TAIJI LABORATORY
FOR GRAVITATIONAL WAVE UNIVERSE

ICTP-AP
International Centre
for Theoretical Physics Asia-Pacific
国际理论物理中心-亚太地区

University of Chinese Academy of Sciences

引力波数据探索：编程与分析实战训练营

第 2 部分 基于 Python 的数据分析基础

数据分析实训之 Pandas

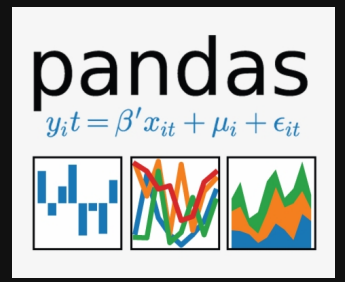
主讲老师：王赫
ICTP-AP, UCAS

2023/12/03

Taiji Program in Space
Gravitational wave physics
空间太极计划

数据分析实训之 Pandas

- Pandas 中的数据结构：Series 和 DataFrame
- DataFrame 的增删改查
- DataFrame 的组合与聚合&透视表
- 实例：GW Catalog 数据分析案例
- 实例：股票数据分析案例 (optional)



“ Talk is cheap. Show me the code. ”

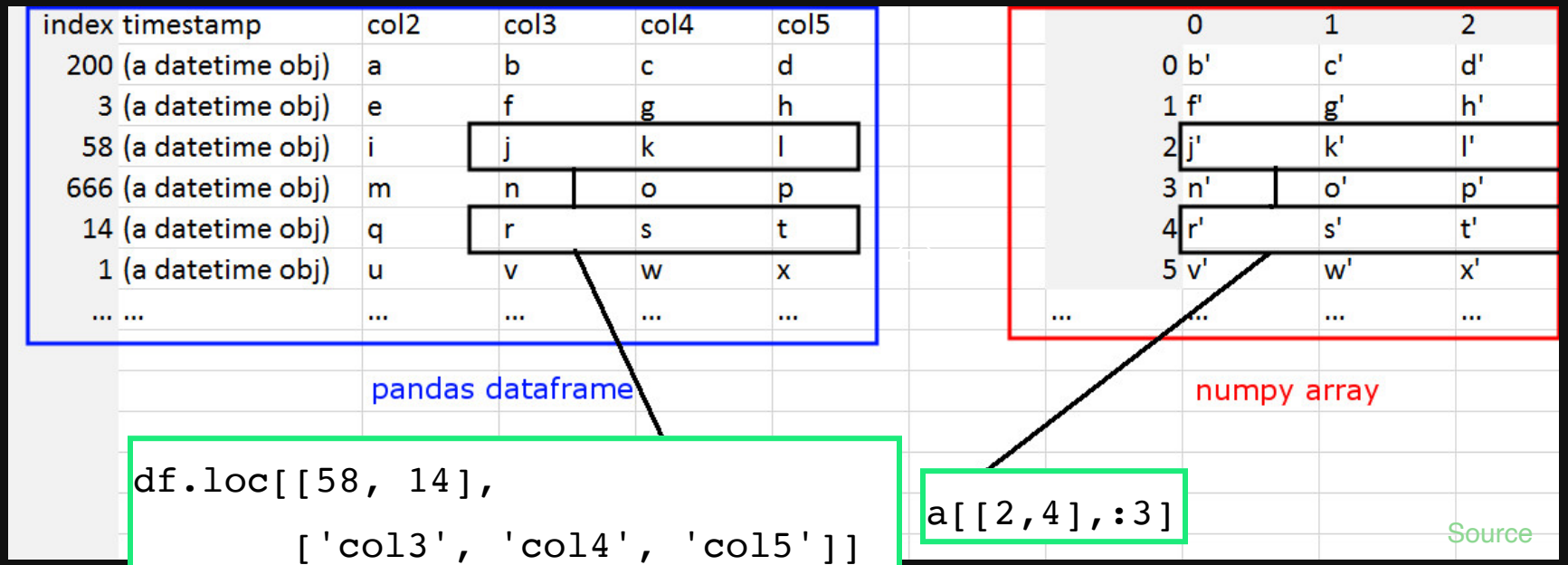
Linus Torvalds

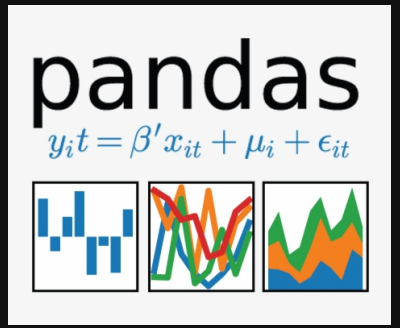
Ndarray → Series/DataFrame (Pandas)

