IS 271 Assignment 1

Due Date
11:59pm April 6, 2016

Setup
As I mentioned in class, you need to install Python 3. You can download it here: https://www.python.org/downloads/

To test it out after installation, open a terminal window and type: `python3` That should start the interactive Python interpreter, in which you can type commands. This interactive shell is very useful for testing out snippets of code while you write your programs.

In class, I used the interactive Python interpreter to show you how to write some basic commands. For your assignments, you will create Python files, which are just plain text files with a `.py` file extension. You can use just about any plain text editor to create these files, such as Notepad, TextWrangler, Vim, Emacs, etc. Some popular text editors offer more features, such as syntax highlighting and autocompletion. These include:

- Sublime Text (I use this one)
- Atom
- Notepad++
- PyCharm

Feel free to try them all out and use whichever one you like best.

Submission
Create a file called `is271_wk1_LAST_FIRST.py`, replacing LAST and FIRST with your name. You will write all your code for this assignment in this file and submit it.

Tasks

0. Example of string formatting and printing.

- create a variable called `course` and assign it the value `'IS271'`
- create a variable called `week` and assign it the value`1`
- now print out the values formatted like so, substituting the parts in angle brackets with the values of your variables...

```
course: <course>
week: <week>
```

To do this, you will need to learn how to format strings and insert the values of variables into them. The official documentation is here:https://docs.python.org/2/library/string.html#format-string-syntax. However, that can be a bit daunting for your first week. For now, you can just look at the following example and copy it. The basic idea is that you create a string literal, and put braces `{}` where you intend to insert a variable, and then call `.format(var1, var2, ...)` on the string and pass in the variables in the same order as the braces.

```
course= 'IS271'
```

```
week = 1
output = 'course: {}\nweek: {}'.format(course, week)
print(output)
```

Later on, we'll see that you can put instructions inside those braces that tell Python how to format the value of the variable, such as padding numbers with 0's, aligning it to the left or right, and so on. For now, we can just use the empty braces to substitute our variables as is.

Creating the variable `output` was not necessary. I did that just to make it a little easier for you to read. We could have put that string literal inside the `print()` statement like so:

```
print('course: {}\nweek: {}'.format(course, week))
```

Be sure to keep track of your parenthesis when nesting functions within functions!

Also, notice the \n in the string literal which represents a line break.

Now save your file, open your terminal, change to the directory in which you saved the file, and try running it. For instance, if you saved it to the desktop you would do the following:

```
my-laptop:~ joshua$ cd Desktop
my-laptop:Desktop joshua$ python3 is271_wk1_gomez_joshua.py
course: IS271
week: 1
```

You can see the results of your print commands are displayed in the terminal. You should run your program after completing each task in the rest of this assignment to make sure it's working. You can use the interactive interpreter to test out pieces of code as well.

1. The "me" dictionary

- Create a variable called `me` and assign it an empty dictionary. We are going to populate it with information about you.
- To the `me` dictionary, assign values for the following keys:
  - `name` (in "last, first" format)
  - `hometown`
  - `undergrad_major`
  - `expected_grad_year` (use an integer, not a string)
- Now print it out like so, substituting the parts in angle brackets with the values in your dictionary. You can use 4 separate print statements or a single one with line breaks in it.

```
My name: <name>
Hometown: <hometown>
Undergrad Major: <undergrad_major>
Expected Graduation: <expected_grad_year>
```

2. Top 5 Authors List

Now we will add a list of your favorite authors and some of their books.

- create an empty list called `fav_auths`
- create a dictionary called `auth1` with the following keys:
  - `auth_name` (last, first)
  - `dob` (year of birth, as an integer)
  - `books` (make this an empty list)
- add values for that dictionary with your favorite author's name and date of birth
- add 2 books to the books list. These should be dictionaries with the following keys:
  - `title`
  - `year` (of publication, as integer)
- add `auth1` to your `fav_auths` list
- repeat the previous 3 steps for 4 more authors

`fav_auth` should now be a list with 5 dictionaries in it. Each of those dictionaries should have 3 keys, including`books`, which is another list of dictionaries containing book titles and years. That's 10 total book dictionaries within 5 books lists within 5 author dictionaries within 1 favorite authors list!

