Big Data Technology and its Applications

Python

张宁 ningzhang@tsinghua.edu.cn

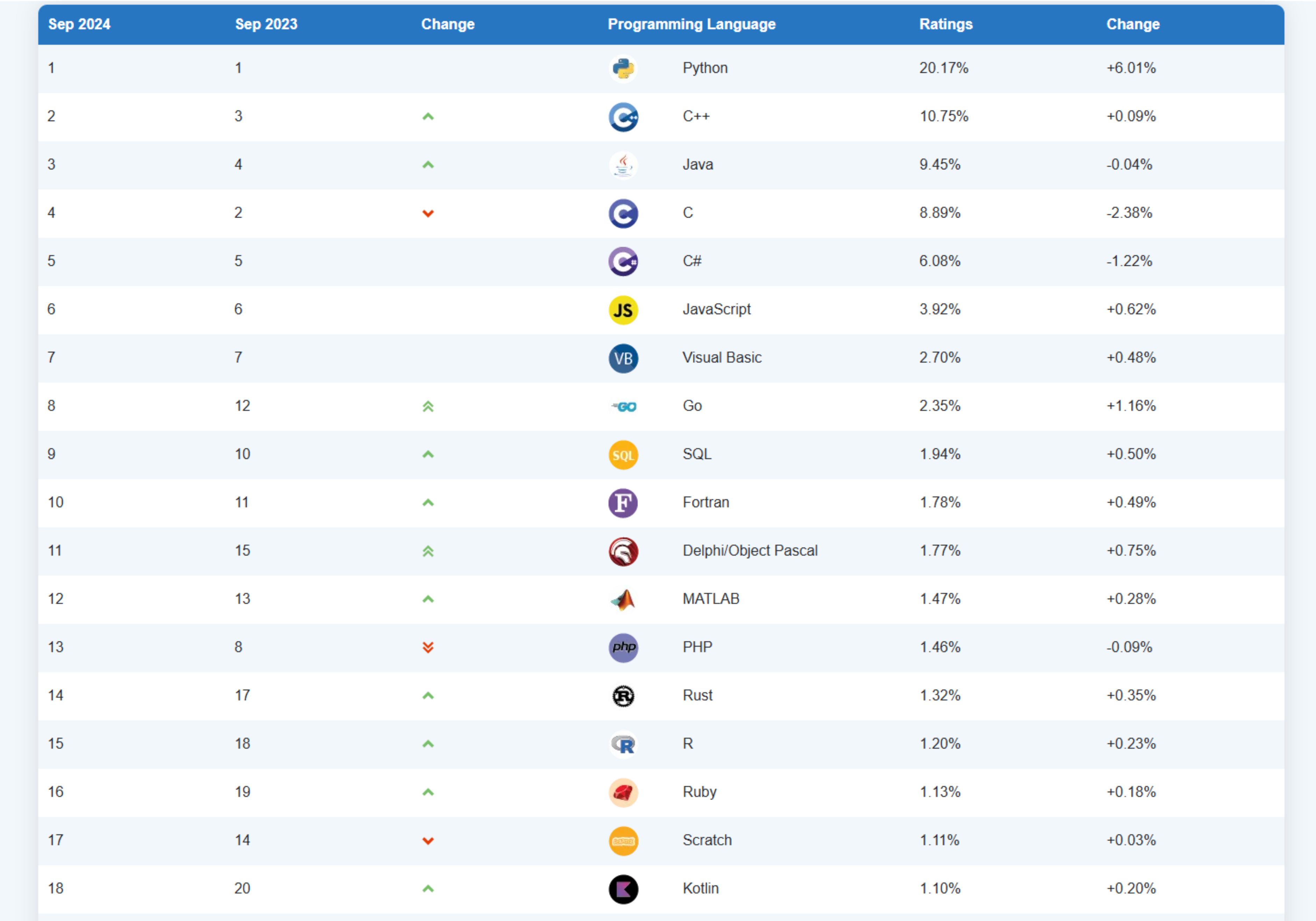
Outline

- Python introduction
- NumPy
- SciPy
- Matplotlib
- Pandas
- Example for long-term load forecasting

Compared with other languages

- Java
  - Python programs are generally expected to run slower than Java programs, but they also take much less time (typically 3-5 times shorter) to develop.
- C++
  - Similar with Java, just more so. One Python programmer can finish in two months what two C++ programmers can't complete in a year. Python can integrate C++.
- Matlab
  - Both of them are easy to use. Python is stronger in ecosystem. They have some well-developed packages such as pytorch, tensorflow, and scikit-learn.

https://www.python.org/doc/essays/comparisons/



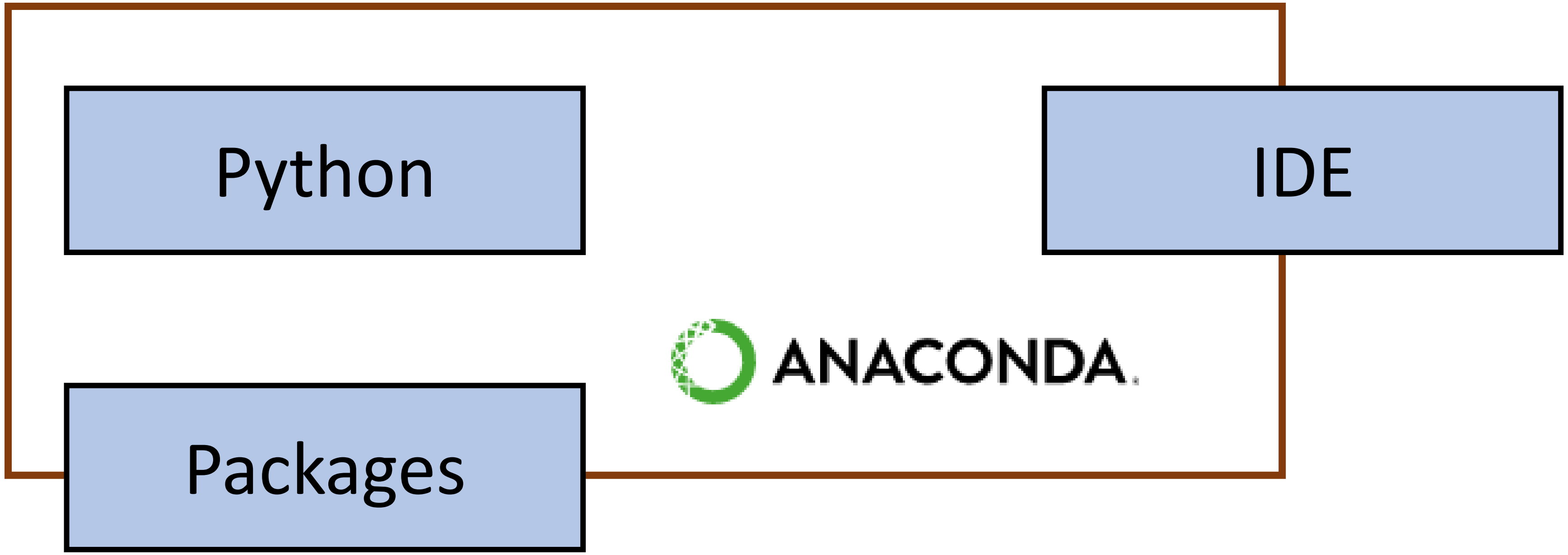
Sep 2024	Sep 2023	Change	Programming Language	Ratings	Change
1	1		Python	20.17%	+6.01%
2	3	↑	C++	10.75%	+0.09%
3	4	↑	Java	9.45%	-0.04%
4	2	↓	C	8.89%	-2.38%
5	5		C#	6.08%	-1.22%
6	6		JavaScript	3.92%	+0.62%
7	7		Visual Basic	2.70%	+0.48%
8	12	⇈	Go	2.35%	+1.16%
9	10	↑	SQL	1.94%	+0.50%
10	11	↑	Fortran	1.78%	+0.49%
11	15	⇈	Delphi/Object Pascal	1.77%	+0.75%
12	13	↑	MATLAB	1.47%	+0.28%
13	8	⇊	PHP	1.46%	-0.09%
14	17	↑	Rust	1.32%	+0.35%
15	18	↑	R	1.20%	+0.23%
16	19	↑	Ruby	1.13%	+0.18%
17	14	↓	Scratch	1.11%	+0.03%
18	20	↑	Kotlin	1.10%	+0.20%

https://www.tiobe.com/tiobe-index/

Overview of python, IDE, packages, and anaconda

Programming language

Integrated development environment



Help you to implement a certain kind of functions

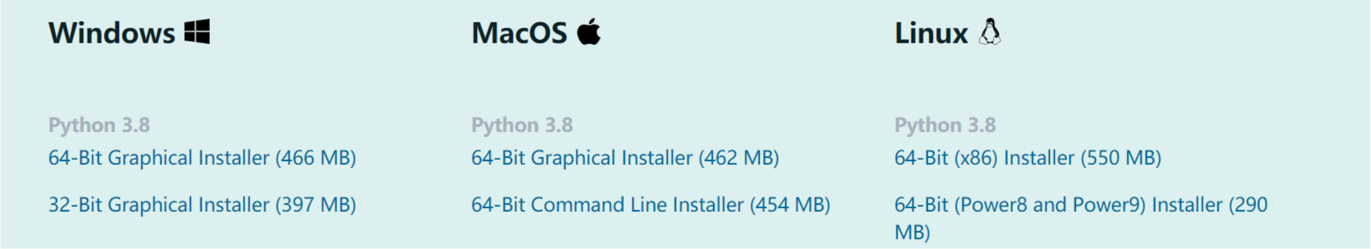
Install Python and other packages

- Install at https://www.python.org/
  - Use python 3.x rather than 2.x
- Or install anaconda instead
  - Get python, IDE (Spyder, Jupyter), and some other packages (NumPy, SciPy, pandas, …) together.
  - https://www.anaconda.com/products/individual
- Choose an IDE and start coding
  - Spyder, Jupyter Notebook
- Install some other packages if necessary
  - pip install **, conda install ** in the cmd/terminal
  - Get instruction from the website of the corresponding packages

Comparison: Spyder, Jupyter Notebook, Pycharm

- Spyder
    - An IDE specifically for scientific programming in Python
    - Interface similar to Matlab
    - Easy to manage packages trough Anaconda
- Jupyter Notebook
    - An interactive Python notebook where you can run code, visualize data and include text all in one document
    - Based on Web
- Pycharm
    - A powerful and featureful IDE for Python applications
    - Need to configure environment for each project
    - https://www.runoob.com/w3cnote/pycharm-windows-install.html

