Python

April 30, 2018

1 Learning Objectives

This notebook will give a brief introduction to the following:

- Python programming and getting familiar with some of its built-in containers

1.1 Built-in Containers

Container - An object that can hold and iterate over other objects. The most important built-in Python containers are lists, dictionaries, tuples and sets.

The containers we will cover are:

- Lists
- Dictionaries
- Tuples
- Sets

1.1.1 1. Lists

- Python's equivalent for an array.

Creation of a List Object

Guess, who's names are these? - Yours!

** Instructions **

- Create a list separated by comma's

```
In [113]: names=['John Smith', 'Adam Dom','Tim Cook', 'Sundar Pichai', 'Elon Musk', 'Sebastian T

In [114]: '''
          Print the items in the list called 'names'
          '''
          for name in names:
              print(name)
```

```
John Smith
Adam Dom
Tim Cook
```

```
Sundar Pichai
Elon Musk
Sebastian Thrun
Steve Jobs
Bill Gates
```

** Instructions **

- Get the length of your list using len()

```
In [115]: len(names)
```

```
Out[115]: 8
```

** Instructions **

- Using .index(''), get the index of the third item in your list

```
In [116]: names.index('Elon Musk')
```

```
Out[116]: 4
```

** Instructions **

- Save the indexes of your list to a variable named 'indexes'

```
In [117]: indexes = range(len(names))
```

```
In [118]: '''
          Print the indexes
          '''
          print(list(indexes))
```

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

** Bonus **

- Lists can store different types of variables like strings and integers

```
In [119]: '''
          Example of list with values of different data types
          '''
          for i in [3,'A',3.5]:
              print(type(i))
```

```
<class 'int'>
<class 'str'>
<class 'float'>
```

1.1.2 2. Dictionaries

- A dictionary is an unordered set of key: value pairs, with the requirement that the keys are unique (within one dictionary).
- A pair of braces creates an empty dictionary: {}.
- Placing a comma-separated list of key:value pairs within the braces adds initial key:value pairs to the dictionary; this is also the way dictionaries are written on output.

Creation of a Dictionary Object
This is how a mapping is created between identity numbers and names

```
In [120]: '''
          Create a manual mapping between the index and value for the items in the list 'names'
          '''
          for index, name in zip(indexes, sorted(names)):
              print(index, name)
```

```
0 Adam Dom
1 Bill Gates
2 Elon Musk
3 John Smith
4 Sebastian Thrun
5 Steve Jobs
6 Sundar Pichai
7 Tim Cook
```

```
In [121]: '''
          Create a dict() object and call it 'dictionary'
          '''
          dictionary = dict()
          for index, name in zip(indexes, sorted(names)):
              dictionary[index]=name
```

```
In [122]: '''
          Print the values in the dictionary
          '''
          dictionary
```

```
Out[122]: {0: 'Adam Dom',
           1: 'Bill Gates',
           2: 'Elon Musk',
           3: 'John Smith',
           4: 'Sebastian Thrun',
           5: 'Steve Jobs',
           6: 'Sundar Pichai',
           7: 'Tim Cook'}
```

The value for a specific key can be extracted as follows

```
In [123]: dictionary[0]

Out[123]: 'Adam Dom'

In [124]: dictionary[6]

Out[124]: 'Sundar Pichai'
```

1.1.3 3. Tuples

- A sequence of immutable Python objects
- Tuples are always enclosed in parentheses, so that nested tuples are interpreted correctly; they may be input with or without surrounding parentheses, although often parentheses are necessary anyway (if the tuple is part of a larger expression). It is not possible to assign to the individual items of a tuple, however it is possible to create tuples which contain mutable objects, such as lists.
- Though tuples may seem similar to lists, they are often used in different situations and for different purposes. Tuples are immutable, and usually contain a heterogeneous sequence of elements that are accessed via unpacking (see later in this section) or indexing (or even by attribute in the case of namedtuples). Lists are mutable, and their elements are usually homogeneous and are accessed by iterating over the list.
- A special problem is the construction of tuples containing 0 or 1 items: the syntax has some extra quirks to accommodate these. Empty tuples are constructed by an empty pair of parentheses; a tuple with one item is constructed by following a value with a comma (it is not sufficient to enclose a single value in parentheses).

