Python Tips for Data Scientist

Release 1.00

Wenqiang Feng

February 28, 2019

CONTENTS

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.

CHAPTER

ONE

PREFACE

Chinese proverb

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

1.1 About

1.1.1 About this tutorial

This document is a summary of my 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, I 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 I am by no means a python programming and Data Scientist expert, I decided that it would be useful for me to share what I learned about Python in the form of easy tutorials with detailed example. I hope those tutorials will be a valuable tool for your studies.

The tutorials assume that the reader has a preliminary knowledge of `python` programing, `LaTex` and `Linux`. And this document is generated automatically by using sphinx.

1.1.2 About the authors

- **Wenqiang Feng**
  - Data Scientist and PhD in Mathematics
  - University of Tennessee at Knoxville

– Email: von198@gmail.com

- **Biography**

  Wenqiang Feng is Data Scientist within DST's Applied Analytics Group. Dr. Feng's responsibilities include providing DST clients with access to cutting-edge skills and technologies, including Big Data analytic solutions, advanced analytic and data enhancement techniques and modeling.

  Dr. Feng has deep analytic expertise in data mining, analytic systems, machine learning algorithms, business intelligence, and applying Big Data tools to strategically solve industry problems in a cross-functional business. Before joining DST, Dr. Feng was an IMA Data Science Fellow at The Institute for Mathematics and its Applications (IMA) at the University of Minnesota. While there, he helped startup companies make marketing decisions based on deep predictive analytics.

  Dr. Feng graduated from University of Tennessee, Knoxville, with Ph.D. in Computational Mathematics and Master's degree in Statistics. He also holds Master's degree in Computational Mathematics from Missouri University of Science and Technology (MST) and Master's degree in Applied Mathematics from the University of Science and Technology of China (USTC).

- **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 author and do not necessarily reflect the views of the IMA, UTK and DST.

1.2 Motivation for this tutorial

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`. Graduately I recognize the elegance of Python and use it as one of my main programming language. But I foud that:

- Most of the `Python` books or tutorials which emphasize on programming will overwhelme the green hand.
- While most of the `Python` books or tutorials `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) for improvements.

