Microsoft

Introduction to Python for Data Science

资料科学入门 – 数据维护与清理

June 2020
Microsoft Reactor | Ryan Chung

Microsoft

Ryan Chung

Instructor / DevelopIntelligence
Founder / MobileDev.TW

@ryanchung403 on WeChat

Microsoft

Reactor

developer.microsoft.com/reactor/

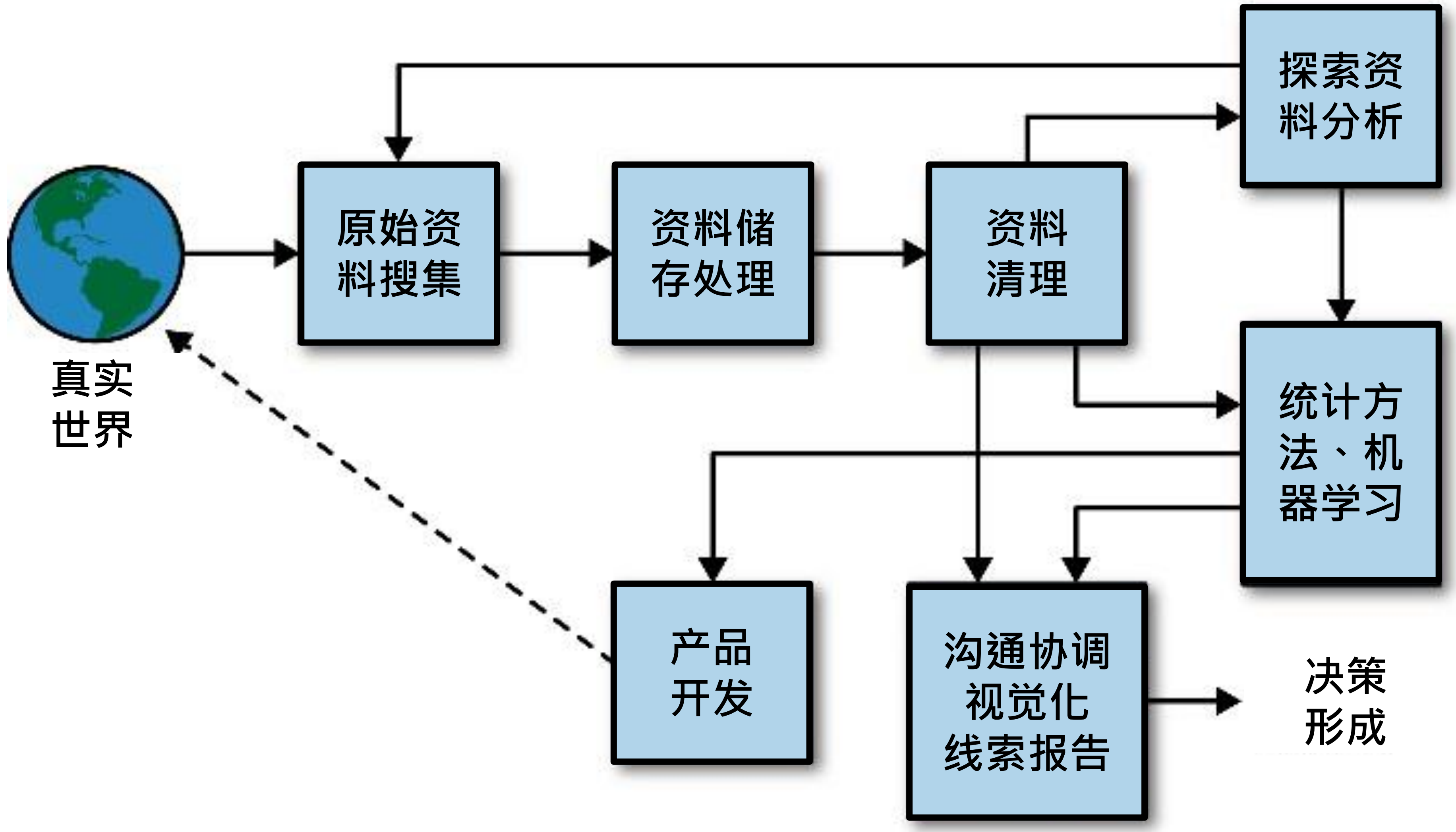
@MSFTReactor on Twitter

