



## **Python Tips for Data Scientist**

**Wenqiang Feng and Jing Yang**

**December 23, 2021**



# CONTENTS

<b>1</b>	<b>Preface</b>	<b>3</b>
1.1	About . . . . .	3
1.1.1	About this note . . . . .	3
1.1.2	About the authors . . . . .	4
1.2	Motivation for this note . . . . .	4
1.3	Feedback and suggestions . . . . .	5
<b>2</b>	<b>Python Environment</b>	<b>7</b>
2.1	Use python on local computer . . . . .	7
2.2	Use python on the cloud (zero setup) . . . . .	7
2.2.1	Google Colab (FREE) . . . . .	9
2.2.2	Kaggle Kernels (FREE) . . . . .	10
2.2.3	Databricks Community Edition (FREE) . . . . .	10
<b>3</b>	<b>Notebooks</b>	<b>13</b>
3.1	Nteract . . . . .	13
3.2	Jupyter Notebook Viewer . . . . .	13
3.3	Apache Zeppelin . . . . .	16
3.4	Jupyter Notebook . . . . .	16
<b>4</b>	<b>Confidential Information</b>	<b>19</b>
<b>5</b>	<b>Primer Functions</b>	<b>21</b>
5.1	* . . . . .	21
5.2	range . . . . .	21
5.3	random . . . . .	22
5.3.1	random.random . . . . .	22
5.3.2	np.random . . . . .	22
5.4	round . . . . .	23
5.5	TODO.. . . . .	23
<b>6</b>	<b>Data Structures</b>	<b>25</b>
6.1	List . . . . .	25

6.1.1	Create list . . . . .	25
6.1.2	Unpack list . . . . .	26
6.1.3	Methods of list objects . . . . .	26
6.1.4	list.append(x) vs. list.extend(iterable) . . . . .	27
6.2	Tuple . . . . .	27
6.3	Dictionary . . . . .	27
6.3.1	Create dict from lists . . . . .	28
6.3.2	dict.get() . . . . .	28
6.3.3	Looping Techniques . . . . .	29
6.3.4	Update Values in Dict . . . . .	29
6.3.5	Update Keys in Dict . . . . .	30
6.4	One line if-else statement . . . . .	30
6.4.1	With filter . . . . .	30
6.4.2	Without filter . . . . .	30
<b>7</b>	<b>Data Read and Ingestion with DataBase</b>	<b>33</b>
7.1	Data Ingestion from Local to DataBase . . . . .	33
7.2	Data Read from DataBase to Local . . . . .	34
7.3	Connect to various DataBases ( <i>pyodbc</i> ) . . . . .	35
7.4	Hive and Impala Table Ingestion . . . . .	35
<b>8</b>	<b>Working with AWS S3</b>	<b>37</b>
8.1	Credentials . . . . .	37
8.2	File Upload to S3 . . . . .	37
8.2.1	s3_file_upload Function . . . . .	38
8.2.2	s3_content_type Function . . . . .	38
8.2.3	Examples . . . . .	50
8.3	File Download from S3 . . . . .	51
8.3.1	s3_file_download Function . . . . .	51
8.3.2	Examples . . . . .	52
8.4	File Management in S3 . . . . .	52
8.4.1	s3_fs Function . . . . .	52
8.4.2	Examples . . . . .	52
<b>9</b>	<b>pd.DataFrame vs PySpark DataFrame</b>	<b>53</b>
9.1	Create DataFrame . . . . .	53
9.1.1	From List . . . . .	53
9.1.2	From Dict . . . . .	54
9.2	Convert between pandas and pyspark DataFrame . . . . .	54
9.2.1	From pandas to pyspark DataFrame . . . . .	54
9.2.2	From pyspark to pandas DataFrame . . . . .	55
9.3	Load DataFrame . . . . .	55
9.3.1	From DataBase . . . . .	55
9.3.2	From .csv . . . . .	56
9.3.3	From .json . . . . .	56

9.4	First n Rows . . . . .	57
9.5	Column Names . . . . .	58
9.6	Data types . . . . .	58
9.7	Replace Data types . . . . .	58
9.8	Fill Null . . . . .	59
9.9	Replace Values . . . . .	60
9.10	Rename Columns . . . . .	61
	9.10.1 Rename all columns . . . . .	61
	9.10.2 Rename one or more columns . . . . .	61
9.11	Drop Columns . . . . .	62
9.12	Filter . . . . .	63
9.13	With New Column . . . . .	64
9.14	Join . . . . .	67
	9.14.1 Left Join . . . . .	67
	9.14.2 Right Join . . . . .	68
	9.14.3 Inner Join . . . . .	68
	9.14.4 Full Join . . . . .	69
9.15	Concat Columns . . . . .	70
9.16	GroupBy . . . . .	71
9.17	Pivot . . . . .	71
9.18	Unixtime to Date . . . . .	72
<b>10</b>	<b>pd.DataFrame manipulation</b>	<b>73</b>
10.1	TODO.. . . . .	73
<b>11</b>	<b>rdd.DataFrame manipulation</b>	<b>75</b>
11.1	TODO.. . . . .	75
<b>12</b>	<b>Online Courses and Useful Websites</b>	<b>77</b>
12.1	Recommended online courses . . . . .	77
12.2	Recommended Online Resources . . . . .	78
<b>13</b>	<b>Package Wrapper</b>	<b>79</b>
13.1	Hierarchical Structure . . . . .	79
13.2	Set Up . . . . .	80
13.3	Requirements . . . . .	81
13.4	ReadMe . . . . .	81
<b>14</b>	<b>Publish Package to PyPI</b>	<b>83</b>
14.1	Register PyPI account . . . . .	83
14.2	Install twine . . . . .	83
14.3	Build Your Package . . . . .	83
14.4	Upload Your Package . . . . .	84
14.5	Package at PyPI . . . . .	84
<b>15</b>	<b>Model Deployment with Flask</b>	<b>85</b>

15.1	Install flask . . . . .	85
15.2	Train and Save your model . . . . .	85
15.3	Deployment with Flask . . . . .	88
15.4	Lunch your app on server . . . . .	90
15.4.1	1. Lunch the APP . . . . .	90
15.4.2	2. Run the APP . . . . .	90
<b>16</b>	<b>API Book</b>	<b>91</b>
16.1	Basics Module . . . . .	91
16.1.1	rnorm . . . . .	91
16.1.2	dnorm . . . . .	92
16.1.3	runif . . . . .	92
16.2	Tests Module . . . . .	92
16.2.1	T-test . . . . .	92
<b>17</b>	<b>Main Reference</b>	<b>95</b>
	<b>Bibliography</b>	<b>97</b>
	<b>Python Module Index</b>	<b>99</b>
	<b>Index</b>	<b>101</b>



Welcome to my **Python Tips for Data Scientist** notes! In those notes, you will learn some useful tips for Data Scientist daily work. The PDF version can be downloaded from [HERE](#).





## PREFACE

---

### Chinese proverb

The palest ink is better than the best memory. – old Chinese proverb

---

## 1.1 About

### 1.1.1 About this note

This document is a summary of our valueable experiences in using Python for Data Scientist daily work. The PDF version can be downloaded from [HERE](#).

**You may download and distribute it. Please be aware, however, that the note contains typos as well as inaccurate or incorrect description.**

In this repository, we try to use the detailed Data Scientist related demo code and examples to share some useful python tips for Data Scientist work. If you find your work wasn't cited in this note, please feel free to let me know.

Although we are by no means a python programming and Data Scientist expert, We decided that it would be useful for us to share what we learned about Python in the form of easy note with detailed example. We hope those notes will be a valuable tool for your studies.

The notes assume that the reader has a preliminary knowledge of python programing, LaTeX and Linux. And this document is generated automatically by using [sphinx](#). More details can be found at [[Georg2018](#)].

### 1.1.2 About the authors

- **Wenqiang Feng**
  - Director of Data Science and PhD in Mathematics
  - University of Tennessee, Knoxville
  - Webpage: <http://web.utk.edu/~wfeng1>
  - Email: [von198@gmail.com](mailto:von198@gmail.com)
- **Jing Yang**
  - Senior Data Scientist and PhD in Physics
  - Harvard University
  - Webpage: <https://scholar.harvard.edu/jingyang/home>
  - Email: [jingyangharvard@gmail.com](mailto:jingyangharvard@gmail.com)

- **Declaration**

The work of Wenqiang Feng was supported by the IMA, while working at IMA. However, any opinion, finding, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the IMA, UTK, DST and Harvard.

## 1.2 Motivation for this note

No matter you like it or not, Python has been one of the most popular programming languages. I have been using Python for almost 4 years. Frankly speaking, I wasn't impressed and attracted by Python at the first using. After starting working in industry, I have to use Python. Gradually I recognize the elegance of Python and use it as one of my main programming language. But I found that:

- Most of the Python books or tutorials which emphasize on programming will overwhelm the green hand.
- While most of the Python books or tutorials for Data Scientist or Data Analysis didn't cover some essential skills from the engineer side.

So I want to keep some of my valuable tips which are heavily applied in my daily work.

## 1.3 Feedback and suggestions

Your comments and suggestions are highly appreciated. I am more than happy to receive corrections, suggestions or feedbacks through email (Wenqiang Feng: [von198@gmail.com](mailto:von198@gmail.com), Jing Yang: [jingyangharvard@gmail.com](mailto:jingyangharvard@gmail.com) ) for improvements.



## **PYTHON ENVIRONMENT**

---

**Note:** This Chapter *Python Environment* is for beginner. If you have some Python programming experience, you may skip this chapter. For beginners, you can choose the hard-core way, installing and setting up python in your own computer. Alternatively, there is easy route to leverage free online data science environment that requires zero setup.

---

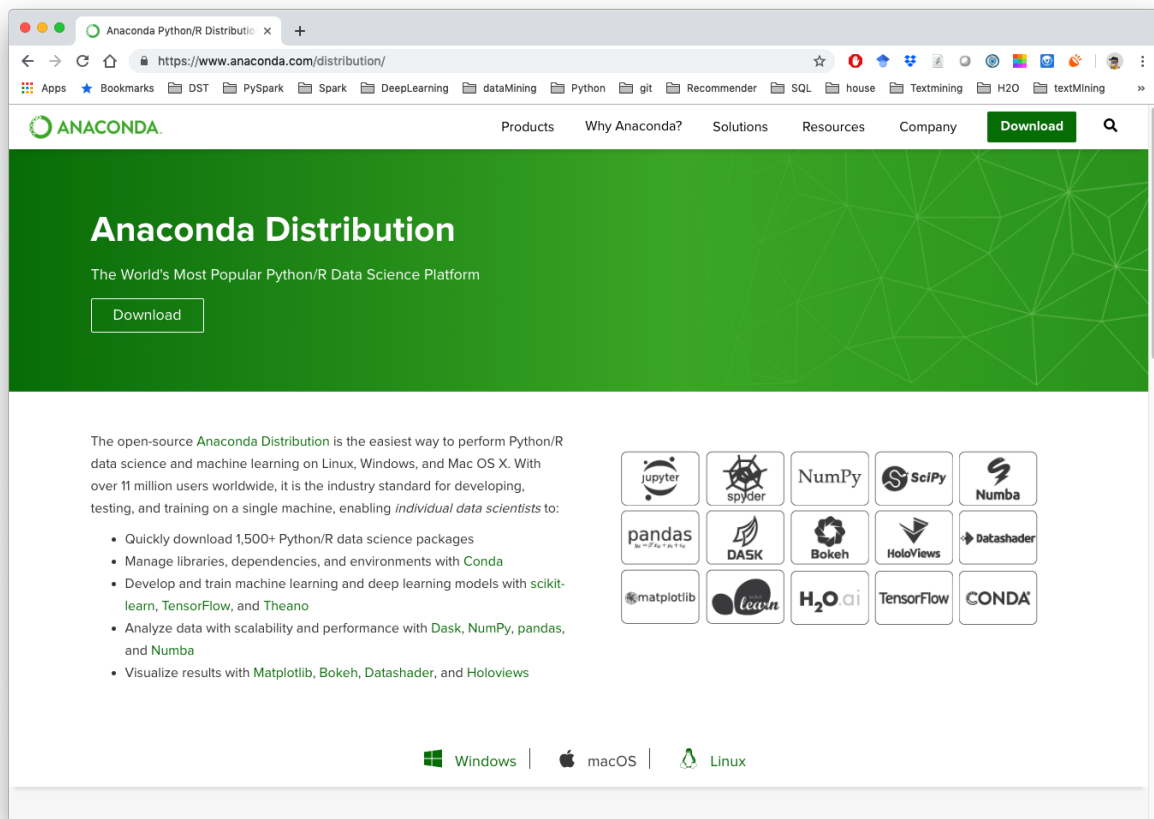
### **2.1 Use python on local computer**

No matter what operator system is, I will strongly recommend you to install Anaconda which contains Python, Jupyter, spyder, Numpy, Scipy, Numba, pandas, DASK, Bokeh, HoloViews, Datashader, matplotlib, scikit-learn, H2O.ai, TensorFlow, CONDA and more.