- Pandas 是基于 Numpy 的强大的分析结构化数据的工具集
  - Pandas 官方网址：https://pandas.pydata.org
  - Pandas 官方中文文档：https://www.pypandas.cn
- 从 Numpy 的 **Ndarray** 到 Pandas 的 **Series / DataFrame**
  - (Numpy) *1-dimensional array* ⇔ *Series* (Pandas)
  - (Numpy) *2-dimensional array* ⇔ *DataFrame* (Pandas)

NumPy

https://numpy.org



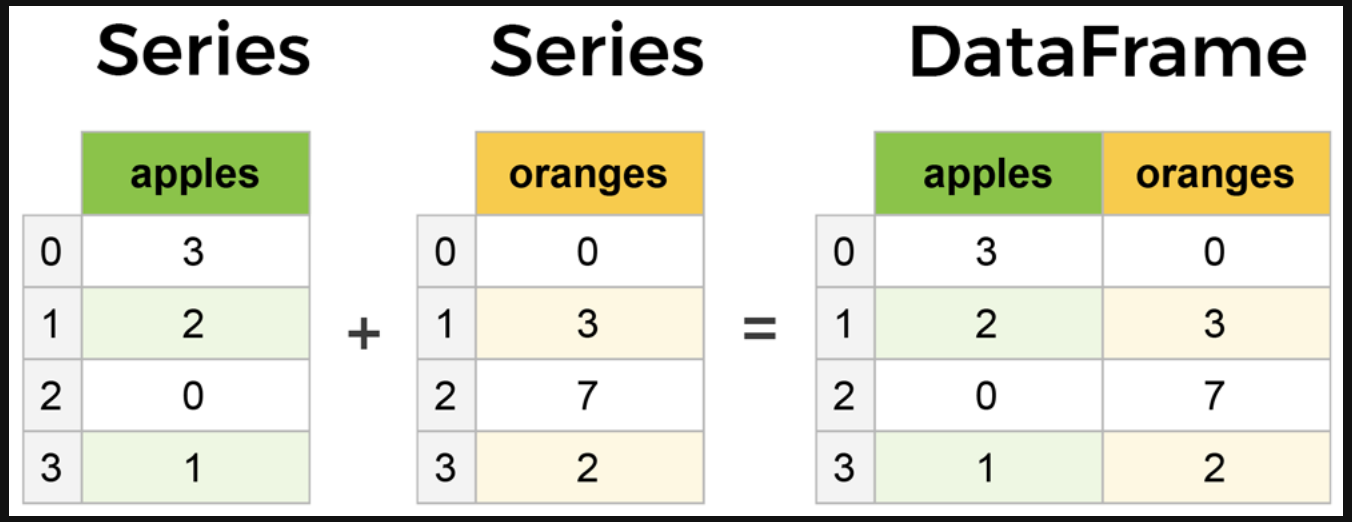


Series vs DataFrame

- **Series** 是一种类似于*一维数组*的对象，是由一组数据 (各种 NumPy 数据类型) 以及一组与之相关的数据标签(即索引)组成。仅由一组数据也可产生简单的 Series 对象。
- **DataFrame** 是 Pandas 中的一个*表格型*的数据结构，包含有一组有序的列，每列可以是不同的值类型(数值、字符串、布尔型等)，DataFrame 即有行索引也有列索引，可以被看做是由 Series 组成的字典。

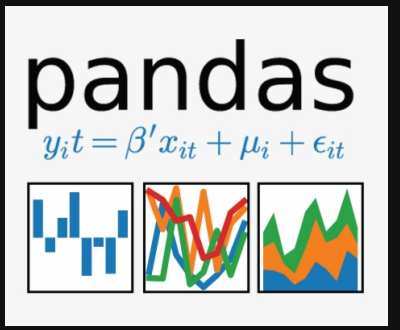
NumPy

https://numpy.org



	Series			Series			DataFrame	
	apples			oranges			apples	oranges
0	3	+	0	0	=	0	3	0
1	2		1	3		1	2	3
2	0		2	7		2	0	7
3	1		3	2		3	1	2