We can print only a slice of the list

```
In [125]: '''
          Print the first five elements from the list 'names'
          '''
          for name in names[:5]:
              print(name)

John Smith
Adam Dom
Tim Cook
Sundar Pichai
Elon Musk


In [126]: '''
          Using replace(), replace the white space between the first and last name with a '-'
          '''
          for name in names[:5]:
              print(name.replace(' ', '-'))

John-Smith
Adam-Dom
```

```
Tim-Cook
Sundar-Pichai
Elon-Musk
```

```
In [127]: '''
          Using split(), seperate the first name and the last name, Also, using replace(), get r
          '''
          for name in names[:5]:
              print(name.replace('-', ' ').split(' '))
```

```
['John', 'Smith']
['Adam', 'Dom']
['Tim', 'Cook']
['Sundar', 'Pichai']
['Elon', 'Musk']
```

```
In [128]: '''
          The first and last names can be accessed from the list by selecting the appropriate el
          '''
          for name in names[:5]:
              full_name = name.replace('.', ' ').split(' ')
              # First name
              print(full_name[0])
              # Last name
              print(full_name[-1])
```

```
John
Smith
Adam
Dom
Tim
Cook
Sundar
Pichai
Elon
Musk
```

Creation of a Tuple Object

```
In [129]: '''
          Create a list of tuples called 'first_last_names', with each tuple containing the firs
          '''
          first_last_names = []
          for name in sorted(names):
              full_name = name.replace('.', ' ').split(' ')
              first_last_names.append((full_name[0], full_name[-1]))
```

```
In [130]: '''
          Print the values in the tuple first_last_names
          '''
          first_last_names
```

```
Out[130]: [('Adam', 'Dom'),
           ('Bill', 'Gates'),
           ('Elon', 'Musk'),
           ('John', 'Smith'),
           ('Sebastian', 'Thrun'),
           ('Steve', 'Jobs'),
           ('Sundar', 'Pichai'),
           ('Tim', 'Cook')]
```

```
In [131]: '''
          Print the first element in the tuple 'first_last_names'
          '''
          first_last_names[0]
```

```
Out[131]: ('Adam', 'Dom')
```

```
In [132]: '''
          Print the first element of the first tuple object
          '''
          first_last_names[0][0]
```

```
Out[132]: 'Adam'
```

```
In [133]: '''
          Print the second element of the second tuple object
          '''
          first_last_names[1][1]
```

```
Out[133]: 'Gates'
```

1.2 4. Sets

- An unordered collection of distinct elements

We will use sets to identify unique last names

```
In [134]: '''
          Create a list called 'last_names' and store the last names from the tuple 'first_last_
          '''
          last_names=[]
          for name in first_last_names:
              last_names.append(name[1])
```

```
In [135]: '''
          Print the values of the 'last_names' list
          '''
          last_names
```

```
Out[135]: ['Dom', 'Gates', 'Musk', 'Smith', 'Thrun', 'Jobs', 'Pichai', 'Cook']
```

NOTE: The above `for` loop can also be condensed into a single statement using list comprehensions.

```
In [136]: '''
          Repeat the above step using list comprehension and store the last names in 'last_names
          '''
          last_names_lc = [name[1] for name in first_last_names]

In [38]: '''
         Print the values of 'last_names_lc'
         '''
         last_names_lc

Out[38]: ['Dom', 'Musk', 'Musk', 'Smith', 'Thrun', 'Jobs', 'Pichai', 'Cook']
```

Creation of a Set Object

```
In [39]: '''
         Create a set called 'unique_last_names' object using () and store the last_names values
         '''
         unique_last_names = set(last_names)

In [40]: '''
         Print the values of the set named 'unique_last_names'
         '''
         unique_last_names

Out[40]: {'Cook', 'Dom', 'Jobs', 'Musk', 'Pichai', 'Smith', 'Thrun'}

In [137]: print('No. of records - {}\nNo. of unique last names - {}'.format(len(last_names), len

No. of records - 8
No. of unique last names - 7
```

** Creation of a Function **

```
In [138]: def print_names(li):
              for name in li:
                  if name[1]=='Thrun':
                      print(name)

In [139]: print_names(first_last_names)

('Sebastian', 'Thrun')
```