Data Science Workshop agenda 资料科学在线研讨会议程

How to manipulate and clean your data 数据维护与清理

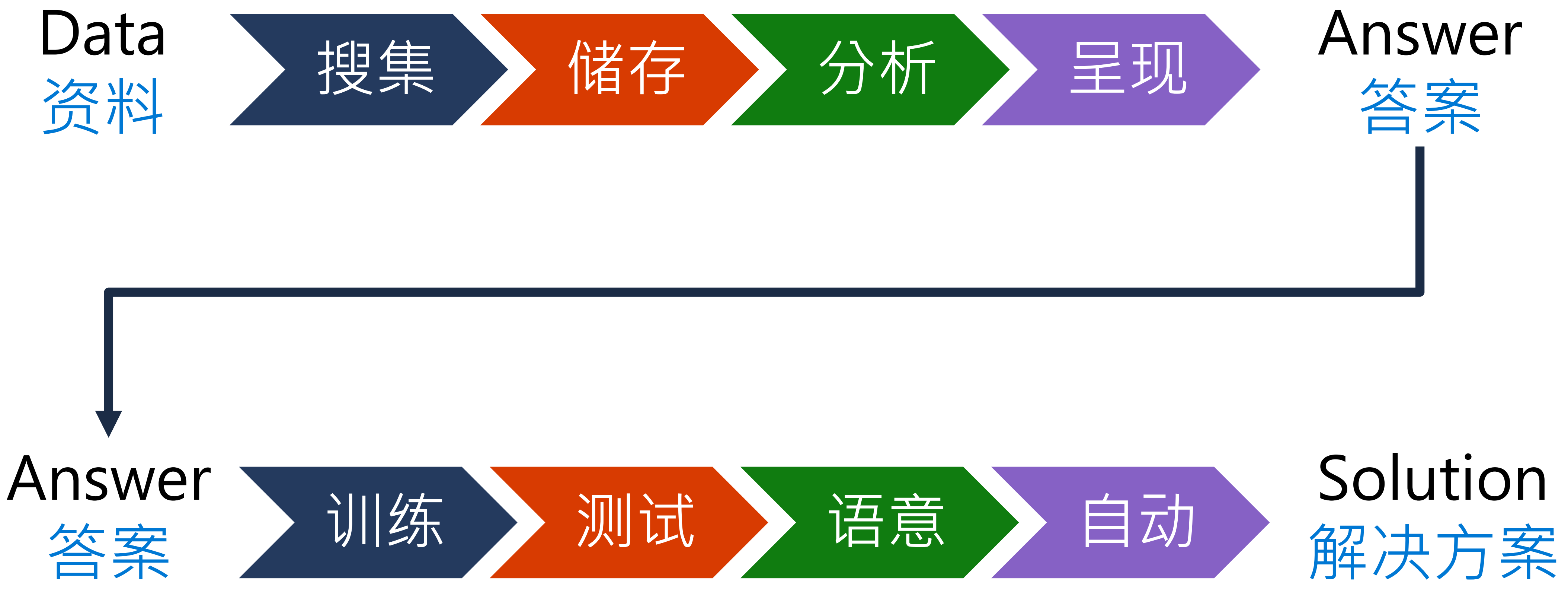
19:30	Welcome 开场
19:35	Introduction to Data Science 资料科学介绍
20:10	Exploring information in DataFrame 信息探索
20:30	10-minute break 中场休息
20:40	Working with missing data values 遗漏值处理
21:20	Dataset Combination 数据集合并
21:30	Event end 研讨会结束

资料科学家在做哪些事?



