
Read chapter 9 and 11 in [Fundamentals of Python Programming](#) and complete the following exercises.

You may need to reference the following as well:

[Python tutorial: string slicing](#)

[matplotlib.pyplot.plot](#)

[matplotlib.lines.Line2D](#)

[matplotlib.pyplot.legend](#)

[matplotlib.pyplot.hist](#)

[matplotlib.patches.Patch](#)

Complete computer setup by following the directions here:

<http://cs.appstate.edu/~rmp/cs5245/setup.pdf>

Create a folder called `cs05` to store your files for this lab.

Complete the following activities using the **exact variable names** and save the files using the **exact file names** as specified.

To test your programs, run them from IDLE and check their output.

1 Using Pandas to Load CSV files

First an introduction to loading CSV files. A CSV file one line per row of a table. Each line separates the different values for that row using a comma. For example, a CSV file might look like this:

```
Name, Age, Height
Roger, 19, 72
Kelly, 23, 64
Brian, 21, 75
Joy, 20, 68
```

You can download this file here: [data.csv](#)

The following example loads a CSV file into a Python object called a Data Frame. You can think of the data frame as something like a spreadsheet, where each column has a label and each row has an index.

```
>>> import pandas as pd
>>> df = pd.read_csv('data.csv')
>>> df
   Name  Age  Height
0  Roger   19     72
1  Kelly   23     64
2  Brian   21     75
3    Joy   20     68
```

The top row shows the names of columns (Name, Age, and Height). The index for each row is shown on the left margin (0, 1, 2, 3). To access the first Name column:

```
>>> df['Name']
0    Roger
1    Kelly
2    Brian
3     Joy
Name: Name, dtype: object
>>>
```

Like the audio data we will convert a column of the dataframe into a list in order to process it:

```
>>> height = list(df['Height'])
>>> height
[72, 64, 75, 68]
```

Then, we can use that list to compute something else, perhaps the height in feet:

```
>>> feet = []
>>> for h in height:
...     feet.append(h / 12)
...
>>> feet
[6.0, 5.333333333333333, 6.25, 5.666666666666667]
```

We can create a new column in the data frame and store the height in feet in it:

```
>>> df['Feet'] = feet
>>> df
```

	Name	Age	Height	Feet
0	Roger	19	72	6.000000
1	Kelly	23	64	5.333333
2	Brian	21	75	6.250000
3	Joy	20	68	5.666667

Finally, we can save this data frame as another CSV file:

```
>>> df.to_csv('new_file.csv', index=False)
```

`index=False` means that we do not want to put the row indexes (0, 1, ...) in the CSV file because they weren't there to begin with.

We will use data from CSV files because it is way easier to load a file than ask the user to enter everything in from the command line. Good luck!

2 file_bmi

In a file called `cs05.py`, write a Python function named `file_bmi` that takes an input argument containing the name of the CSV file as a string, loads that file, computes the BMI, adds a column to the data frame, and saves it with a new name, `<file_name>_bmi.csv` where `<file_name>` is the name of the original CSV (without the extension).

See [Section 1](#) for how to use pandas with CSV files. To make the new file name, you can use [string slicing](#) to get all but the last 4 characters from the file name then append the new ending to the file name.

For reference, here is the equation for BMI:

$$\text{BMI} = 703 \frac{\text{weight}}{\text{height}^2}$$

2.1 Example Usage

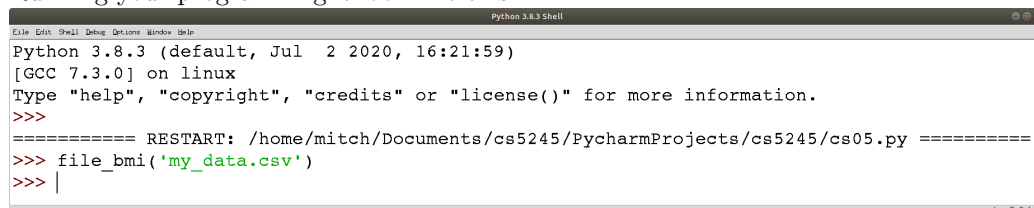
```
def bmi(...): # replace '...' with the appropriate input argument(s).  
    # your code here...  
    pass
```

For example, consider the following file named `my_data.csv`:

```
Name, Age, Height, Weight  
Roger, 19, 72, 200  
Kelly, 23, 64, 140  
Brian, 21, 75, 220  
Joy, 20, 68, 160
```

You can download this file here: [my_data.csv](#)

