Programming Fundamental

Lab#10

Contents

Nested List / List of List

Lists can contain elements of different types, including other lists, as illustrated here:

```
>>> small_birds = ['hummingbird', 'finch']
>>> extinct_birds = ['dodo', 'passenger pigeon', 'Norwegian Blue']
>>> carol_birds = [3, 'French hens', 2, 'turtledoves']
>>> all_birds = [small_birds, extinct_birds, 'macaw', carol_birds]
```

So what does `all_birds`, a list of lists, look like?

```
>>> all_birds
[['hummingbird', 'finch'], ['dodo', 'passenger pigeon', 'Norwegian Blue'], 'macaw',
[3, 'French hens', 2, 'turtledoves']]
```

It's the second item we specified, extinct_birds. If we want the first item of extinct_birds, we can extract it from all_birds by specifying two indexes:

Nested for Loop

```
In [1]: a=[[1,2],[3,4]]
        for i in range(2):
            for j in range(2):
                print a[i][j]
```

```
1
2
3
4
```

List Comprehension

Syntax:

```
[expression for item in list]
```

FOR loop:

```
In [2]: digits=[d for d in range(0,10)]
        print digits
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

If-else Statement:

```
In [3]: even =[i for i in digits if i%2 == 0]
        print even
```

```
[0, 2, 4, 6, 8]
```

```
In [4]: obj = ["Even" if i%2==0 else "Odd" for i in digits]
        print(obj)
```

```
['Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd']
```

Tuple

A tuple is a collection which is **unchangeable**. In Python tuples are written with round brackets. It is also known as constant list

Create Tuple

```
In [8]: a=()
        print a
```

```
()
```

```
In [10]: a=("a","b")
         print a
```

```
('a', 'b')
```

Accessing Tuple Item

You can access tuple items by referring to the index number:

```
In [11]: a=("a","b")
         print a[1]

b
```

Multiple assignment through tuple

You can do multiple assignment using tuple

```
In [2]: t=(1,2,3)
        a,b,c=t
        print "Value of a:",a
        print "Value of b:",b
        print "Value of c:",c

Value of a: 1
Value of b: 2
Value of c: 3
```

Loop through Tuple

```
In [12]: thistuple = ("apple", "banana", "cherry")
         for x in thistuple:
             print(x)

apple
banana
cherry
```

Change, Add and Remove Tuple Item

Once a tuple is created, you cannot change it's values. Tuples are **unchangeable**.

```
In [14]: thistuple = ("apple", "banana", "cherry")
         thistuple[1] = "blackcurrant"
         # The values will remain the same:
         print(thistuple)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-14-e5050d128a30> in <module>()
      1 thistuple = ("apple", "banana", "cherry")
----> 2 thistuple[1] = "blackcurrant"
      3 # The values will remain the same:
      4 print(thistuple)

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

Tuples are **unchangeable**, so you cannot remove items from it, but you can delete the tuple completely:

```
In [16]: thistuple = ("apple", "banana", "cherry")
         del thistuple
         print(thistuple) #this will raise an error because the tuple no longer exists
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-16-67c5b2f930ca> in <module>()
      1 thistuple = ("apple", "banana", "cherry")
      2 del thistuple
----> 3 print(thistuple) #this will raise an error because the tuple no longer exists

NameError: name 'thistuple' is not defined
```

Basic Function use on Tuple

1. Whether item is in tuple or not:

```
In [18]: thistuple = ("apple", "banana", "cherry")
         if "apple" in thistuple:
             print("Yes, 'apple' is in the fruits tuple")
```

```
Yes, 'apple' is in the fruits tuple
```

2. Count

```
In [19]: thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)

         x = thistuple.count(5)

         print(x)
```

```
2
```

3. Length

```
In [20]: thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)

         x = len(thistuple)

         print(x)
```

```
10
```

4. Index

```
In [21]: thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)

         x = thistuple.index(8)

         print(x)
```

```
3
```

Dictionaries

A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and they have keys and values.