原始资料搜集
资料储存处理
资料清理
探索资料分析
统计方法、机器学习
真实世界
产品开发
沟通协调视觉化线索报告
决策形成

资料科学家 & AI解决方案架构师

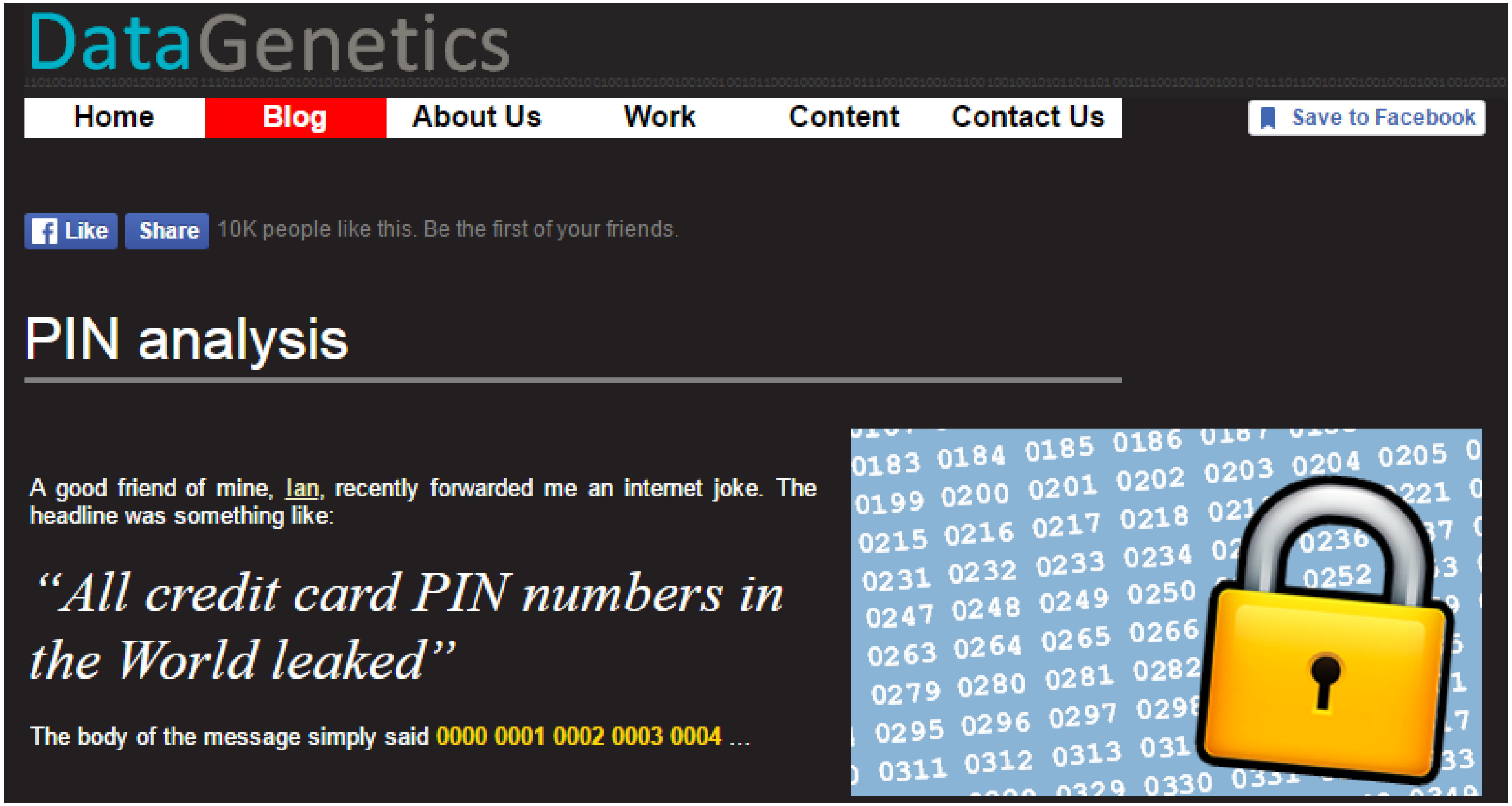


案例：用算法卖衣服冲出近十亿美元业绩

我们将你可能会喜欢的服饰寄给你；你留下你喜欢的品项，其他免费退回。

- 善用资料科学，大规模提供个人化服务，因而超越传统的实体和电子商务零售体验。顾客乐于有专业造型师为他们采买，而且赞赏这种服务既便利又简单。
- Stitch Fix每一次交货，是用一个盒子装着专门为你挑选的五样服饰。
- 挑选来源：根据顾客与数百万其他人提供的资讯，第一个来源就是顾客在注册时填写的详细问卷，接下来就是每次交运之后，顾客提供的回馈意见。

资料探索实例分享-信用卡PIN码



10K people like this. Be the first of your friends.

PIN analysis

A good friend of mine, Ian, recently forwarded me an internet joke. The headline was something like:

"All credit card PIN numbers in the World leaked"

The body of the message simply said 0000 0001 0002 0003 0004 ...