Using anaconda



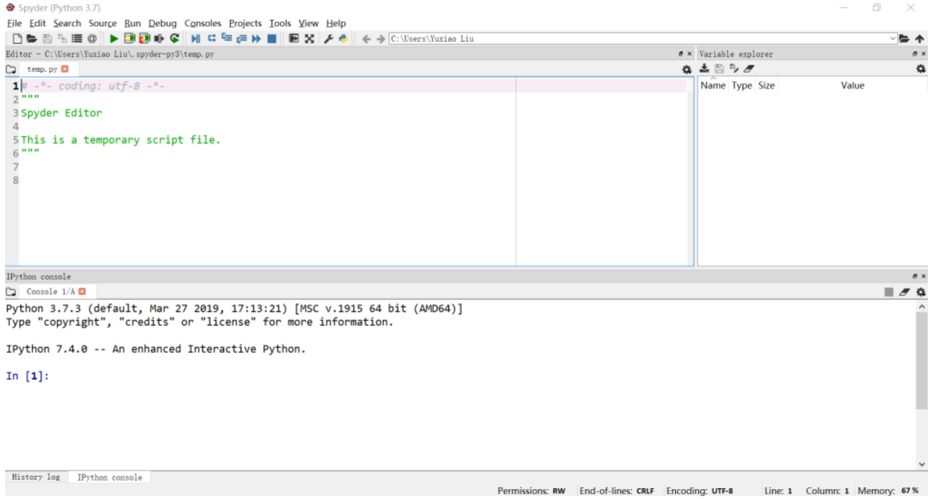
Windows	MacOS	Linux
Python 3.8	Python 3.8	Python 3.8
64-Bit Graphical Installer (466 MB)	64-Bit Graphical Installer (462 MB)	64-Bit (x86) Installer (550 MB)
32-Bit Graphical Installer (397 MB)	64-Bit Command Line Installer (454 MB)	64-Bit (Power8 and Power9) Installer (290 MB)

- Download and install at https://www.anaconda.com/products/individual

OR 清华大学开源镜像站

https://mirrors.tuna.tsinghua.edu.cn/anaconda/archive

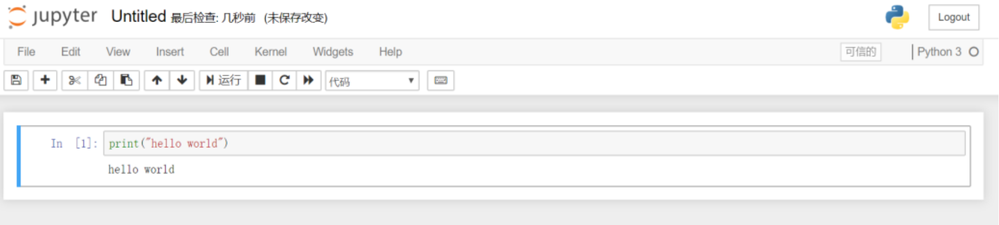
- Start spyder


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Yuxiao Liu
Editor - C:\Users\Yuxiao Liu\.spyder-py3\temp.py
temp.py
Variable explorer
Name Type Size Value
IPython console
Console 1/A
History log IPython console
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 1 Column: 1 Memory: 67 %

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
```

```
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.4.0 -- An enhanced Interactive Python.

In [1]:
```

- Or type jupyter notebook in the cmd/terminal
- Start a new notebook

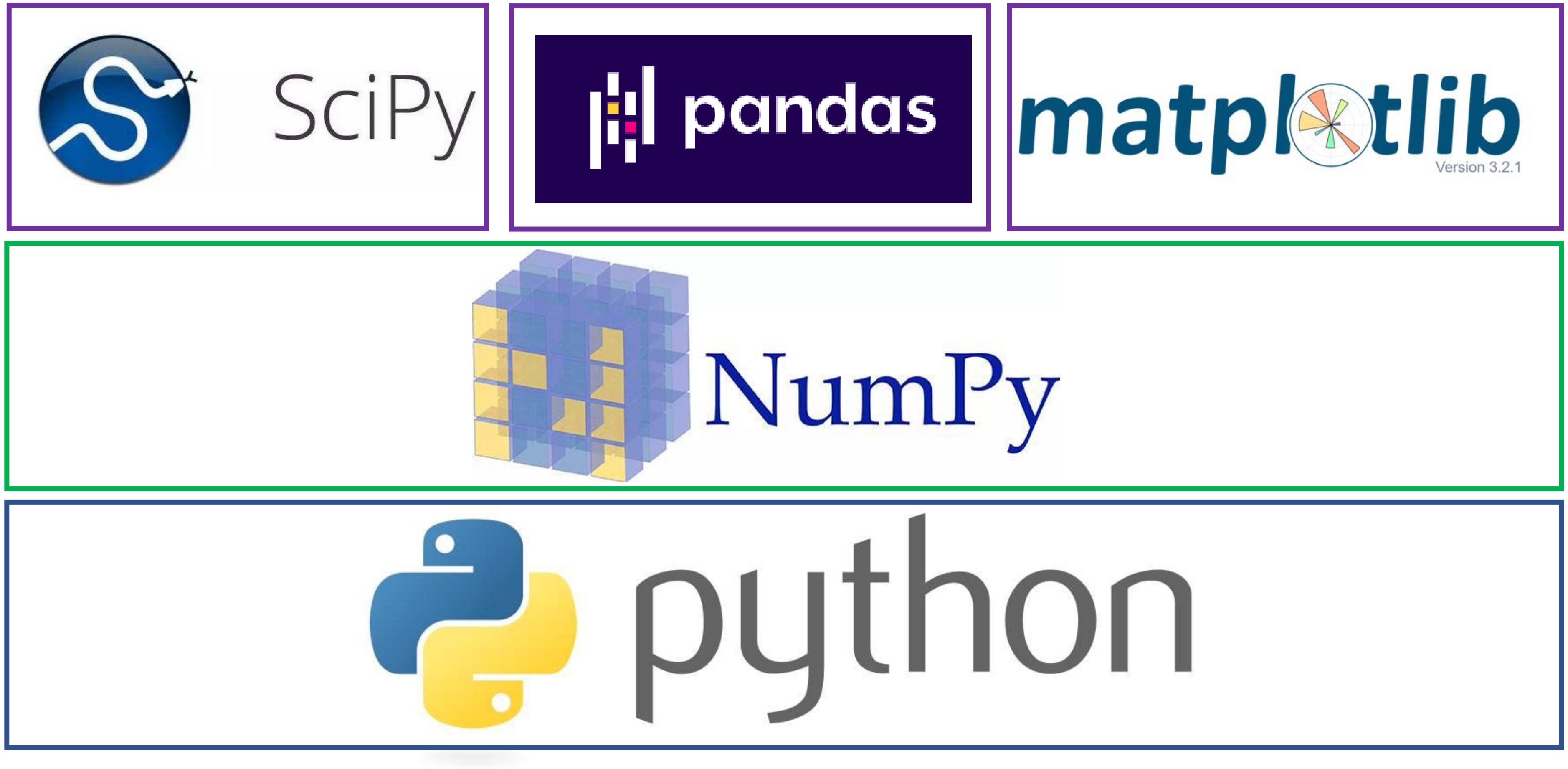

jupyter Untitled 最后检查: 几秒前 (未保存改变)
Logout
File Edit View Insert Cell Kernel Widgets Help
可信的
Python 3
运行
代码

```
In [1]: print("hello world")

        hello world
```

Python features

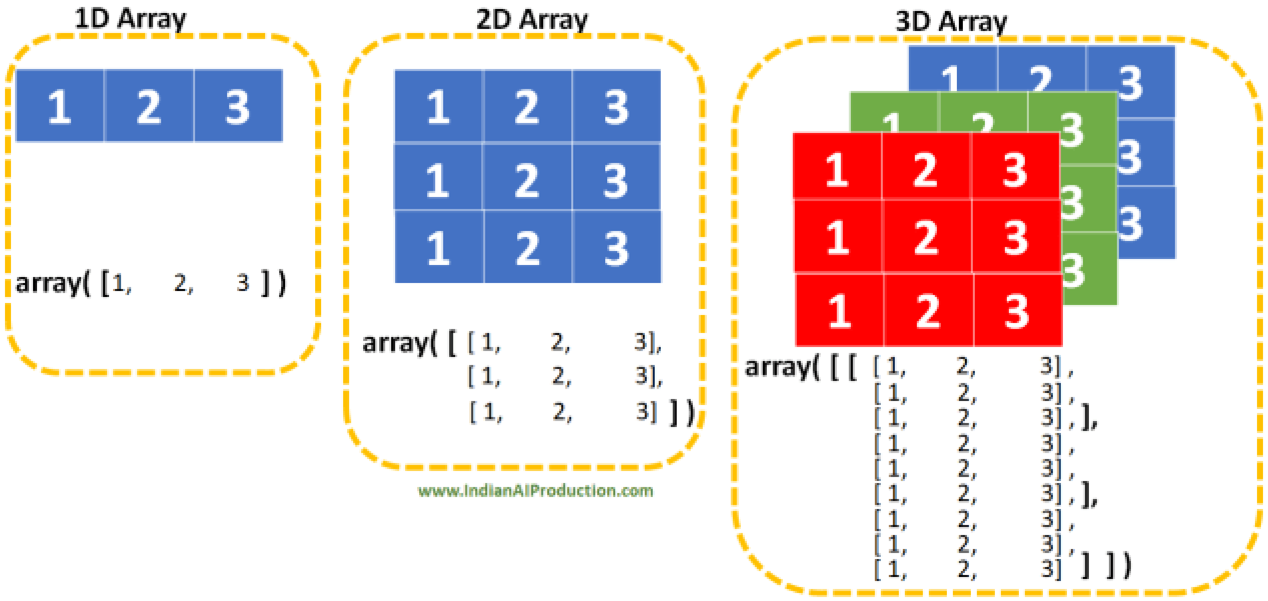
- No compiling
- Rapid development cycle
- No type declarations
- High-level data types and operations
- Simpler, shorter, more flexible
- Automatic memory management
- Garbage collection
- Object-oriented programming
- Mixed language systems

Useful python packages



Numpy

- NumPy is the fundamental package for scientific computing with Python. It contains among other things:
  - a powerful N-dimensional array object (ndarray)
  - tools for integrating C/C++ and Fortran code
  - useful linear algebra, Fourier transform, and random number capabilities



```
array( [1,    2,    3 ] )
```

```
array( [ [ 1,    2,    3],
         [ 1,    2,    3],
         [ 1,    2,    3] ] )
```

