****

**LIsTs**

Remember to **startlab** and follow the directions. We are working on day 5!

Remember to **endlab** and follow the directions when you are done.

## **Part 1 - Fun with Indexing**

1. Run this block of code on the shell.

|  |
| --- |
| >>> a = ['she','sells','sea','shells','by','the','sea','shore'] >>> b = "selfish shellfish" >>> c = [1, 1, 2, 3, 5, 8, 13] |

Fill in the blanks below according to the example. Some may have **more than one** correct **answer**. Fill in the **guess** column before you try it on your computer.

|  |  |  |
| --- | --- | --- |
| **List or String Indexing** | **Guess** | **Result** |
| *a[1]* |  | *'sells'* |
| b[3:4] | lf | ‘f’ |
| c[6] | 8 | 13 |
| c[:-2] | 8 | 8 |
| a[2] | ‘sea’ | 'sea' |
| a[2:4] | ‘2, 3’(i look at c by accedent) | [‘sea’, ‘shells’] |
| a[3][0:6] | ‘shell’ | 'shells' |
| B[7:13] | ‘ shell’ | ' shell' |
| c[1] + c[2] | 2 | 3 |

Now, with a partner, type the following into your shells and read the output of this code to each other **very fast**. You can film yourselves doing it! Save the video and show it to an instructor:

>>> a\*3

2. Write **True** or **False** for each expression below. **First** try to **guess**, **then** check in the **shell**:

'by' in a True [1,2,5] in c True .. False

'self' in b True 'sh' in c True .. False

a[2] == a[6] True a[3] == b[8:13] False

3. Try out this code, which uses **len()**. Fill in the blanks with the results that would appear:

|  |
| --- |
| >>> dog = 'dalmatian' >>> len(dog) 9 >>> dogs = [dog,'poodle','boxer'] >>> len(dogs)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

**Part 2 - Shelli**

Run these two lines of code by **typing** them in the **shell**.

>>> one = [1,2,3,4]

>>> two = [7,6,5,4]

>>> three = ["y1", "friends", "fun"]

What gets printed to the screen after running the code in each box? (some boxes produce errors). First, you should **guess**, and then check the answer in the **shell**.

|  |  |  |
| --- | --- | --- |
| Input | Guess | Shell Output |
| >>> print (one + two) |  |  |
| >>> print (one[3]) |  |  |
| >>> one.remove(4)  >>> print(one) |  |  |
| >>> one.append(4)  >>> print(one) |  |  |
| >>> one.pop(1) |  |  |
| >>> print(one) |  |  |
| >>> four = one + three  >>> print(four) |  |  |
| >>> one = [1,2,3,4]  >>> print(four) |  |  |
| >>> 'fun' in three |  |  |
| >>> three.pop(2) |  |  |
| >>> three.remove('fun') |  |  |
| >>> 'fun' in three |  |  |
| >>> one[3] = two[0] |  |  |
| >>> one[3] == two[0] |  |  |
| >>> one.sort()  >>> print(one) |  |  |
| >>> two.sort()  >>> print(two) |  |  |
| >>> three.sort()  >>> print(three) |  |  |
| >>> one[3] == two[0] |  |  |
| >>> five = two + three  >>> print(five) |  |  |
| >>> five.sort()  >>> print(five) |  |  |
| >>> print(len(five)) |  |  |
| >>> five.pop(8) |  |  |

**Part 3 - Fun with LIsTs**

Make a new file called stringfun.py

Copy the following code into your file and follow the instructions in the comments.

|  |
| --- |
| #PROBLEM ONE  #You will be given a list named our\_list  #Print True if our\_list is longer than 2 elements AND  #the first and last elements are the same.  #If the list is 2 elements OR shorter OR the first and  #last elements are not the same, print False.  #Hint: to test your code, define an example value for  #our\_list at the top of your code.  #PROBLEM TWO  #It may be helpful to comment out the code from  #problem one while you are working on this section.list#and list\_two.  #Print a list containing elements for which the same  #index in both lists has the same element. |

Here are examples of how the functions should work:

|  |
| --- |
| >>> our\_list = [1,2,1] #Example for problem one  \*Run your code for problem one\*  True  >>> our\_list = [2,2] #Example for problem one  \*Run your code for problem one\*  False  >>> our\_list = [1,2,3] #Example for problem one  \*Run your code from problem one\*  False  >>> list\_one = [1,2,3] #Example for problem two  >>> list\_two = [3,2,1]  \*Run your code from problem two\*  [2] |

**YOU: **

**Bonus One:**

Still unsure about list operations, do this bonus: shelli on your own. Pick your own lists and try out the commands we learned on them. You can also try the following operations we didn't get to.