基本统计

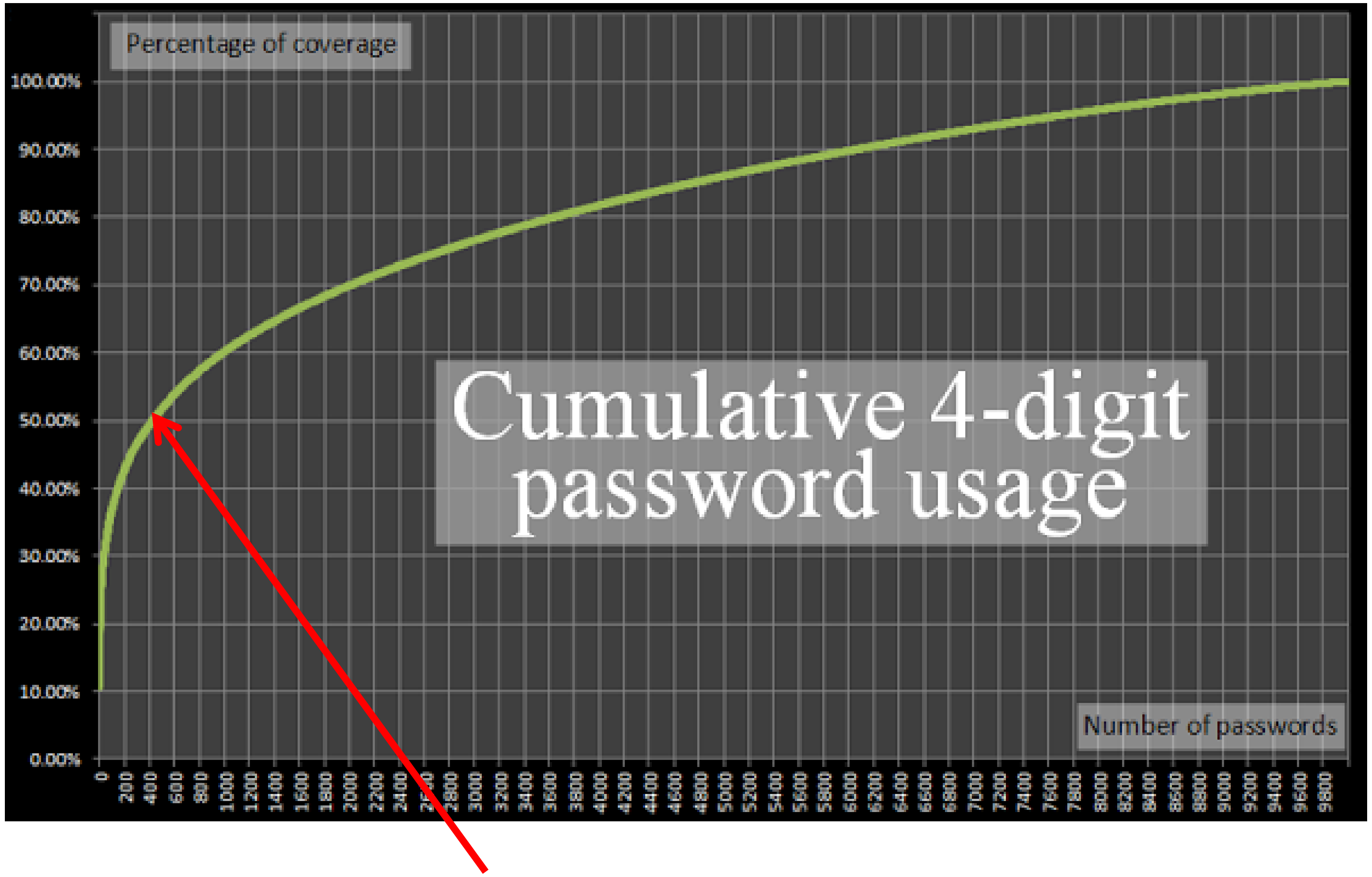
PIN	Freq		PIN	Freq	
#1	1234	10.71%	#9980	8557	0.00%
#2	1111	6.02%	#9981	9047	0.00%
#3	0000	1.88%	#9982	8438	0.00%
#4	1212	1.20%	#9983	0439	0.00%
#5	7777	0.75%	#9984	9539	0.00%
#6	1004	0.62%	#9985	8196	0.00%
#7	2000	0.61%	#9986	7063	0.00%
#8	4444	0.53%	#9987	6093	0.00%
#9	2222	0.52%	#9988	6827	0.00%
#10	6969	0.51%	#9989	7394	0.00%
#11	9999	0.45%	#9990	0859	0.00%
#12	3333	0.42%	#9991	8957	0.00%
#13	5555	0.40%	#9992	9480	0.00%
#14	6666	0.39%	#9993	6793	0.00%
#15	1122	0.37%	#9994	8398	0.00%
#16	1313	0.30%	#9995	0738	0.00%
#17	8888	0.30%	#9996	7637	0.00%
#18	4321	0.29%	#9997	6835	0.00%
#19	2001	0.29%	#9998	9629	0.00%
#20	1010	0.29%	#9999	8093	0.00%
	……	……	#10000	8068	0.00%