Download link: <https://www.anaconda.com/distribution/>

### **2.2 Use python on the cloud (zero setup)**

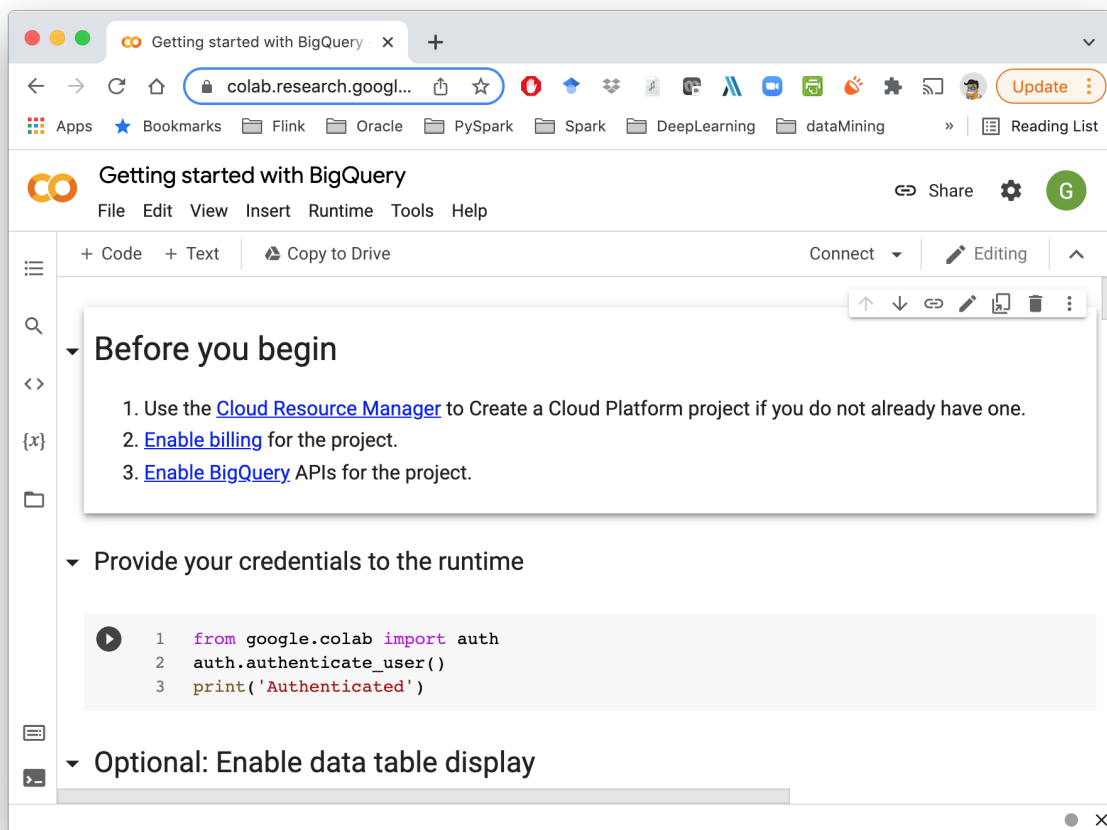
Learning a new programming language is not easy. Luckily, nowadays there are many web-based data science environment available that allows one to learn python without downloading or installing python in local laptop, you can do almost anything online, including free access to GPUs!



## 2.2.1 Google Colab (FREE)

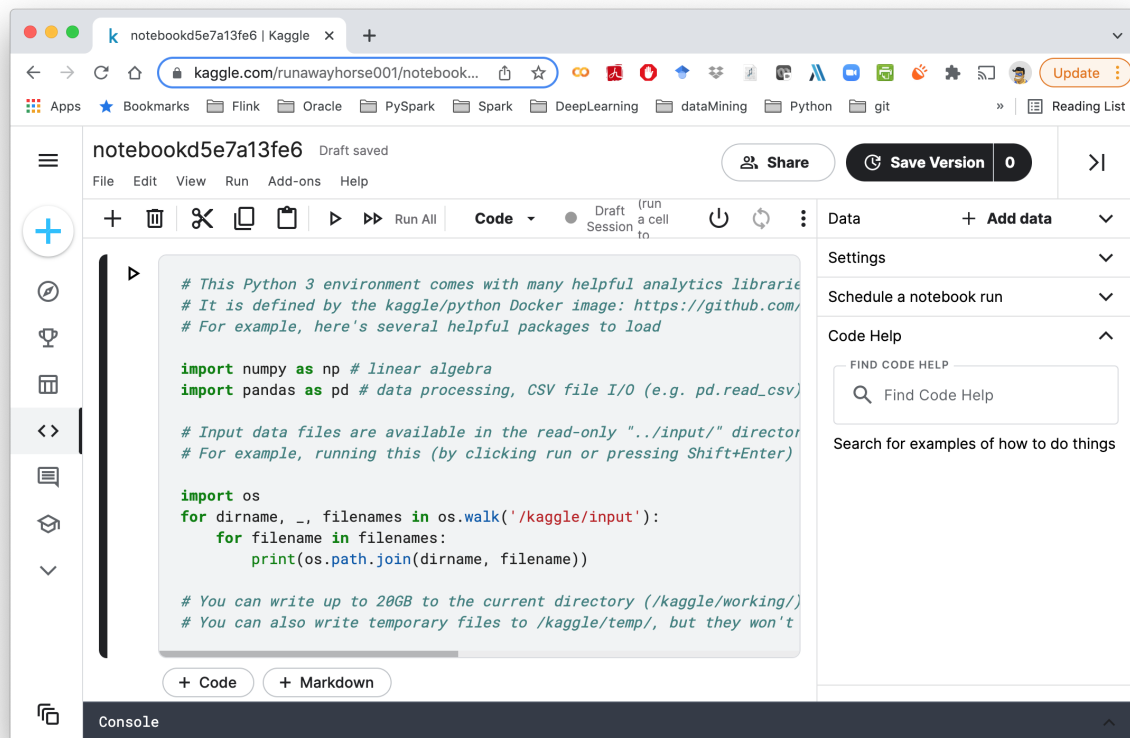
Quote from the official introduction on Google, Colaboratory, or “Colab” for short, is a free Jupyter notebook environment that requires no setup, and runs entirely (writing, running, & sharing code) within Google Drive.

- Zero configuration/setup required on your own machine!
- Free access to GPUs and TPUs: code execute on Google’s cloud servers
- Search and use built-in code snippets
- Easy sharing (like Google doc)



### 2.2.2 Kaggle Kernels (FREE)

Kaggle is best known as a platform for data science competitions. They also provide a free service called Kernels that can be used independently of their competitions.



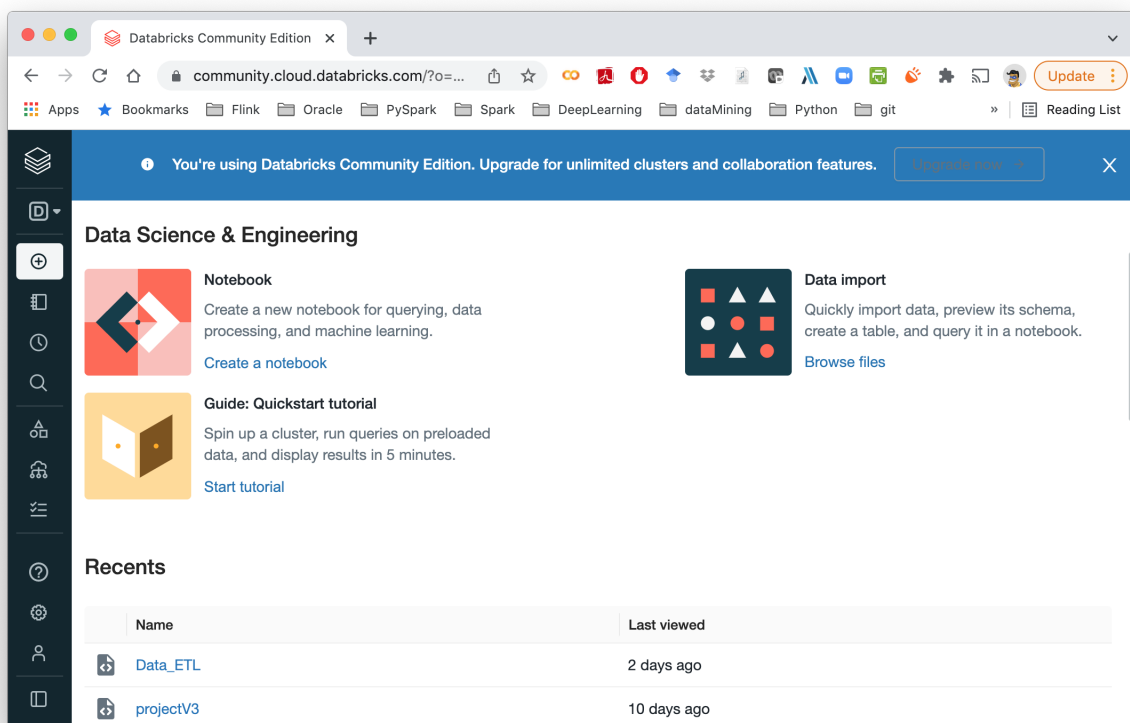
There are a few more choices to run the Jupyter Notebook in the cloud. Feel free to checkout this article below: <https://www.dataschool.io/cloud-services-for-jupyter-notebook/> This blog is posted in March 2019, content maybe a little out of date. It does offer great in-depth comparison of the different platforms.

### 2.2.3 Databricks Community Edition (FREE)

The Databricks Community Edition is the free version of Databricks cloud-based big data platform. The users can access a micro-cluster as well as a cluster manager and notebook environment. All users can share their notebooks and host them free of charge with Databricks.

The Databricks Community Edition also comes with a rich portfolio of award-winning training resources that will be expanded over time, making it ideal for developers, data scientists, data engineers and other IT professionals to learn Apache Spark. More details can be found at: <https://community.cloud.databricks.com>







## NOTEBOOKS

---

**Note:** This Chapter *Notebooks* is for beginner. If you have already know Nteract, Zeppelin and Python, you may skip this chapter.

---

If you are a Data Scientist, it's not enough to just know Jupyter Notebook. You should also take a look at nbviewer, Nteract and Zeppelin notebooks.

### 3.1 Nteract

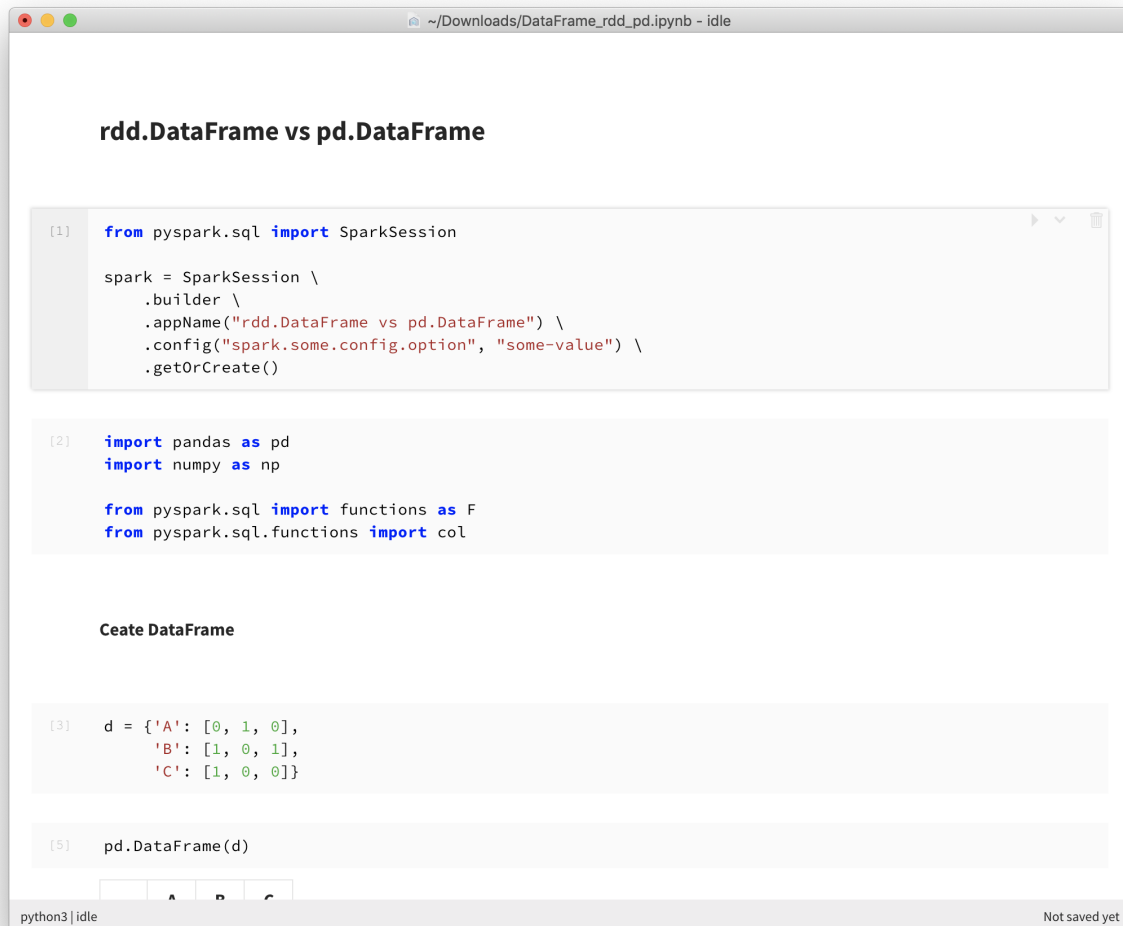
Nteract is an amazing .ipynb reader. You can open and run the .ipynb by just double clicking the .ipynb file.

Download from: <https://nteract.io/>

### 3.2 Jupyter Notebook Viewer

If you are a MAC user, you can also install the Jupyter Notebook Viewer `nbviewer-app` which is much faster than Nteract.