Source



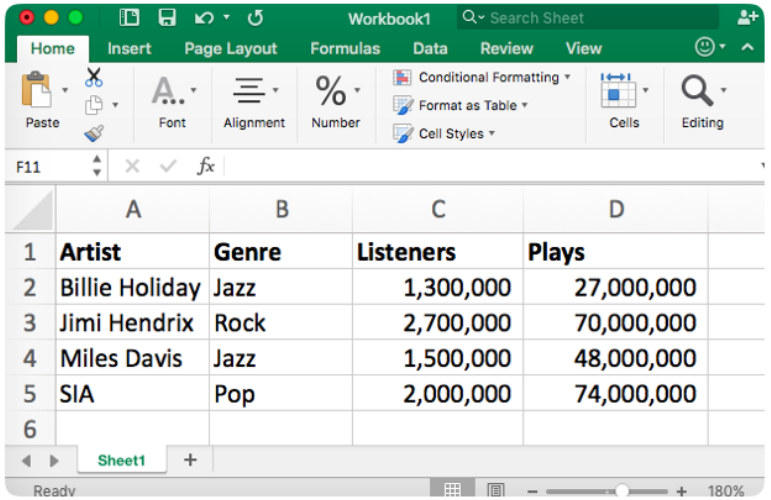
Excel vs DataFrame

- **Series** 是一种类似于*一维数组*的对象，是由一组数据 (各种 NumPy 数据类型) 以及一组与之相关的数据标签(即索引)组成。仅由一组数据也可产生简单的 Series 对象。
- **DataFrame** 是 Pandas 中的一个*表格型*的数据结构，包含有一组有序的列，每列可以是不同的值类型(数值、字符串、布尔型等)，DataFrame 即有行索引也有列索引，可以被看做是由 Series 组成的字典。

NumPy

https://numpy.org

music.csv



	A	B	C	D
1	Artist	Genre	Listeners	Plays
2	Billie Holiday	Jazz	1,300,000	27,000,000
3	Jimi Hendrix	Rock	2,700,000	70,000,000
4	Miles Davis	Jazz	1,500,000	48,000,000
5	SIA	Pop	2,000,000	74,000,000
6				

pandas.read_csv('music.csv')

	Artist	Genre	Listeners	Plays
0	Billie Holiday	Jazz	1,300,000	27,000,000
1	Jimi Hendrix	Rock	2,700,000	70,000,000
2	Miles Davis	Jazz	1,500,000	48,000,000
3	SIA	Pop	2,000,000	74,000,000

pandas

$y_i t = \beta' x_{it} + \mu_i + \epsilon_{it}$

Source

DataFrame 的“增删改查”

- DataFrame 构建与初始化：1. 按列构建 2. 按行构建

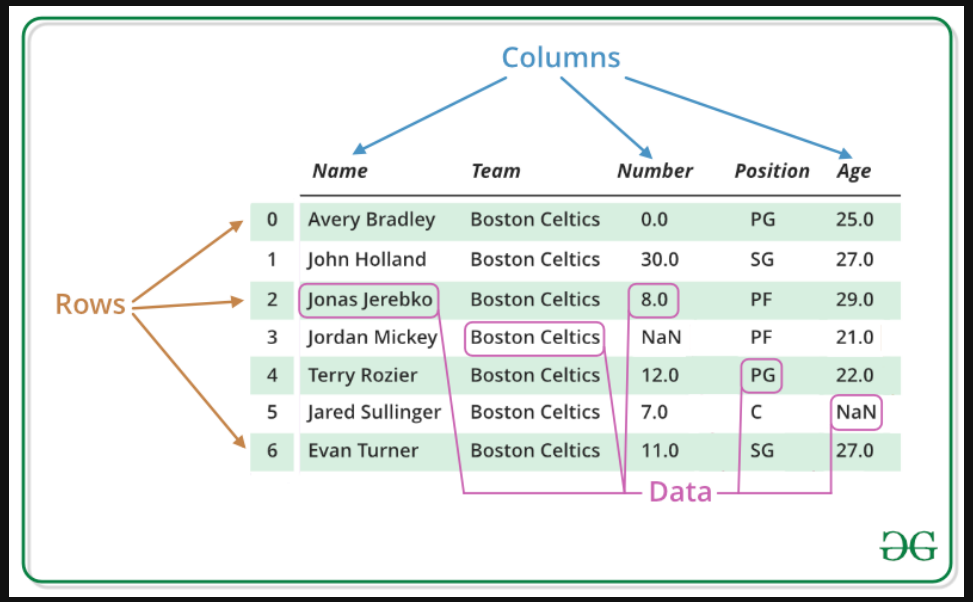
	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
df = pd.DataFrame(
          {"a" : [4 ,5, 6],
           "b" : [7, 8, 9],
           "c" : [10, 11, 12]},
         index = [1, 2, 3])
```

Specify values for each column.