不意外!
1234, 1111, 0000, 1212, 7777

「2580」名列第22？

资料视觉化

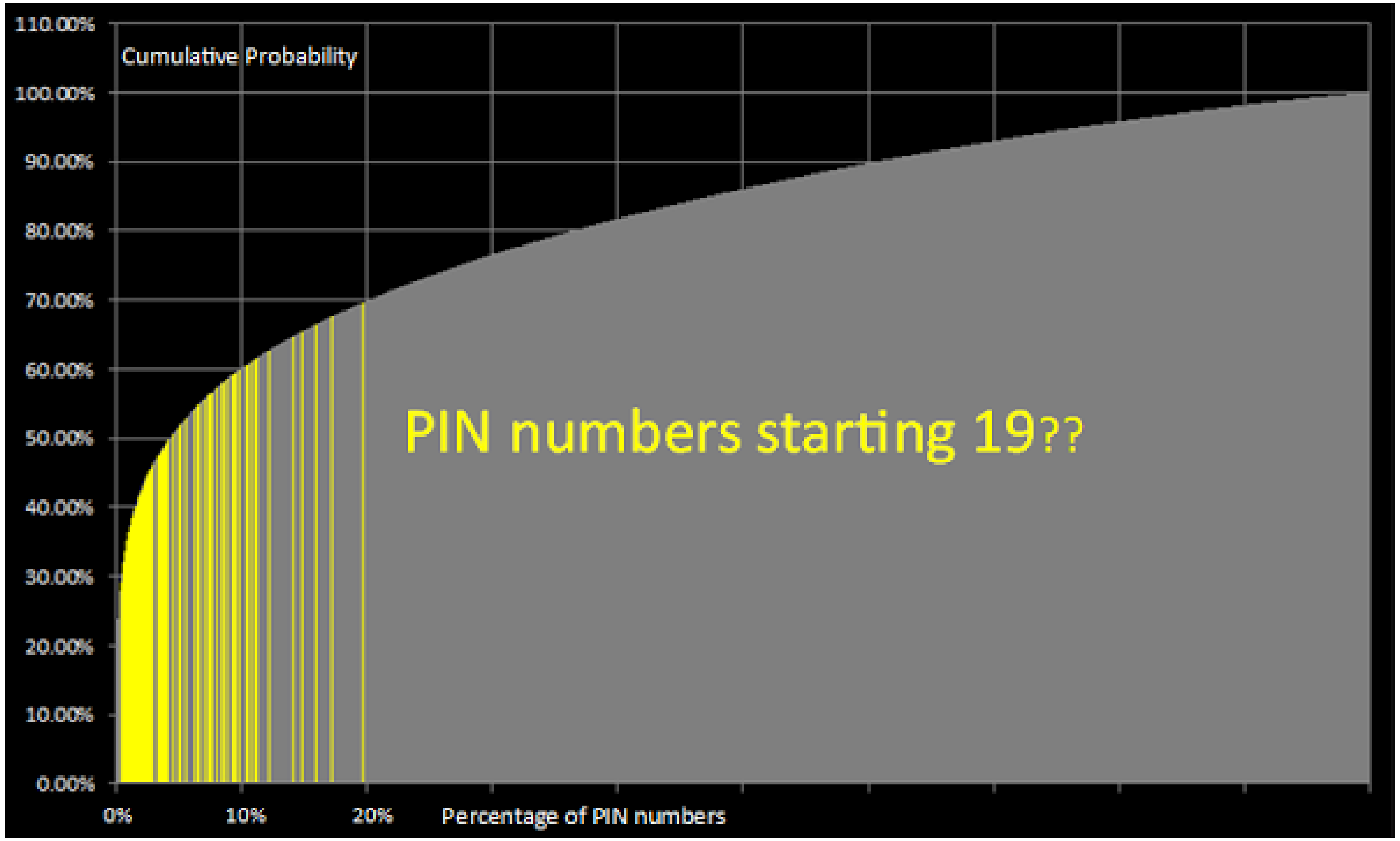
Cumulative Frequency