Download from: <https://github.com/tuxu/nbviewer-app>



The screenshot shows a Jupyter Notebook window with the title bar indicating the file path is `~/Downloads/DataFrame_rdd_pd.ipynb` and it is in 'idle' state. The notebook content is as follows:

### rdd.DataFrame vs pd.DataFrame

```
[1] from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .appName("rdd.DataFrame vs pd.DataFrame") \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()
```

```
[2] import pandas as pd
import numpy as np

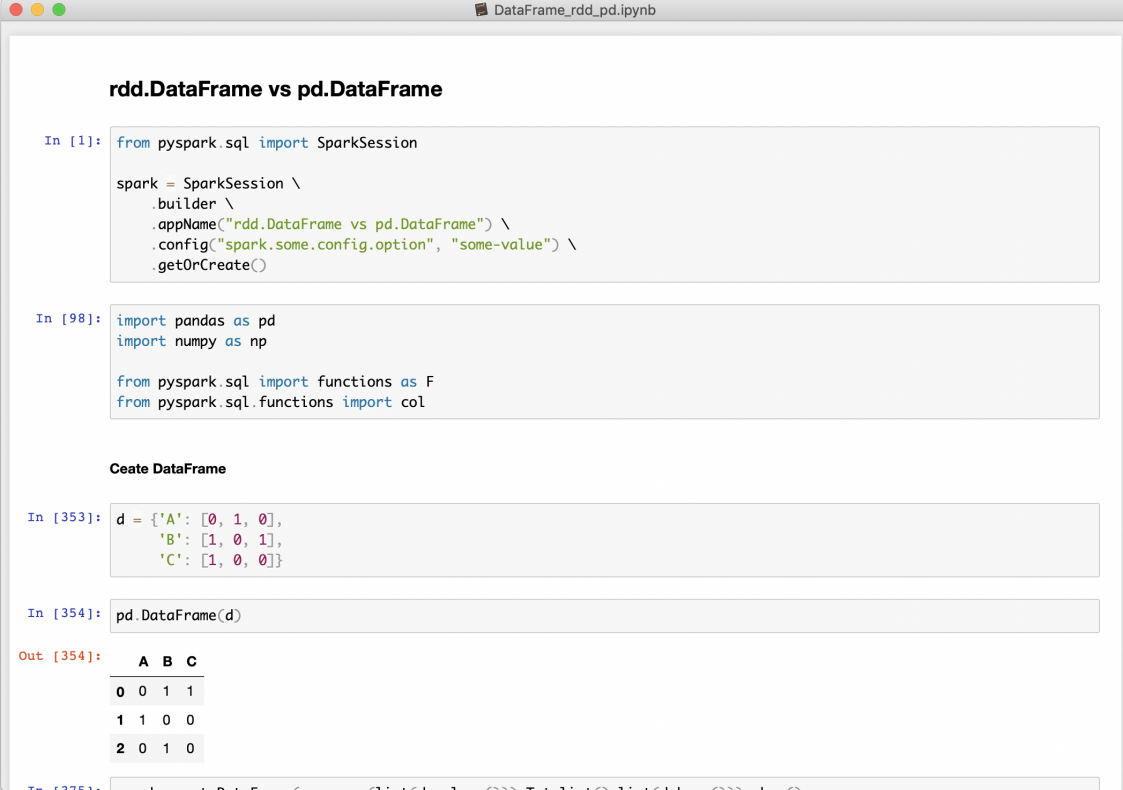
from pyspark.sql import functions as F
from pyspark.sql.functions import col
```

### Ceate DataFrame

```
[3] d = {'A': [0, 1, 0],
      'B': [1, 0, 1],
      'C': [1, 0, 0]}
```

```
[5] pd.DataFrame(d)
```

At the bottom of the notebook, there is a table with three columns labeled A, B, and C. The status bar at the very bottom shows 'python3 | idle' on the left and 'Not saved yet' on the right.



**rdd.DataFrame vs pd.DataFrame**

```
In [1]: from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .appName("rdd.DataFrame vs pd.DataFrame") \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()
```

```
In [98]: import pandas as pd
import numpy as np

from pyspark.sql import functions as F
from pyspark.sql.functions import col
```

**Create DataFrame**

```
In [353]: d = {'A': [0, 1, 0],
              'B': [1, 0, 1],
              'C': [1, 0, 0]}
```

```
In [354]: pd.DataFrame(d)
```

```
Out [354]:
```

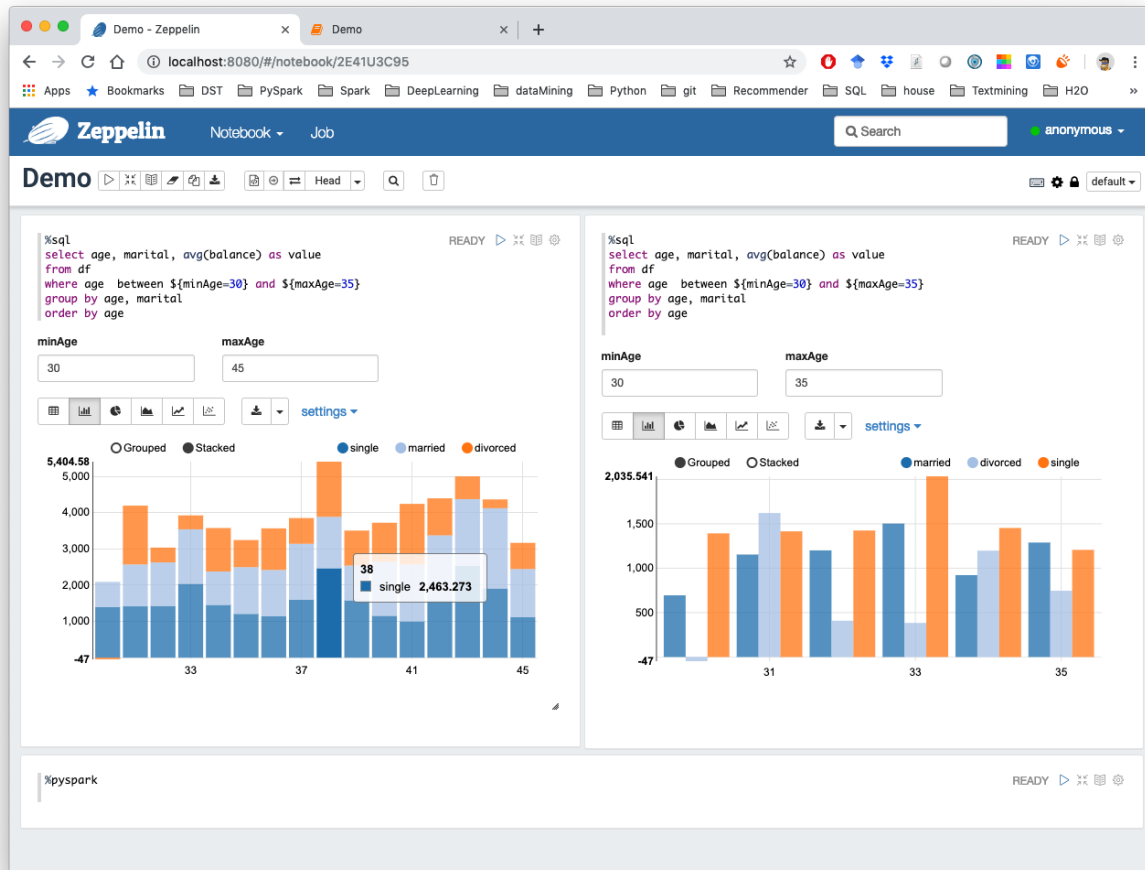
	A	B	C
0	0	1	1
1	1	0	0
2	0	1	0

```
In [375]: spark.createDataFrame(sc.parallelize(list(d.values())), F.toListCol(list(d.keys()))).show()
```

### 3.3 Apache Zeppelin

The Zeppelin (Apache Zeppelin) is an open-source Web-based notebook that enables data-driven, interactive data analytics and collaborative documents with Python, PySpark, SQL, Scala and more.

Download from: <https://zeppelin.apache.org/>



### 3.4 Jupyter Notebook

The Jupyter Notebook (Ipython Notebook) is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

The screenshot shows a Jupyter Notebook titled "Demo" running on a local host. The notebook contains several code cells for setting up a Spark session and comparing RDD and Pandas DataFrames.

**Code Cell 1:**

```
In [1]: from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .appName("rdd.DataFrame vs pd.DataFrame") \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()
```

**Code Cell 2:**

```
In [98]: import pandas as pd
import numpy as np

from pyspark.sql import functions as F
from pyspark.sql.functions import col
```

**Code Cell 3:**

```
Create DataFrame

In [353]: d = {'A': [0, 1, 0],
              'B': [1, 0, 1],
              'C': [1, 0, 0]}
```

**Code Cell 4:**

```
In [354]: pd.DataFrame(d)
```

**Output of Cell 4:**

```
Out[354]:
```

	A	B	C
0	0	1	1
1	1	0	0
2	0	1	0

**Code Cell 5:**

```
In [375]: spark.createDataFrame(np.array(list(d.values())).T.tolist(), list(d.keys())).show()
```

**Output of Cell 5:**

```
+---+---+---+
| A | B | C |
+---+---+---+
```





## CONFIDENTIAL INFORMATION

---

### Chinese proverb

Be mindful of guarding against harm from others, and stay away from placing harming upon others.

---

If you are a real Data Scientist, you have to share your code with your colleagues or release your code for Code Review or Quality assurance(QA). You will definitely do not want to have your User Information in the code. So you can save them in login.txt in a safe folder:

```
runawayhorse001  
PythonTips
```

and use the following code to import your User Information:

```
#User Information  
try:  
    login = pd.read_csv(r'login.txt', header=None)  
    user = login[0][0]  
    pw = login[0][1]  
    print('User information is ready!')  
except:  
    print('Login information is not available!!!')
```

You may also want to get the User Information by using `os.environ` in Python:

```
try:  
    user = os.environ['LOGNAME']  
except OSError:  
    user = os.environ['USER']  
except OSError:  
    user = os.environ['USERNAME']  
    print(err)  
except OSError as err:  
    print('The user information is not available!!!')
```



## PRIMER FUNCTIONS

---

**Note:** This Chapter *Primer Functions* is for beginner. If you have some Python programming experience, you may skip this chapter.

---

The following functions have been heavily used in my daily Data Scientist work.

### 5.1 \*

Single asterisk as used in function declaration allows variable number of arguments passed from calling environment. Inside the function it behaves as a tuple.

:: Python Code:

```
my_list = [1,2,3]
print(my_list)
print(*my_list)
```

:: Ouput:

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

### 5.2 range

:: Python Code:

```
print(range(5))
print(*range(5))
print(*range(3,8))
```

:: Ouput:

```
range(0, 5)
0 1 2 3 4
3 4 5 6 7
```

## 5.3 random

More details can be found at:

- random: <https://docs.python.org/3/library/random.html#random.randint>
- np.random: <https://docs.scipy.org/doc/numpy/reference/routines.random.html>

### 5.3.1 random.random

:: Python Code:

```
import random
random.random()

# (b - a) * random() + a
random.uniform(3, 8)
```

:: Ouput:

```
0.33844051243073625
7.772024014335885
```

### 5.3.2 np.random

:: Python Code:

```
np.random.random_sample()
np.random.random_sample(4)
np.random.random_sample([2, 4])

# (b - a) * random_sample() + a
a = 3; b = 8
(b-a)*np.random.random_sample([2, 4])+a
```

:: Ouput:

```
0.11919402208670005
array([0.07384755, 0.9005251 , 0.30030561, 0.38221819])
array([[0.76851156, 0.56973309, 0.47074505, 0.7814957 ],
       [0.5778028 , 0.94653057, 0.51193493, 0.48693931]])

array([[4.65799262, 6.32702018, 6.55545234, 5.45877784],
       [7.69941994, 4.68709357, 5.49790728, 4.60913966]])
```

## 5.4 round

Sometimes, we really do not need the scientific decimals for output results. So you can use this function to round an array to the given number of decimals.

:: Python Code:

```
np.round(np.random.random_sample([2,4]),2)
```

:: Ouput:

```
array([[0.76, 0.06, 0.41, 0.4 ],
       [0.07, 0.51, 0.84, 0.76]])
```

## 5.5 TODO..

:: Python Code:

:: Ouput:

:: Python Code:

:: Ouput:

:: Python Code:

:: Output:

:: Python Code:

:: Output:

## DATA STRUCTURES

---

**Note:** This Chapter *Data Structures* is for beginner. If you have some Python programming experience, you may skip this chapter.

---

### 6.1 List

`list` is one of data structures which is heavily using in my daily work.

#### 6.1.1 Create list

##### 1. Create empty list

The empty list is used to initialize a list.

:: Python Code:

```
my_list = []  
type(my_list)
```

:: Output:

```
list
```

I applied the empty list to initialize my `silhouette_score` list when I try to find the optimal number of the clusters.

:: Example:

```
min_cluster = 3  
max_cluster = 8
```

(continues on next page)

(continued from previous page)

```
# silhouette_score
scores = []

for i in range(min_cluster, max_cluster):
    score = np.round(np.random.random_sample(), 2)
    scores.append(score)

print(scores)
```

:: Output:

```
[0.16, 0.2, 0.3, 0.87, 0.59]
```

### 6.1.2 Unpack list

:: Example:

```
num = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(*num)
```

:: Output:

```
1 2 3 4 5 6 7 8 9 10
```

### 6.1.3 Methods of list objects

Methods of list objects:

Name	Description
<code>list.append(x)</code>	Add an item to the end of the list
<code>list.extend(iterable)</code>	Extend the list by appending all
<code>list.insert(i, x)</code>	Insert an item at a given position
<code>list.remove(x)</code>	Remove the first item
<code>list.pop([i])</code>	Remove the item at given position
<code>list.clear()</code>	Remove all items from the list
<code>list.index(x[, s[, e]])</code>	Return zero-based index in the list
<code>list.count(x)</code>	Return the number of times x
<code>list.sort(key, reverse)</code>	Sort the items of the list
<code>list.reverse()</code>	Reverse the elements of the list
<code>list.copy()</code>	Return a shallow copy <sup>1</sup> of list

---

<sup>1</sup> Shallow Copy vs Deep Copy Reference: <https://stackoverflow.com/posts/184780/revisions>



### 6.1.4 list.append(x) vs. list.extend(iterable)

The difference of `list.append(x)` vs. `list.extend(iterable)` is easy to understand from the example below:

:: Example:

```
list1 = ['A', 'B', 'C']
list2 = ['D', 'E', 'F']
list1.append(list2)
print(list1)
```

:: Output:

```
['A', 'B', 'C', ['D', 'E', 'F']]
```

:: Example:

```
list1 = ['A', 'B', 'C']
list2 = ['D', 'E', 'F']
list1.extend(list2)
print(list1)
```

:: Output:

```
['A', 'B', 'C', 'D', 'E', 'F']
```

## 6.2 Tuple

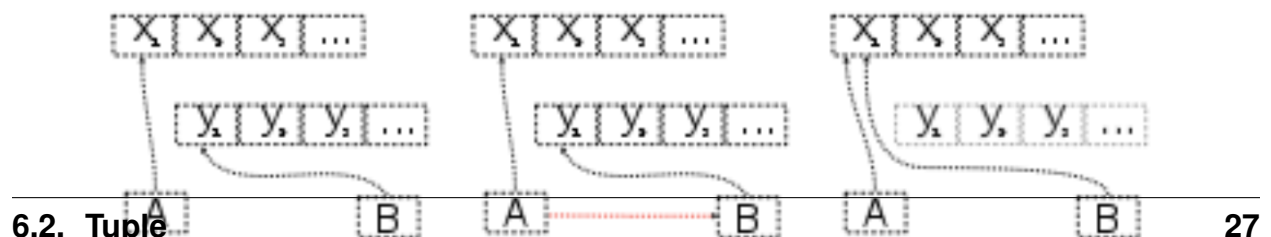
A tuple is an assortment of data, separated by commas, which makes it similar to the Python list, but a tuple is fundamentally different in that a tuple is “immutable.” This means that it cannot be changed, modified, or manipulated.