```
df = pd.DataFrame(
     [[4, 7, 10],
      [5, 8, 11],
      [6, 9, 12]],
     index=[1, 2, 3],
     columns=['a', 'b', 'c'])
```

Specify values for each row.



Source

DataFrame 的“增删改查”

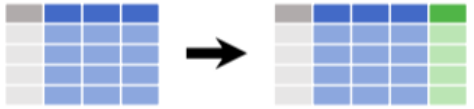
- DataFrame 的构建与初始化：1. 按列构建 2. 按行构建

		a	b	c
n	v			
d	1	4	7	10
	2	5	8	11
e	2	6	9	12

```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
index = pd.MultiIndex.from_tuples(
        [('d',1),('d',2),('e',2)],
          names=['n','v']))
```

Create DataFrame with a MultiIndex

Make New Variables



```
df=df.assign(Area=lambda df: df.Length*df.Height)
```

Compute and append one or more new columns.

```
df['Volume'] = df.Length*df.Height*df.Depth
```

Add single column.

```
pd.qcut(df.col, n, labels=False)
```

Bin column into n buckets.

Creating Pandas DataFrames from Python Lists and Dictionaries

Row Oriented

Dictionary

```
sales = [{'account': 'Jones LLC', 'Jan': 150, 'Feb': 200, 'Mar': 140},
         {'account': 'Alpha Co', 'Jan': 200, 'Feb': 210, 'Mar': 215},
         {'account': 'Blue Inc', 'Jan': 50, 'Feb': 90, 'Mar': 95 }]
df = pd.DataFrame(sales)
```

default

List

```
sales = [('Jones LLC', 150, 200, 50),
         ('Alpha Co', 200, 210, 90),
         ('Blue Inc', 140, 215, 95)]
labels = ['account', 'Jan', 'Feb', 'Mar']
df = pd.DataFrame.from_records(sales, columns=labels)
```

from_records

	account	Jan	Feb	Mar
0	Jones LLC	150	200	140
1	Alpha Co	200	210	215
2	Blue Inc	50	90	95

Column Oriented

```
sales = {'account': ['Jones LLC', 'Alpha Co', 'Blue Inc'],
         'Jan': [150, 200, 50],
         'Feb': [200, 210, 90],
         'Mar': [140, 215, 95]}
df = pd.DataFrame.from_dict(sales)
```

from_dict