```
array( [ [ [1,    2,    3] ,
           [1,    2,    3] ,
           [1,    2,    3] , ],
           [1,    2,    3] ,
           [1,    2,    3] ,
           [1,    2,    3] , ],
           [1,    2,    3] ,
           [1,    2,    3] ,
           [1,    2,    3] ] ] )
```

www.IndianAIProduction.com

https://www.runoob.com/numpy/numpy-ndarray-object.html

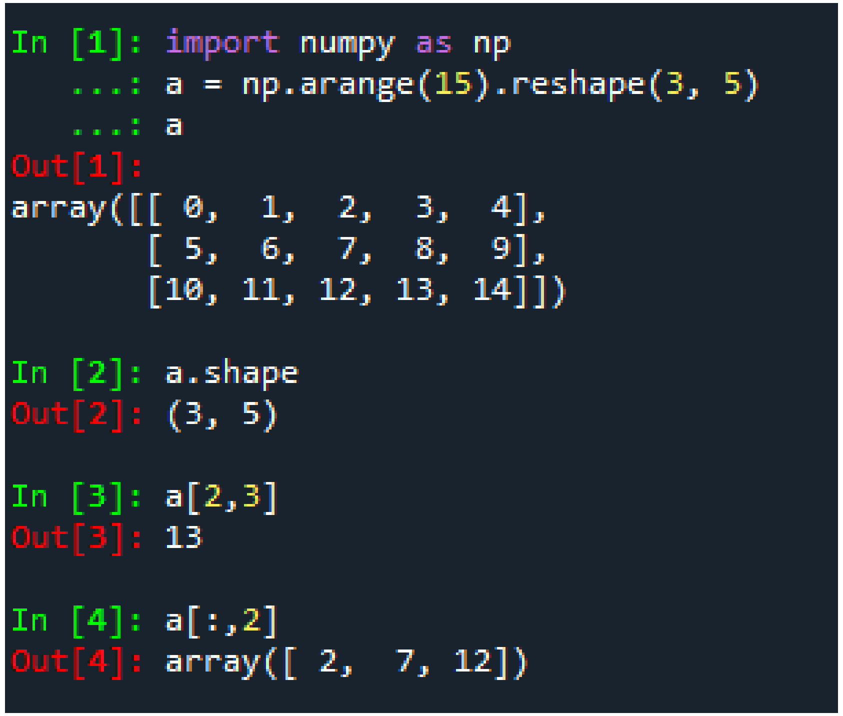
What can NumPy do

Constants	np.inf, np.pi, np.nan, …
Mathematical functions	Math (np.sqrt, …), Trigonometric (np.sin, …), …
Array creation	np.ones, np.ones_like, …
Array manipulation	np.transpose, np.stack, np.tile, np.flip, …
Functional programming	np.apply_along_axis, np.piecewise, …
Indexing	np.where, np.take, np.select, …
I/O	np.loadtxt, np.load, np.save, …
Logic functions	Comparison (np.greater, …), …
Polynomials	np.poly1d, np.polyfit, …
Random sampling	Simple random data (np.rand, …), Permutations (np.shuffle, …), Distributions (np.normal, …), Random generator (np.seed, …)
Statistics	np.mean, np.sum, np.correlate, np.histogram, …
Financial functions	np.fv, np.ppmt, np.rate, …

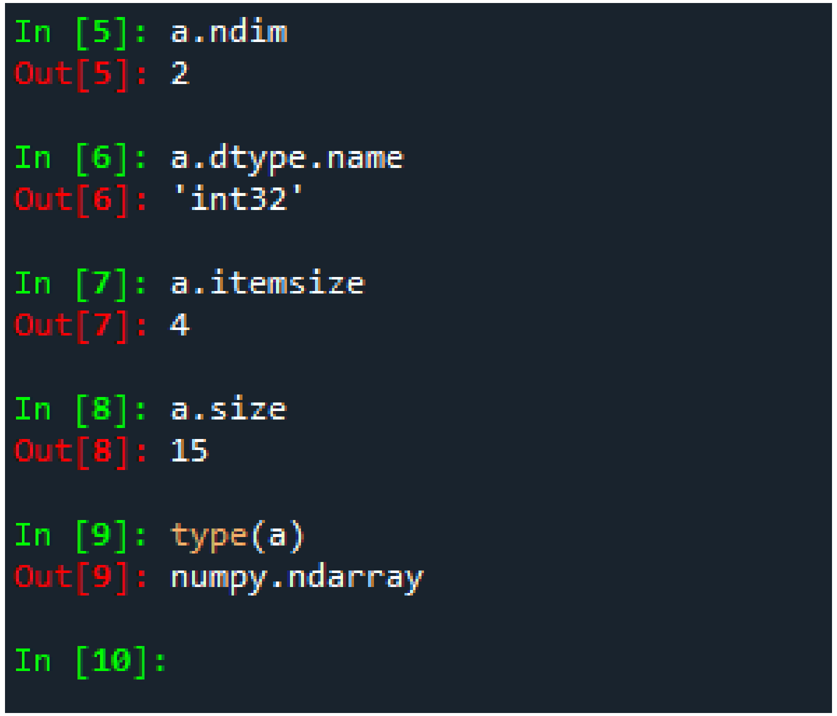
https://numpy.org/
https://numpy.org/doc/stable/user/quickstart.html

What can NumPy do

Type and run in the Console of Spyder



```
In [1]: import numpy as np
   ...: a = np.arange(15).reshape(3, 5)
   ...: a
Out[1]:
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])

In [2]: a.shape
Out[2]: (3, 5)

In [3]: a[2,3]
Out[3]: 13

In [4]: a[:,2]
Out[4]: array([ 2,  7, 12])
```

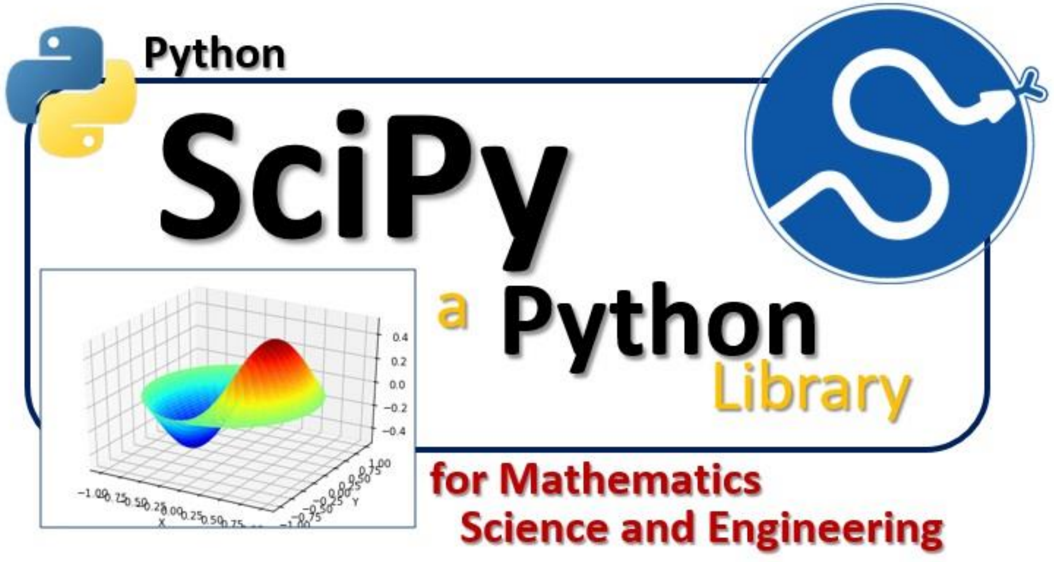


```
In [5]: a.ndim
Out[5]: 2

In [6]: a.dtype.name
Out[6]: 'int32'