CHAPTER

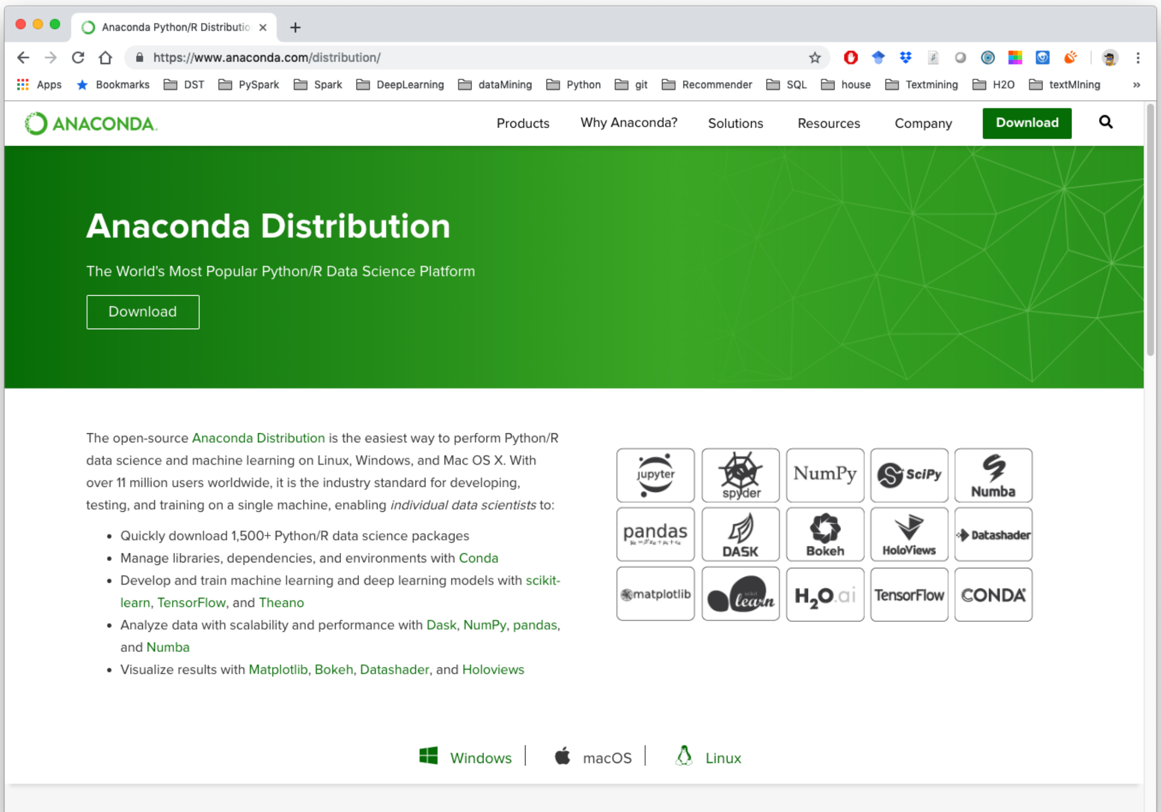
TWO

PYTHON INSTALLATION

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

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/


Anaconda Python/R Distributio
https://www.anaconda.com/distribution/
ANACONDA
Products
Why Anaconda?
Solutions
Resources
Company
Download
Anaconda Distribution
The World's Most Popular Python/R Data Science Platform
Download
The open-source Anaconda Distribution is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 11 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling individual data scientists to:
Quickly download 1,500+ Python/R data science packages
Manage libraries, dependencies, and environments with Conda
Develop and train machine learning and deep learning models with scikit-learn, TensorFlow, and Theano
Analyze data with scalability and performance with Dask, NumPy, pandas, and Numba
Visualize results with Matplotlib, Bokeh, Datashader, and Holoviews
Windows
macOS
Linux

CHAPTER THREE

NOTEBOOKS

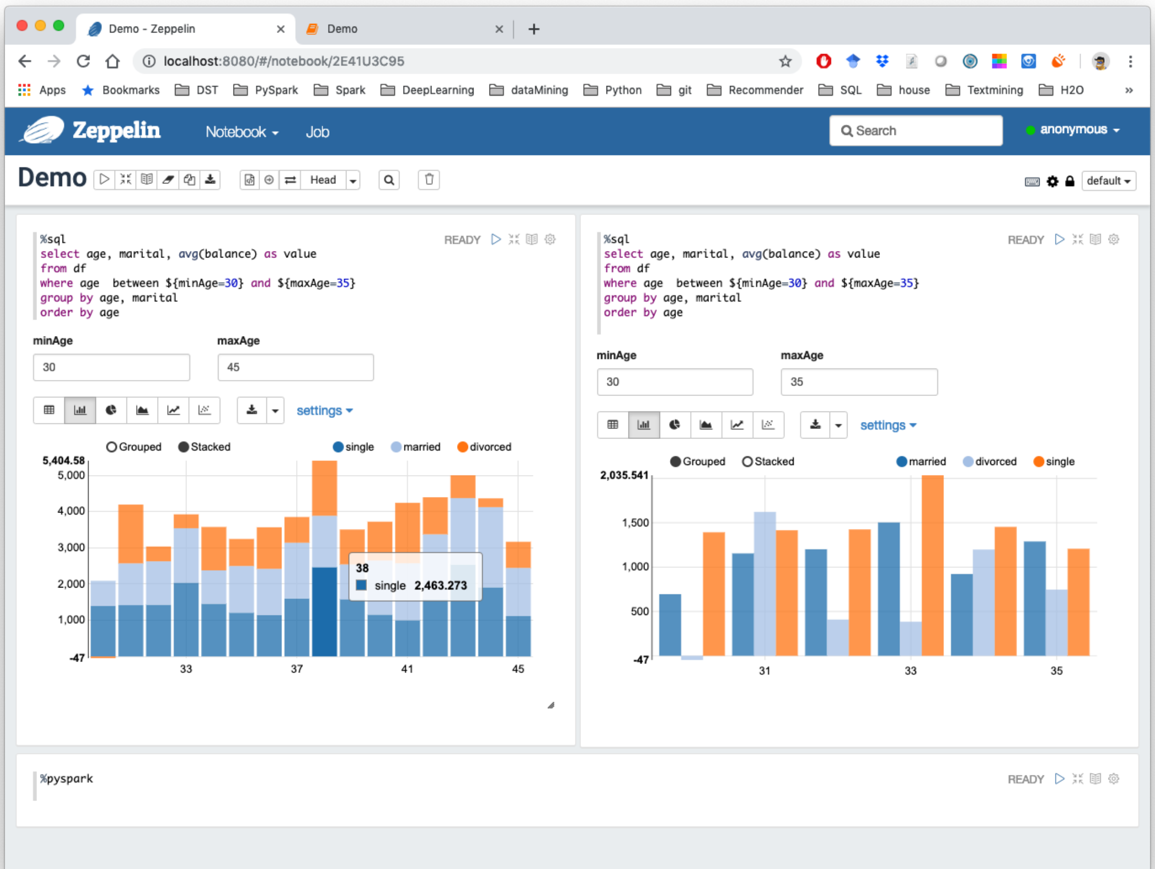
Note: This Chapter *Notebooks* is for beginner. If you have some `Python` programming experience, you may skip this chapter.