```
sales = [('account', ['Jones LLC', 'Alpha Co', 'Blue Inc']),
         ('Jan', [150, 200, 50]),
         ('Feb', [200, 210, 90]),
         ('Mar', [140, 215, 95]) ]
df = pd.DataFrame.from_items(sales)
```

from_items

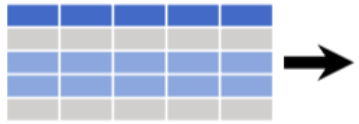
When using a dictionary, column order is not preserved.
Explicitly order them:
`df = df[['account', 'Jan', 'Feb', 'Mar']]`

Practical Business Python - pbpython.com

DataFrame 的“增删改查”

- DataFrame 的索引与切片：

Subset Observations (Rows)



`df[df.Length > 7]`
Extract rows that meet logical criteria.

`df.drop_duplicates()`
Remove duplicate rows (only considers columns).

`df.head(n)`
Select first n rows.

`df.tail(n)`
Select last n rows.

`df.sample(frac=0.5)`
Randomly select fraction of rows.

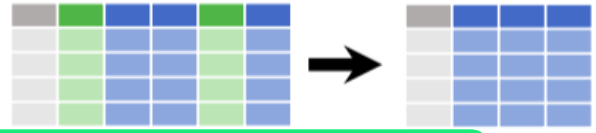
`df.sample(n=10)`
Randomly select n rows.

`df.iloc[10:20]`
Select rows by position.

`df.nlargest(n, 'value')`
Select and order top n entries.

`df.nsmallest(n, 'value')`
Select and order bottom n entries.

Subset Variables (Columns)



`df[['width','length','species']]`
Select multiple columns with specific names.

`df['width']` *or* `df.width`
Select single column with specific name.

⇒ DataFrame

⇒ Series

`df.filter(regex='regex')`
Select columns whose name matches regular expression *regex*.