## 6.3 Dictionary

`dict` is one of another data structures which is heavily using in my daily work. I heavily applied the `dict` in my `PyAudit` package, more details can be found at [PyAudit](#).

---

Shallow copy:



The variables A and B refer to different areas of memory, when B is assigned to A the two variables refer to the same memory. In this case, the memory is shared between the two variables. This is a shallow copy.

### 6.3.1 Create dict from lists

:: Example:

```
col_names = ['name', 'Age', 'Sex', 'Car']
col_values = ['Michael', '30', 'Male', ['Honda', 'Tesla']]
#
d = {key: value for key, value in zip(col_names, col_values)}
print(d)
#
import pandas as pd

df = pd.DataFrame(d)
print(df)
```

:: Output:

```
{'name': 'Michael', 'Age': '30', 'Sex': 'Male', 'Car': ['Honda
→', 'Tesla']}
```

	name	Age	Sex	Car
0	Michael	30	Male	Honda
1	Michael	30	Male	Tesla

### 6.3.2 dict.get()

When `get()` is called, Python checks if the specified key exists in the dict. If it does, then `get()` returns the value of that key. If the key does not exist, then `get()` returns the value specified in the second argument to `get()`. A good application of `get()` can be found at [Update Keys in Dict](#).

:: Example:

```
data1 = d.get("name", "best")
data2 = d.get("names", "George")
print(data1)    # Michael
print(data2)    # George
```

:: Output:

```
Michael
George
```

### 6.3.3 Looping Techniques

:: Example:

```
print([(key, val) for key, val in d.items()])
```

:: Output:

```
[('name', 'Michael'), ('Age', '30'), ('Sex', 'Male'), ('Car', 'Honda'), ('Car', 'Tesla')]
```

### 6.3.4 Update Values in Dict

#### 1. Replace values in dict

:: Example:

```
replace = {'Car': ['Tesla S', 'Tesla X']}
print(d)
d.update(replace)
print(d)
```

:: Output:

```
{'name': 'Michael', 'Age': '30', 'Sex': 'Male', 'Car': ['Honda', 'Tesla']}
{'name': 'Michael', 'Age': '30', 'Sex': 'Male', 'Car': ['Tesla S', 'Tesla X']}
```

#### 2. Add key and values in dict

:: Example:

```
# add key and values in dict
added = {'Kid': ['Tom', 'Jim']}
print(d)
d.update(added)
print(d)
```

:: Output:

```
{'name': 'Michael', 'Age': '30', 'Sex': 'Male', 'Car': ['Tesla S', 'Tesla X']}
{'name': 'Michael', 'Age': '30', 'Sex': 'Male', 'Car': ['Tesla S', 'Tesla X'], 'Kid': ['Tom', 'Jim']}
```

### 6.3.5 Update Keys in Dict

:: Example:

```
# update keys in dict
mapping = {'Car': 'Cars', 'Kid': 'Kids'}
#
print({mapping.get(key, key): val for key, val in d.items()})
```

:: Output:

```
{'name': 'Michael', 'Age': '30', 'Sex': 'Male', 'Car': [
→ 'Tesla S', 'Tesla X'], 'Kid': ['Tom', 'Jim']}
{'name': 'Michael', 'Age': '30', 'Sex': 'Male', 'Cars': [
→ 'Tesla S', 'Tesla X'], 'Kids': ['Tom', 'Jim']}
```

## 6.4 One line if-else statement

### 6.4.1 With filter

::syntax:

```
[ RESULT for x in seq if COND ]
```

:: Python Code:

```
num = [1,2,3,4,5,6,7,8,9,10]
[x for x in num if x%2 ==0]
```

:: Output:

```
[2, 4, 6, 8, 10]
```

### 6.4.2 Without filter

::syntax:

```
[ RESULT1 if COND1 else RESULT2 if COND2 else RESULT3 for x_
→ in seq]
```

:: Python Code:

```
num = [1,2,3,4,5,6,7,8,9,10]

['Low' if 1<= x <=3 else 'Median' if 3<x<8 else 'High' for x_
→in num]
```

:: Output:

```
['Low',
 'Low',
 'Low',
 'Median',
 'Median',
 'Median',
 'Median',
 'High',
 'High',
 'High']
```

[VanderPlas2016] [McKinney2013]



## DATA READ AND INGESTION WITH DATABASE

### 7.1 Data Ingestion from Local to DataBase

```
# User Information
try:
    login = pd.read_csv(r'login.txt', header=None)
    user = login[0][0]
    pw = login[0][1]
    print('User information is ready!')
except:
    print('Login information is not available!!!')

# Database information
host = '##.###.###.##'
db_name = 'db_name'
table_name = 'table_name'

# Setup connection
conn = psycopg2.connect(host=host, database=db_name, user=user,
    ↪password=pw)
cur = conn.cursor()

# Creat table in DataBase
conn.commit()
query = """
    DROP TABLE IF EXISTS {table_name};
    CREATE TABLE {table_name}
    (
        id character varying(20)
        , val1 double precision
        , val2 double precision
        , val3 double precision
        , val4 text
    )
    DISTRIBUTED BY (id);
```

(continues on next page)

(continued from previous page)

```
GRANT SELECT ON TABLE {table_name} TO xxxx;
""".format(table_name=table_name)
cur.execute(query)
conn.commit()

# load the data
df = pd.read_csv('xx.csv')

# Write dataframe to memory as csv
csv_io = io.StringIO()
df.to_csv(csv_io, sep='\t', header=True, index=False)
csv_io.seek(0)

# Copy the dataframe in memory to GP
conn.commit()
copy_sql = """
    COPY {table_name} FROM stdin WITH CSV HEADER
    DELIMITER as '\t'
    """.format(table_name=table_name)
cur.copy_expert(sql=copy_sql, file=csv_io)
conn.commit()
```

---

**Note:** You can also use `copy_to` to copy the dataframe from local memory to GP

```
cur.copy_to(df, table_name)
```

---

## 7.2 Data Read from DataBase to Local

```
# User information
try:
    login = pd.read_csv(r'login.txt', header=None)
    user = login[0][0]
    pw = login[0][1]
    print('User information is ready!')
except:
    print('Login information is not available!!!')

# Database information
host = '##.###.###.##'
db_name = 'db_name'
```

(continues on next page)



(continued from previous page)

```

table_name = 'table_name'

# Setup connection
conn = psycopg2.connect(host=host, database=db_name, user=user,
    ↳ password=pw)
cur = conn.cursor()

# Read table
sql = """
    select *
    from {table_name}
    """.format(table_name=table_name)
dp = pd.read_sql(sql, conn)

```

## 7.3 Connect to various DataBases (*pyodbc*)

One open source python library *pyodbc* makes accessing ODBC databases simple. For example, it can connect with Google BigQuery, Hive from Ubuntu / Debian, Microsoft Excel, Microsoft SQL Server etc.

```

# set up DSN (database source name) connection
import pyodbc
conn = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL Server};SERVER=
    ↳ {server};DATABASE={db};UID={user};PWD={password}')
query_string = "SQL QUERY"

import pandas as pd
df = pd.read_sql(query_string, conn)

```

## 7.4 Hive and Impala Table Ingestion

Data Lake Analy...

Overview

Account

Schema **New**

Execute **New**

Endpoint

Execute

Search schema name

"Double click" to switch

dataworks\_demo (current)

dia\_orders\_db

dia\_oss\_db

dia\_oss\_demo

dia

prod\_name

cnt

y

m

dia\_prod\_db

hangzhou\_ots\_test

hello\_mysql\_vpc\_rds

Syntax manuals

Function manuals

<<

Sync execute(F8)

Async execute(F9)

Format(F10)

Log on in DMS to execute SQL operations

1 SELECT \* from select \* from 'dia\_oss\_demo'. 'dia' limit 20

2

History

Result set

▼ Hide

Start time	SQL	Status
15:08:28	select infinity();	COMPLETE
2019-04-16 15:08:24	select infinity();	COMPLETE
2019-04-16 15:08:24	select tanh(8);	COMPLETE

## WORKING WITH AWS S3

Many companies chose to save sensitive data in AWS S3. So you may have to deal with data with python in Databricks, while python will have many issues to operate in Databricks(PySpark will not have problems if the environment was set up correctly in Databricks): such as the `os.system` command-like function can not use any more; no unified way to upload or download different type files, etc. Here, I will provide my way to work in AWS S3 with python in Databricks:

### 8.1 Credentials

I use Security Token Service (STS) to create the credentials to access AWS S3. STS enables you to request temporary, limited-privilege credentials for Identity and Access Management (IAM) users or for users that you authenticate (federated users). More details can be found at: <https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/sts.html>

```
import boto3

response = boto3.client('sts')\
    .assume_role(RoleArn='arn:aws:iam::123456789012:role/\
↪demo',
                RoleSessionName='your_role_session_name')
credentials = response['Credentials']
```

### 8.2 File Upload to S3

The main idea at here is to save the file in memory or a temporary file, then use `put_object` to put the file in S3. It's a little bit tricky to save the corresponding formatted file in memory, I will list several common types in the examples.

### 8.2.1 s3\_file\_upload Function

```
def s3_file_upload(data, path):

    file_type = path.split('/')[ -1 ].split('.')[ -1 ].lower()
    try:
        content_type = s3_content_type(file_type)
    except:
        print('Do not support the current input type!!!')

    s3_path = path.replace('s3://', '').replace('s3a://', '')
    bucket = s3_path.split('/')[ 0 ]
    key = s3_path.split('/', 1)[ 1 ]

    try:
        s3_resource.Bucket(bucket) \
            .put_object(Key=key,
                        Body=data,
                        ContentType=content_type,
                        ACL='public-read')
        print(f"{key.split('/')[ -1 ]} has been successfully saved in s3!
→")
    except Exception as err:
        print(err)
```

### 8.2.2 s3\_content\_type Function

```
def s3_content_type(file_type):
    mapping = {png:image/png}

    return mapping[file_type]
```

The full mapping list can be found as follows:

```
3dm:x-world/x-3dmf
3dmf:x-world/x-3dmf
a:application/octet-stream
aab:application/x-authorware-bin
aam:application/x-authorware-map
aas:application/x-authorware-seg
abc:text/vnd.abc
acgi:text/html
afl:video/animaflex
ai:application/postscript
```

(continues on next page)

(continued from previous page)

```
aif:audio/aiff
#aif:audio/x-aiff
aifc:audio/aiff
#aifc:audio/x-aiff
aiff:audio/aiff
#aiff:audio/x-aiff
aim:application/x-aim
aip:text/x-audio-software-intra
ani:application/x-navi-animation
aos:application/x-nokia-9000-communicator-add-on-software
aps:application/mime
arc:application/octet-stream
arj:application/arj
art:image/x-jg
asf:video/x-ms-asf
asm:text/x-asm
asp:text/asp
asx:application/x-mplayer2
#asx:video/x-ms-asf
#asx:video/x-ms-asf-plugin
au:audio/basic
#au:audio/x-au
#avi:video/avi
#avi:video/msvideo
avi:video/x-msvideo
avs:video/avs-video
bcpio:application/x-bcpio
#bin:application/mac-binary
#bin:application/macbinary
#bin:application/octet-stream
bin:application/x-binary
#bin:application/x-macbinary
bm:image/bmp
bmp:image/bmp
boo:application/book
book:application/book
boz:application/x-bzip2
bsh:application/x-bsh
bz:application/x-bzip
bz2:application/x-bzip2
c:text/plain
c++:text/plain
cat:application/vnd.ms-pki.seccat
cc:text/plain
ccad:application/clariscad
```

(continues on next page)

(continued from previous page)

```
cco:application/x-cocoa
cdf:application/cdf
cer:application/pkix-cert
cha:application/x-chat
chat:application/x-chat
class:application/java
com:application/octet-stream
conf:text/plain
cpio:application/x-cpio
cpp:text/x-c
cpt:application/mac-compactpro
crl:application/pkcs-crl
crt:application/pkix-cert
csh:application/x-csh
css:text/css
cxx:text/plain
dcr:application/x-director
deepv:application/x-deepv
def:text/plain
der:application/x-x509-ca-cert
dif:video/x-dv
dir:application/x-director
dl:video/dl
doc:application/msword
dot:application/msword
dp:application/commonground
drw:application/drafting
dump:application/octet-stream
dv:video/x-dv
dvi:application/x-dvi
dwf:model/vnd.dwf
dwg:application/acad
dxf:application/dxf
dxr:application/x-director
el:text/x-script.elisp
elc:application/x-bytecode.elisp
env:application/x-envoy
eps:application/postscript
es:application/x-esrehber
etx:text/x-setext
evy:application/envoy
exe:application/octet-stream
f:text/plain
f77:text/x-fortran
f90:text/plain
```

(continues on next page)