Running your program might look like this:



```
Python 3.8.3 (default, Jul 2 2020, 16:21:59)  
[GCC 7.3.0] on linux  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/cs05.py =====  
>>> file_bmi('my_data.csv')  
>>> |
```

afterward there should be another file named `my_data_bmi.csv`. Your results need only be correct within 6 decimal places:

```
Name, Age, Height, Weight, BMI  
Roger, 19, 72, 200, 27.121913580246915  
Kelly, 23, 64, 140, 24.0283203125  
Brian, 21, 75, 220, 27.495111111111111  
Joy, 20, 68, 160, 24.325259515570934
```

3 plot_signals

In the same `cs05.py` file, write a Python function named `plot_signals` that takes the name of a CSV file as an input argument, plots the second through fourth columns as y -coordinates versus the first column (x), saving it as a new file named `<file_name>_plot.png`, where `<file_name>` is the name of the CSV file (without the extension).

3.1 Data Format

First, you will need to get the data from the CSV file.

For this program, you can expect a CSV file formatted as follows. The first column will be named `'x'` and there will be three columns that follow. For example, the columns might be `'x'`, `'a'`, `'b'`,

'c'. You are to use the 'x' column as the x -coordinates and plot the other columns as y -coordinates with different labels.

Here's an example with one specific set of column labels:

```
x,y1,y2,y3
0,1,0,0
0.1,0.809,0.588,0.727
0.2,0.309,0.951,3.078
0.3,-0.309,0.951,-3.078
0.4,-0.809,0.588,-0.727
0.5,-1,0,0
0.6,-0.809,-0.588,0.727
0.7,-0.309,-0.951,3.078
0.8,0.309,-0.951,-3.078
0.9,0.809,-0.588,-0.727
1,1,0,0
```

You can download this file here: [data2.csv](#)

The labels used by Web-CAT are not 'y1', 'y2', and 'y3', so your code will need to get these from the data frame:

```
>>> # load the data frame first, then:
>>> list(df.columns)
['x', 'y1', 'y2', 'y3']
```

3.2 Plotting

Second, you will need to visualize it.

Use the `plt.plot` function to generate each line plot. Style each plot using the `matplotlib.lines.Line2D` properties and set the `xlabel`, `ylabel`, and `title` to the designated values. Create a `legend` with the correct column names. The line properties are as follows:

- Use the **color** property to set the line and marker colors:
 1. The first signal (y1) is green (0% red, 50% green, 0% blue).
 2. The second signal (y2) is cyan (0% red, 75% green, 75% blue).
 3. The third signal (y3) is orange (100% red, 50% green, 0% blue).
- Use the **marker** property to set the marker shapes:
 1. The first signal (y1) has a **circle** shape.
 2. The second signal (y2) has a **plus** shape.
 3. The third signal (y3) has a **diamond** shape.
- Use the **linestyle** property to set each line style:
 1. The first signal (y1) has a **dashed** line.
 2. The second signal (y2) has a **dotted** line.
 3. The third signal (y3) has a **dashdot** line.
- Use the **linewidth** property to set each line width to 2 points.

- Make sure to label your axes.
 1. Use the `plt.legend` function to place the legend with labels provided from the columns in the data frame.
 2. Use the `plt.xlabel` function to label the x -axis, `'time'`.
 3. Use the `plt.ylabel` function to label the y -axis, `'amplitude'`.
 4. Use the `plt.title` function to set the title to the name of the PNG file that you save.
- Save the plot as a new file named `<file_name>_plot.png` where `<file_name>` is the name of the CSV file (without the extension).

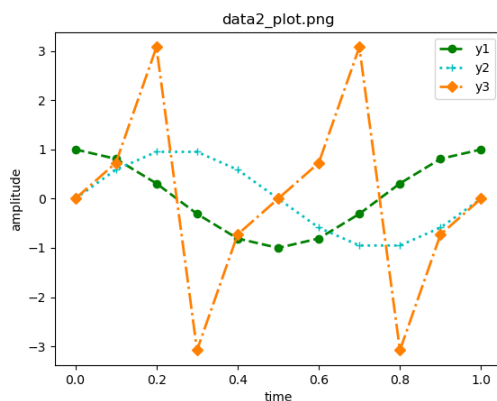
3.3 Example Usage

```
def plot_signals(...): # replace "..." with the appropriate argument(s).
    # your code here...
```

Running the program for the above CSV should produce the following program output:

```
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
Python 3.8.3 (default, Jul 2 2020, 16:21:59)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/cs05.py =====
>>> plot_signals('data2.csv')
>>> |
```

and produce the following PNG:



4 grade_histogram

In the same `cs05.py` file, write a Python function named `grade_histogram` that takes an input argument specifying the name of the CSV file, load the file, plot a **histogram**, and save the histogram as a PNG file named `<file_name>_hist.png` where `<file_name>` is the name of the CSV file (without the extension).

