





























 **jupyter** Lab 2 Last Checkpoint: 10/05/2017 (autosaved)  Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
<ipython-input-57-063f1f0653> in <module>()
----> 1 '4' < 3
```

```
TypeError: unorderable types: str() < int()
```

```
In [58]: number = 4
        number_str = '4'
```

```
In [59]: type(number)
```

```
Out[59]: int
```

```
In [60]: type(number_str)
```

```
Out[60]: str
```

Length

```
In [61]: a = 'abc'
        b = 'whats your name?'
```

```
In [62]: len(a)
```

```
Out[62]: 3
```

```
In [63]: len(b)
```

```
Out[63]: 16
```

Indexing & Slicing

```
In [74]: alphabets = 'abcdefghijklmnopqrstuvwxyz'
```

```
In [75]: alphabets[0]
```

```
Out[75]: 'a'
```

Important: Indexing starts from 0

```
In [76]: alphabets[-1]
```

```
Out[76]: 'z'
```

```
In [85]: alphabets[0] = 'a'
```

```
-----
TypeError                                Traceback (most recent call last)
<ipython-input-85-40f6d5b512b3> in <module>()
----> 1 alphabets[0] = 'a'
```

```
TypeError: 'str' object does not support item assignment
```

```
In [77]: alphabets[1]
```

```
Out[77]: 'b'
```

```
In [78]: alphabets[5]
```

```
Out[78]: 'f'
```

```
In [79]: alphabets[1:5]
```

```
Out[79]: 'bcde'
```

Note: when [a:b] means, a is inclusive and b is exclusive

```
In [80]: #Entire string
        alphabets[:]
```

```
Out[80]: 'abcdefghijklmnopqrstuvwxyz'
```

```
In [81]: #Alternate characters
        alphabets[::2]
```

```
Out[81]: 'acegikmoqsuwy'
```

```
In [82]: #Evry 3rd character
        alphabets[::3]
```

```
Out[82]: 'adgjmpsvy'
```

```
In [83]: #Reversing the string
        alphabets[::-1]
```

```
Out[83]: 'zyxwvutsrqponmlkjihgfedcba'
```

```
In [84]: alphabets[::-2]
```

```
Out[84]: 'zxvtrpnljhfdb'
```

```
In [5]: str1 = 'Apple'
        str2 = 'Apple'

        str1 == str2
```

```
Out[5]: True
```

```
In [7]: # Question : Print Last Letter of the string 'Duke'
```

```
str1 = "Duke"
length = len(str1)
print (str1[length -1 ])
```

```
e
```

Lower and Upper case characters

```
In [10]: str1 = "lower"
         str1.upper()
```

```
Out[10]: 'LOWER'
```

```
In [11]: str2 = "UPPER"
         str2.lower()
```

```
Out[11]: 'upper'
```

Lists

A list is an ordered set of values, where each value is identified by an index. The values that make up a list are called its elements. Lists are similar to strings, which are ordered sets of characters, except that the elements of a list can have any type.

Lists are mutable.

Creating List

```
In [13]: list1 = [1, 2.5, 'a', 'b', 'physics', 'chemistry']
```

Indexing and accesing elements of list

```
In [20]: list1 = ['1','2','3','4','5','a','b','c','d','e']
```

```
In [88]: list1
```

```
Out[88]: ['1', '2', '3', '4', '5', 'a', 'b', 'c', 'd', 'e']
```

```
In [89]: list1[0]
```

```
Out[89]: '1'
```

```
In [90]: list1[1:6]
```

```
Out[90]: ['2', '3', '4', '5', 'a']
```

```
In [91]: list1[::-2]
```

```
Out[91]: ['1', '3', '5', 'b', 'd']
```

Updating List

```
In [28]: list1 = [1,2,3,4,5,6]
```

```
In [29]: list1
```

```
Out[29]: [1, 2, 3, 4, 5, 6]
```

```
In [30]: list1[0] = 'a'
```

```
In [31]: list1
```

```
Out[31]: ['a', 2, 3, 4, 5, 6]
```

```
In [32]: list1 + ['f']
```

```
Out[32]: ['a', 2, 3, 4, 5, 6, 'f']
```

```
In [33]: #ask what should ne the output
         list1.index('f')
```

```
-----
ValueError                                Traceback (most recent call last)
<ipython-input-33-1141df27eb2b> in <module>()
      1 #ask what should ne the output
----> 2 list1.index('f')

ValueError: 'f' is not in list
```

```
In [34]: list1
```

```
Out[34]: ['a', 2, 3, 4, 5, 6]
```

```
In [35]: list1 = list1 + ['f']
```

```
In [36]: list1
```

```
Out[36]: ['a', 2, 3, 4, 5, 6, 'f']
```

```
In [37]: list1.index('f')
```

```
Out[37]: 6
```

```
In [38]: del list1[5]
```

Which element do you expect to be deleted?