In [7]: a.itemsize
Out[7]: 4

In [8]: a.size
Out[8]: 15

In [9]: type(a)
Out[9]: numpy.ndarray

In [10]:
```

https://numpy.org/
https://numpy.org/doc/stable/user/quickstart.html

SciPy

- The scipy package contains various toolboxes dedicated to common issues in scientific computing.
- It provides many user-friendly and efficient numerical routines, such as routines for numerical integration, interpolation, optimization, linear algebra, and statistics.



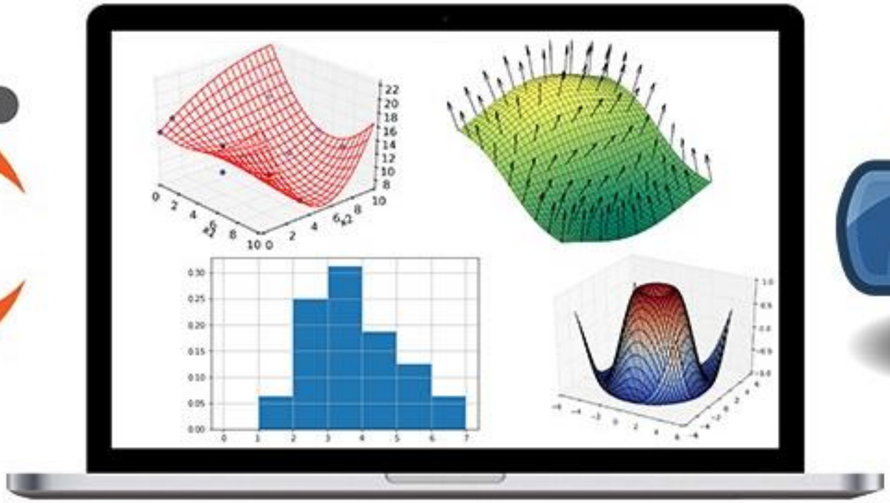
What can SciPy do

Physical and mathematical constants	scipy.constants
Vector quantization/Kmeans	scipy.cluster
Fourier transform	scipy.fft
Integration	scipy.integrate
Interpolation	scipy.interpolate
I/O	scipy.io
Linear algebra	scipy.linalg
N-dimensional image package	scipy.ndimage
Optimization	scipy.optimize
Signal processing	scipy.signal
Sparse matrices	scipy.sparse
Spatial data structures and algorithms	scipy.spatial
Any special mathematical functions	scipy.special
Statistics	scipy.stats

https://docs.scipy.org/doc/scipy/reference/

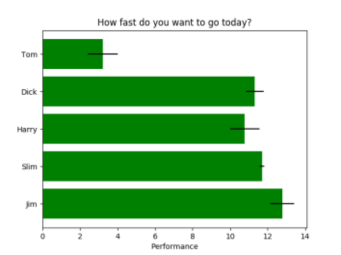
Matplotlib

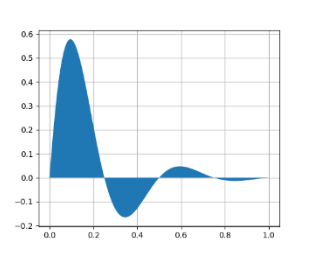
- Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations.
- You can develop publication quality plots; take full control of line styles, axes properties; explore functionality provided by third party packages, etc.

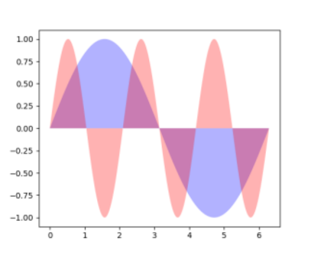


What can matplotlib do

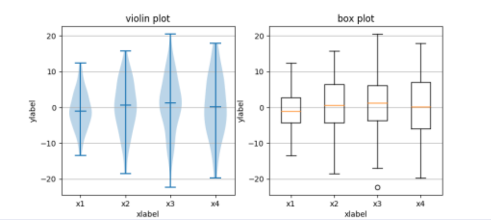
Lines, bars, and markers

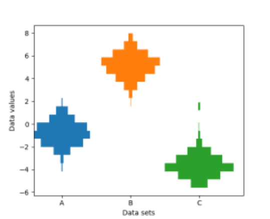




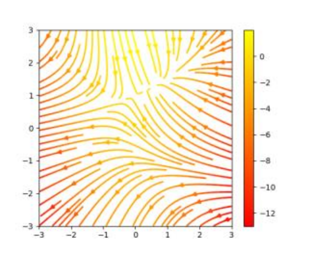


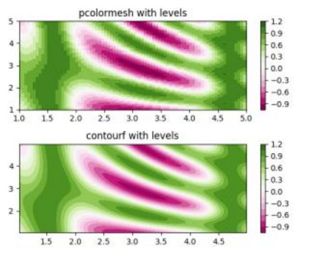
Statistical plots



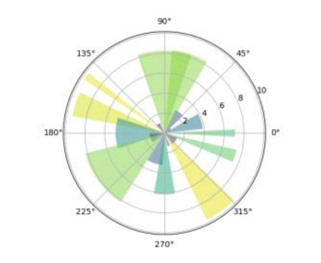


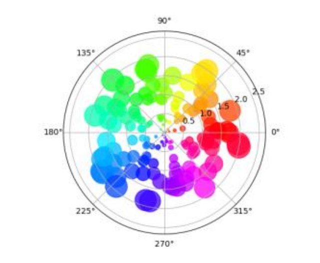
Images, contours, and fields





Pie and polar charts

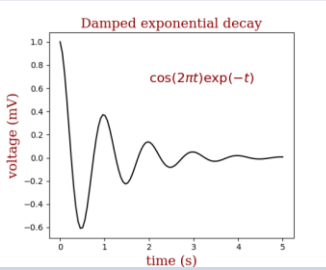


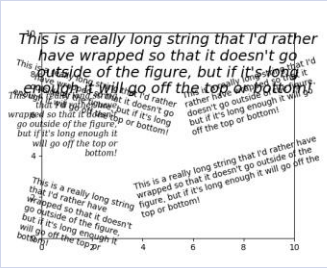


https://matplotlib.org/stable/gallery/index.html

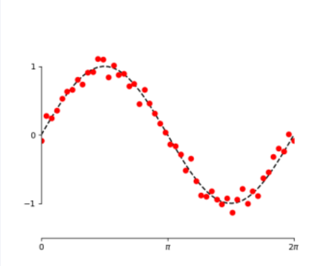
What can matplotlib do

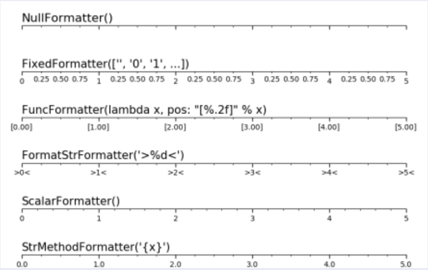
Text, labels, and annotations



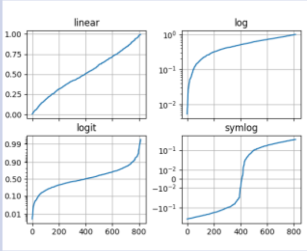


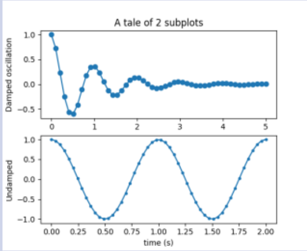
Ticks and spines



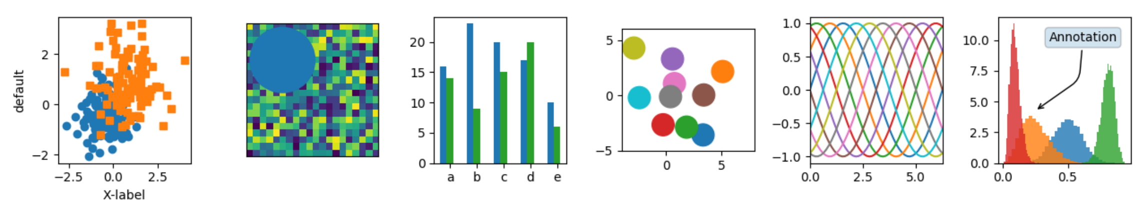


Subplots, axes, and figures



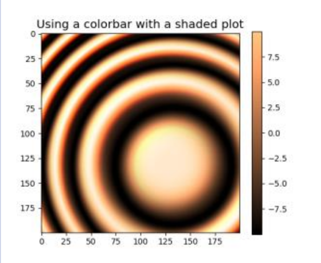
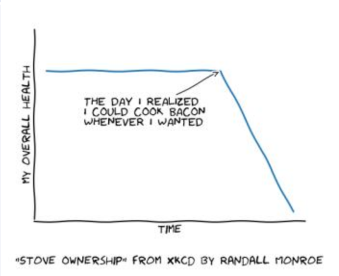
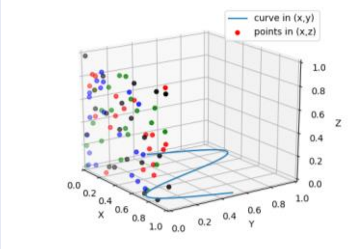
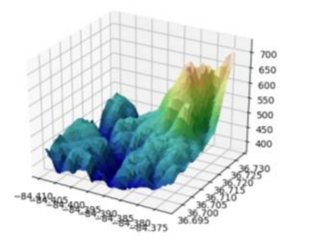
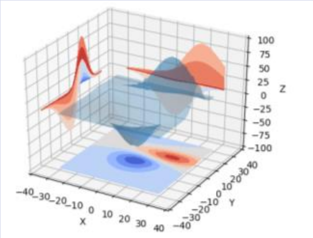
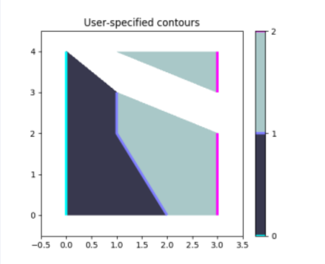
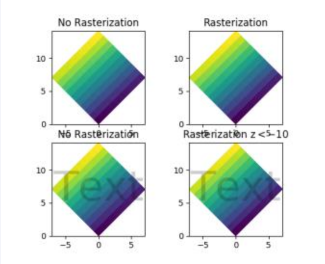


Style sheets



https://matplotlib.org/stable/gallery/index.html

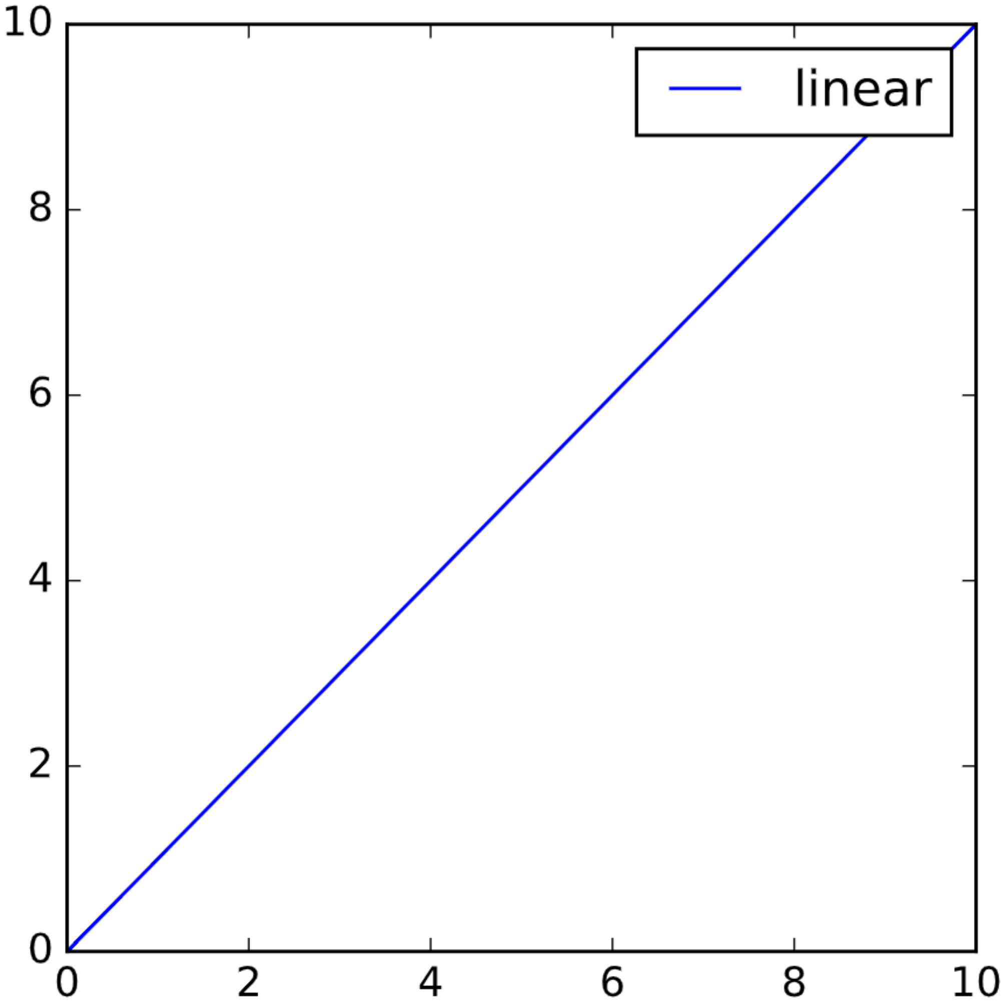
What can matplotlib do

Specialty plots		 	
Showcase		 	
mplot3d toolkit	 		
Miscellaneous	 	 	

https://matplotlib.org/stable/gallery/index.html

What can matplotlib do

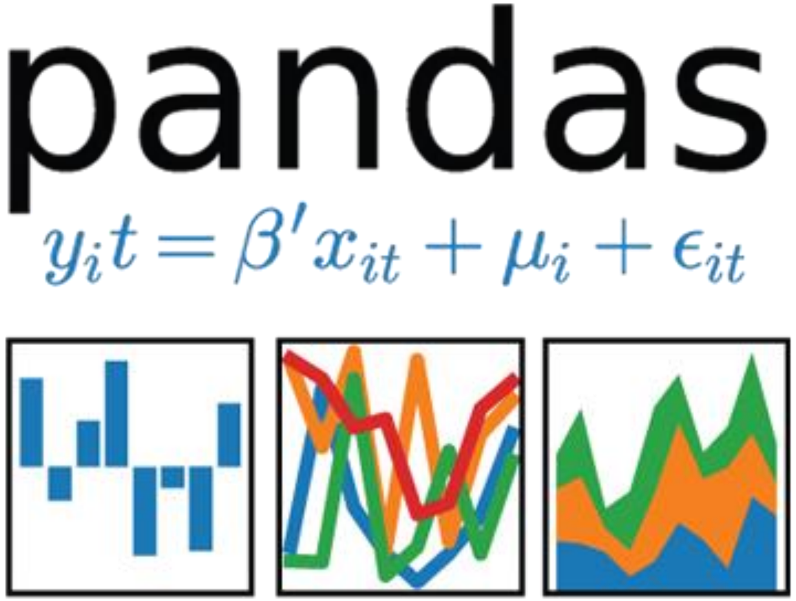
```
# Import the necessary packages and modules
import matplotlib.pyplot as plt
import numpy as np