(continued from previous page)

```
fdf:application/vnd.fdf
fif:application/fractals
fli:video/fli
flo:image/florian
flx:text/vnd.fmi.flexstor
fmf:video/x-atomic3d-feature
for:text/plain
fpx:image/vnd.fpx
frl:application/freelader
funk:audio/make
g:text/plain
g3:image/g3fax
gif:image/gif
gl:video/gl
gsd:audio/x-gsm
gsm:audio/x-gsm
gsp:application/x-gsp
gss:application/x-gss
gtar:application/x-gtar
gz:application/x-gzip
gzip:application/x-gzip
h:text/plain
hdf:application/x-hdf
help:application/x-helpfile
hgl:application/vnd.hp-hpgl
hh:text/plain
hlp:application/hlp
hpg:application/vnd.hp-hpgl
hpgl:application/vnd.hp-hpgl
hqx:application/binhex
hta:application/hta
htc:text/x-component
htm:text/html
html:text/html
htmls:text/html
htt:text/webviewhtml
htx:text/html
ice:x-conference/x-cooltalk
ico:image/x-icon
idc:text/plain
ief:image/ief
iefs:image/ief
iges:application/iges
igs:application/iges
ima:application/x-ima
```

(continues on next page)

(continued from previous page)

```
imap:application/x-httpd-imap
inf:application/inf
ins:application/x-internett-signup
ip:application/x-ip2
isu:video/x-isvideo
it:audio/it
iv:application/x-inventor
ivr:i-world/i-vrml
ivy:application/x-livescreen
jam:audio/x-jam
java:text/plain
jcm:application/x-java-commerce
jfif:image/jpeg
jpeg:image/jpeg
jpg:image/jpeg
jps:image/x-jps
js:application/x-javascript
jut:image/jutvision
kar:audio/midi
ksh:text/x-script.ksh
la:audio/nsaudio
lam:audio/x-liveaudio
latex:application/x-latex
lha:application/octet-stream
lhx:application/octet-stream
list:text/plain
lma:audio/nsaudio
log:text/plain
lst:text/plain
lsx:text/x-la-asf
ltx:application/x-latex
lzh:application/octet-stream
lzx:application/octet-stream
m:text/plain
m1v:video/mpeg
m2a:audio/mpeg
m2v:video/mpeg
m3u:audio/x-mpequrl
m4v:video/x-m4v
man:application/x-troff-man
mht:message/rfc822
mhtml:message/rfc822
midi:audio/midi
mif:application/x-frame
mjf:audio/x-vnd.audioexplosion.mjuicemediafile
```

(continues on next page)



(continued from previous page)

```
mjpg:video/x-motion-jpeg
mod:audio/mod
mov:video/quicktime
movie:video/x-sgi-movie
mp2:audio/mpeg
mp3:audio/mpeg
#mpa:audio/mpeg
mpa:video/mpeg
mpc:application/x-project
mpeg:video/mpeg
mpg:video/mpeg
mpga:audio/mpeg
ogg:video/ogg
ogv:video/ogg
p:text/x-pascal
p10:application/pkcs10
#p12:application/pkcs-12
p12:application/x-pkcs12
p7a:application/x-pkcs7-signature
p7c:application/x-pkcs7-mime
p7m:application/pkcs7-mime
p7r:application/x-pkcs7-certreqresp
p7s:application/pkcs7-signature
part:application/pro_eng
pas:text/pascal
pbm:image/x-portable-bitmap
pcl:application/x-pcl
pct:image/x-pict
pcx:image/x-pcx
pdb:chemical/x-pdb
pdf:application/pdf
pfunk:audio/make
pgm:image/x-portable-graymap
pic:image/pict
pict:image/pict
pkg:application/x-newton-compatible-pkg
pko:application/vnd.ms-pki.pko
pl:text/plain
plx:application/x-pixclscript
pm:image/x-xpixmap
pm4:application/x-pagemaker
pm5:application/x-pagemaker
png:image/png
pnm:image/x-portable-anymap
pot:application/mspowerpoint
```

(continues on next page)

(continued from previous page)

```
ppa:application/vnd.ms-powerpoint
ppm:image/x-portable-pixmap
pps:application/mspowerpoint
ppt:application/mspowerpoint
#ppt:application/powerpoint
#ppt:application/vnd.ms-powerpoint
#ppt:application/x-mspowerpoint
ppz:application/mspowerpoint
pre:application/x-freelance
prt:application/pro_eng
ps:application/postscript
psd:application/octet-stream
pvu:paleovu/x-pv
pwz:application/vnd.ms-powerpoint
py:text/x-script.python
pyc:applicaiton/x-bytecode.python
qcp:audio/vnd.qcelp
qd3:x-world/x-3dmf
#qd3d:x-world/x-3dmf
qif:image/x-quicktime
qt:video/quicktime
qtc:video/x-qtc
qti:image/x-quicktime
qtif:image/x-quicktime
ra:audio/x-pn-realaudio
#ra:audio/x-pn-realaudio-plugin
#ra:audio/x-realaudio
ram:audio/x-pn-realaudio
ras:application/x-cmu-raster
#ras:image/cmu-raster
#ras:image/x-cmu-raster
#rast:image/cmu-raster
#rexx:text/x-script.rexx
#rf:image/vnd.rn-realflash
rgb:image/x-rgb
rm:application/vnd.rn-realmedia
#rm:audio/x-pn-realaudio
rmi:audio/mid
rmm:audio/x-pn-realaudio
rmp:audio/x-pn-realaudio
#rmp:audio/x-pn-realaudio-plugin
rng:application/ringing-tones
#rng:application/vnd.nokia.ringing-tone
rnx:application/vnd.rn-realplayer
roff:application/x-troff
```

(continues on next page)

(continued from previous page)

```
rp:image/vnd.rn-realpix
rpm:audio/x-pn-realaudio-plugin
rt:text/richtext
#rt:text/vnd.rn-realtex
rtf:application/rtf
#rtf:application/x-rtf
#rtf:text/richtext
#rtx:application/rtf
#rtx:text/richtext
rv:video/vnd.rn-realvideo
s:text/x-asm
s3m:audio/s3m
#saveme:application/octet-stream
sbk:application/x-tbook
scm:application/x-lotusscreencam
#scm:text/x-script.guile
#scm:text/x-script.scheme
#scm:video/x-scm
sdml:text/plain
sdp:application/sdp
#sdp:application/x-sdp
sdr:application/sounder
sea:application/sea
#sea:application/x-sea
set:application/set
sgm:text/sgml
#sgm:text/x-sgml
sgml:text/sgml
#sgml:text/x-sgml
sh:application/x-bsh
#sh:application/x-sh
#sh:application/x-shar
#sh:text/x-script.sh
shar:application/x-bsh
#shar:application/x-shar
shtml:text/html
#shtml:text/x-server-parsed-html
sid:audio/x-psid
#sit:application/x-sit
sit:application/x-stuffit
skd:application/x-koan
skm:application/x-koan
skp:application/x-koan
skt:application/x-koan
sl:application/x-seelogo
```

(continues on next page)

(continued from previous page)

```
smi:application/smil
smil:application/smil
#snd:audio/basic
snd:audio/x-adpcm
sol:application/solids
#spc:application/x-pkcs7-certificates
spc:text/x-speech
spl:application/futuresplash
spr:application/x-sprite
sprite:application/x-sprite
src:application/x-wais-source
ssi:text/x-server-parsed-html
ssm:application/streamingmedia
sst:application/vnd.ms-pki.certstore
step:application/step
stl:application/sla
#stl:application/vnd.ms-pki.stl
#stl:application/x-navistyle
stp:application/step
sv4cpio:application/x-sv4cpio
sv4crc:application/x-sv4crc
svf:image/vnd.dwg
#svf:image/x-dwg
svr:application/x-world
#svr:x-world/x-svr
swf:application/x-shockwave-flash
t:application/x-troff
talk:text/x-speech
tar:application/x-tar
tbk:application/toolbook
#tbk:application/x-tbook
tcl:application/x-tcl
#tcl:text/x-script.tcl
tcsh:text/x-script.tcsh
tex:application/x-tex
texi:application/x-texinfo
texinfo:application/x-texinfo
#text:application/plain
text:text/plain
#tgz:application/gnutar
tgz:application/x-compressed
tif:image/tiff
#tif:image/x-tiff
tiff:image/tiff
#tiff:image/x-tiff
```

(continues on next page)

(continued from previous page)

```
tr:application/x-troff
tsi:audio/tsp-audio
tsp:application/dsptype
#tsp:audio/tsplayer
tsv:text/tab-separated-values
turbot:image/florian
txt:text/plain
uil:text/x-uil
uni:text/uri-list
unis:text/uri-list
unv:application/i-deas
uri:text/uri-list
uris:text/uri-list
ustar:application/x-ustar
#ustar:multipart/x-ustar
uu:application/octet-stream
#uu:text/x-uuencode
uue:text/x-uuencode
vcd:application/x-cdlink
vcs:text/x-vcalendar
vda:application/vda
vdo:video/vdo
vew:application/groupwise
viv:video/vivo
#viv:video/vnd.vivo
vivo:video/vivo
#vivo:video/vnd.vivo
vmd:application/vocaltec-media-desc
vmf:application/vocaltec-media-file
voc:audio/voc
#voc:audio/x-voc
vos:video/vosaic
vox:audio/voxware
vqe:audio/x-twinvq-plugin
vqf:audio/x-twinvq
vql:audio/x-twinvq-plugin
vrml:application/x-vrml
#vrml:model/vrml
#vrml:x-world/x-vrml
vrt:x-world/x-vrt
vsd:application/x-visio
vst:application/x-visio
vsw:application/x-visio
w60:application/wordperfect6.0
w61:application/wordperfect6.1
```

(continues on next page)

(continued from previous page)

```
w6w:application/msword
wav:audio/wav
#wav:audio/x-wav
wbl:application/x-qpro
wbmp:image/vnd.wap.wbmp
web:application/vnd.xara
wiz:application/msword
wk1:application/x-123
wmf:windows/metafile
wml:text/vnd.wap.wml
wmlc:application/vnd.wap.wmlc
wmls:text/vnd.wap.wmlscript
wmlsc:application/vnd.wap.wmlscriptc
word:application/msword
wp:application/wordperfect
wp5:application/wordperfect
#wp5:application/wordperfect6.0
wp6:application/wordperfect
wpd:application/wordperfect
#wpd:application/x-wpwin
wq1:application/x-lotus
wri:application/mswrite
#wri:application/x-wri
#wrl:application/x-world
wrl:model/vrml
#wrl:x-world/x-vrml
#wrz:model/vrml
#wrz:x-world/x-vrml
#wsc:text/scriplet
wsr:application/x-wais-source
wtk:application/x-wintalk
#xbm:image/x-xbitmap
#xbm:image/x-xbm
xbm:image/xbm
xdr:video/x-amt-demorun
xgz:xgl/drawing
xif:image/vnd.xiff
xl:application/excel
xla:application/excel
#xla:application/x-excel
#xla:application/x-msexcel
#xlb:application/excel
#xlb:application/vnd.ms-excel
xlb:application/x-excel
#xlc:application/excel
```

(continues on next page)

(continued from previous page)

```
#xlc:application/vnd.ms-excel
#xlc:application/x-excel
xld:application/excel
#xld:application/x-excel
#xlk:application/excel
xlk:application/x-excel
#xll:application/excel
#xll:application/vnd.ms-excel
xll:application/x-excel
#xlm:application/excel
#xlm:application/vnd.ms-excel
xlm:application/x-excel
#xls:application/excel
#xls:application/vnd.ms-excel
#xls:application/x-excel
xls:application/x-msexcel
#xlt:application/excel
xlt:application/x-excel
#xlv:application/excel
xlv:application/x-excel
#xlw:application/excel
#xlw:application/vnd.ms-excel
#xlw:application/x-excel
xlw:application/x-msexcel
xm:audio/xm
#xml:application/xml
xml:text/xml
xmz:xgl/movie
xpix:application/x-vnd.ls-xpix
#xpm:image/x-xpixmap
xpm:image/xpm
x-png:image/png
xsr:video/x-amt-showrun
#xwd:image/x-xwd
xwd:image/x-xwindowdump
xyz:chemical/x-pdb
#z:application/x-compress
z:application/x-compressed
#zip:application/x-compressed
#zip:application/x-zip-compressed
zip:application/zip
#zip:multipart/x-zip
zoo:application/octet-stream
zsh:text/x-script.zsh
```

### 8.2.3 Examples

#### 1. .csv file

save csv file in memory:

```
csv_io = io.StringIO()
df.to_csv(csv_io, sep='\t', header=True, index=False)
csv_io.seek(0)
# the csv data need encode
csv_data = io.BytesIO(csv_io.getvalue().encode())
```

---

#### Note:

The alternative way by using tempfile:

```
with tempfile.TemporaryFile(mode='r+') as fp:
    df.to_csv(fp, sep='\t', header=True, index=False)
    fp.seek(0)
#
s3_file_upload(csv_data, file_path)
```

---

#### Upload file

```
>>> file_path = 'my_bucket/~/~/test/test.csv'
>>> s3_file_upload(csv_data, file_path)
test.csv has been successfully saved in S3!
```

#### 2. .json file

```
>>> json_object = """ your json content """
>>> json_data = json.dumps(json_object)
>>> file_path = 'my_bucket/~/~/test/test.json'
>>> s3_file_upload(json_data, file_path)
test.json has been successfully saved in S3!
```

---

#### Note:

The alternative way by using tempfile:

```
with tempfile.TemporaryFile() as fp:
    joblib.dump(json_data, fp)
    fp.seek(0)
#
s3_file_upload(json_data, file_path)
```



3. .png, .jpeg or .pdf

Save the image in memory:

```
flights = sns.load_dataset("flights")
may_flights = flights.query("month == 'May'")
fig = plt.figure(figsize=(20,8))
sns.lineplot(data=may_flights, x="year", y="passengers")

img_data = io.BytesIO()
plt.savefig(img_data, format='png')
img_data.seek(0)
```

Save the in-memory image data in S3:

```
path = 'my_bucket/my_key'
s3_file_save(img_data, path)
```

---

**Note:** The above method also works for .jpeg and .pdf format.

---

## 8.3 File Download from S3

The main idea is using the s3 resource function to download the file and save it at /temp as a temporary file, then use the corresponding formatted functions to read it.

