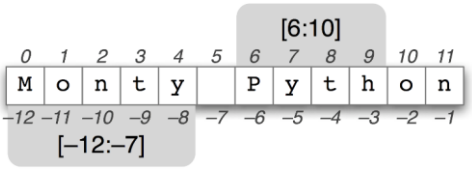
Level 3 Intro Review Exercises



In case you haven't been coding on your own in Python during the last few months here is a few exercises to get you back into the swing of things:

1. Create program that asks the user for a string of any length and then prints out the first 3 characters of the string and then the last 3 characters of the string.

2. Create a program that asks the user to create a **list** of words that is **at least** 7 words long and then prints out the following words in the following order:

    1. Last 2 words of the list
    2. The first 3 words of the list
    3. The fourth word of the list.

3. Create a program that asks the user for a **list of numbers** and then reverses it. You can't use the .reversed() method. **You must do it with slicing**…important!...look it up if you have to…easy!

4. Use the **range** method to create a list of number from 1 to 100 and then use slicing to print out the numbers from100 back to 50 but only every second number. Example: 100,98,96,94,9,…..

5. Use the **range** method to create a list of number from 1 to 100 and then use slicing to print out the numbers from 20 to 80 but only every 4th number example: 20,24,28,32,36,40,……

6. First, create the following two strings:

```
s1 = "   Hey buddy, what's up?  "
s2 = "        Not much Bro!     "
```
 # make sure spaces are part of string!

    Create a program that will **remove the any whitespace at the beginning or the end of each string**. Then add the strings together but include the two spaces before the first and second sting.

    Final product: `Hey buddy, what's up?  Not much Bro!`

Hint: Use the **.strip()** method... and then *concatenate* the stings. (look it up).

7. Create a program that:

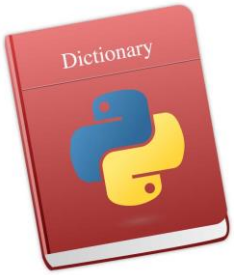
   a) **repeatedly** takes the name of a student and their overall mark in Computer Science 12 from the user (until the use types "done"). Example:

```
student and grade:  Jim 87
student and grade:  Sarah 92
student and grade:  Bill 67
student and grade:  Spencer 100
student and grade:  done
```

   b) stores the input in a **.txt** file that contains each student entered and their grade but in *order of highest to lowest grade*.
   c) asks the user if they want to **add** to the file
   d) asks the user if they want to **view/print** the file on the screen.

   Make sure you program is in the form of a user-friendly interface that prompts the user appropriately and guides them through the available choices.

8. Create a **dictionary** that contains 10 company names as **keys** and the name of their CEOs as **values**. They can be fictional or real…whatever you like. Create a program that will allow the user to:



   a) Add companies and CEOs to the dictionary.
   b) Remove company records from the dictionary.
   c) View all the dictionary keys and values in an easy-to-read printed table (not just printing the dictionary).
   d) Select a company and see what the CEOs name is.

   Make sure you program is in the form of a user-friendly interface that prompts the user appropriately and guides them through the available choices.

9. Create a **dictionary** that consists of 10 keys that are **student names** and paired **values** that are **lists**. Each **list** should contain the following items, in order, for that student: [age, grade, student#, locker#, email address]

Example:
```
students = {
            "Feff Galk":  [17,11,02345,24,"jgalk@sd48.bc.ca"],
            "Mary Finel": [16,10,02119,39,"mfinel@sd48.bc.ca"]
           }
```

Create a program that will allow the user to:

a) Add new students and their data to the dictionary.
b) Remove students and their data from the dictionary.
c) View all the dictionary keys and values in an easy to read printed table (not just printing the dictionary).
d) Allow the user to select an individual student to see only their data
e) Allow the user to select any student *and* the individual data they wish to see.

Make sure you program is in the form of a user-friendly interface that prompts the user appropriately and guides them through the available choices.

10. Write a program containing a **nested for loop** that will allow the user to input 2 numbers: **Columns** and **Rows**. Then the program will output a table of zeros based on the input given. Example:

6 and 4

000000
000000
000000
000000

11. Use a **nested** for loop to print out the following display (tricky!).

1
22
333
4444
55555
666666

12. You are hired by a logging company to keep track of a section of forest that you are harvesting. Use a **nested for loop** to create a 10 x 10 **2D list** that represents a square area of forest. Initially has each element in the 2D list will be the string '**NL**' (not logged).


```
LoggingBlock = [[ 'NL', 'NL', 'NL', 'NL', 'NL', 'NL', 'NL', 'NL', 'NL', 'NL'],
                 [ 'NL', 'NL', 'NL', 'NL', 'NL', 'NL', 'NL', 'NL', 'NL', 'NL'],
                 .
                 .
                 .
                ]
```

Then create a program that will allow the **user** to:

a) *any* section the wish from *not logged* 'NL' to 'L' *logged* in the 2D list.
b) print out the updated 2D list in a neat format
c) count the number of logged sections.
d) Find out if the **entire *left* half** of the Logging block has been harvested.

Make sure you program is in the form of a user-friendly interface that prompts the user appropriately and guides them through the available choices.

13. Write a **function** that accepts two **arguments**:

a) a list called: (list1)
b) and a positive integer called: (chop)

the function will divides list1 into new lists of lengths of "chop" and puts them into a new list called: (list2)….creating a "*list of lists*". For example,

```
[1,2,3,4,5,6,7,8], 4 => [[1,2,3,4], [5,6,7,8]]
[1,2,3,4,5,6,7,8], 3 => [[1,2,3], [4,5,6], [7,8]]
[1,2,3,4], 1         =>  [[1],[2],[3],[4]]
```

14. Use the **PyGame** Module to create a game that allows the user to select randomly appearing circle on the screen of a certain color using the mouse.

the game should have the following features:

a) The game begins with different colored circles being continually added to the screen in random spots (circles should not overlap).
b) Some text will appear to indicate to the user the color of circles that need to be selected.
c) Once the text has appeared. The user will use the mouse to click on as many circles as they can in 6 seconds. During this time new circles will be continued to be generated.
d) After 6 seconds has ended, the game will end and the number of circles captured will be reported to the user.

Make sure you program is in the form of a user-friendly interface that prompts the user appropriately and guides them through the objective of the game.