# Prepare the data
x = np.linspace(0, 10, 100)

# Plot the data
plt.plot(x, x, label='linear’)

# Add a legend
plt.legend()

# Show the plot
plt.show()
```



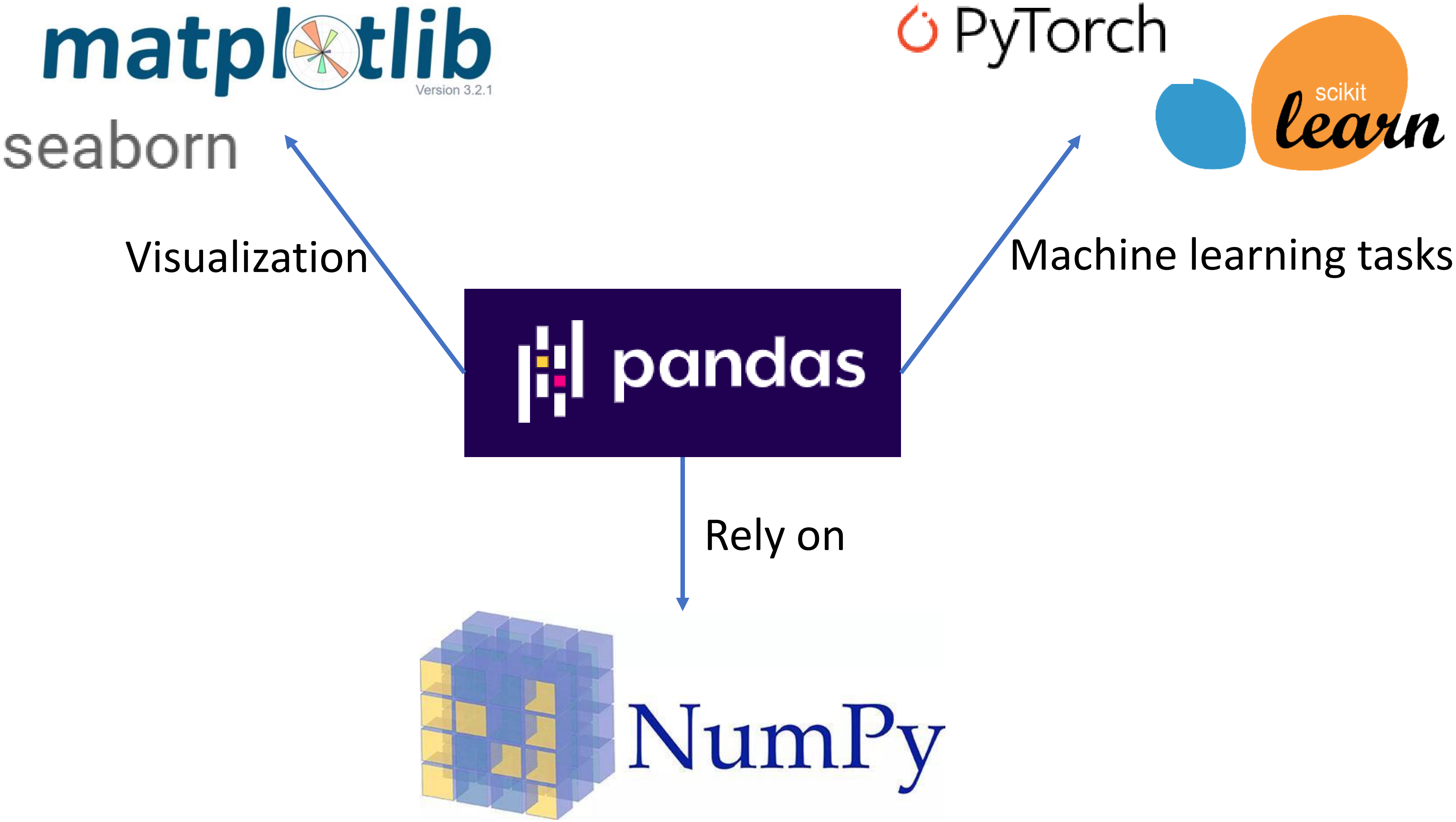
Let’s try this code!

Pandas

- Pandas is an open source library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.



Relations with other packages



matplotlib

Version 3.2.1

seaborn

PyTorch

scikit learn

Visualization

Machine learning tasks

pandas

Rely on

NumPy

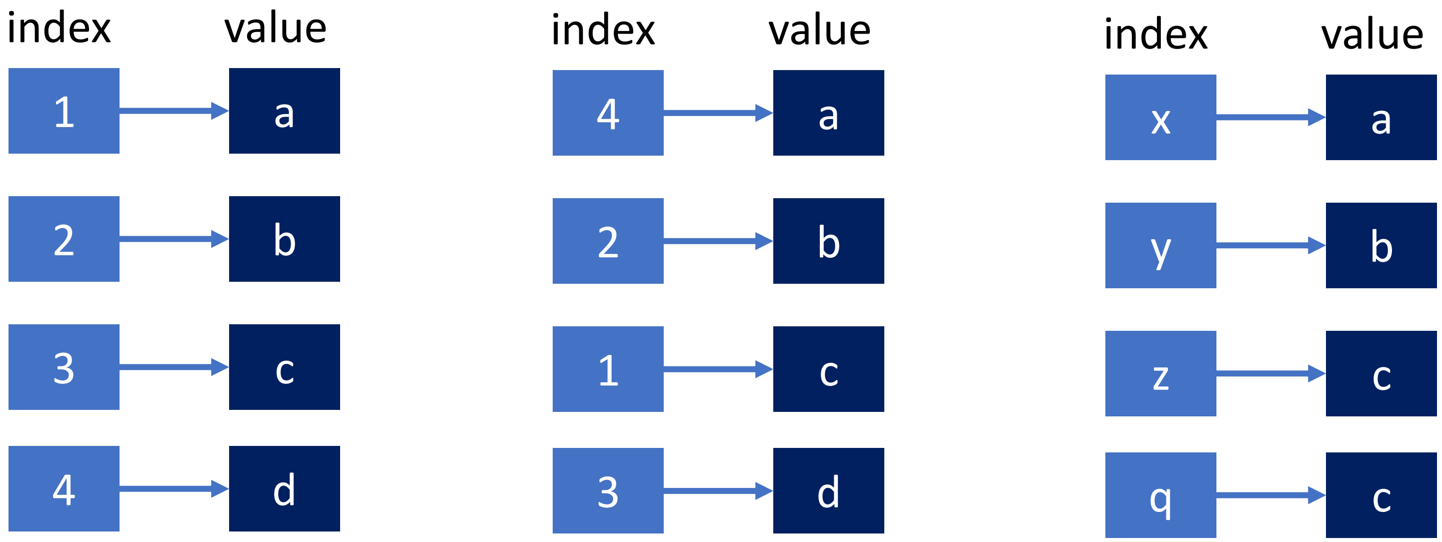
What can pandas do

I/O	pd.read_csv, pd.read_excel, …
General functions	pd.merge, pd.isna, pd.pivot, …
Series	Attributes (Series.shape), Conversion (Series.copy), Indexing (Series.loc), Binary operator (Series.mul), Function application&GroupBy (Series.aggregate), Selection (Series.where), Sorting (Series.sort_values), Plotting (Series.plot.bar), I/O (Series.to_csv), …
DataFrame	Similar with Series
Index	Numeric Index, CategoricalIndex, IntervalIndex, MultiIndex, DatetimeIndex, TimedeltaIndex, PeriodIndex
Scalars	Period, Timestamp, Interval
Window	Moving window, expanding window, Exponentially-weighted moving window
Resampling	Resampler.groups, Resampler.ffill, Resampler.count, …

https://pandas.pydata.org/pandas-docs/stable/getting_started/tutorials.html

Pandas: data structures--Series

- One-dimensional ndarray with axis labels (including time series).
- Index labels need not to be ordered
- Duplicates are possible



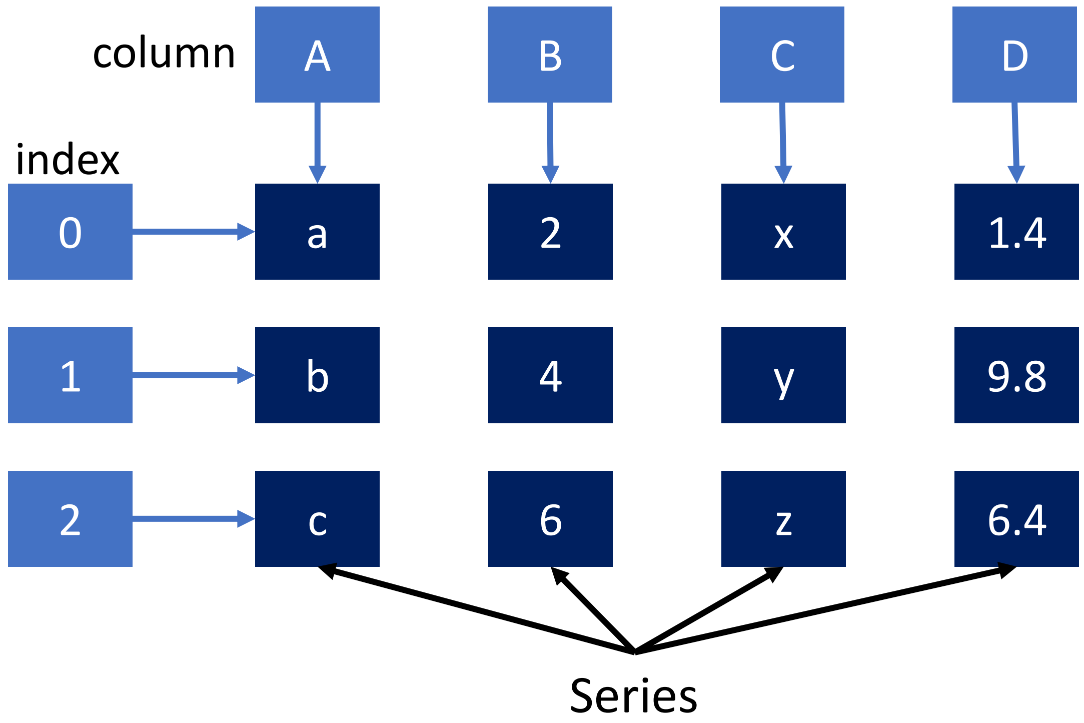
index	value
1	a
2	b
3	c
4	d

index	value
4	a
2	b
1	c
3	d

index	value
x	a
y	b
z	c
q	c

Pandas: data structures--DataFrame

- Two-dimensional, size-mutable, potentially heterogeneous tabular data.
- Data structure also contains labeled axes (rows and columns).
- Can be thought of as a dict-like container for Series objects.



index \ column	A	B	C	D
0	a	2	x	1.4
1	b	4	y	9.8
2	c	6	z	6.4

All data in a column must be the same kind

DataFrame: overview

Creation	Filtering	Update	Insertion	Deletion	Sorting	Statistics	Missing Value
Basic					Advanced		

DataFrame: creation

Dictionary

List

Row oriented

```
salesRowDict = [
{'Account': 'Jones LLC', 'Jan': 15,
'Feb': 22, 'Mar': 10},
…
]
dfRowDict = pd.DataFrame(salesRowDict)
```