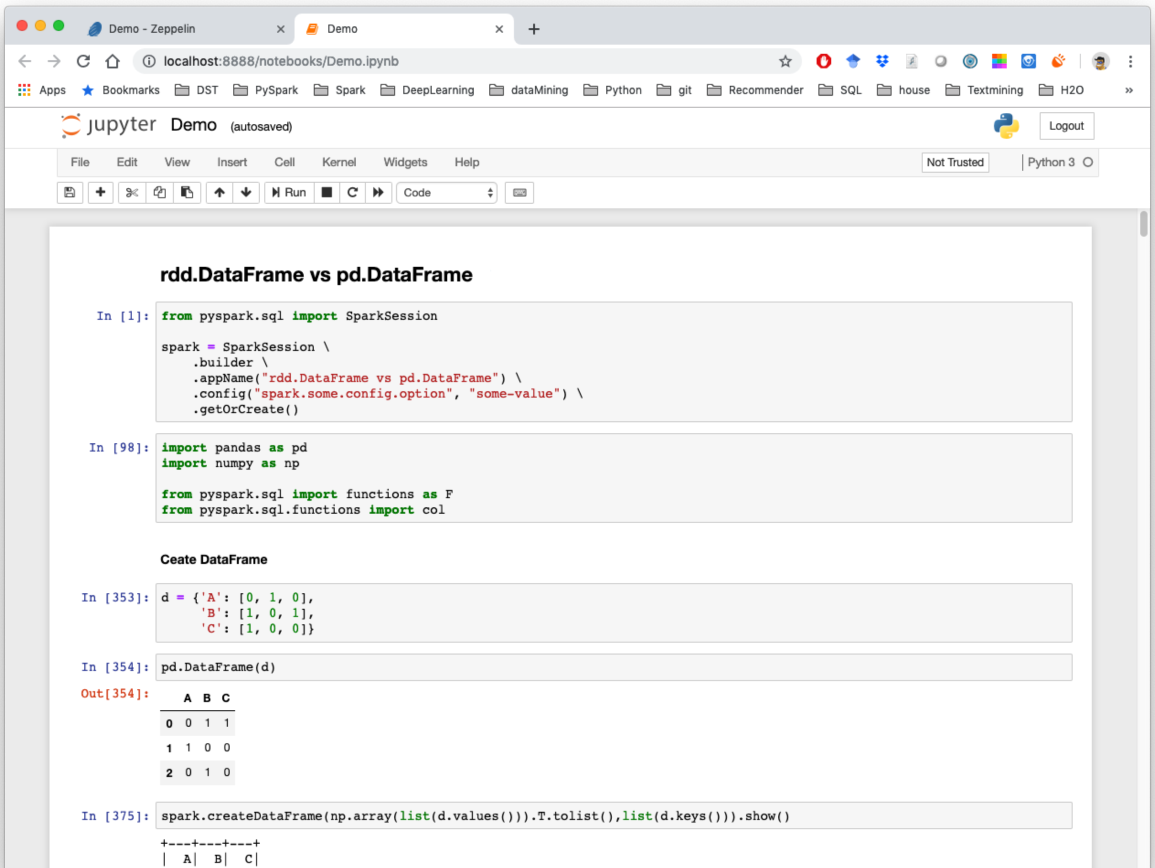
3.1 Apache Zeppelin

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.

3.2 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.




rdd.DataFrame vs pd.DataFrame
Ceate DataFrame

CHAPTER

FOUR

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.

