**EXERCISE 5: LISTS AND FOR-LOOPS**

Just when you had become an expert at string slicing, you discovered another sliceable data type: lists. However, lists are diﬀerent from strings in that they are mutable. Not only can we slice a list, but we can also change its contents. The purpose of the lab is to introduce you to these new features, and demonstrate just how powerful the list type can be.

This lab will also give you some experience writing functions with for-loops

1. **List Expressions and Statements**

The first part of the lab will take place in the Python interactive shell, much like the first two labs. You do not need to create a module. Instead, you will be filling in tables in response to certain expressions and statements. These tables are very similar to exercises you have seen before.

Before starting the first table, enter the following assignment statement into the interactive shell:

>>> alist = ['H','e','l','l','o',' ','W','o','r','l','d','!']

Like a string, this is a list of individual characters. Unlike a string, however, the contents of this list can be changed. This makes lists a very important data type.

Now that you have the variable alist, enter the following statements in the order they are presented. Many of the commands below are statements, not expressions. Hence, they are all followed by a print statement showing some output. In each case, write down what you see in the second column. If the command results in an error, simply write Error.

|  |  |
| --- | --- |
| **Statements** | **Output** |
| alist.remove('o')  print(alist) | ['H', 'e', 'l', 'l', ' ', 'W', 'o', 'r', 'l', 'd', '!'] |
| alist.remove('x') | Error |
| pos = alist.index('o')  print(pos) | 6 |
| pos = alist.index('B') | Error |
| alist[0] = 'J'  print(alist) | ['J', 'e', 'l', 'l', ' ', 'W', 'o', 'r', 'l', 'd', '!'] |
| alist.insert(4,'o')  print(alist) | ['J', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd', '!'] |
| s = alist[:]  print(s) | ['J', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd', '!'] |
| s[0] = 'C'  print(s)  print(alist) | ['C', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd', '!'] |
| s[0] = 'C'  print(s)  print(alist) | ['C', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd', '!'] |
| a = '-'.join(s)  print(a) | 'C-e-l-l-o- -W-o-r-l-d-!' |
| a = ''.join(s)  print(a) | 'Cello World!' |
| t = list(a)  print(t) | ['C', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd', '!'] |

For the next table, we want you to reassign alist, as follows:

>>> alist = list('CS1110')

In the table below, the commands are all missing something, represented by a box. That something may be a literal or a variable. The second column displays the output. You need to fill in the box to give the desired output. If you are confused, go back and look at your answers in the first table.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Statements | | | | | | | | |  | Output | Missing |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| alist.remove( | | | |  |  | ) | |  |  |  |  | ['C','S','1','1','0'] | ‘1’ |  |
| print(alist) | | | | | |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  | | | |  |  |  |  |  |  |  |  |
|  |  |  |  | | | |  |  |  |  |  |  |  |  |
| pos = alist.index( | | | | | | | | |  |  | ) | 4 | ‘0’ |  |
| print(pos) | | | | | |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| pos = | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| alist.index('1', | | | | | |  |  |  | ) |  |  | 3 | 2 |  |
| print(pos) | | | | | |  |  |  |  |  |  |  |  |  |
|  |  |  |  | | |  | | | | |  |  |  |  |
|  |  |  |  | | |  | | | | |  |  |  |  |
| alist.insert( | | | |  |  | ,'I') | | | | |  | ['C','I','S','1','1','0'] | 1 |  |
| print(alist) | | | | | |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  | | | | | | | |  |  |  |  |
|  |  |  |  | | | | | | | |  |  |  |  |
| a = |  | .join(alist) | | | | | | | | |  | 'C.I.S.1.1.0' | ‘.’ |  |
| print(a) | | | | | |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  | |  |  | | |  |  |  |  |  |  |  |  |  |
|  | |  |  | | |  |  |  |  |  |  |  |  |  |
| alist[ | |  | ]='2' | | |  |  |  |  |  |  | ['C','I','S','1','1','2'] | 5 |  |
| print(alist) | | | | | |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1. **List Functions**

In the file lab05.py are the stubs of several functions. In addition, we have already provided you with test cases in test05.py. So all you need to do is implement the functions.

These implementations will require for-loops. You may find the following list methods useful.

|  |  |
| --- | --- |
| **Method** | **Description** |
| x.index(c) | Returns: the first position of c in list x; error if not there. |
| x.count(c) | Returns: the number of times that c appears in the list x. |
| x.append(c) | Adds the value c to the end of the list. This method alters the list; it does not make a new list. |

Lists do not have a find() method like strings do. They only have index(). To check if an element is in a list, use the in operator (e.g. x in thelist).

Implement the functions specified on the next page within the module lab05.py. This time we do not need you to list any test cases (they are already provided in test05.py). However, it might be a good idea to test the functions before attempting to get credit.

Function lesser\_than(alist,value). This first function should not alter alist. If you need to call a method that might alter the contents of alist, you should make a copy of it first.

def lesser\_than(alist,value):

"""

Returns: number of elements in alist strictly less than value

Example: lesser\_than([5, 9, 1, 7], 6) evaluates to 2

Precondition: alist is a list of ints

Precondition: value is an int

"""

Function uniques(alist). Once again, this function should not alter alist.

def uniques(alist):

"""

Returns: The number of unique elements in the list.

Example: uniques([5, 9, 5, 7]) evaluates to 3

Example: uniques([5, 5, 1, 'a', 5, 'a']) evaluates to 3

Precondition: alist is a list.

"""

Function clamp(alist,min,max). Unlike the other two functions, this last function does alter alist. This function is a procedure with no return value. You might want to look at test05.py to see how we would test a procedure like this.

def clamp(alist,min,max):

"""

Modifies the list so that every element is between min and max.

Any number in the list less than min is replaced with min. Any number in the list greater than max is replaced with max. Any number between min and max is left unchanged.

This is a PROCEDURE. It modifies alist, but does not return a new list.

Example: if alist is [-1, 1, 3, 5], then clamp(thelist,0,4) changes alist to have [0,1,3,4] as its contents.

Precondition: alist is a list of numbers (float or int) Precondition: min <= max is a number Precondition: max >= min is a number """

4