Lists in Python

Leigh J. Little SUNY Brockport November 30, 2018

Contents

1 Introduction

This document will discuss the list structure in Python and how to perform basic operations with it.

2 Lists

In MATLAB, the most basic non-scalar data type is an array. In Python, there is no direct equivalent to an array that is built into the language; rather Python works with *lists*. In some respects, a list is similar to an array, but has much more flexibility in the list elements

Over the years, many arguments about they way computers should behave have emerged. One of the oldest is: Should indexed variables (like arrays and lists) start indexing at 1 or 0? This argument is probably 60 years old and is still going on despite the fact that everything that can be said has been said. Both approaches have good and bad merits, but in the long run the issue should not be that significant of a concern. In any case, the key thing to be aware of when working with lists is that the first element of the list is element 0, not element 1 as in MATLAB. For comparison, the first element of a Fortran array is element 1 (though this can be changed). Most other languages use element 0 as the first element.

To generate a simple list in Python, use the [ ] constructors

```
>>> x = [4, 6, 7, -3, 2]    # Note the commas
>>> x
[4, 6, 7, -3, 2]
>>> x[0]
4
>>> x[4]
2
```

Some important items to note from the above example:

a) The commas between list elements are required.

b) The list doesn't have an orientation (row or column).

c) The elements are referenced using [ ] instead of ( ).

d) The first element in the list is `x[0]`.

e) The list has 5 elements and the last one is `x[4]`.

2.1 Example 1 - Summing the elements in a list

We can use a list of numerical values in much the same way we have used vectors in MATLAB. Enter the following in a script called `testlists.py`

```
x = [4, 6, 7, -3, 2]
n = len(x)
total = 0
for i in range(0,n):
    total = total + x[i]
print('Sum of vector = ',total)
```

Note that the `len` function is used to get the length of the list.

2.2 The `range` Function

The basic form of the range function is

```
range(n)
```

This will generate a list of `n` consecutive integers, starting at 0. Thus, `range(5)` will give the list 0,1,2,3,4. When you modify the `range` function with an optional initial argument like

```
range(1,n)
```

The list will start at 1, but still end at `n-1`. The reason for this unusual behavior is tied into the fact that the first element of a list is element 0.

3 Unusual Lists

What makes lists so flexible in Python is that an element of a list can be anything. In this aspect, a list is like a cell array in MATLAB. For example, the list `a` below is valid

```
>>> a = [4, 5.2, 'Car', [1,2,3]]
```

This list contains 4 elements. `a[0]` is the integer 4, `a[1]` is the double 5.2, `a[2]` is the string 'Car' and `a[3]` is the list [1, 2, 3].

Because lists like this commonly occur, there is an alternative method to loop over the elements in a list

```
a = [4, 5.2, 'Car', [1,2,3]]
for i in a:
    print(i)

4
5.2
Car
[1, 2, 3]
```

3.1 Adding Elements to an Existing List

In MATLAB, if we have an array that we need to add elements to, we can do

```
>> b = [1 2 3 4 5]
b =
     1     2     3     4     5
>> b(6) = 6;
>> b(7) = 7
b =
     1     2     3     4     5     6     7
```

If you attempt to do this in Python, an error will be generated

```
>>> b = [1,2,3,4,5]
>>> b
[1, 2, 3, 4, 5]
>>> b[5] = 6
Traceback (most recent call last):
  File "<pyshell#29>", line 1, in <module>
    b[5] = 6
IndexError: list assignment index out of range
```

In order to add an element to an array, you need to use the `append` command.

```
>>> b = [1,2,3,4,5]
>>> b
[1, 2, 3, 4, 5]
>>> b.append(6)
>>> b
[1, 2, 3, 4, 5, 6]
```

3.2 Reading Data from Files

The native file capabilities for working with data files in Python are very primitive. As a result, there are numerous modules that be created to make reading data files easier.

4 String Variables

Python has string variables, which are denoted by either single or double quotes, for example,

```
>>> s = "Today is Friday"
>>> s
'Today is Friday'
>>> len(s)
15
>>> s[0]
'T'
>>> s[13]
'a'
>>> for i in s: print(i)
T
o
d
a
y

i
s

F
r
i
d
a
y
```

String variables are basically a special type of list. In particular, string variables are *immutable*. This means that once they are assigned, they cannot be modified.

```
>>> s[10] = 'r'
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
```

```
    s[10] = 'r'
TypeError: 'str' object does not support item assignment
>>>
```

Otherwise they behave like any other list.

5 Reading Columns from Text Files

The file reading capabilities native to Python are fairly primitive. This has given rise to a number of modules designed to make this easier. Fortunately, if all you need is to extract a few columns from a text data file this can be done with native Python commands. To extract a column from a data file, use the process below:

```
colnum = 2                          # Column number to extract
varname = []                        # Variable you want the column stored in
with open(filename) as fnum:        # Filename is the name of the file stored
                                    # in a string variable
    for currline in fnum:           # Loop over lines of the file
        linesplit = currline.strip().split()    # For each line, break the line into
                                                # individual list elements
        varname.append(int(linesplit[colnum-1]))   # Append column element to output list
                                                   # Change type as necessary
```