```
salesRowList = [('Jones LLC', 15, 22, 10),…]
labels = ['Account', 'Jan', 'Feb', 'Mar']
dfRowList =
pd.DataFrame.from_records(salesRowList,columns=l
abels)
```

	Account	Jan	Feb	Mar
0	Jones LLC	15	22	10
1	Alpha Go	20	28	14
2	Blue Inc	10	9	13

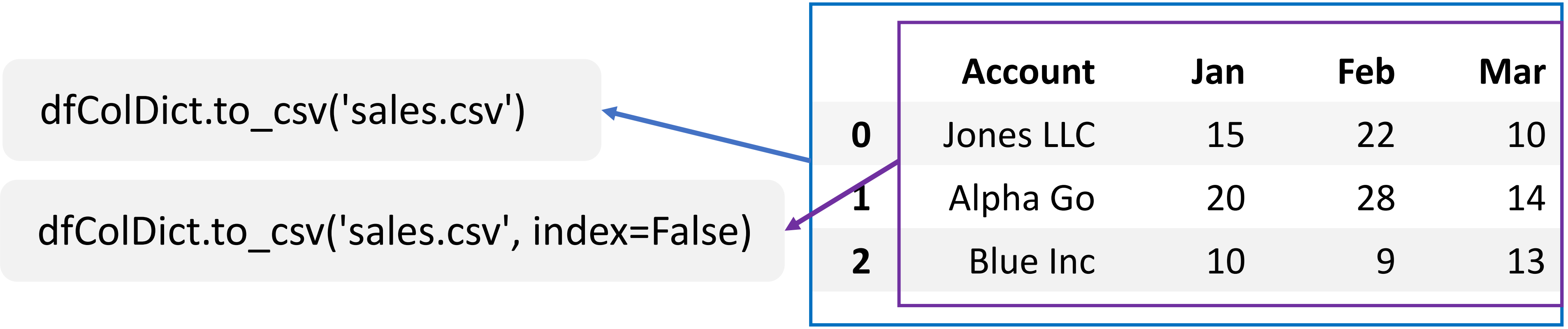
Column oriented

```
salesColDict = {
'Account': ['Jones LLC','Alpha Go','Blue
Inc'],
'Jan': [15, 22, 10],
…}
dfColDict =
pd.DataFrame.from_dict(salesColDict)
```

```
salesColList = [
   ('account', ['Jones LLC','Alpha Go','Blue Inc']),
   ('Jan',[15, 22, 10]),
   ('Feb',[20, 28, 14]),
   ('Mar',[10, 9, 13])]
dfColList = pd.DataFrame.from_items(salesColList)
```

Deprecated since version 0.23.0

DataFrame: save and load data



```
dfColDict.to_csv('sales.csv')
```

```
dfColDict.to_csv('sales.csv', index=False)
```

	Account	Jan	Feb	Mar
0	Jones LLC	15	22	10
1	Alpha Go	20	28	14
2	Blue Inc	10	9	13

```
df = pd.read_csv('sales.csv')
```

```
df = pd.read_excel('sales.xls')
```

```
df = pd.read_excel('sales.xlsx')
```

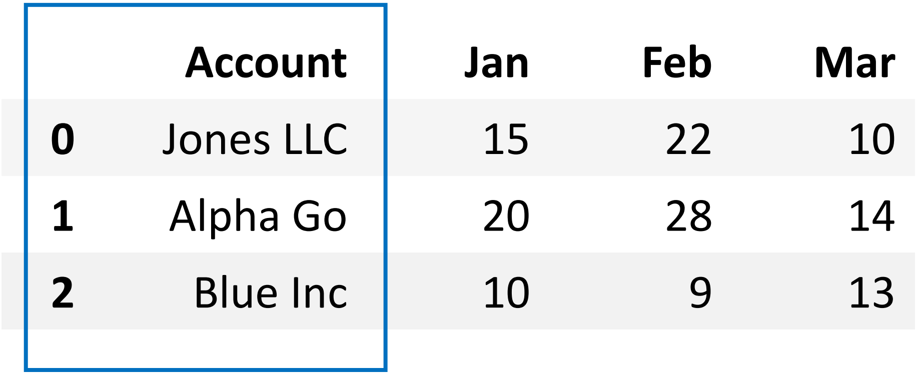
Support csv, xls, and xlsx format

Creation | Filtering | Update | Insertion | Deletion | Sorting | Statistics | Missing Value

DataFrame: filtering

Filter by column name(by default)

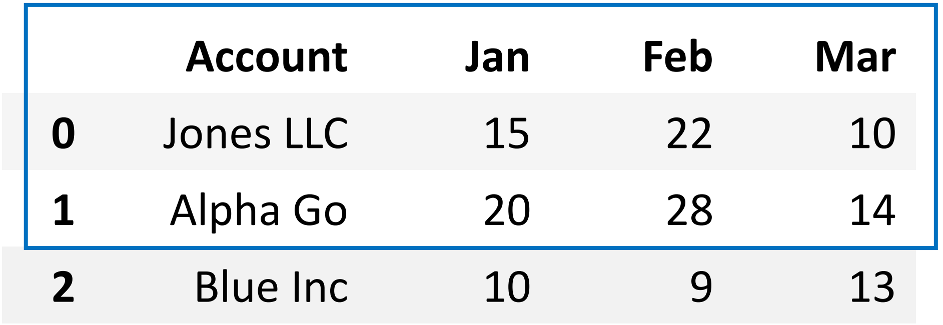
```
df['Account']
```



	Account	Jan	Feb	Mar
0	Jones LLC	15	22	10
1	Alpha Go	20	28	14
2	Blue Inc	10	9	13

Filter by index(when index is integer)

```
df[0:2]
```



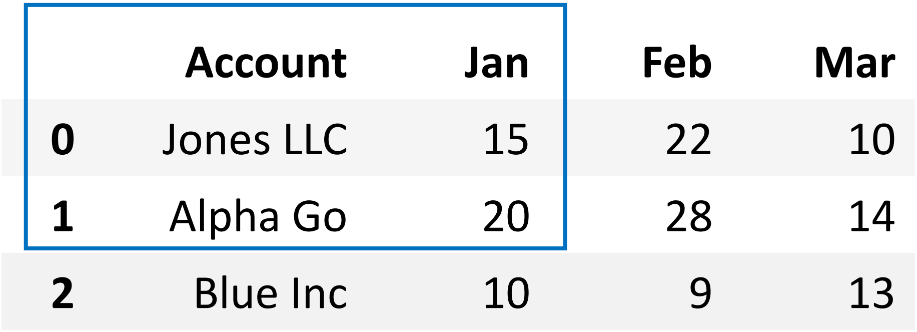
	Account	Jan	Feb	Mar
0	Jones LLC	15	22	10
1	Alpha Go	20	28	14
2	Blue Inc	10	9	13

DataFrame: filtering

Filter by iloc (index location)

```
df.iloc[0:2, 0:2]
```

left-closed,
right-open

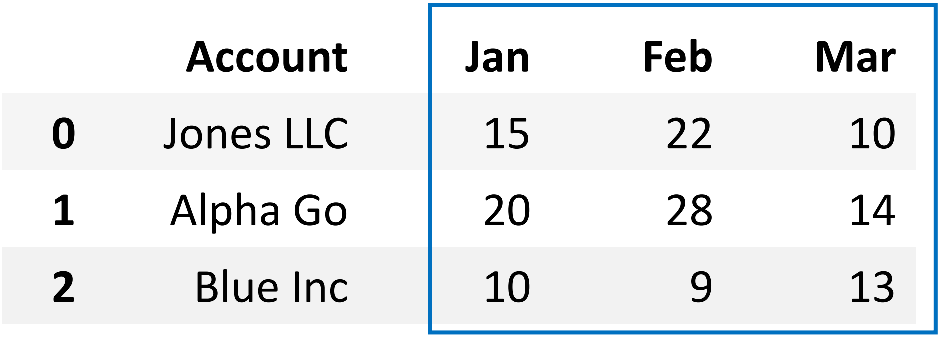


	Account	Jan	Feb	Mar
0	Jones LLC	15	22	10
1	Alpha Go	20	28	14
2	Blue Inc	10	9	13

Filter by loc

left-closed,
right-closed

```
df.loc[0:2, 'Jan':'Mar']
```



	Account	Jan	Feb	Mar
0	Jones LLC	15	22	10
1	Alpha Go	20	28	14
2	Blue Inc	10	9	13

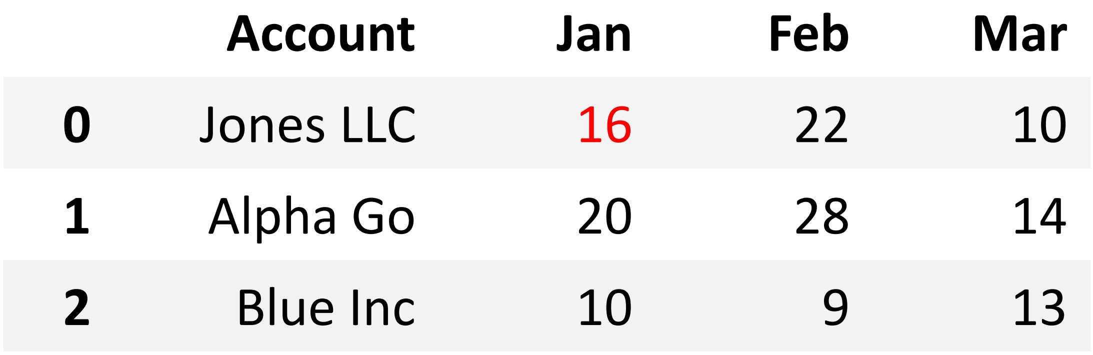
iloc和loc的区别： https://blog.csdn.net/Leon_Kbl/article/details/97492966

DataFrame: update

All the filtering techniques can be used.