### 8.3.1 s3\_file\_download Function

```
def s3_file_download(path):

    # extract bucket and key from the given path
    s3_path = path.replace('s3://', '').replace('s3a://', '')
    bucket = s3_path.split('/')[0]
    key = s3_path.split('/', 1)[1]

    # download file and save it as a temp file
    file_name = os.path.join('/tmp', path.split('/')[-1])
    s3_resource.Bucket(bucket).download_file(key, file_name)

    # return saved file path
    return file_name
```

### 8.3.2 Examples

```
>>> file_path = 'my_bucket/***/***/test/test.json'
>>> file_name = s3_file_download(file_path)
'/temp/test.json'
>>> joblib.load(filename)
```

## 8.4 File Management in S3

I mainly use my `s3_fs` to help me do the file management in S3.

### 8.4.1 `s3_fs` Function

The `s3_fs` is mainly based on `s3fs` package. The top-level class `s3fs` holds connection information and allows typical file-system style operations like `cp`, `mv`, `ls`, `walk`, `du`, `glob`, etc. More details can be found at: <https://s3fs.readthedocs.io/en/latest/index.html>

```
import s3fs

s3_fs = s3fs.S3FileSystem(anon=False,
                           key=credentials['AccessKeyId'],
                           secret=credentials['SecretAccessKey'],
                           token=credentials['SessionToken'])
```

### 8.4.2 Examples

Simple locate and read a file:

```
>>> s3_fs.ls('my-bucket')
['demo-file.csv']
>>> with fs.open('my-bucket/demo-file.csv', 'rb') as f:
...     print(f.read())
b'UserId\tdate\nuser_id1\t2019-05-02\nuser_id2\t2019-12-02\n'
```

## PD.DATAFRAME VS PYSPARK DATAFRAME

### 9.1 Create DataFrame

#### 9.1.1 From List

```
my_list = [['a', 1, 2], ['b', 2, 3], ['c', 3, 4]]
col_name = ['A', 'B', 'C']
```

:: Python Code:

```
# caution for the columns=
pd.DataFrame(my_list, columns= col_name)
#
spark.createDataFrame(my_list, col_name).show()
```

:: Comparison:

	A	B	C
0	a	1	2
1	b	2	3
2	c	3	4

**Attention:** Pay attention to the parameter `columns=` in `pd.DataFrame`. Since the default value will make the list as rows.

:: Python Code:

```
# caution for the columns=
pd.DataFrame(my_list, columns= col_name)
#
pd.DataFrame(my_list, col_name)
```

:: Comparison:

	A	B	C			0	1	2
0	a	1	2	A	a	1	2	
1	b	2	3	B	b	2	3	
2	c	3	4	C	c	3	4	

### 9.1.2 From Dict

```
data = {'A': [0, 1, 0],
        'B': [1, 0, 1],
        'C': [1, 0, 0]}
```

:: Python Code:

```
pd.DataFrame(data)
# Tedious for PySpark
spark.createDataFrame(np.array(list(d.values())).T.tolist(), list(d.
    ↪keys())).show()
```

:: Comparison:

	A	B	C					
0	0	1	1		0	1	1	
1	1	0	0		1	0	0	
2	0	1	0		0	1	0	

## 9.2 Convert between pandas and pyspark DataFrame

### 9.2.1 From pandas to pyspark DataFrame

:: Example:

```
# pd.DataFrame pandas_df: DataFrame pandas
# rdd.DataFrame. spark_df: DataFrame spark
pandas_df = pd.read_csv('Advertising.csv')
spark_df = spark.createDataFrame(pandas_df)
spark_df.printSchema()
```

## 9.2.2 From pyspark to pandas DataFrame

:: Example:

```
pandas_df = spark_df.toPandas()
pandas_df.info()
```

## 9.3 Load DataFrame

### 9.3.1 From DataBase

Most of time, you need to share your code with your colleagues or release your code for Code Review or Quality assurance(QA). You will definitely do not want to have your User Information in the code. So you can save them in login.txt:

```
runawayhorse001
PythonTips
```

and use the following code to import your User Information:

```
#User Information
try:
    login = pd.read_csv(r'login.txt', header=None)
    user = login[0][0]
    pw = login[0][1]
    print('User information is ready!')
except:
    print('Login information is not available!!!')

#Database information
host = '##.###.###.##'
db_name = 'db_name'
table_name = 'table_name'
```

:: Comparison:

```
conn = psycopg2.connect(host=host, database=db_name, user=user,
    ↪password=pw)
cur = conn.cursor()

sql = """
    select *
    from {table_name}
```

(continues on next page)

(continued from previous page)

```
"".format(table_name=table_name)
pandas_df = pd.read_sql(sql, conn)
```

```
# connect to database
url = 'jdbc:postgresql://'+host+':5432/'+db_name+'?user='+user+'&
    ↳password='+pw
properties={'driver': 'org.postgresql.Driver', 'password': pw, 'user':
    ↳user}
spark_df = spark.read.jdbc(url=url, table=table_name,
    ↳properties=properties)
```

**Attention:** Reading tables from Database with PySpark neespark\_df the proper drive for the corresponding Database. For example, the above demo neespark\_df org.postgresql.Driver and you need to download it and put it in jars folder of your spark installation path. I download postgresql-42.1.1.jar from the official website and put it in jars folder.

### 9.3.2 From .csv

:: Comparison:

```
# pd.DataFrame pandas_df: DataFrame pandas
pandas_df = pd.read_csv('Advertising.csv')
#rdd.DataFrame. spark_df: DataFrame spark
spark_df = spark.read.csv(path='Advertising.csv',
#                               sep=', ',
#                               encoding='UTF-8',
#                               comment=None,
                               header=True,
                               inferSchema=True)
```

### 9.3.3 From .json

Data from: <http://api.luftdaten.info/static/v1/data.json>

```
pandas_df = pd.read_json("data/data.json")
spark_df = spark.read.json('data/data.json')
```

:: Python Code:

```
pandas_df[['id', 'timestamp']].head(4)
#
spark_df[['id', 'timestamp']].show(4)
```

:: Comparison:

	id	timestamp
0	2994551481	2019-02-28 17:23:52
1	2994551482	2019-02-28 17:23:52
2	2994551483	2019-02-28 17:23:52
3	2994551484	2019-02-28 17:23:52

only showing top 4 rows

## 9.4 First n Rows

:: Python Code:

```
pandas_df.head(4)
#
spark_df.show(4)
```

:: Comparison:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5

only showing top 4 rows

### 9.5 Column Names

:: Python Code:

```
pandas_df.columns  
#  
spark_df.columns
```

:: Comparison:

```
Index(['TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')  
['TV', 'Radio', 'Newspaper', 'Sales']
```

### 9.6 Data types

:: Python Code:

```
pandas_df.dtypes  
#  
spark_df.dtypes
```

:: Comparison:

TV	float64	(('TV', 'double'),
Radio	float64	('Radio', 'double'),
Newspaper	float64	('Newspaper', 'double'),
Sales	float64	('Sales', 'double')]
dtype:	object	

### 9.7 Replace Data types

```
my_list = [('a', 2, 3),  
           ('b', 5, 6),  
           ('c', 8, 9),  
           ('a', 2, 3),  
           ('b', 5, 6),  
           ('c', 8, 9)]  
col_name = ['col1', 'col2', 'col3']  
  
pandas_df = pd.DataFrame(my_list, columns=col_name)
```

(continues on next page)



(continued from previous page)

```
spark_df = spark.createDataFrame(pandas_df)

pandas_df.dtypes
```

```
col1      object
col2      int64
col3      int64
dtype: object
```

:: Python Code:

```
d = {'col2': 'string', 'col3': 'string'}
pandas_df = pandas_df.astype({'col2': 'str', 'col3': 'str'})
spark_df = spark_df.select(*list(set(spark_df.columns)-set(d.keys()))),
                          *(col(c[0]).astype(c[1]).alias(c[0]) for c in d.
                          →items()))
```

:: Comparison:

```
col1      object
col2      object      [('col1', 'string'), ('col2', 'string'), (
→ 'col3', 'string')]
col3      object
dtype: object
```

## 9.8 Fill Null

```
my_list = [['a', 1, None], ['b', 2, 3], ['c', 3, 4]]
pandas_df = pd.DataFrame(my_list, columns=['A', 'B', 'C'])
spark_df = spark.createDataFrame(my_list, ['A', 'B', 'C'])
#
pandas_df.head()
spark_df.show()
```

:: Comparison:

	A	B	C
0	male	1	NaN
1	female	2	3.0
2	male	3	4.0

	A	B	C
0	male	1	null
1	female	2	3
2	male	3	4

:: Python Code:

```
pandas_df.fillna(-99)
#
spark_df.fillna(-99).show()
```

:: Comparison:

	A	B	C
0	male	1	-99
1	female	2	3.0
2	male	3	4.0

	A	B	C
0	male	1	-99
1	female	2	3
2	male	3	4

## 9.9 Replace Values

:: Python Code:

```
# caution: you need to chose specific col
pandas_df.A.replace(['male', 'female'],[1, 0], inplace=True)
pandas_df
#caution: Mixed type replacements are not supported
spark_df.na.replace(['male','female'],['1','0']).show()
```

:: Comparison:

	A	B	C
0	1	1	NaN
1	0	2	3.0
2	1	3	4.0

	A	B	C
0	1	1	null
1	0	2	3
2	1	3	4

## 9.10 Rename Columns

### 9.10.1 Rename all columns

:: Python Code:

```
pandas_df.columns = ['a', 'b', 'c', 'd']
pandas_df.head(4)
#
spark_df.toDF('a', 'b', 'c', 'd').show(4)
```

:: Comparison:

	a	b	c	d
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5

	a	b	c	d
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5

only showing top 4 rows

### 9.10.2 Rename one or more columns

```
mapping = {'Newspaper': 'C', 'Sales': 'D'}
```

:: Python Code:

```
pandas_df.rename(columns=mapping).head(4)
#
new_names = [mapping.get(col, col) for col in spark_df.columns]
spark_df.toDF(*new_names).show(4)
```

:: Comparison:

	TV	Radio	C	D
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5

	TV	Radio	C	D
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5

(continues on next page)

(continued from previous page)

```
+-----+-----+-----+-----+
only showing top 4 rows
```

---

**Note:** You can also use `withColumnRenamed` to rename one column in PySpark.

:: Python Code:

```
spark_df.withColumnRenamed('Newspaper', 'Paper').show(4
```

:: Comparison:

```
+-----+-----+-----+-----+
|    TV|Radio|Paper|Sales|
+-----+-----+-----+-----+
|230.1| 37.8| 69.2| 22.1|
| 44.5| 39.3| 45.1| 10.4|
| 17.2| 45.9| 69.3|  9.3|
|151.5| 41.3| 58.5| 18.5|
+-----+-----+-----+-----+
only showing top 4 rows
```

---

## 9.11 Drop Columns

```
drop_name = ['Newspaper', 'Sales']
```

:: Python Code:

```
pandas_df.drop(drop_name,axis=1).head(4)
#
spark_df.drop(*drop_name).show(4)
```

:: Comparison:

```

      TV  Radio
0  230.1   37.8
1   44.5   39.3
2   17.2   45.9
3  151.5   41.3

+-----+-----+
|    TV|Radio|
+-----+-----+
|230.1| 37.8|
| 44.5| 39.3|
| 17.2| 45.9|
|151.5| 41.3|
+-----+-----+
only showing top 4 rows
```

## 9.12 Filter

```
pandas_df = pd.read_csv('Advertising.csv')
#
spark_df = spark.read.csv(path='Advertising.csv',
                           header=True,
                           inferSchema=True)
```

:: Python Code:

```
pandas_df[pandas_df.Newspaper<20].head(4)
#
spark_df[spark_df.Newspaper<20].show(4)
```

:: Comparison:

	TV	Radio	Newspaper	Sales
7	120.2	19.6	11.6	13.2
6	13.2			
8	8.6	2.1	1.0	4.8
0	4.8			
11	214.7	24.0	4.0	17.4
0	17.4			
13	97.5	7.6	7.2	9.7
2	9.7			

only showing top 4 rows

:: Python Code:

```
pandas_df[(pandas_df.Newspaper<20) & (pandas_df.TV>100)].head(4)
#
spark_df[(spark_df.Newspaper<20) & (spark_df.TV>100)].show(4)
```

:: Comparison:

	TV	Radio	Newspaper	Sales
--	----	-------	-----------	-------

(continues on next page)

(continued from previous page)

	TV	Radio	Newspaper	Sales	
↪+-----+					+-----+-----+-----
7	120.2	19.6	11.6	13.2	120.2   19.6   11.
↪6   13.2					
11	214.7	24.0	4.0	17.4	214.7   24.0   4.
↪0   17.4					
19	147.3	23.9	19.1	14.6	147.3   23.9   19.
↪1   14.6					
25	262.9	3.5	19.5	12.0	262.9   3.5   19.
↪5   12.0					
					+-----+-----+-----
↪+-----+					

only showing top 4 rows

## 9.13 With New Column

:: Python Code:

```
pandas_df['tv_norm'] = pandas_df.TV/sum(pandas_df.TV)
pandas_df.head(4)
#
spark_df.withColumn('tv_norm', spark_df.TV/spark_df.groupBy().agg(F.
↪sum("TV")).collect()[0][0]).show(4)
```

:: Comparison:

↪+-----+-----+-----+								+-----+-----+-----
↪TV Radio Newspaper Sales					tv_norm			
TV Radio Newspaper Sales					tv_norm			+-----+-----+-----
↪+-----+-----+-----+								
0	230.1	37.8	69.2	22.1	0.007824			230.1   37.8   69.
↪2   22.1   0.007824268493802813								
1	44.5	39.3	45.1	10.4	0.001513			44.5   39.3   45.
↪1   10.4   0.001513167961643...								
2	17.2	45.9	69.3	9.3	0.000585			17.2   45.9   69.
↪3   9.3   5.848649200061207E-4								
3	151.5	41.3	58.5	18.5	0.005152			151.5   41.3   58.
↪5   18.5   0.005151571824472517								
								+-----+-----+-----
↪+-----+-----+-----+								

only showing top 4 rows

:: Python Code:

```
pandas_df['cond'] = pandas_df.apply(lambda c: 1 if ((c.TV>100)&(c.Radio
→<40)) else 2 if c.Sales> 10 else 3,axis=1)
#
spark_df.withColumn('cond',F.when((spark_df.TV>100)&(spark_df.Radio
→<40),1)\
                                .when(spark_df.Sales>10, 2)\
                                .otherwise(3)).show(4)
```

