1`, and returns the sum of transaction volumes between these dates, inclusively, as an integer.

   Notes:

   - If `date1` $\geq$ `date2` return `None`.
   - Assume the dates exist in the files.

4. **[15 pts]** Write a function called `getBestGain(date)` that takes in a date, and returns the highest positive gain percentage, where the gain percentage is calculated as:

$$G = \frac{\text{close} - \text{open}}{\text{open}} \times 100$$

   on that date, across all companies provided.

   Note: Assume the date exists in the files, and that at least one company will have a positive gain.

5. **[10 pts]** Write a function called `getAveragePrice(symbol, year)` that takes in a company symbol and a year, as an integer, and returns the average price of the company's stock over that year. Note that the average of the price in a day is calculated as:

$$\text{Daily Avg} = \frac{\text{close} + \text{open}}{2}$$

and the annual average is the average of that over all days of the year.

Note: Assume the year exists in the files.

6. **[10 pts]** Write a function called `getCountOver(symbol, price)` that takes in a company symbol and a stock price, as a float, and returns the number of days where all trade values (opening price, closing price, low and high prices,) are above or equal to the given price.

## Final Check!

You are given a file that checks for the exact spelling of the required functions. Make sure you run this file before your final commit. Remember that **you will lose some points for any function that does not match the expected name.**

### Python File Structure

The following is the expected file structure that you need to conform to:

```
#########################################################
#   Author:     <Your Full Name>
#   email:      <Your Email>
#   ID:         <Your course ID, e.g. ee364j20>
#   Date:       <Start Date>
#########################################################

import os     # List of module import statements
import sys    # Each one on a line


#########################################################
# No Module-Level Variables or Statements!
# ONLY FUNCTIONS BEYOND THIS POINT!
#########################################################

def functionName1(a: float, b: float) -> float:
    return 0.


def functionName2(c: str, d: str) -> int:
    return 1


# This block is optional and can be used for testing.
# We will NOT look into its content.
#########################################################
if __name__ == "__main__":
    # Write anything here to test your code.
    z = functionName1(1, 2)
    print(z)
```