4.1 *

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
```

4.2 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
```

4.3 random

More details can be found at:

a. random: https://docs.python.org/3/library/random.html#random.randint

b. np.random: https://docs.scipy.org/doc/numpy/reference/routines.random.html

4.3.1 random.random

:: Python Code:

```
import random
random.random()

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

:: Ouput:

```
0.33844051243073625
7.772024014335885
```

4.3.2 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]])
```

4.4 `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]])
```

4.5 TODO..

`:: Python Code:`

```

```

`:: Ouput:`

```

```

`:: Python Code:`

```

```

`:: Ouput:`

```

```

`:: Python Code:`

```
```

```
:: Ouput:
```

```
```

```
:: Python Code:
```

```
```

```
:: Ouput:
```

```
```

CHAPTER

FIVE

DATA STRUCTURES

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

5.1 List

`List` is one of data sctructures which is heavily using in my daily work.

5.1.1 Create list

1. Create empty list

The empty list is used to initialize a list.

`::  Python Code:`

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

`::  Ouput:`

```
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)
```

:: Ouput:

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

5.1.2 Unpack list

5.2 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.

[VanderPlas2016] [McKinney2013] [Georg2018]

CHAPTER

SIX

`PD.DATAFRAME` MANIPULATION

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

6.1 TODO..

CHAPTER

SEVEN

RDD.DATAFRAME MANIPULATION

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

7.1 TODO..

CHAPTER

EIGHT

PD.DATAFRAME VS PD.DATAFRAME

8.1 Create DataFrame

8.1.1 From DataBase

login.txt:

```
runawayhorse001
PythonTips
```

```
#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 avaliable!!!')

#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)
dp = 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}
ds = spark.read.jdbc(url=url, table=table_name, properties=properties)
```

8.1.2 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|
   A  B  C         +---+---+---+
0  a  1  2         |  a|  1|  2|
1  b  2  3         |  b|  2|  3|
2  c  3  4         |  c|  3|  4|
                   +---+---+---+
```

Attention: Pay attentation 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
```

8.1.3 From Dict

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

```
::  Python Code:
```

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

```
::  Comparison:
```

```
                         +---+---+---+
                         |  A|  B|  C|
   A  B  C               +---+---+---+
0  0  1  1               |  0|  1|  1|
1  1  0  0               |  1|  0|  0|
2  0  1  0               |  0|  1|  0|
                         +---+---+---+
```

8.2 Load DataFrame

8.2.1 From .csv

8.2.2 From .json

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

8.2.3 From DataBase

```
dp = pd.read_json("data/data.json")
ds = spark.read.json('data/data.json')
```

```
:: Python Code:
```

```
dp[['id','timestamp']].head(4)
#
ds[['id','timestamp']].show(4)
```

```
:: Comparison:
```

```
                                                            +----------+-----------
↪--------+
                                                            |        id|           ␣
↪timestamp|
              id  timestamp                                 +----------+-----------
↪--------+
0   2994551481  2019-02-28 17:23:52                         |2994551481|2019-02-28␣
↪17:23:52|
1   2994551482  2019-02-28 17:23:52                         |2994551482|2019-02-28␣
↪17:23:52|
2   2994551483  2019-02-28 17:23:52                         |2994551483|2019-02-28␣
↪17:23:52|
3   2994551484  2019-02-28 17:23:52                         |2994551484|2019-02-28␣
↪17:23:52|
                                                            +----------+-----------
↪--------+
                                                            only showing top 4 rows
```

8.3 First n Rows

```
:: Python Code:
```

```
dp.head(4)
#
ds.show(4)
```

```
:: Comparison:
```

```
                                                    +-----+-----+---------+-----+
                                                    |   TV|Radio|Newspaper|Sales|
       TV  Radio  Newspaper  Sales                  +-----+-----+---------+-----+
```

(continues on next page)

(continued from previous page)

```
0  230.1   37.8        69.2   22.1        |230.1| 37.8|      69.2| 22.1|
1   44.5   39.3        45.1   10.4        | 44.5| 39.3|      45.1| 10.4|
2   17.2   45.9        69.3    9.3        | 17.2| 45.9|      69.3|  9.3|
3  151.5   41.3        58.5   18.5        |151.5| 41.3|      58.5| 18.5|
                                          +-----+-----+---------+-----+
                                          only showing top 4 rows
```

8.4 Column Names

```
::  Python Code:
```

```
dp.columns
#
ds.columns
```

```
::  Comparison:
```

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

8.5 Data types

```
::  Python Code:
```

```
dp.dtypes
#
ds.dtypes
```

```
::  Comparison:
```

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

8.6 Fill Null

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