Create

Creating a dictionary is as simple as placing items inside curly braces {} separated by comma.
An item has a key and the corresponding value expressed as a pair, key: value. While values can be of any data type and can repeat, keys must be of immutable type (string, number or tuple with immutable elements) and must be unique

```
In [13]: thisdict ={}
         print(thisdict)
```

```
{}
```

```
In [ ]: thisdict ={
          "brand": "Ford",
          "model": "Mustang",
          "year": 1964
        }
        print(thisdict)
```

Accessing Item

You can access the items of a dictionary by referring to its key name, inside square brackets:

```
In [6]: x = thisdict["model"]
        print x
```

```
Mustang
```

You can also access the items of dictionary using get()

```
In [7]: x = thisdict.get("model")
        print x
```

```
Mustang
```

Change or Add values

Dictionary are mutable. We can add new items or change the value of existing items using assignment operator.

If the key is already present, value gets updated, else a new key: value pair is added to the dictionary.

Change:

```
In [9]: thisdict = {
          "brand": "Ford",
          "model": "Mustang",
          "year": 1964
        }
        thisdict["year"] = 2018
        print thisdict
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 2018}
```

Add:

```
In [11]: thisdict = {
           "brand": "Ford",
           "model": "Mustang",
           "year": 1964
         }
         thisdict["color"] = "Black"
         print thisdict
```

```
{'color': 'Black', 'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

Remove values

Here are several methods to remove items from a dictionary:

1. **Pop()**

```
In [12]: thisdict = {
           "brand": "Ford",
           "model": "Mustang",
           "year": 1964
         }
         thisdict.pop("model")
         print(thisdict)
```

```
{'brand': 'Ford', 'year': 1964}
```

2. **Popitem()**

The popitem() method removes the last inserted item (in versions before 3.7, a random item is removed instead):

```
In [13]: thisdict = {
           "brand": "Ford",
           "model": "Mustang",
           "year": 1964
         }
         thisdict.popitem()
         print(thisdict)
```

```
{'model': 'Mustang', 'year': 1964}
```

3. **Del**

The del keyword removes the item with the specified key name:

```
In [14]: thisdict = {
           "brand": "Ford",
           "model": "Mustang",
           "year": 1964
         }
         del thisdict["model"]
         print(thisdict)
```

```
{'brand': 'Ford', 'year': 1964}
```

The del keyword can also delete the dictionary completely:

```
In [15]: thisdict = {
           "brand": "Ford",
           "model": "Mustang",
           "year": 1964
         }
         del thisdict
         print(thisdict)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-15-45a6f6e38541> in <module>()
      5 }
      6 del thisdict
----> 7 print(thisdict)

NameError: name 'thisdict' is not defined
```

4. Clear()

The clear() keyword empties the dictionary:

```
In [16]: thisdict = {
           "brand": "Ford",
           "model": "Mustang",
           "year": 1964
         }
         thisdict.clear()
         print(thisdict)
```

```
{}
```

Loop through Dictionary

You can loop through a dictionary by using a `for` loop.

When looping through a dictionary, the return value are the *keys* of the dictionary, but there are methods to return the *values* as well.

1. Print all key names in the dictionary, one by one:

```
In [17]: thisdict = {
           "brand": "Ford",
           "model": "Mustang",
           "year": 1964
         }
         for x in thisdict:
             print(x)
```

```
brand
model
year
```

2. Print all *values* in the dictionary, one by one:

```
In [18]: for x in thisdict:
             print(thisdict[x])
```

```
Ford
Mustang
1964
```

3. You can also use the values() function to return values of a dictionary:

```
In [19]: for x in thisdict.values():
             print(x)
```

```
Ford
Mustang
1964
```

4. Loop through both *keys* and *values*, by using the items() function:

```
In [20]: for x, y in thisdict.items():
             print(x, y)
```

```
('brand', 'Ford')
('model', 'Mustang')
('year', 1964)
```

Basic Function use on Dictionaries

1. Whether item is in Dictionary or not:

To determine if a specified key is present in a dictionary use the `in` keyword:

```
In [23]: thisdict = {
           "brand": "Ford",
           "model": "Mustang",
           "year": 1964
         }
         if "model" in thisdict:
             print("Yes, 'model' is one of the keys in the thisdict dictionary")

Yes, 'model' is one of the keys in the thisdict dictionary
```

2. Len()

```
In [25]: thisdict = {
           "brand": "Ford",
           "model": "Mustang",
           "year": 1964
         }
         len(thisdict)

Out[25]: 3
```