What if we want to print out the whole thing at look at it? If we just do `print(fav_auths)` it will print it as one long string and it will be difficult for human eyes to parse out the information. Try it out for yourself. Doing a bunch of print statements and formatting to get it looking nice and pretty can be laborious. Luckily for us, there is a convenience function called `pprint()` (aka pretty print), that will do that for us.

- add the following command to your program: `from pprint import pprint`
- now print out your dictionary by using`pprint(fav_auth)`

3. String Manipulations

- create a new variable (**not** part of your `me` dictionary) called `flname`. This will be your name represented in "first last" format. But do not just assign the string value yourself! Instead, `split()` the `name` value from your `me` dictionary on the comma and arrange the resulting output in the right order and assign them to the new variable `flname`. (Don't forget to strip the white space)
  - print it out as `flname: <flname>` (substituting the variable value for the angle brackets as we did above)
- now create another variable called `FLname` and do the same as above, but make your name all uppercase using `upper()`.
  - print it out in the same manner.
- create a variable called `rev_name` that is the`flname` value spelled backwards. (hint: use string slicing with a negative step)
  - print it out.
- find the length of your name with `len()`, and assign it to `name_length`.
  - print it out.
- using split(), list indexes, and dictionary keys, create a list called `auths_last` that contains just the last names of each of the authors in your `fav_auths`list.
  - print it out.
- Using the `join()` method, create a variable called`auths_piped` that is a single string of all the names in `auths_last` joined by a pipe: `'|'`.

    - print it out.
- similar to above, create a list called `pub_dates` that contains the publication date of every book in your`fav_auths` data structure.
    - print it out.
- now create a variable called `avg_pub` that is the average of all the dates in `pub_dates`.
    - print it out.

4. Text Processing

- copy the code snippet below which assigns a poem by John Keats to the variable `ode`.
- Using `count()` create a variable `comma_count` and assign the total number of commas in the poem to it.
    - print it out.
- create a variable `line_count` and assign it the total number of lines in the poem.
    - print it out.
- create a variable `word_count` and assign it the total number of words in the poem.
    - print it out.
- Using `replace()` create a variable `EVER_ode` and assign it the same poem but replace all the instances of the word 'ever' with 'EVER' (but don't mess with words like 'never').
    - print it out.

```
ode = """Thou still unravish'd bride of quietness,
Thou foster-child of silence and slow time,
Sylvan historian, who canst thus express
A flowery tale more sweetly than our rhyme:
What leaf-fring'd legend haunts about thy shape
Of deities or mortals, or of both,
In Tempe or the dales of Arcady?
What men or gods are these? What maidens loth?
What mad pursuit? What struggle to escape?
What pipes and timbrels? What wild ecstasy?

Heard melodies are sweet, but those unheard
Are sweeter; therefore, ye soft pipes, play on;
Not to the sensual ear, but, more endear'd,
Pipe to the spirit ditties of no tone:
Fair youth, beneath the trees, thou canst not leave
Thy song, nor ever can those trees be bare;
Bold Lover, never, never canst thou kiss,
Though winning near the goal yet, do not grieve;
She cannot fade, though thou hast not thy bliss,
For ever wilt thou love, and she be fair!

Ah, happy, happy boughs! that cannot shed
Your leaves, nor ever bid the Spring adieu;
And, happy melodist, unwearied,
For ever piping songs for ever new;
```

```
More happy love! more happy, happy love!
For ever warm and still to be enjoy'd,
For ever panting, and for ever young;
All breathing human passion far above,
That leaves a heart high-sorrowful and cloy'd,
A burning forehead, and a parching tongue.

Who are these coming to the sacrifice?
To what green altar, O mysterious priest,
Lead'st thou that heifer lowing at the skies,
And all her silken flanks with garlands drest?
What little town by river or sea shore,
Or mountain-built with peaceful citadel,
Is emptied of this folk, this pious morn?
And, little town, thy streets for evermore
Will silent be; and not a soul to tell
Why thou art desolate, can e'er return.

O Attic shape! Fair attitude! with brede
Of marble men and maidens overwrought,
With forest branches and the trodden weed;
Thou, silent form, dost tease us out of thought
As doth eternity: Cold Pastoral!
When old age shall this generation waste,
Thou shalt remain, in midst of other woe
Than ours, a friend to man, to whom thou say'st,
"Beauty is truth, truth beauty,—that is all
Ye know on earth, and all ye need to know."""
```