```
:: Comparison:
```

```
                                                +------+---+----+
                                                |     A|  B|   C|
         A  B    C                              +------+---+----+
0     male  1  NaN                              |  male|  1|null|
1   female  2  3.0                              |female|  2|   3|
2     male  3  4.0                              |  male|  3|   4|
                                                +------+---+----+
```

```
:: Python Code:
```

```
dp.fillna(-99)
#
ds.fillna(-99).show()
```

```
:: Comparison:
```

```
                                                +------+---+----+
                                                |     A|  B|   C|
         A  B    C                              +------+---+----+
0     male  1  -99                              |  male|  1| -99|
1   female  2  3.0                              |female|  2|   3|
2     male  3  4.0                              |  male|  3|   4|
                                                +------+---+----+
```

8.7 Replace Values

```
:: Python Code:
```

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

```
:: Comparison:
```

```
                                +---+---+----+
                                |  A|  B|   C|
   A  B    C                    +---+---+----+
0  1  1  NaN                    |  1|  1|null|
1  0  2  3.0                    |  0|  2|   3|
2  1  3  4.0                    |  1|  3|   4|
                                +---+---+----+
```

8.8 Rename Columns

8.8.1 Rename all columns

```
:: Python Code:
```

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

```
:: Comparison:
```

```
                                                +-----+----+----+----+
                                                |    a|   b|   c|   d|
       a     b     c     d                      +-----+----+----+----+
0  230.1  37.8  69.2  22.1                      |230.1|37.8|69.2|22.1|
1   44.5  39.3  45.1  10.4                      | 44.5|39.3|45.1|10.4|
2   17.2  45.9  69.3   9.3                      | 17.2|45.9|69.3| 9.3|
3  151.5  41.3  58.5  18.5                      |151.5|41.3|58.5|18.5|
                                                +-----+----+----+----+
                                                only showing top 4 rows
```

8.8.2 Rename one or more columns

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

```
:: Python Code:
```

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

:: Comparison:

```
                                                  +-----+-----+----+----+
                                                  |   TV|Radio|   C|   D|
      TV  Radio     C     D                       +-----+-----+----+----+
0  230.1   37.8  69.2  22.1                       |230.1| 37.8|69.2|22.1|
1   44.5   39.3  45.1  10.4                       | 44.5| 39.3|45.1|10.4|
2   17.2   45.9  69.3   9.3                       | 17.2| 45.9|69.3| 9.3|
3  151.5   41.3  58.5  18.5                       |151.5| 41.3|58.5|18.5|
                                                  +-----+-----+----+----+
                                                  only showing top 4 rows
```

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

:: Python Code:

```
ds.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
```

8.9 Drop Columns

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

:: Python Code:

```
dp.drop(drop_name,axis=1).head(4)
#
ds.drop(*drop_name).show(4)
```

:: Comparison:

```
                                                +-----+-----+
                                                |   TV|Radio|
      TV  Radio                                 +-----+-----+
0  230.1   37.8                                 |230.1| 37.8|
1   44.5   39.3                                 | 44.5| 39.3|
2   17.2   45.9                                 | 17.2| 45.9|
3  151.5   41.3                                 |151.5| 41.3|
                                                +-----+-----+
                                                only showing top 4 rows
```

8.10 Filter

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

```
:: Python Code:
```

```
dp[dp.Newspaper<20].head(4)
#
ds[ds.Newspaper<20].show(4)
```

```
:: Comparison:
```

```
                                                                +-----+-----+---------
↪+------+
                                                                |  ␣
↪TV|Radio|Newspaper|Sales|
       TV  Radio  Newspaper  Sales                              +-----+-----+---------
↪+------+
7   120.2   19.6       11.6   13.2                              |120.2| 19.6|      11.
↪6| 13.2|
8     8.6    2.1        1.0    4.8                              |  8.6|  2.1|       1.
↪0|  4.8|
11  214.7   24.0        4.0   17.4                              |214.7| 24.0|       4.
↪0| 17.4|
13   97.5    7.6        7.2    9.7                              | 97.5|  7.6|       7.
↪2|  9.7|
                                                                +-----+-----+---------
↪+------+
                                                                only showing top 4 rows
```

```
:: Python Code:
```

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

```
:: Comparison:
```

```
                                                            +-----+-----+---------
↪+-----+
                                                            |   ␣
↪TV|Radio|Newspaper|Sales|
       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
```

8.11 With New Column

