

# CSCI 478: Data Science, Lab 1: Python I

## Goals:

- To gain hands-on experience creating and running simple python files
- To set up your git environment for submitting labs
- To set up your Python3 environment
- To implement a python program that reads in a data file in a specified format compute specified values
- To demonstrate basic error handling in python
- To demonstrate basic testing in python

## What to hand in:

- Submit the following to your lab 1 git repo by Jan 22/25 23:59:
  1. Main program:
    - lab1.py
  2. Additional Test files:
    - lab1Data4.txt
    - lab1Data5.txt
    - lab1Data6.txt
  3. Test output files:
    - data1output.txt
    - data2output.txt
    - data3output.txt
    - data4output.txt
    - data5output.txt
    - data6output.txt
  4. git log
    - lab1gitLog.txt

## What to do:

### git setup

In lab, we'll go over how to set up your git repositories for the labs.

### python setup

In lab, we'll go over how to:

- check what version of python is installed
- use interpreter mode to use python to solve simple problems
- create a python file and run it

Please follow along in lab or follow the course notes on how to do this.

## Basics of python

At the start of lab, we'll go over some basic concepts in Python:

- declaring variables
- scope
- defining the main entry point
- basic data types (numerics, strings, lists)
- reading in files

- basic exception handling

Armed with this knowledge, you will write a simple program to compute some values based on an input data file.

lab1.py

Your job is to write a python program that meets the following specifications:

- program is saved in a file named lab1.py
- prompts the user for the name of an input file
- based on the file name provided, reads in the data *using the* `with open` statement and:
  - based on the meta data in line 1, and the number of data entries on line 2, computes the appropriate value
  - output meaningful output for both valid and invalid inputs (see tests and error handline, below)
- write a function to handle the computation of the value that takes as input:
  - a list of data
  - the size of the list
  - the type of computation to be done
- error handling:
  - If the datafile doesn't exist:
    - Catch the `FileNotFound` exception, output a meaningful error message, and terminate the program
  - If the input file contains an invalid entry on line 1:
    - Output a clear error message and terminate the program
  - If the input file contains an invalid value on line 2
    - Non-numeric value
      - Output a meaningful error message and terminate the program
    - The number doesn't match the number of data entries:
      - Output a meaningful error message, and either:
        - If there are too few values, terminate the program
        - If there are more values than needed, ignore any extra ones and compute the appropriate value
  - If the data values contain non-numeric values or if the type of any of the data values doesn't match the specified type on line one:
    - Catch the `ValueError` exception, output a meaningful error message, and terminate the program

format of data files

The input data files (if correct) should be of the following format:

- line 1: datatype, computation
  - datatype can be either "int" or "float"
  - computation can be either: "mean", "max", "min", "sum"
- line 2: number of entries
  - This number should match the number of numeric values that follow in lines 3 onwards

Sample input file:

The following file contains 3 floating point numbers, and when your program runs, it should output 3340.3 (the max value in the list):

```
float,max
```

3  
45.6  
3340.3  
123.6

### testing

You will run your program on the three instructor-provided data files, and save the output to its own output file: `data1output.txt`, `data2output.txt`, `data3output.txt`

To demonstrate basic testing, you will also create and submit three additional input data files and submit their outputs.

- create test files that include the following:
  1. `lab1Data4.txt` with output: `data4output.txt`
    - includes 1000 randomly generated numbers (you can write another python file that creates this for you)
  2. `lab1Data5.txt` with output: `data5output.txt`
    - a file that includes an incorrect number on line 2 (the number of data entries differs from the number included)
  3. `lab1Data6.txt` with output: `data6output.txt`
    - a file that includes an invalid command