`df.drop(['length','species'], axis=1)`

Just like ndarray ...

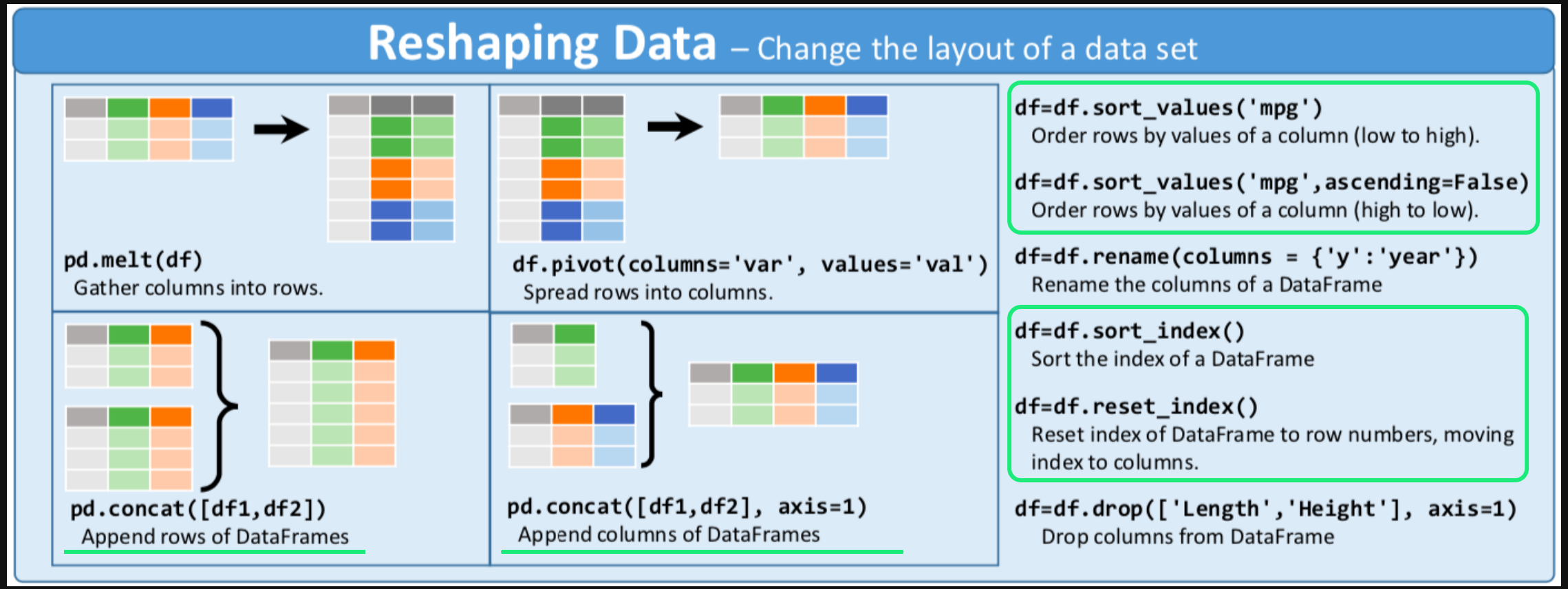
`df.loc[:,'x2':'x4']`
Select all columns between x2 and x4 (inclusive).

`df.iloc[:,[1,2,5]]`
Select columns in positions 1, 2 and 5 (first column is 0).

`df.loc[df['a'] > 10, ['a','c']]`
Select rows meeting logical condition, and only the specific columns .

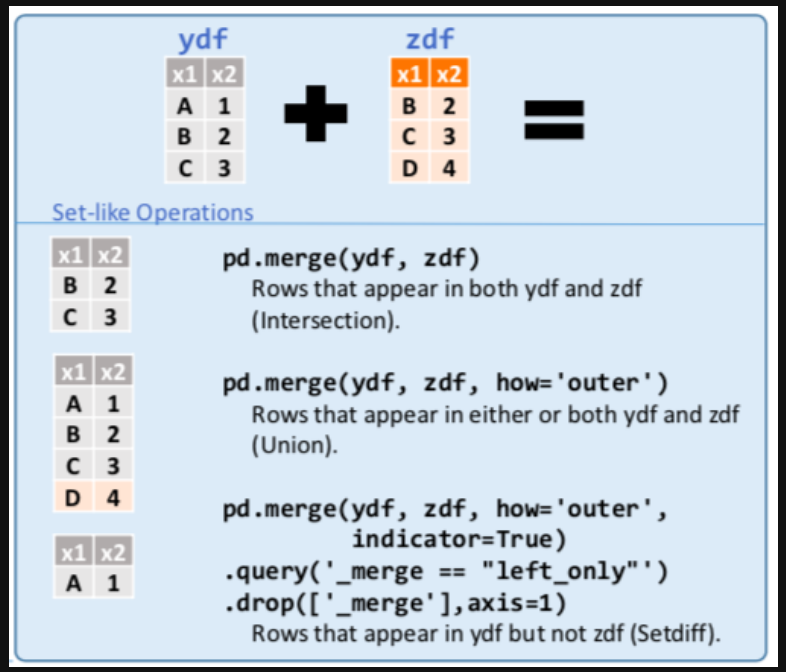
DataFrame 的“增删改查”

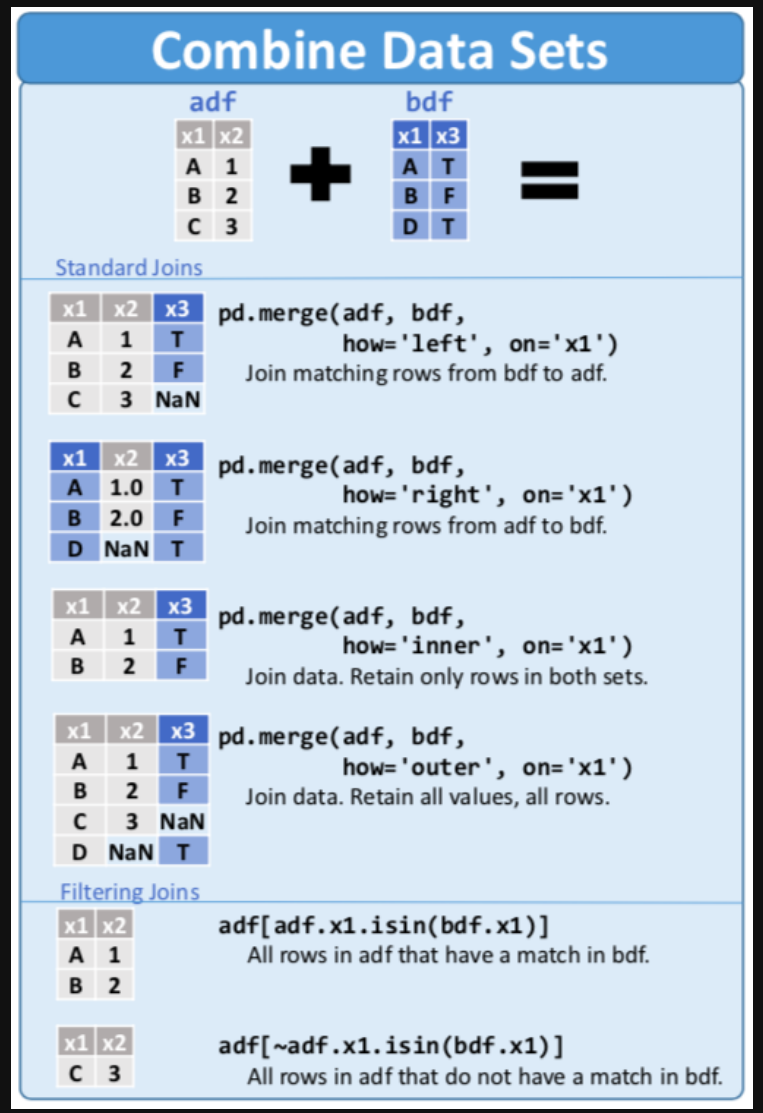
- DataFrame 的改造与合并：



DataFrame 的“增删改查”

- DataFrame 的改造与合并：
  - merge vs concat ? (Ref)
    - concat ⇒ "append"
    - concat < merge





DataFrame 的“增删改查”

- Series / DataFrame 常用的属性与方法
  - `shape / size / index / dtype / astype() / ...`
  - `head() / tail() / describe() / values` ⇒ ndarray (Numpy)
  - `max() / min() / mean() / median() / ...`
  - `to_*() / sort_index() / sort_values() / ...`
  - `apply() / drop() / drop_duplicates() / ...`
  - `isin() / isna() / isnull() / fillna() / ...`

- Series 的常用属性与方法
  - `unique()`
  - `tolist()`
  - `value_counts()`
  - `map()`
  - `is_unique() / is_monotonic()`

- DataFrame 的常用属性与方法
  - `columns`
  - `info()`
  - `stack()`
  - `insert()`
  - ...

DataFrame 的组合与聚合&透视表

—— 数据高级统计分析

- Pandas 中常用的组合和聚合方法 Ref
  - groupby() / apply() / agg()
  - stack() / unstack()
  - melted()
  - pivot_table()