```
:: Python Code:
```

```
dp['tv_norm'] = dp.TV/sum(dp.TV)
dp.head(4)
#
ds.withColumn('tv_norm', ds.TV/ds.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|
```

(continues on next page)

(continued from previous page)

```
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:

```
dp['cond'] = dp.apply(lambda c: 1 if ((c.TV>100)&(c.Radio<40)) else 2␣
↪if c.Sales> 10 else 3,axis=1)
#
ds.withColumn('cond',F.when((ds.TV>100)&(ds.Radio<40),1)\
                      .when(ds.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:

```
dp['log_tv'] = np.log(dp.TV)
dp.head(4)
#
ds.withColumn('log_tv',F.log(ds.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| 5.43851399704132|
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:
```

```
dp['tv+10'] = dp.TV.apply(lambda x: x+10)
dp.head(4)
#
ds.withColumn('tv+10', ds.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
```

8.12 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                     A   F   G   H
0  A0  B0  C0  D0                4  A0  B4  C4  D4
1  A1  B1  C1  D1                5  A1  B5  C5  D5
2  A2  B2  C2  D2                6  A6  B6  C6  D6
3  A3  B3  C3  D3                7  A7  B7  C7  D7
```

8.12.1 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|
    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|
```

(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|
                                                 +---+---+---+---+----+----+----
↪+
```

8.12.2 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|
                                                 +---+----+----+----+---+---+---
↪+
```

8.12.3 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|
    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|
                                       +---+---+---+---+---+---+---+
```

8.12.4 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|
    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  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|
4  A6  NaN  NaN  NaN   B6   C6   D6     | A6|null|null|null|  B6|  C6|␣
↪ D6|
5  A7  NaN  NaN  NaN   B7   C7   D7     | A7|null|null|null|  B7|  C7|␣
↪ D7|
                                        +---+----+----+----+----+----+-
↪---+
```

8.13 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']
#
dp = pd.DataFrame(my_list,columns=col_name)
ds = 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:
```

```
dp['concat'] = dp.apply(lambda x:'%s%s'%(x['col1'],x['col2']),axis=1)
dp
#
ds.withColumn('concat',F.concat('col1','col2')).show()
```

```
:: Comparison:
```

```
                                                 +----+----+----+------+
                                                 |col1|col2|col3|concat|
  col1  col2  col3 concat                        +----+----+----+------+
0    a     2     3     a2                        |   a|   2|   3|    a2|
1    b     5     6     b5                        |   b|   5|   6|    b5|
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|
                                                 +----+----+----+------+
```

8.14 GroupBy

```
:: Python Code:
```

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

:: Comparison:

```
                                                        +----+---------+---------+
      col2  col3                                        |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|
                                                        +----+---------+---------+
```

8.15 Pivot

:: Python Code:

```
pd.pivot_table(dp, values='col3', index='col1', columns='col2',␣
↪aggfunc=np.sum)
#
ds.groupBy(['col1']).pivot('col2').sum('col3').show()
```

:: Comparison:

```
                                 +----+----+----+----+
col2     2     5     8           |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|
                                 +----+----+----+----+
```

CHAPTER NINE

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 PySpark Package. The hierarchical structure and the directory structure of this package are as follows.

9.1 Hierarchical Structure

```
PySparkTools/
├── __init__.py
├── PySparkTools
│   ├── __init__.py
│   ├── Manipulation
│   │   ├── DataManipulation.py
│   │   └── __init__.py
│   └── Visualization
│       ├── __init__.py
│       ├── PyPlots.py
│       └── PyPlots.pyc
├── README.md
├── requirements.txt
├── setup.py
└── test
    ├── spark-warehouse
    ├── test1.py
    └── test2.py
```

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:

9.2 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='PySParkTools',
          install_requires=requirements,
      version='1.0',
      description='Python Spark Tools',
      author='Wenqiang Feng',
      author_email='von198@gmail.com',
      url='https://github.com/runawayhorse001/PySparkTools',
      packages=find_packages(),
      long_description=long_description
     )
```

9.3 ReadMe

```
# PySparkTools

This is my PySpark Tools. If you want to colne and install it, you can␣
↪use

- clone

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

- install

```{bash}
```

(continues on next page)
```

(continued from previous page)

```
cd PySparkTools
pip install -r requirements.txt
python setup.py install
```


- test

```{bash}
cd PySparkTools/test
python test1.py
```
```
```

CHAPTER

TEN

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.