lists

January 14, 2021

1 Module 0 - List, Tuples and Dictionaries

A PDF version of this notebook is available at Module 0 - List, Tuples and Dictionaries

1.1 Tuples

Tuples are an **unchangeable** container type. They contain a collection of objects. The tuple is a sequence type. They are used extensively in ML because they use considerable less memory than other data objects

```
[ ]: ('a', 10, True)
```

```
[ ]: ('a', 10, True)
```

There is no need for parenthesis for a tuple

```
[ ]: a = ('a', 10, True)
     b = 'b', 20, False
```

```
[ ]: type(a)
```

```
[ ]: tuple
```

```
[ ]: type(b)
```

```
[ ]: tuple
```

Since tuples are sequence types, we can access items by index:

```
[ ]: a = 'a', 10, True
```

```
[ ]: a[0] # note that index starts at 0
```

```
[ ]: 'a'
```

```
[ ]: a[2]
```

```
[ ]: True
[ ]: ## we can slice a tuple with :
     ## The index with slice stops at the item before the number
     ## a slice of tuple is also a tuple
     a = 1, 2, 3, 4, 5
     a[2:4]
```

```
[ ]: (3, 4)
```

Tuples are iterable objects

```
[ ]: a = 1, 2, 3, 4, 5
     for element in a:
         print(element)
```

```
1
2
3
4
5
```

Tuples are immutable. Objects within a tuple cannot be changed.

```
[ ]: a
```

```
[ ]: (1, 2, 3, 4, 5)
```

```
[ ]: a[1] = 6 ## an exception (error) will result when trying to change a tuple
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-12-9c4c3b2df379> in <module>()
----> 1 a[1] = 6 ## an exception (error) will result when trying to change a␣
 ↪tuple

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

You can change a different object to a tuple with the `tuple()` constructor.

```
[ ]: ## this is a list
     a = [1,2,3]
     type(a)
```

```
[ ]: list
[ ]: ## change to a tuple
     a = tuple(a)
     type(a)
```

```
[ ]: tuple
```

1.2 Lists

Lists are used to store multiple items in a single variable. They are **mutable** or changeable objects. Lists are also iterable. Lists are created using square brackets.

```
[ ]: myfirstlist = [3,45,40,732]
```

```
[ ]: type(myfirstlist)
```

```
[ ]: list
```

Lists are iterable, and index also starts at 0

```
[ ]: myfirstlist[0]
```

```
[ ]: 3
```

```
[ ]: myfirstlist[0:2]
```

```
[ ]: [3, 45]
```

Lists can also contain multiple object types

```
[ ]: mylist1 = ['thanks', 'isa630', 630, (20,21), [20, 21]]
```

```
[ ]: type(mylist1)
```

```
[ ]: list
```

```
[ ]: mylist1[3]
```

```
[ ]: (20, 21)
```

```
[ ]: mylist1[4]
```

```
[ ]: [20, 21]
```

```
[ ]: ## an object within a list can be of different type
     type(mylist1[3])
```

```
[ ]: tuple
```

You can change a different object to a list with the `list()` constructor.

```
[ ]: a = (1,2,3)
     type(a)
```

```
[ ]: tuple
```

```
[ ]: a = list(a)
     type(a)
```

```
[ ]: list
```

```
[ ]: print(a)
```

```
[1, 2, 3]
```

```
[ ]: list(range(10))
```

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

Because lists are mutable, they individual objects can be changed

```
[2]: a = [1,2,3,4]
     print(a)
```

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

```
[3]: a[0] = 5
     print(a)
```

```
[5, 2, 3, 4]
```

1.3 Dictionaries

A Dictionary in Python is an unordered and changeable collection of data values that holds key-value pairs. A dictionary is created with a curly bracket.

```
[ ]: ## Dictionary with name and age
     Dict = {'Tim': 18,'Charlie':12,'Tiffany':22,'Robert':25}
```

```
[4]: ## Multiple lines are ok in Python
     Dict = {'Tim': 18,
             'Charlie':12,
             'Tiffany':22,
             'Robert':25}
```

We can extract an item directly with the key

```
[5]: Dict['Tim']
```

[5]:
```
18
```

We can update a dictionary with the `update` method.

[ ]:
```
Dict.update({'Tim': 20})
Dict
```

[ ]:
```
{'Charlie': 12, 'Robert': 25, 'Tiffany': 22, 'Tim': 20}
```

We can extract the keys

[ ]:
```
Dict.keys()
```

[ ]:
```
dict_keys(['Tim', 'Charlie', 'Tiffany', 'Robert'])
```

We can extract the values

[ ]:
```
Dict.values()
```

[ ]:
```
dict_values([20, 12, 22, 25])
```