```
pivoted = pd.pivot_table(s4g, index=['Symbol', 'Year'],
                         values=['Open','Close'], aggfunc='mean',
                         columns=['Month'], fill_value = 0)
```

⇕

```
table = s4g.groupby(['Symbol', 'Year', 'Month'])['Open', 'Close'].mean()
table = table.unstack('Month')
table = table.fillna(0)
```

	Account	Name	Rep	Manager	Product	Quantity	Price	Status
0	714466	Trantow-Barrows	Craig Booker	Debra Henley	CPU	1	30000	presented
1	714466	Trantow-Barrows	Craig Booker	Debra Henley	Software	1	10000	presented
2	714466	Trantow-Barrows	Craig Booker	Debra Henley	Maintenance	2	5000	pending
3	737550	Fritsch, Russel and Anderson	Craig Booker	Debra Henley	CPU	1	35000	declined
4	146832	Kiehn-Spinka	Daniel Hilton	Debra Henley	CPU	2	65000	won

```
pd.pivot_table(df,
    index=["Manager","Status"],
    columns=["Product"],
    aggfunc=[np.sum],
    values=["Price"],
    fill_value=0,
    margins=True,
    dropna=True)
```

Can also use a dictionary:
aggfunc={"Quantity":len,
"Price":[np.sum,np.mean]}

		sum				
		Price				
	Product	CPU	Maintenance	Monitor	Software	All
Manager	Status					
Debra Henley	declined	70000	0	0	0	70000
	pending	40000	10000	0	0	50000
	presented	30000	0	0	20000	50000
	won	65000	0	0	0	65000
Fred Anderson	declined	65000	0	0	0	65000
	pending	0	5000	0	0	5000
	presented	30000	0	5000	10000	45000
	won	165000	7000	0	0	172000
All		465000	22000	5000	30000	522000

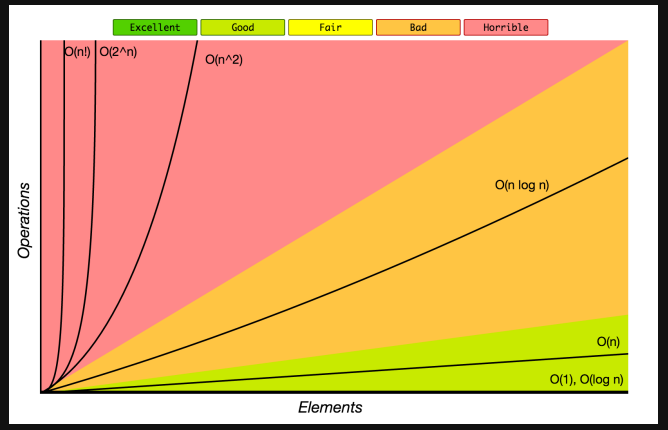
Repo of the course: https://github.com/iphysresearch/GWData-Bootcamp

Homework

1. Python、Numpy 和 Pandas 的基础作业题目（单选题）位于：