:: Comparison:

```
→+-----+-----+
|
→TV|Radio|Newspaper|Sales|cond|
   TV  Radio  Newspaper  Sales  cond
→+-----+-----+
0  230.1   37.8         69.2   22.1    1  |230.1| 37.8|    69.
→2| 22.1|    1|
1  44.5   39.3         45.1   10.4    2  | 44.5| 39.3|    45.
→1| 10.4|    2|
2   17.2   45.9         69.3    9.3    3  | 17.2| 45.9|    69.
→3|  9.3|    3|
3  151.5   41.3         58.5   18.5    2  |151.5| 41.3|    58.
→5| 18.5|    2|
→+-----+-----+
only showing top 4 rows
```

:: Python Code:

```
pandas_df['log_tv'] = np.log(pandas_df.TV)
pandas_df.head(4)
#
spark_df.withColumn('log_tv',F.log(spark_df.TV)).show(4)
```

:: Comparison:

```
→+-----+-----+
|
→TV|Radio|Newspaper|Sales|log_tv|
   TV  Radio  Newspaper  Sales  log_tv
→+-----+-----+
0  230.1   37.8         69.2   22.1  5.438514  |230.1| 37.8|    69.
→2| 22.1|    1|  5.43851399704132|
```

(continues on next page)

(continued from previous page)

```

1  44.5  39.3      45.1  10.4  3.795489 | 44.5| 39.3|      45.
↪1 | 10.4| 3.7954891891721947|
2  17.2  45.9      69.3   9.3  2.844909 | 17.2| 45.9|      69.
↪3 |  9.3| 2.8449093838194073|
3 151.5  41.3      58.5  18.5  5.020586 |151.5| 41.3|      58.
↪5 | 18.5| 5.020585624949423|

+-----+-----+-----+
↪+-----+-----+-----+
only showing top 4 rows

```

:: Python Code:

```

pandas_df['tv+10'] = pandas_df.TV.apply(lambda x: x+10)
pandas_df.head(4)
#
spark_df.withColumn('tv+10', spark_df.TV+10).show(4)

```

:: Comparison:

```

↪+-----+-----+
TV|Radio|Newspaper|Sales|tv+10|
  TV  Radio  Newspaper  Sales  tv+10
↪+-----+-----+
0 230.1  37.8      69.2  22.1  240.1 |230.1| 37.8|      69.
↪2 | 22.1| 240.1|
1  44.5  39.3      45.1  10.4   54.5 | 44.5| 39.3|      45.
↪1 | 10.4| 54.5|
2  17.2  45.9      69.3   9.3   27.2 | 17.2| 45.9|      69.
↪3 |  9.3| 27.2|
3 151.5  41.3      58.5  18.5  161.5 |151.5| 41.3|      58.
↪5 | 18.5| 161.5|

+-----+-----+-----+
↪+-----+-----+
only showing top 4 rows

```



## 9.14 Join

```
leftp = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
                      'B': ['B0', 'B1', 'B2', 'B3'],
                      'C': ['C0', 'C1', 'C2', 'C3'],
                      'D': ['D0', 'D1', 'D2', 'D3']},
                      index=[0, 1, 2, 3])

rightp = pd.DataFrame({'A': ['A0', 'A1', 'A6', 'A7'],
                       'F': ['B4', 'B5', 'B6', 'B7'],
                       'G': ['C4', 'C5', 'C6', 'C7'],
                       'H': ['D4', 'D5', 'D6', 'D7']},
                       index=[4, 5, 6, 7])

lefts = spark.createDataFrame(leftp)
rights = spark.createDataFrame(rightp)
```

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

	A	F	G	H
4	A0	B4	C4	D4
5	A1	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

### 9.14.1 Left Join

:: Python Code:

```
leftp.merge(rightp,on='A',how='left')
#
lefts.join(rights,on='A',how='left')
      .orderBy('A',ascending=True).show()
```

:: Comparison:

	A	B	C	D	F	G	H
0	A0	B0	C0	D0	B4	C4	D4
1	A1	B1	C1	D1	B5	C5	D5

(continues on next page)

(continued from previous page)

```

2  A2  B2  C2  D2  NaN  NaN  NaN          | A2| B2| C2|_
  ↳D2|null|null|null|
3  A3  B3  C3  D3  NaN  NaN  NaN          | A3| B3| C3|_
  ↳D3|null|null|null|

+---+---+---+---+---+---+---+
  ↳+

```

## 9.14.2 Right Join

:: Python Code:

```

leftp.merge(rightp,on='A',how='right')
#
lefts.join(rights,on='A',how='right')
      .orderBy('A',ascending=True).show()

```

:: Comparison:

```

  ↳+
                                     +---+---+---+---+---+---+---+
                                     |  A|  B|  C|  D|  F|  G|  _
  ↳H|
      A      B      C      D      F      G      H      +---+---+---+---+---+---+---+
  ↳+
0  A0      B0      C0      D0      B4      C4      D4      | A0| B0| C0| D0| B4| C4|_
  ↳D4|
1  A1      B1      C1      D1      B5      C5      D5      | A1| B1| C1| D1| B5| C5|_
  ↳D5|
2  A6      NaN      NaN      NaN      B6      C6      D6      | A6|null|null|null| B6| C6|_
  ↳D6|
3  A7      NaN      NaN      NaN      B7      C7      D7      | A7|null|null|null| B7| C7|_
  ↳D7|

+---+---+---+---+---+---+---+
  ↳+

```

## 9.14.3 Inner Join

:: Python Code:

```

leftp.merge(rightp,on='A',how='inner')
#
lefts.join(rights,on='A',how='inner')
      .orderBy('A',ascending=True).show()

```

:: Comparison:

	A	B	C	D	F	G	H
0	A0	B0	C0	D0	B4	C4	D4
1	A1	B1	C1	D1	B5	C5	D5

### 9.14.4 Full Join

:: Python Code:

```
leftp.merge(rightp,on='A',how='full')
#
lefts.join(rights,on='A',how='full')
        .orderBy('A',ascending=True).show()
```

:: Comparison:

	A	B	C	D	F	G	H
0	A0	B0	C0	D0	B4	C4	D4
1	A1	B1	C1	D1	B5	C5	D5
2	A2	B2	C2	D2	NaN	NaN	NaN
3	A3	B3	C3	D3	NaN	NaN	NaN
4	A6	NaN	NaN	NaN	B6	C6	D6
5	A7	NaN	NaN	NaN	B7	C7	D7

## 9.15 Concat Columns

```
my_list = [('a', 2, 3),
           ('b', 5, 6),
           ('c', 8, 9),
           ('a', 2, 3),
           ('b', 5, 6),
           ('c', 8, 9)]
col_name = ['col1', 'col2', 'col3']
#
pandas_df = pd.DataFrame(my_list, columns=col_name)
spark_df = spark.createDataFrame(my_list, schema=col_name)
```

	col1	col2	col3
0	a	2	3
1	b	5	6
2	c	8	9
3	a	2	3
4	b	5	6
5	c	8	9

:: Python Code:

```
# one (or both) of the columns are not string typed, convert it (them)
→ first to string,
    pandas_df['concat'] = pandas_df['col1'] + pandas_df['col2'].
→ astype(str)
    # alternatively
    pandas_df['concat'] = pandas_df.apply(lambda x: '%s%s'%(x['col1'], x[
→ 'col2']), axis=1)

# don't use pandas_df['concat'] = pandas_df[['col1', 'col2']].
→ apply(lambda x: ''.join(x), axis=1)
# note it will error out as expecting both fields are strings
#
    pandas_df
    #
    spark_df.withColumn('concat', F.concat('col1', 'col2')).show()
```

:: Comparison:

	col1	col2	col3	concat
0	a	2	3	a2
1	b	5	6	b5

(continues on next page)

(continued from previous page)

2	c	8	9	c8		c	8	9	c8
3	a	2	3	a2		a	2	3	a2
4	b	5	6	b5		b	5	6	b5
5	c	8	9	c8		c	8	9	c8
					+-----+-----+-----+-----+				

## 9.16 GroupBy

:: Python Code:

```
pandas_df.groupby(['col1']).agg({'col2':'min','col3':'mean'})
#
spark_df.groupBy(['col1']).agg({'col2': 'min', 'col3': 'avg'}).show()
```

:: Comparison:

			+-----+-----+-----+		
			col1	min(col2)	avg(col3)
col1			+-----+-----+-----+		
a	2	3	c	8	9.0
b	5	6	b	5	6.0
c	8	9	a	2	3.0
			+-----+-----+-----+		

## 9.17 Pivot

:: Python Code:

```
pd.pivot_table(pandas_df, values='col3', index='col1', columns='col2',
→aggfunc=np.sum)
#
spark_df.groupBy(['col1']).pivot('col2').sum('col3').show()
```

:: Comparison:

				+-----+-----+-----+			
				col1	2	5	8
col1				+-----+-----+-----+			
a	6.0	NaN	NaN	c	null	null	18
b	NaN	12.0	NaN	b	null	12	null
c	NaN	NaN	18.0	a	6	null	null
				+-----+-----+-----+			

## 9.18 Unixtime to Date

```
from datetime import datetime

my_list = [['a', int("1284101485")], ['b', int("2284101485")], ['c',
    ↪int("3284101485")]]
col_name = ['A', 'ts']

pandas_df = pd.DataFrame(my_list, columns=col_name)
spark_df = spark.createDataFrame(pandas_df)
```

:: Python Code:

```
pandas_df['datetime'] = pd.to_datetime(pandas_df['ts'], unit='s').dt.
    ↪tz_localize('UTC')
pandas_df

spark.conf.set("spark.sql.session.timeZone", "UTC")
from pyspark.sql.types import DateType
spark_df.withColumn('date', F.from_unixtime('ts')).show() #.
    ↪cast(DateType())
```

:: Comparison:

	date	ts	datetime
0	a	1284101485	2010-09-10 06:51:25+00:00
1	b	2284101485	2042-05-19 08:38:05+00:00
2	c	3284101485	2074-01-25 10:24:45+00:00

## PD . DATAFRAME MANIPULATION

---

**Note:** This Chapter *Notebooks* is for beginner. If you have some Python programming experience, you may skip this chapter.

---

### 10.1 TODO..





## RDD . DATAFRAME MANIPULATION

---

**Note:** This Chapter *Notebooks* is for beginner. If you have some Python programming experience, you may skip this chapter.

---

### 11.1 TODO..



## ONLINE COURSES AND USEFUL WEBSITES

---

### Chinese proverb

practice makes perfect.

---

### 12.1 Recommended online courses

There are many open source DS courses. Of course,

Name	Description (Level)	Link
Harvard CS 109a	Intro to Data Science (Easy)	<a href="https://harvard-iacs.github.io/2021-CS109A/">https://harvard-iacs.github.io/2021-CS109A/</a>
NYU DS-GA 1003	Machine Learning (Intermediate)	<a href="https://nyu-ds1003.github.io/spring2021/">https://nyu-ds1003.github.io/spring2021/</a>
UC Berkeley	Full Stack Deep Learning (Advanced)	<a href="https://fullstackdeeplearning.com/">https://fullstackdeeplearning.com/</a>
Stanford CS 224N	Natural Language Processing with Deep Learning (Advanced)	<a href="http://web.stanford.edu/class/cs224n/">http://web.stanford.edu/class/cs224n/</a>
Harvard AC215	Advanced practical data science, MLOps. (Advanced)	<a href="https://harvard-iacs.github.io/2021-AC215/">https://harvard-iacs.github.io/2021-AC215/</a>
Stanford CS 329S	Machine Learning Systems Design (Advanced)	<a href="https://stanford-cs329s.github.io/">https://stanford-cs329s.github.io/</a>

## 12.2 Recommended Online Resources

Link	Description
<a href="https://pythontutor.com/">https://pythontutor.com/</a>	Visualize Code Execution
<a href="https://visualgo.net/en">https://visualgo.net/en</a>	Visualising data structures and algorithms through animation
<a href="https://jalammar.github.io/">https://jalammar.github.io/</a>	Jay's Visualizing machine learning concept
<a href="https://explained.ai/">https://explained.ai/</a>	Deep explanations of machine learning and related topics.

## PACKAGE WRAPPER

It's super easy to wrap your own package in Python. I packed some functions which I frequently used in my daily work. You can download and install it from [My ststspy library](#). The hierarchical structure and the directory structure of this package are as follows.

### 13.1 Hierarchical Structure

```
├── README.md
├── __init__.py
├── requirements.txt
├── setup.py
├── statspy
│   ├── __init__.py
│   ├── basics.py
│   └── tests.py
├── test
│   ├── nb
│   │   └── t.test.ipynb
│   └── test1.py
3 directories, 9 files
```

From the above hierarchical structure, you will find that you have to have `__init__.py` in each directory. I will explain the `__init__.py` file with the example below:

## 13.2 Set Up

```
from setuptools import setup, find_packages

try:
    with open("README.md") as f:
        long_description = f.read()
except IOError:
    long_description = ""

try:
    with open("requirements.txt") as f:
        requirements = [x.strip() for x in f.read().splitlines() if x.
→strip()]
except IOError:
    requirements = []

setup(name='statspy',
      install_requires=requirements,
      version='1.0',
      description='Statistics python library',
      author='Wenqiang Feng',
      author_email='von198@gmail.com',
      license="MIT",
      url='git@github.com:runawayhorse001/statspy.git',
      packages=find_packages(),
      long_description=long_description,
      long_description_content_type="text/markdown",
      classifiers=[
          "License :: OSI Approved :: MIT License",
          "Programming Language :: Python",
          "Programming Language :: Python :: 2",
          "Programming Language :: Python :: 3",
      ],
      include_package_data=True
    )
```

---

**Note:** If you want to compile the conda package, you can add the following code in your setup.py:

1. import import distutils.command.bdist\_conda:

```
import distutils.command.bdist_conda
```

2. pass the options to setup():

```
distclass=distutils.command.bdist_conda.CondaDistribution)

# eg
setup(
    name="conda_example",
    version="1.0",
    distclass=distutils.command.bdist_conda.CondaDistribution,
    conda_buildnum=1,)
```

Then, you can use the following code in terminal to compile the conda package:

```
python setup.py bdist_conda
```

---

## 13.3 Requirements

```
pandas
numpy
scipy
patsy
matplotlib
```

## 13.4 ReadMe

```
# StatsPy

This is my statistics python library repositories.
The ``API`` can be found at: https://runawayhorse001.github.io/statspy.
If you want to clone and install it, you can use

- clone

```{bash}
git clone git@github.com:runawayhorse001/statspy.git
```

- install

```{bash}
cd statspy
pip install -r requirements.txt
python setup.py install
```

(continues on next page)

(continued from previous page)

```
```\n- uninstall\n\n```${bash}\npip uninstall statspy\n```\n\n- test\n\n```${bash}\ncd statspy/test\npython test1.py\n```\n
```



## PUBLISH PACKAGE TO PYPI

In this chapter, you'll learn how to upload your own package to PyPI.