```
df.iloc[0,1]=16
```

```
df.loc[0, 'Jan'] = 16
```



	Account	Jan	Feb	Mar
0	Jones LLC	16	22	10
1	Alpha Go	20	28	14
2	Blue Inc	10	9	13

DataFrame: update

Update a single value

```
df.iloc[0,1]=16
```

```
df.loc[0, 'Jan'] = 16
```

	Account	Jan	Feb	Mar
0	Jones LLC	16	22	10
1	Alpha Go	20	28	14
2	Blue Inc	10	9	13

Update a single column

```
df['Jan'] = 20
```

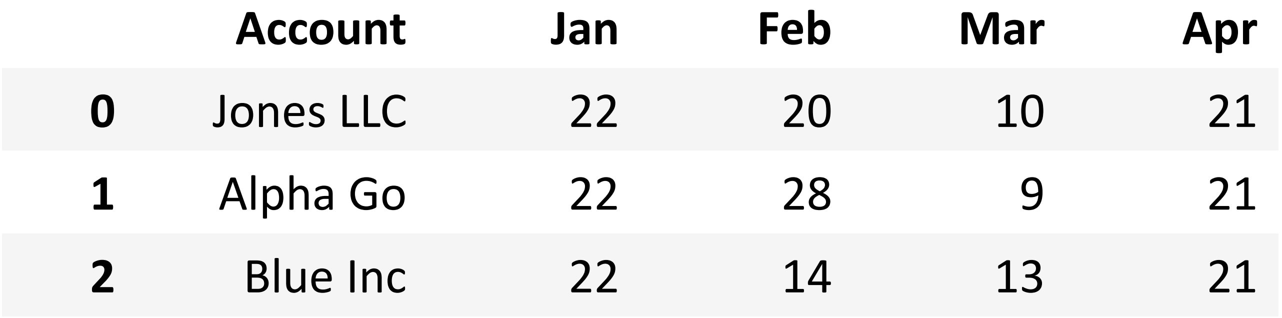
	Account	Jan	Feb	Mar
0	Jones LLC	20	22	10
1	Alpha Go	20	28	14
2	Blue Inc	20	9	13

Creation | Filtering | Update | Insertion | Deletion | Sorting | Statistics | Missing Value

DataFrame: insertion

Insert a column

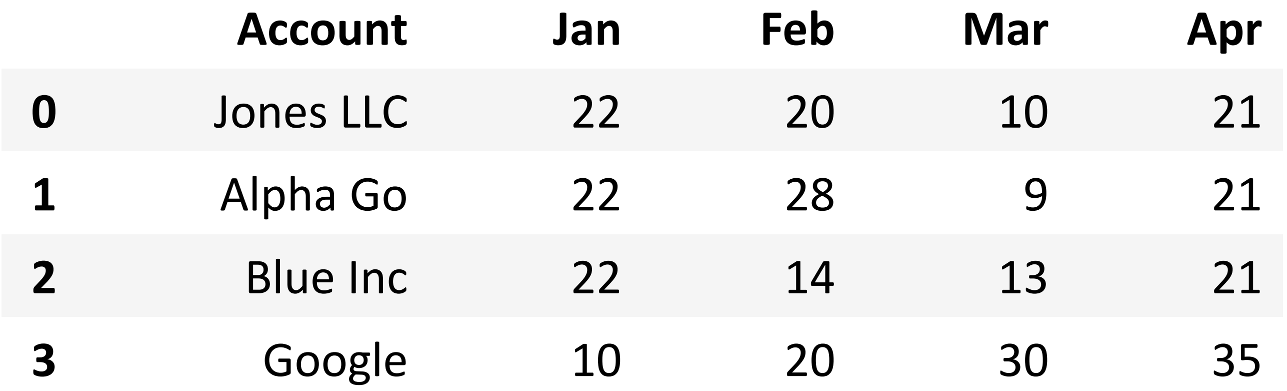
```
df['Apr'] = 21
```



	Account	Jan	Feb	Mar	Apr
0	Jones LLC	22	20	10	21
1	Alpha Go	22	28	9	21
2	Blue Inc	22	14	13	21

Insert a row

```
df.loc[3] = \
('Google', 10, 20, 30, 35)
```



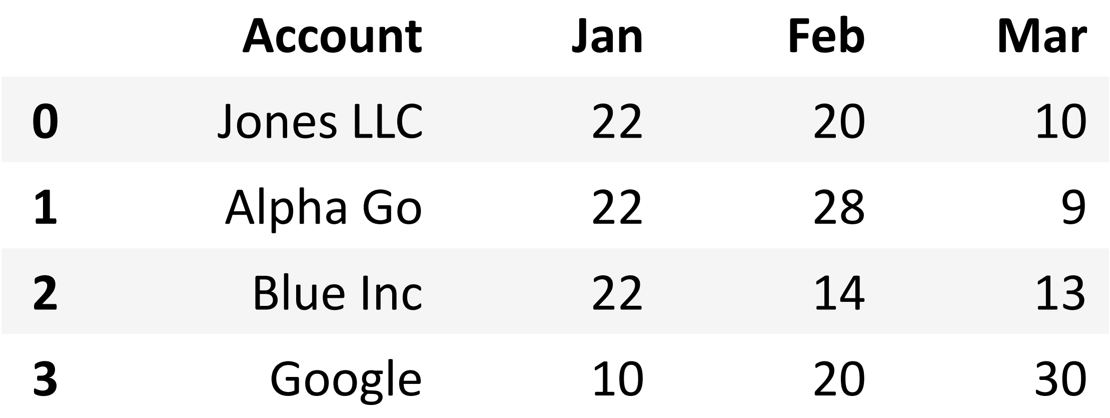
	Account	Jan	Feb	Mar	Apr
0	Jones LLC	22	20	10	21
1	Alpha Go	22	28	9	21
2	Blue Inc	22	14	13	21
3	Google	10	20	30	35

DataFrame: deletion

Delete columns

```
df1 = df.drop(columns=['Apr’],
inplace=False)
```

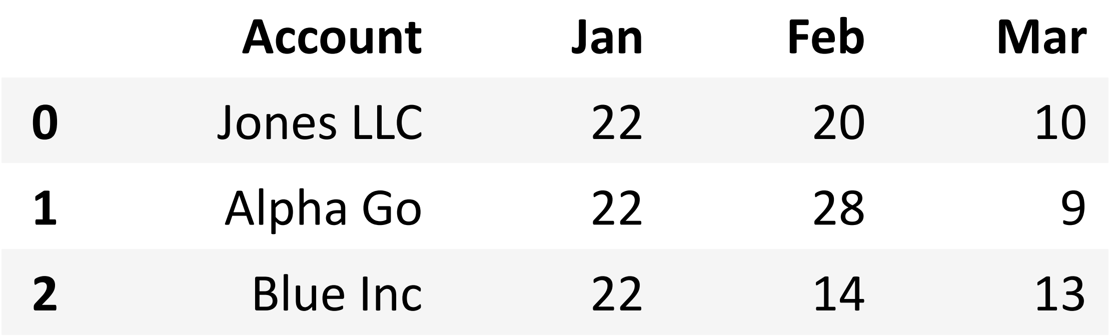
Set inplace=True if you want to modify the table directly.



	Account	Jan	Feb	Mar
0	Jones LLC	22	20	10
1	Alpha Go	22	28	9
2	Blue Inc	22	14	13
3	Google	10	20	30

Delete rows

```
df1.drop(index=3, inplace=True)
```



	Account	Jan	Feb	Mar
0	Jones LLC	22	20	10
1	Alpha Go	22	28	9
2	Blue Inc	22	14	13

DataFrame: sorting

Sort by index

```
df.sort_index(axis=1,
ascending=False)
```

	Mar	Jan	Feb	Account
0	10	22	20	Jones LLC
1	9	22	28	Alpha Go
2	13	22	14	Blue Inc

Sort by column value

```
df.sort_values('Feb',
ascending=True)
```

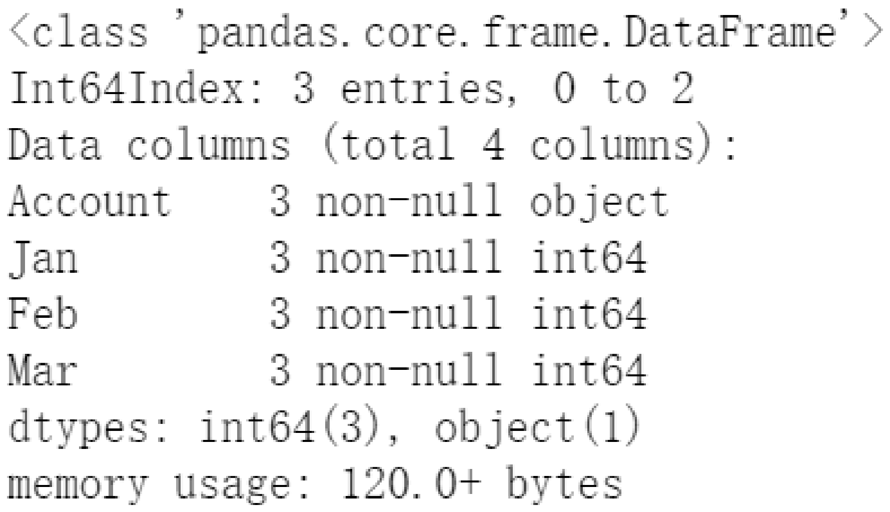
	Account	Jan	Feb	Mar
2	Blue Inc	22	14	13
0	Jones LLC	22	20	10
1	Alpha Go	22	28	9

DataFrame: statistics

Basic information

df.shape
df.index
df.columns

df.info()
df.count()

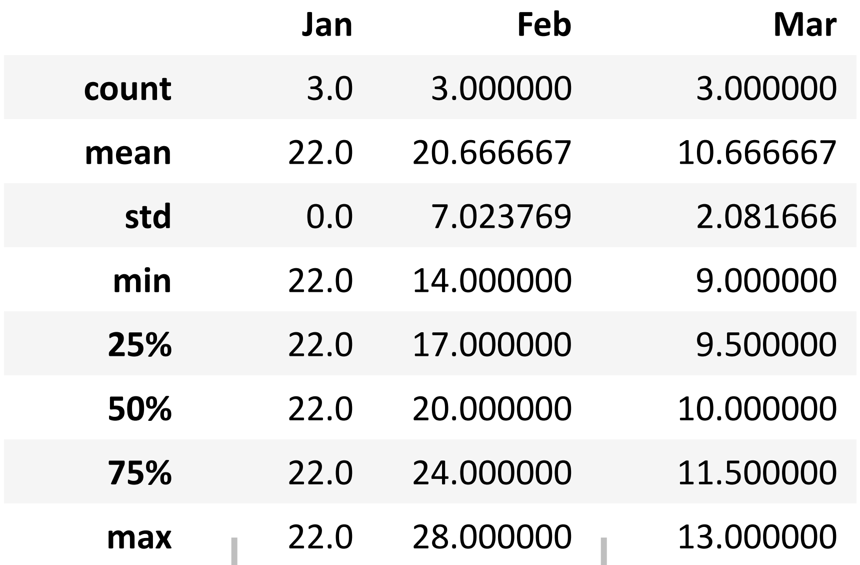


