

Data Science

Chapter 2: Python basics

Chapter outline

This chapter covers

- Benefits of Python in Data Science
- Jupyter Notebook
- Python basics – **to be done by the students**
- Python: import, lists, tuples, dictionaries

Why python?

- Easy-to-read and learn
- Vibrant community
- Growing and evolving set of libraries
 - Data management
 - Analytical processing
 - Visualization
- Applicable to each step in the data science process
- Notebooks

What to look forward to !

Jupyter notebooks:

- Interactive Coding and Visualization of Output

NumPy, SciPy, pandas:

- Numerical computation

Matplotlib, Seaborn:

- Data visualization

Scikit-Learn:

- Machine learning

Jupyter notebook

- HTML & Markdown
- LaTeX (equations)
- Code
- Code is divided into cells to control execution
- Enables interactive development

The screenshot displays a Jupyter notebook titled "Python basics" with a last checkpoint of "09/24/2020 (autosaved)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, cell navigation, and execution. The notebook content is divided into sections:

- Variables**: A text block explaining variable naming rules:
 - You can name a variable anything as long as it obeys the following three rules:
 - 1. It can be only one word.
 - 2. It can use only letters, numbers, and the underscore (`_`) character.
 - 3. It can't begin with a number.
 - 4. Variable name starting with an underscore (`_`) are considered as "unuseful".
- Code Cell 1**: A code cell with the following code:

```
In [2]: 1 a=10
        2 print(a)
        3
```

The output of this cell is the number `10`.
- String Concatenation**: A text block with a code cell below it. The code cell contains:

```
In [3]: 1 a='aa' 'bb' #concatenation
        2 print ('it\'s a concatenation:',a)
```

The output of this cell is the string `it's a concatenation: aabb`.

Green arrows from the list on the left point to these elements: "HTML & Markdown" points to the "Variables" section; "LaTeX (equations)" points to the "Variables" section; "Code" points to the first code cell; "Code is divided into cells to control execution" points to the second code cell; and "Enables interactive development" points to the "String Concatenation" section.

Special instructions!

- **%matplotlib inline** - display plots inline in Jupyter notebook
- **%%timeit** - time how long a cell takes to execute
- **%run filename.ipynb** - execute code from another notebook or python file

Python basics

- Course + direct applications:
 - python Notebook : [Python basics.ipynb](#)
 - Or HTML: [Python basics.html](#)
- Crash course :
 - [Python basics pdfs](#)

To be done by the students

Import Libraries

```
1 from random import * #you can use directly all the functions
2 for i in range(5):
3     print(randint(1, 10)) # from 1 to 10
```

7
2
2
2
2

```
1 import random as rand
2 for i in range(5):
3     print(rand.randint(1, 10)) # from 1 to 10
```

6
3
5
4
1

Main libraries for data science:

- Numpy – work with arrays
- Matplotlib – plot charts
- Pandas – create matrix of features and the dependent variable vector

Lists

- A list stores a series of items in a particular order. Lists allow you to store sets of information in one place, whether you have just a few items or millions of items.

- **Define a list**

```
1 #define a list
2 animals = ['cat', 'bat', 'rat', 'elephant', 'rat']
3 another_example = [x**2 for x in range(1, 11)] # from 1 to 10
4 print (another_example)
5 print (type(animals))
6 print (len(animals)) #len of the list
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
<class 'list'>
```

```
5
```

- **Access elements**

```
1 #access elements and indexes
2 print (animals[0])
3 print (animals[3])
4 print (animals[-2])
5 print (animals.index('rat'))
```

```
cat
elephant
elephant
2
```

Lists

➤ Add element

```
1 #add element in a list
2 animals.append('tiger')
3 print(animals)
4
5 #insert in specific index
6 animals.insert(2,'lion')
7 print(animals)
```

```
['cat', 'bat', 'rat', 'elephant', 'rat', 'tiger']
['cat', 'bat', 'lion', 'rat', 'elephant', 'rat', 'tiger']
```

➤ Remove element

```
1 #Removing elements from Lists by index
2 del animals[2]
3 print(animals)
4
5 #Removing elements from Lists by value
6 animals.remove('elephant')
7 print(animals)
8 #note: Python removes only the first item that has that value.
```

```
['cat', 'bat', 'rat', 'elephant', 'rat', 'tiger']
['cat', 'bat', 'rat', 'rat', 'tiger']
```

➤ Concatenate & Replicate

```
1 #List Concatenation and List Replication
2 concat= [1, 2, 3] + ['A', 'B', 'C']
3 repl= concat*3
4
5 print (concat)
6 print(repl)
```

```
[1, 2, 3, 'A', 'B', 'C']
[1, 2, 3, 'A', 'B', 'C', 1, 2, 3, 'A', 'B', 'C', 1, 2, 3, 'A', 'B', 'C']
```