### 14.1 Register PyPI account

If you do not have a PyPI account, you need to register an account at <https://pypi.org/account/register>.

### 14.2 Install twine

```
pip install twine
```

### 14.3 Build Your Package

```
python setup.py sdist bdist_wheel
```

Then you will get a new folder dist:

```
.
├── PyAudit-1.0-py3-none-any.whl
├── PyAudit-1.0-py3.6.egg
└── PyAudit-1.0.tar.gz
```

## 14.4 Upload Your Package

```
twine upload dist/*
```

During the uploading processing, you need to provide your PyPI account username and password:

```
Enter your username: runawayhorse001
Enter your password:
```

## 14.5 Package at PyPI

Here is my PyAudit package at [PyPI](<https://pypi.org/project/PyAudit>). You can install PyAudit using:

```
pip install PyAudit
```

## MODEL DEPLOYMENT WITH FLASK

In this chapter, you'll learn how to deployment your model with `flask`. The main idea and code (I made some essential modification to make it work for Python 3) are from the Git repo:[https://github.com/IlSourceCell/how\\_to\\_deploy\\_a\\_keras\\_model\\_to\\_production](https://github.com/IlSourceCell/how_to_deploy_a_keras_model_to_production). So the copy-right belongs to the original author.

### 15.1 Install flask

```
pip install Flask
```

### 15.2 Train and Save your model

You can use the following code to train and save your CNN model:

```
#python 2/3 compatibility
from __future__ import print_function
#simplified interface for building models
import keras
#our handwritten character labeled dataset
from keras.datasets import mnist
#because our models are simple
from keras.models import Sequential
#dense means fully connected layers, dropout is a technique to improve_
→convergence, flatten to reshape our matrices for feeding
#into respective layers
from keras.layers import Dense, Dropout, Flatten
#for convolution (images) and pooling is a technique to help choose_
→the most relevant features in an image
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
```

(continues on next page)

(continued from previous page)

```

#mini batch gradient descent ftw
batch_size = 128
#10 difference characters
num_classes = 10
#very short training time
epochs = 12

#input image dimensions
#28x28 pixel images.
img_rows, img_cols = 28, 28

#the data downloaded, shuffled and split between train and test sets
#if only all datasets were this easy to import and format
(x_train, y_train), (x_test, y_test) = mnist.load_data()

#this assumes our data format
#For 3D data, "channels_last" assumes (conv_dim1, conv_dim2, conv_dim3,
→ channels) while
#"channels_first" assumes (channels, conv_dim1, conv_dim2, conv_dim3).
if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

#more reshaping
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

#convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

#build our model
model = Sequential()

```

(continues on next page)

(continued from previous page)

```

#convolutional layer with rectified linear unit activation
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))

#again
model.add(Conv2D(64, (3, 3), activation='relu'))
#choose the best features via pooling
model.add(MaxPooling2D(pool_size=(2, 2)))
#randomly turn neurons on and off to improve convergence
model.add(Dropout(0.25))
#flatten since too many dimensions, we only want a classification_
→output
model.add(Flatten())
#fully connected to get all relevant data
model.add(Dense(128, activation='relu'))
#one more dropout for convergence' sake :)
model.add(Dropout(0.5))
#output a softmax to squash the matrix into output probabilities
model.add(Dense(num_classes, activation='softmax'))
#Adaptive learning rate (adaDelta) is a popular form of gradient_
→descent rivaled only by adam and adagrad
#categorical ce since we have multiple classes (10)
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

#train
model.fit(x_train, y_train,
        batch_size=batch_size,
        epochs=epochs,
        verbose=1,
        validation_data=(x_test, y_test))

#how well did it do?
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

#Save the model
# serialize model to JSON
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
# serialize weights to HDF5
model.save_weights("model.h5")
print("Saved model to disk")

```

## 15.3 Deployment with Flask

```

#our web app framework!

#you could also generate a skeleton from scratch via
#http://flask-appbuilder.readthedocs.io/en/latest/installation.html

#Generating HTML from within Python is not fun, and actually pretty
→cumbersome because you have to do the
#HTML escaping on your own to keep the application secure. Because of
→that Flask configures the Jinja2 template engine
#for you automatically.
#requests are objects that flask handles (get set post, etc)
from flask import Flask, render_template,request
#scientific computing library for saving, reading, and resizing images
#from scipy.misc import imsave, imread, imresize
# import cv2 library for saving, reading, and resizing images
import cv2
#for matrix math
import numpy as np
#for importing our keras model
import keras.models
#for regular expressions, saves time dealing with string data
import re
# for convert base64 string to image
import base64

#system level operations (like loading files)
import sys
#for reading operating system data
import os
#tell our app where our saved model is
sys.path.append(os.path.abspath("./model"))
from load import *
#inititalize our flask app
app = Flask(__name__)
#global vars for easy reusability
global model, graph
#initialize these variables
model, graph = init()

#decoding an image from base64 into raw representation
def convertImage(imgData1):
    imgData1 = imgData1.decode("utf-8")
    imgstr = re.search(r'base64, (.*)',imgData1).group(1)
    #print(imgstr)

```

(continues on next page)

(continued from previous page)

```

imgstr_64 = base64.b64decode(imgstr)
with open('output/output.png','wb') as output:
    output.write(imgstr_64)

@app.route('/')
def index():
    #initModel()
    #render out pre-built HTML file right on the index page
    return render_template("index.html")

@app.route('/predict/',methods=['GET','POST'])
def predict():
    #whenever the predict method is called, we're going
    #to input the user drawn character as an image into the model
    #perform inference, and return the classification
    #get the raw data format of the image
    imgData = request.get_data()
    #print(imgData)
    #encode it into a suitable format
    convertImage(imgData)
    print("debug")
    #read the image into memory
    x = cv2.imread('output/output.png',0)
    #compute a bit-wise inversion so black becomes white and vice_
    →versa
    x = np.invert(x)
    #make it the right size
    x = cv2.resize(x,(28,28))
    #imshow(x)
    #convert to a 4D tensor to feed into our model
    x = x.reshape(1,28,28,1)
    print("debug2")
    #in our computation graph
    with graph.as_default():
        #perform the prediction
        out = model.predict(x)
        #print(out)
        print(np.argmax(out,axis=1))
        print("debug3")
        #convert the response to a string
        response = np.array_str(np.argmax(out,axis=1))
        return response

```

(continues on next page)

(continued from previous page)

```
if __name__ == "__main__":  
    #decide what port to run the app in  
    port = int(os.environ.get('PORT', 5000))  
    #run the app locally on the givn port  
    app.run(host='0.0.0.0', port=port)  
    #optional if we want to run in debugging mode  
    #app.run(debug=False)
```

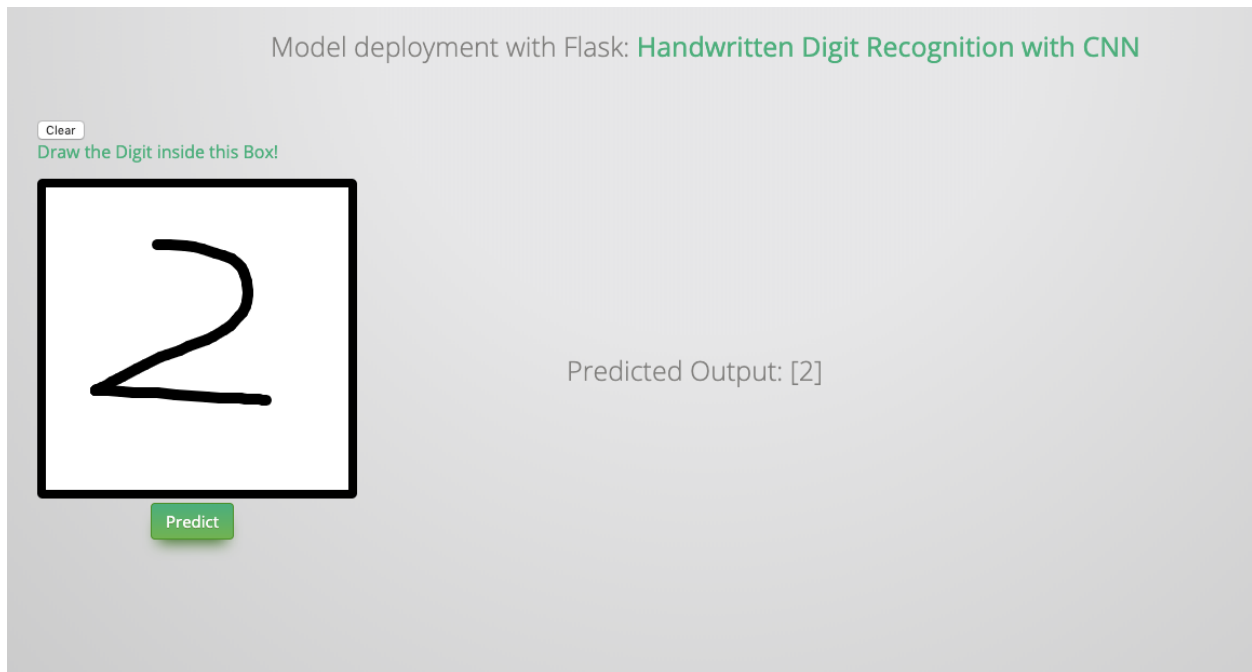
## 15.4 Lunch your app on server

### 15.4.1 1. Lunch the APP

```
python app.py
```

### 15.4.2 2. Run the APP

Open the browser with: <http://0.0.0.0:5000>





## API BOOK

If you developed an amazing library or tool, you need to teach the users how to use it. Now a API book is necessary and a good API book will save a lot of time for the users. The Sphinx provides an awesome auto API book generator. The followings are my statistics python library: `statspy` API demo book:

### 16.1 Basics Module

#### 16.1.1 `rnorm`

`statspy.basics.rnorm(n, mean=0, sd=1)`

Random generation for the normal distribution with mean equal to mean and standard deviation equal to sd same functions as `rnorm` in R: `rnorm(n, mean=0, sd=1)`

##### Parameters

- **n** – the number of the observations
- **mean** – vector of means
- **sd** – vector of standard deviations

**Returns** the vector of the random numbers

**Author** Wenqiang Feng

**Email** [von198@gmail.com](mailto:von198@gmail.com)

### 16.1.2 `dnorm`

`statspy.basics.dnorm(x, mean=0, sd=1, log=False)`

Density of the normal distribution with mean equal to mean and standard deviation equal to sd same functions as `rnorm` in R: `dnorm(x, mean=0, sd=1, log=FALSE)`

#### Parameters

- **x** – the vector of quantiles
- **mean** – vector of means
- **sd** – vector of standard deviations

**Returns** the list of the density

**Author** Wenqiang Feng

**Email** [von198@gmail.com](mailto:von198@gmail.com)

### 16.1.3 `runif`

`statspy.basics.runif(n, min=0, max=1)`

Random generation from the uniform distribution same functions as `rnorm` in R: `runif(n, min=0, max=1)`

#### Parameters

- **n** – the number of the observations
- **min** – the lower limit of the distribution
- **max** – the upper limit of the distribution

**Returns** the list of n uniform random numbers

**Author** Wenqiang Feng

**Email** [von198@gmail.com](mailto:von198@gmail.com)

## 16.2 Tests Module

### 16.2.1 T-test

`statspy.tests.t_test(x, y=None, mu=0.0, conf_level=0.95)`

Performs one and two sample t-tests on vectors of data.

same functions as `t.test` in R: `t.test(x, ...)`

`t.test(x, y = NULL,`

```
alternative = c("two.sided", "less", "greater"),  
mu = 0, paired = FALSE, var.equal = FALSE,  
conf.level = 0.95, ...)
```

**Parameters**

- **x** – a (non-empty) numeric vector of data values.
- **y** – an optional (non-empty) numeric vector of data values.
- **mu** – vector of standard deviations.
- **conf\_level** – confidence level of the interval.

**Returns** the vector of the random numbers.

**Author** Wenqiang Feng

**Email** [von198@gmail.com](mailto:von198@gmail.com)



---

CHAPTER  
**SEVENTEEN**

---

**MAIN REFERENCE**



## BIBLIOGRAPHY

- [VanderPlas2016] Jake VanderPlas. [Python Data Science Handbook: Essential Tools for Working with Data](#), 2016.
- [McKinney2013] Wes McKinney. [Python for Data Analysis](#), 2013.
- [Georg2018] Georg Brandl. [Sphinx Documentation](#), Release 1.7.10+, 2018.





## PYTHON MODULE INDEX

### S

`statspy.basics`, [91](#)  
`statspy.tests`, [92](#)



## M

module

statspy.basics, [91](#)

statspy.tests, [92](#)

## R

`rnorm()` (*in module statspy.basics*), [91](#)

## S

statspy.basics

module, [91](#)

statspy.tests

module, [92](#)

## T

`t_test()` (*in module statspy.tests*), [92](#)