```
In [39]: list1
```

```
Out[39]: ['a', 2, 3, 4, 5, 'f']
```

6 is not anymore in the list

Basic List operations

```
In [41]: list1
```

```
Out[41]: ['a', 2, 3, 4, 5, 'f']
```

```
In [47]: # Checking Length  
len(list1)
```

```
Out[47]: 6
```

```
In [48]: # adding 2 Lists  
list2 = ['A', 'B', 'C']  
list1 + list2
```

```
Out[48]: ['a', 2, 3, 4, 5, 'f', 'A', 'B', 'C']
```

```
In [49]: #Multiplying Lists  
list2 * 3
```

```
Out[49]: ['A', 'B', 'C', 'A', 'B', 'C', 'A', 'B', 'C']
```

```
In [50]: #Chekcing if the element is in the list or not  
3 in list1
```

```
Out[50]: True
```

```
In [51]: '3' in list1
```

```
Out[51]: False
```

Built-in List Functions

```
In [121]: list1
```

```
Out[121]: ['A', '2', '3', '4', '5', 'a', 'b', 'c', 'd', 'e', 'One', 'One']
```

```
In [52]: #Appending element in front of list  
list1.append('One')  
list1
```

```
Out[52]: ['a', 2, 3, 4, 5, 'f', 'One']
```

```
In [53]: #Counting number of occurances of particular element  
list1.count('One')
```

```
Out[53]: 1
```

```
In [54]: #Checking index of element  
list1.index('One')
```

```
Out[54]: 6
```

```
In [55]: #Removing the Last index element  
list1.pop()  
list1
```

```
Out[55]: ['a', 2, 3, 4, 5, 'f']
```

```
In [57]: #Removing one particular element from list  
list1.remove('f')  
list1
```

```
Out[57]: ['a', 2, 3, 4, 5]
```

```
In [58]: #Reversing the list  
list1.reverse()  
list1
```

Out[58]: [5, 4, 3, 2, 'a']

Nested Lists

Can a list have another list as element? YES!

```
In [69]: list1 = ['a', 'b', 'c']
list2 = [1, 2, 3, list1]
list2
```

Out[69]: [1, 2, 3, ['a', 'b', 'c']]

```
In [71]: # Ask for output
list2[3][2]
```

Out[71]: 'c'

Dictionaries

Dictionaries are similar to other compound types except that they can use any immutable type as an index. As an example, we will create a dictionary to translate English words into Spanish. For this dictionary, the indices are strings.

General

```
In [72]: #String as keys
dict_1 = {'key1': 1, 'key2': 2, 'key3': 3}
```

```
In [26]: dict_1['key2']
```

Out[26]: 2

```
In [27]: dict_1[0]
```

```
-----
KeyError                                Traceback (most recent call last)
<ipython-input-27-faffe0d0771d> in <module>()
----> 1 dict_1[0]

KeyError: 0
```

```
In [37]: dict_2 = {'k1':1, 'k2':3.1416, 'k3':'string'}
```

```
In [38]: dict_2['k3']
```

Out[38]: 'string'

```
In [39]: dict_2
```

Out[39]: {'k1': 1, 'k2': 3.1416, 'k3': 'string'}

```
In [40]: dict_2['k1'] + 99
```

Out[40]: 100

```
In [41]: dict_2['k1']
```

Out[41]: 1

```
In [34]: dict_2['k1'] = dict_2['k1'] + 99
```

```
In [35]: dict_2['k1']
```

Out[35]: 100

```
In [43]: # Empty Dictionary
dict_3 = {}
```

```
In [44]: dict_3
```

Out[44]: {}

```
In [45]: dict_3['key1'] = 'value1'
```

```
In [46]: dict_3['key2'] = 'value2'
```

```
In [47]: dict_3
```

Out[47]: {'key1': 'value1', 'key2': 'value2'}

```
In [53]: #Dictionaries are very flexible in data types they hold.
#for e.g. Following dictionary has 1 list, and 1 nested dictionary in it
```

```
In [54]: dict_4 = {'k1': 123, 'k2':[11,12,13], 'k3':{'nestkey':{'subnestkey':'value'}}
```

```
In [55]: dict_4['k3']
```

Out[55]: {'nestkey': {'subnestkey': 'value'}}

```
In [56]: dict_4['k3']['nestkey']
```

```
Out[56]: {'subnestkey': 'value'}
```

```
In [57]: dict_4['k3']['nestkey']['subnestkey']
```

```
Out[57]: 'value'
```

Operations with dictionaries

```
In [1]: inventory = {'apples':430 , 'bananas' : 312, 'oranges' : 525}
```

```
In [2]: print (inventory)
```

```
{'apples': 430, 'bananas': 312, 'oranges': 525}
```

```
In [5]: inventory.keys()
```

```
Out[5]: dict_keys(['apples', 'oranges'])
```

```
In [6]: inventory.values()
```

```
Out[6]: dict_values([430, 525])
```

someone bought all bananas

```
In [3]: del inventory['bananas']
```

```
In [4]: inventory
```

```
Out[4]: {'apples': 430, 'oranges': 525}
```