Lists

➤ Looping through a list

```
1  #looping the elements of the list
2  for x in animals:
3      print (x)
4  print()
5  for i in range(len(animals)):
6      print (animals[i])
7  print()
8  #checking if an element exists in the list/
9  if 'bat' in animals:
10     print ('OK')
```

```
cat
bat
rat
rat
tiger
```

```
cat
bat
rat
rat
tiger
```

```
OK
```

Lists

➤ Slicing a list

You can work with any set of elements from a list. A portion of a list is called a slice. To slice a list start with the index of the first item you want, then add a colon and the index after the last item you want. Leave off the first index to start at the beginning of the list, and leave off the last index to slice through the end of the list.

```
1  #slicing
2  print(animals)
3
4  a_slice1 = animals[:2] #the elements at the index <2
5  a_slice2 = animals[2:] #the elements starting from index 2
6  a_slice3 = animals[1:8]
7
8  a_slice2[0]=100
9  print(a_slice1)
10 print(a_slice2)
11 print(a_slice3)
```

```
['cat', 'bat', 'rat', 'rat', 'tiger']
```

```
['cat', 'bat']
```

```
[100, 'rat', 'tiger']
```

```
['bat', 'rat', 'rat', 'tiger']
```

Lists

➤ Append vs Extend

```
2  #append: Appends object at end
3  x=[1,2,3]
4  y=[4,5,6]
5  x.append(y)
6  print(x)
```

[1, 2, 3, [4, 5, 6]]

```
12 #extend: Extends list by appending elements from the iterable.
13 x=[1,2,3]
14 y=[4,5,6]
15 x.extend(y)
16 print(x)
```

[1, 2, 3, 4, 5, 6]

Lists

➤ Slice vs equal

```
2  #using slice:
3  print (animals)
4  copy_animals=animals[:] #a copy
5  copy_animals[2]='DOG1'
6  print (animals)
```

```
['cat', 'bat', 'rat', 'rat', 'tiger']
['cat', 'bat', 'rat', 'rat', 'tiger']
```

```
8  #using equal:
9  print (animals)
10 equal_animals=animals #same reference
11 equal_animals[3]='DOG2'
12 print (animals)
```

```
['cat', 'bat', 'rat', 'rat', 'tiger']
['cat', 'bat', 'rat', 'DOG2', 'tiger']
```

Tuples

- Tuples are similar to lists, but the items in a tuple can't be modified.

```
1  #create a tuple
2  animals_t = ('cat', 'bat', 'rat', 'elephant')
3  #animals[2]='DOG'
4  print(animals_t[2])
5
6  #slicing
7  animals_copy=animals_t[:2]
8  print(type(animals_copy),animals_copy)
9
10 #length
11 print (len(animals_t))
12
13 #looping the elements
14 for x in animals_t:
15     print (x)
16
17 for i in range(len(animals_t)):
18     print (animals_t[i])
```

```
rat
<class 'tuple'> ('cat', 'bat')
4
cat
bat
rat
elephant
cat
bat
rat
elephant
```

Dictionaries

➤ Define / add elements

```
1 alien = {'color': 'green', 'points': 5} # a dictionary
2 print("The alien's color is "+ alien['color'])
3 print(alien)
4
5 #add an element.
6 alien['x_position'] = 0
7 print(alien)
8
9 #keys are unique.
10 #If you add a new item with existing key, the old value will be replaced
11 alien['x_position'] = 10
12 print(alien)
13
14 #length
15 print (len(alien))
```

```
The alien's color is green
{'color': 'green', 'points': 5}
{'color': 'green', 'points': 5, 'x_position': 0}
{'color': 'green', 'points': 5, 'x_position': 10}
3
```

Dictionaries

➤ Define / add elements

```
1 alien = {'color': 'green', 'points': 5} # a dictionary
2 print("The alien's color is " + alien['color'])
3 print(alien)
4
5 #add an element.
6 alien['x_position'] = 0
7 print(alien)
8
9 #keys are unique.
10 #If you add a new item with existing key, the old value will be replaced
11 alien['x_position'] = 10
12 print(alien)
13
14 #length
15 print (len(alien))
```

```
The alien's color is green
{'color': 'green', 'points': 5}
{'color': 'green', 'points': 5, 'x_position': 0}
{'color': 'green', 'points': 5, 'x_position': 10}
3
```

Sets

- Mathematically a set is a collection of items not in any particular order. A Python set is similar to this mathematical definition with below additional conditions.
- The elements in the set cannot be duplicates.
- The elements in the set are immutable(cannot be modified) but the set as a whole is mutable.
- There is no index attached to any element in a python set. So they do not support any indexing or slicing operation.