   - 2023/python/homework-python.*
   - 2023/python/homework-numpy.*
   - 2023/python/homework-pandas.*
   - （分别有 ipynb, html, md 三种文档格式，以方便阅读）
2. 将你完成的作业添加到你在上一步中创建的个人作业目录中。根据作业的类型，应将完成的作业分别命名为 python_submit.txt、numpy_submit.txt 和 pandas_submit.txt。其中每个 txt 文档的每行对应于 A,B,C,D,... 等选项当中的一个（注意：行号对应于题号）
3. 在 homework 分支上把你完成的作业 push 到你自己的关于本课程的远程仓库中，即：$ git push origin homework ；最后，在GitHub上你的远程仓库中，在 homework 分支下发起 Pull Request (PR) 至本课程远程仓库的 homework 分支。
4. GitHub Actions 工作流将自动检查你的提交，并将 modified 的 python_submit.txt、numpy_submit.txt 和 pandas_submit.txt 与 solution 进行比较。
5. 基础作业的最终成绩，根据PR的最新commit来定，记得到时候 @ 我记录成绩。
6. 不要修改其他学员的作业目录和作业内容！

Repo of the course: https://github.com/iphysresearch/GWData-Bootcamp

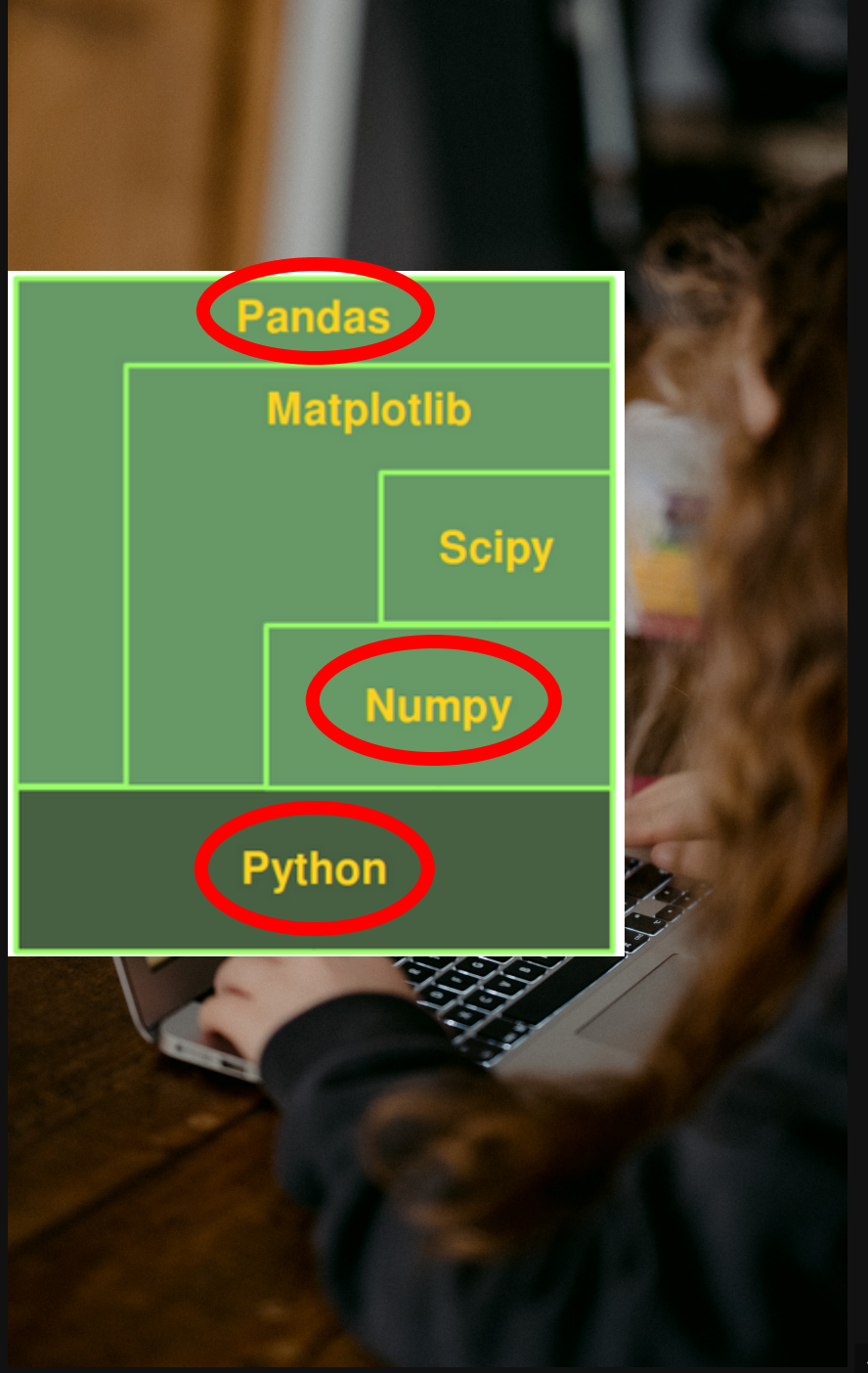
Homework



扩展作业

1. 完成以下 Leetcode 中的算法题目：
   - 434. Number of Segments in a String (初级)
   - 1869. Longer Contiguous Segments of Ones than Zeros (初级)
   - 1784. Check if Binary String Has at Most One Segment of Ones (初级)
   - 852. Peak Index in a Mountain Array (中级)
   - 162. Find Peak Element (中级)
2. 把上面5道算法题目的结果 comment 在你完成的**基础作业**的PR里，要求：
   算法的每一行都写好中文说明注释。

```python
class Solution:
    def twoSum(self, nums: List[int], target: int) -> List[int]:
        return [
            # 取出两个索引
            [index_i,index_j]
            # 第一层循环
            for index_i, i in enumerate(nums)
            # 第二层循环
            for index_j, j in enumerate(nums)
            # 跳过相同的索引
            if index_i != index_j
            # 要求两层循环的下的值满足要求
            if target == i+j][0]
```



Want more? see: Advent of Code