```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3 entries, 0 to 2
Data columns (total 4 columns):
Account     3 non-null object
Jan         3 non-null int64
Feb         3 non-null int64
Mar         3 non-null int64
dtypes: int64(3), object(1)
memory usage: 120.0+ bytes
```

Summary

df.sum()
df.cumsum()
df.min()
df.max()

df.mean()
df.median()
df.describe()



	Jan	Feb	Mar
count	3.0	3.000000	3.000000
mean	22.0	20.666667	10.666667
std	0.0	7.023769	2.081666
min	22.0	14.000000	9.000000
25%	22.0	17.000000	9.500000
50%	22.0	20.000000	10.000000
75%	22.0	24.000000	11.500000
max	22.0	28.000000	13.000000

DataFrame: missing value

Find and replace missing values

df.isnull()

df.notnull()

	Account	Jan	Feb	Mar
0	False	True	False	False
1	False	True	False	False
2	False	False	False	False

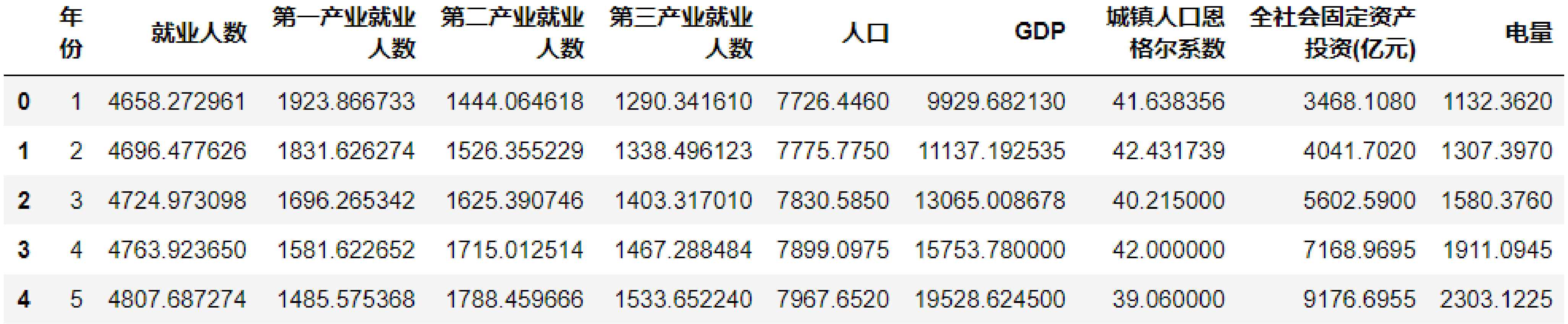
df.fillna(0)

	Account	Jan	Feb	Mar
0	Jones LLC	0.0	20	10
1	Alpha Go	0.0	28	9
2	Blue Inc	22.0	14	13

Example for long-term load forecasting

```
# We first import packages
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.metrics import mean_squared_error
```

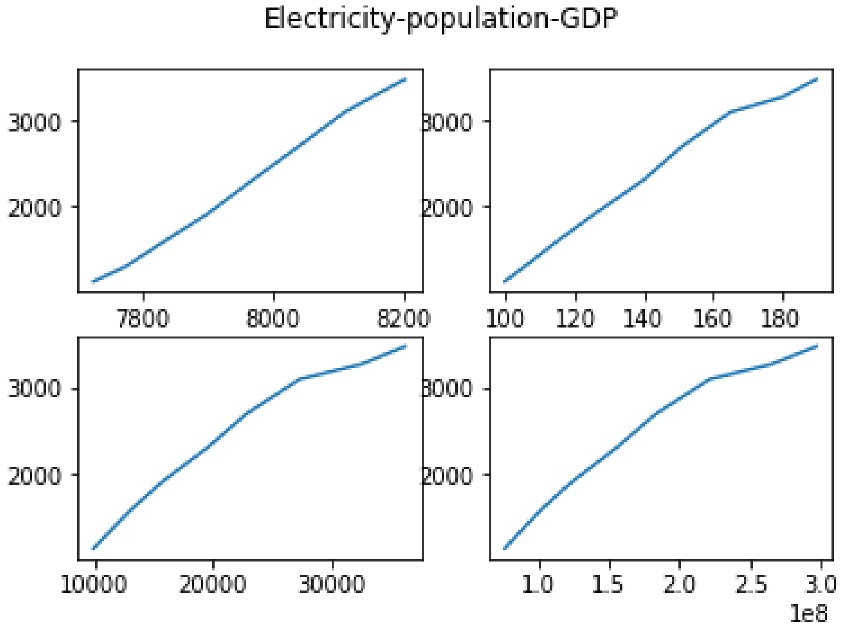
```
# We then read the data
data = pd.read_excel('data.xls')
data.head()
```



	年份	就业人数	第一产业就业人数	第二产业就业人数	第三产业就业人数	人口	GDP	城镇人口恩格尔系数	全社会固定资产投资(亿元)	电量
0	1	4658.272961	1923.866733	1444.064618	1290.341610	7726.4460	9929.682130	41.638356	3468.1080	1132.3620
1	2	4696.477626	1831.626274	1526.355229	1338.496123	7775.7750	11137.192535	42.431739	4041.7020	1307.3970
2	3	4724.973098	1696.265342	1625.390746	1403.317010	7830.5850	13065.008678	40.215000	5602.5900	1580.3760
3	4	4763.923650	1581.622652	1715.012514	1467.288484	7899.0975	15753.780000	42.000000	7168.9695	1911.0945
4	5	4807.687274	1485.575368	1788.459666	1533.652240	7967.6520	19528.624500	39.060000	9176.6955	2303.1225

Example for long-term load forecasting

```
# plot some figures
year = data['年份']
num_year = year.size
gdp = data['GDP']
population = data['人口']
electricity = data['电量']

fig, axs = plt.subplots(2,2)
fig.suptitle('Electricity-population-GDP')
axs[0,0].plot(population, electricity)
axs[1,0].plot(gdp, electricity)
axs[0,1].plot(gdp**0.5, electricity)
axs[1,1].plot(population*gdp, electricity)
```



Example for long-term load forecasting

```
# we choose population and gdp**0.5 as the regressor
# we also add all ones as the regressor
one = pd.DataFrame(data=np.ones(num_year))
X = pd.concat([population, gdp**0.5, one], axis=1)
y = electricity
# we split the datasets into the training and the testing sets
X_train = X.iloc[2:-2, :]
y_train = y.iloc[2:-2]
X_test = X.iloc[-1, :]
y_test = y.iloc[-1]
X_test.head()
```

Example for long-term load forecasting

```
# create linear regression object
regr = linear_model.LinearRegression()
# Train the model using the training sets
regr.fit(X_train, y_train)
# Make predictions using the testing set
y_pred = regr.predict(X_test.to_numpy().reshape(1, -1))
```

```
# The mean squared error
print(y_pred)
print(y_test)
err = (y_pred[0] - y_test) / y_test
print('Mean squared error: %.4f'% err)
```

Python Conditions and Loops

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

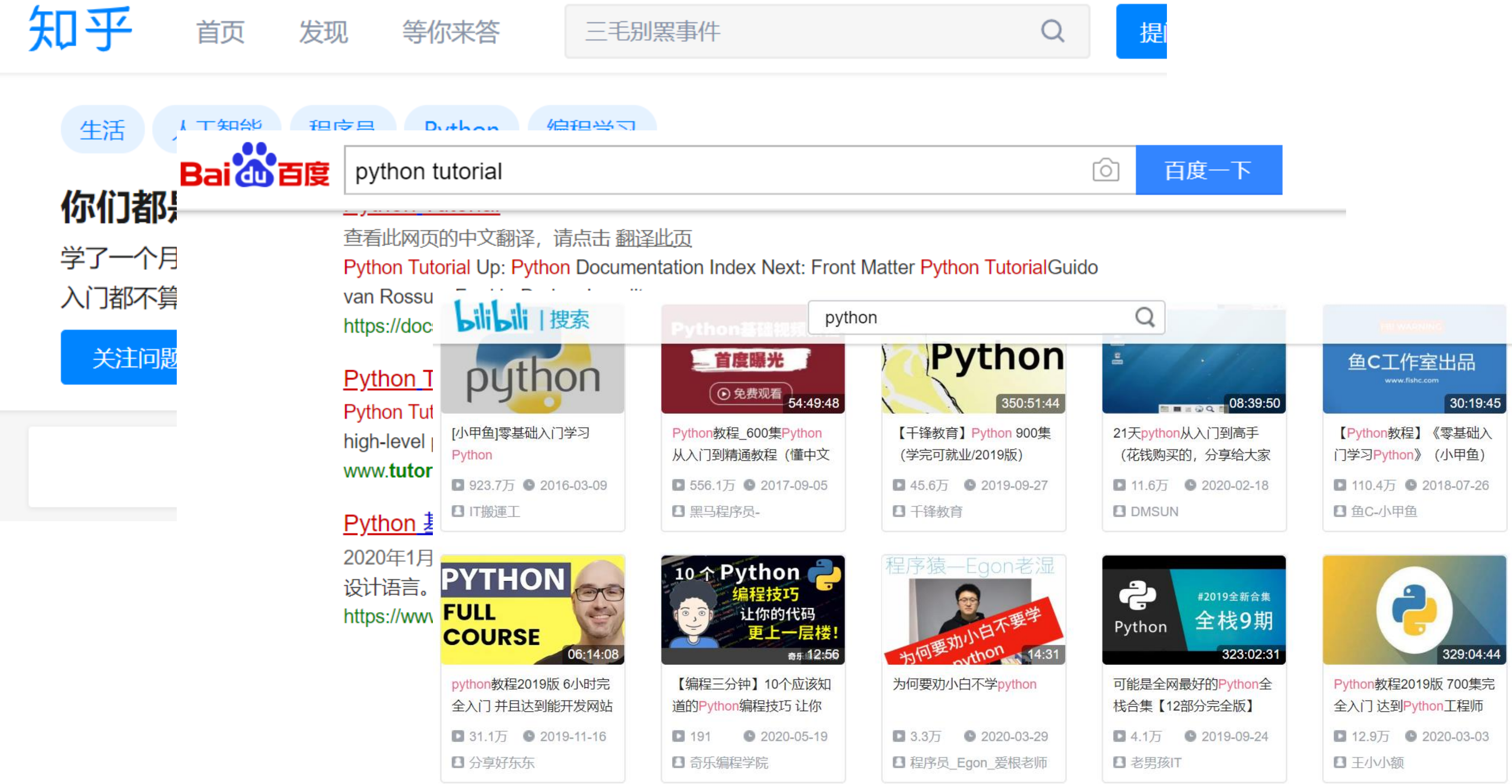
```
i = 1
while i < 6:
  print(i)
  i += 1
```

```
fruits =
["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

Things you may interested in

- List and tuples
- object-oriented programming and functional programming
- scikit-learn
- Pytorch
- …

Make good use of online tutorials!



你们都

学了一个月

入门都不算

python tutorial

查看此网页的中文翻译，请点击 翻译此页

Python Tutorial Up: Python Documentation Index Next: Front Matter Python TutorialGuido van Rossu

https://doc

Python T

Python Tut

high-level

www.tutor

Python 基

2020年1月

设计语言。

https://www

python

[小甲鱼]零基础入门学习Python
923.7万 2016-03-09
IT搬運工

Python教程_600集Python从入门到精通教程（懂中文
556.1万 2017-09-05
黑马程序员-

【千锋教育】Python 900集（学完可就业/2019版）
45.6万 2019-09-27
千锋教育

21天python从入门到高手（花钱购买的，分享给大家
11.6万 2020-02-18
DMSUN

【Python教程】《零基础入门学习Python》（小甲鱼）
110.4万 2018-07-26
鱼C-小甲鱼

python教程2019版 6小时完全入门 并且达到能开发网站
31.1万 2019-11-16
分享好东东

【编程三分钟】10个应该知道的Python编程技巧 让你
191 2020-05-19
奇乐编程学院

为何要劝小白不学python
3.3万 2020-03-29
程序员_Egon_爱根老师

可能是全网最好的Python全栈合集【12部分完全版】
4.1万 2019-09-24
老男孩IT

Python教程2019版 700集完全入门 达到Python工程师
12.9万 2020-03-03
王小小额

Q&A