4.1 Data Format

Consider the following file named `fakegrades.csv` that has been truncated to the first 10 lines:

```
Grades
87
57
65
78
100
70
93
84
74
:
```

You can download this file here: [fakegrades.csv](#)

4.2 Histogram Specification

Use the `plt.hist` function to create a histogram and use the `Patch` properties to control the style of the bars.

1. Create a histogram with bin edges at 0, 5, 10, 15, ..., 100.
2. Set the edge color of the bars to **black**.
3. Set the face color of the bars to **cyan**.
4. Use **diagonal** hatching.
5. Label the *x*-axis and *y*-axis as **'Grades'** and **'Count'**, respectively.
6. Title the figure with the name of the PNG file you save.
7. Save the resulting figure as a PNG file. If the CSV file is called `file.csv`, you would save the PNG file, `file_hist.png`.

4.3 Example Usage

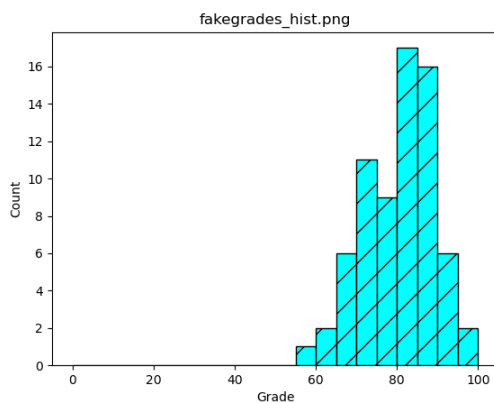
```
def grade_histogram(...): # replace "..." with the appropriate argument(s).
    # your code here...
    return
```

For example, for the `fakegrades.csv` file available here: <http://cs.appstate.edu/~rmp/cs5245/fakegrades.csv>

The program output would be the following:

```
Python 3.8.3 Shell
Python 3.8.3 (default, Jul 2 2020, 16:21:59)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/mitch/Documents/cs5245/PycharmProjects/cs5245/cs05.py =====
>>> grade_histogram('fakegrades.csv')
>>> |
```

and produce the following PNG:



5 time_it

When comparing different computational approaches, often the time it takes them to run is the distinguishing characteristic. If both algorithms provide the same results but one is significantly faster, it should be preferred.

We will compare four data structures for how quickly they can determine whether or not they contain a particular value using Python's `in` operator. All the data structures are fast enough that they won't slow us down in the Python shell executing one command at a time. However, in production they may experience thousands of calls per second and the differences in speed become obvious. To help with this analysis, you will create function that times one algorithm for the task. Then, we will use it to compare the different data structures.

In the same `cs05.py` file, write a Python function named `time_it` that takes the following input arguments in order: a data structure (variable), a number of trials n , and a value that it does not contain.

5.1 time_it Specification

You may add the following imports to the top of `cs05.py`:

```
from time import time
from random import randint
```

In your `time_it` function, check whether or not the data structure (`a`) contains the item (`value`) using the `in` operator. Repeat this n times. Use the `time` function to get the time before and after computing the n repeats and `return` the average time for each trial.

5.2 Example Usage

```
def time_it(...): # Replace "..." with appropriate input arguments
    # your code here...
    return average_time
```

For example, it should look something like this (your times will vary depending on the speed of your processor and its utilization by other processes):

```
>>> # First, create the data structures
>>> a = [randint(1, 100000000) for _ in range(1000000)]
>>> b = tuple(a)
>>> c = set(a)
>>> d = {ai: True for ai in a}
>>> n = 100
>>>
>>> # Now compare the times (in seconds)
>>> time_it(a, n, -1)
0.009019367694854737
>>> time_it(b, n, -1)
0.008962798118591308
>>> time_it(c, n, -1)
1.8835067749023438e-07
>>> time_it(d, n, -1)
2.2172927856445314e-07
>>> speed_up_set = time_it(a, n, -1) / time_it(c, n, -1)
>>> speed_up_dict = time_it(b, n, -1) / time_it(d, n, -1)
>>> print('set is {:.0f} times faster than list.'.format(speedup_set))
set is 140339 times faster than list.
>>> print('dict is {:.0f} times faster than tuple.'.format(speedup_dict))
dict is 132348 times faster than tuple.
```

Submit to Web-CAT to compute your score!

1. Create a ZIP file for your `cs05` folder by right-clicking the folder and selecting:

- **Send to → Compressed (zipped) folder** on Windows
- **Compress Items** on MacOS
- **Compress** on Linux

You should find the new ZIP file in the same directory where `cs05` resides.

2. Login to <http://webcatvm.cs.appstate.edu:8080/Web-CAT> and submit your ZIP file for grading. You may submit as many times as you want before the deadline.