>>> one = [1,2,3,4]

>>> two = [7,6,5,4]

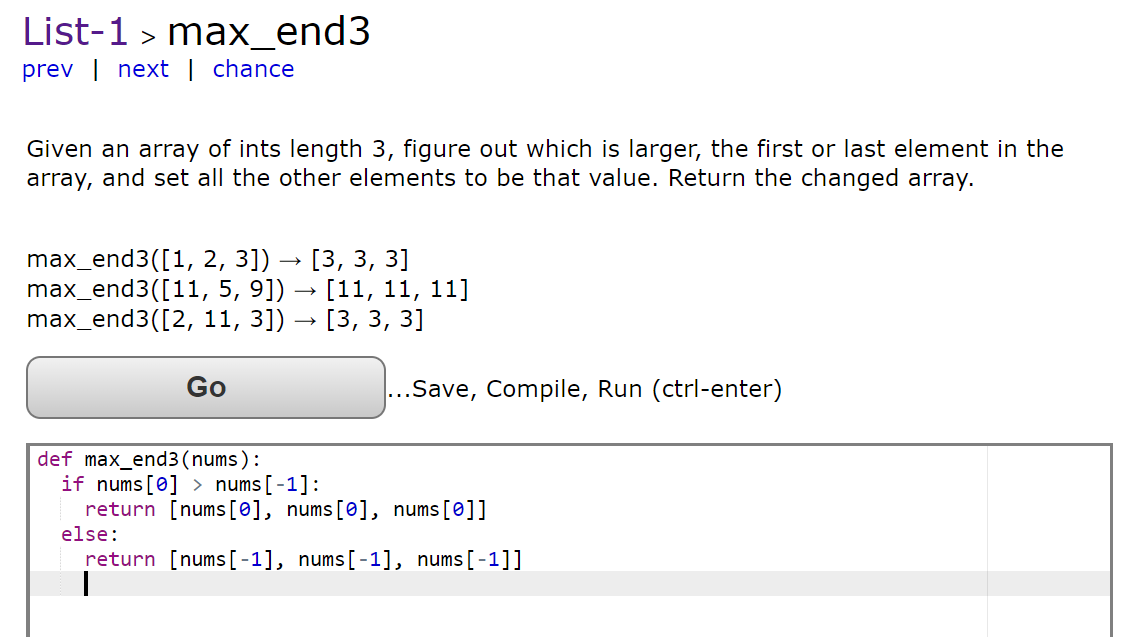
|  |  |  |
| --- | --- | --- |
| Input | Guess | Shell Output |
| >>> one.reverse()  >>> print(one) |  |  |
| >>> one.sort()  >>> print(one) |  |  |
| >>> one.remove(4)  >>> print(one) |  |  |
| >>> three = [3]  >>> four = three\*3  >>> print(four) |  |  |
| >>> five = one.copy()  >>> one.sort()  >>> print(one)  >>> print(five) |  |  |

You're turn! Come up with your own shelli.

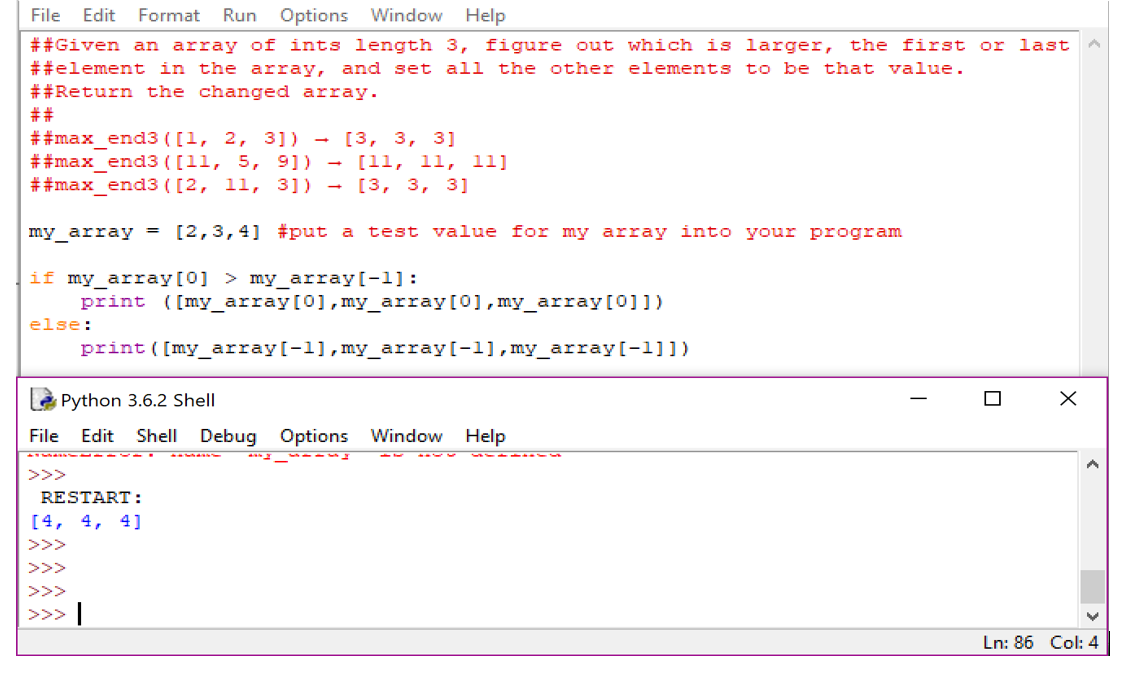
**Bonus Two :**

If you want more practice thinking about how put programs together, do this practice. Work on whichever problem there suits your fancy (whichever problem you want to). This will require you to use functions - a topic we will cover tomorrow. If you haven't seen them before, write a program that will output the response they request. You will need to put an example of the input into the shell or at the top of your function before you run it. For now, think of an array as being the same thing as a list. <http://codingbat.com/python/List-1>.

Example in coding bat:



Or if that looks like gibberish (nonsense):



**Survey Time:**

<https://goo.gl/forms/j1CLWCB6jLC7iOQG3>