不到五百组就把一半密码都破解了 **(全部 10000组)**

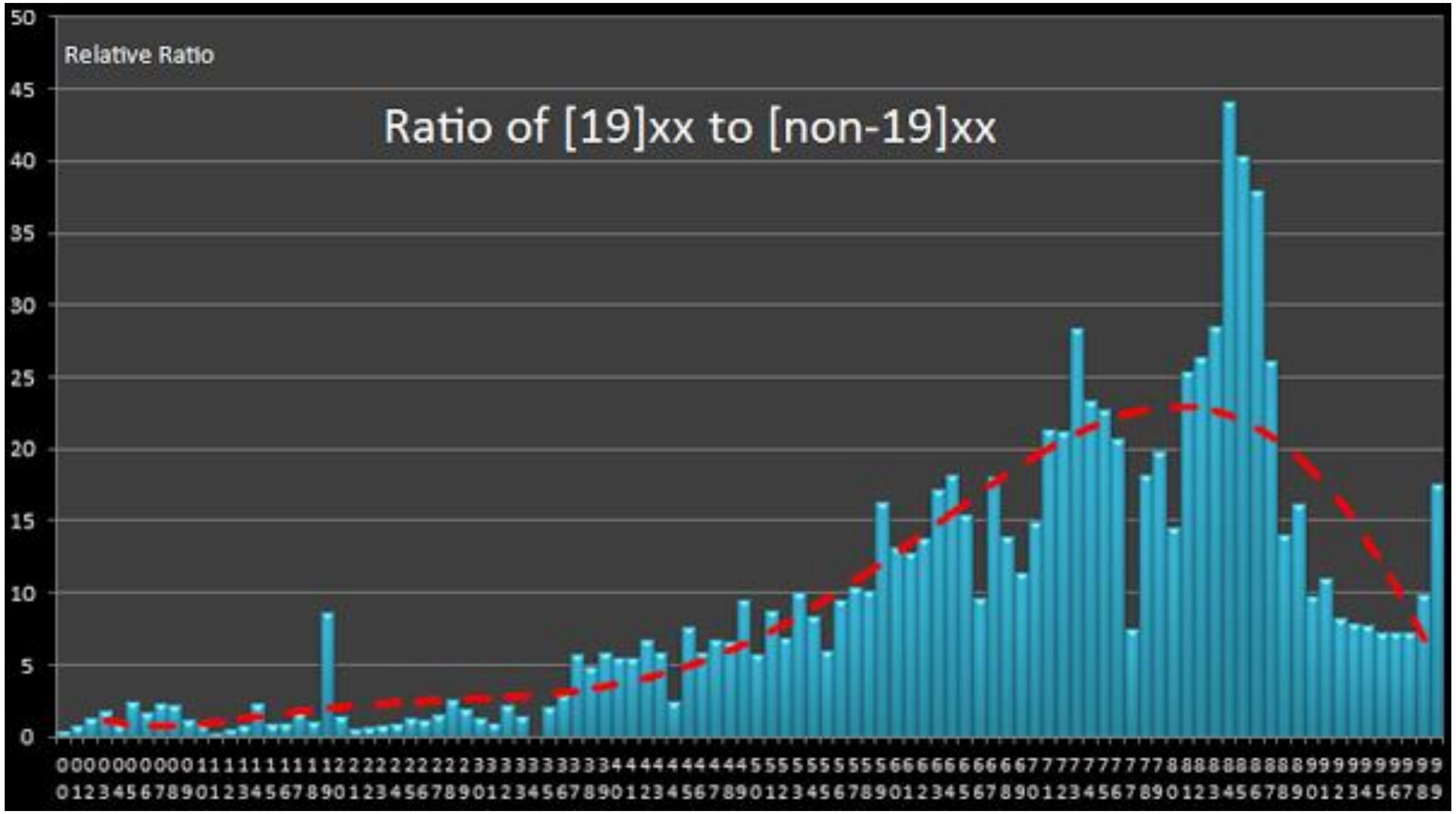
资料视觉化

Cumulative Frequency



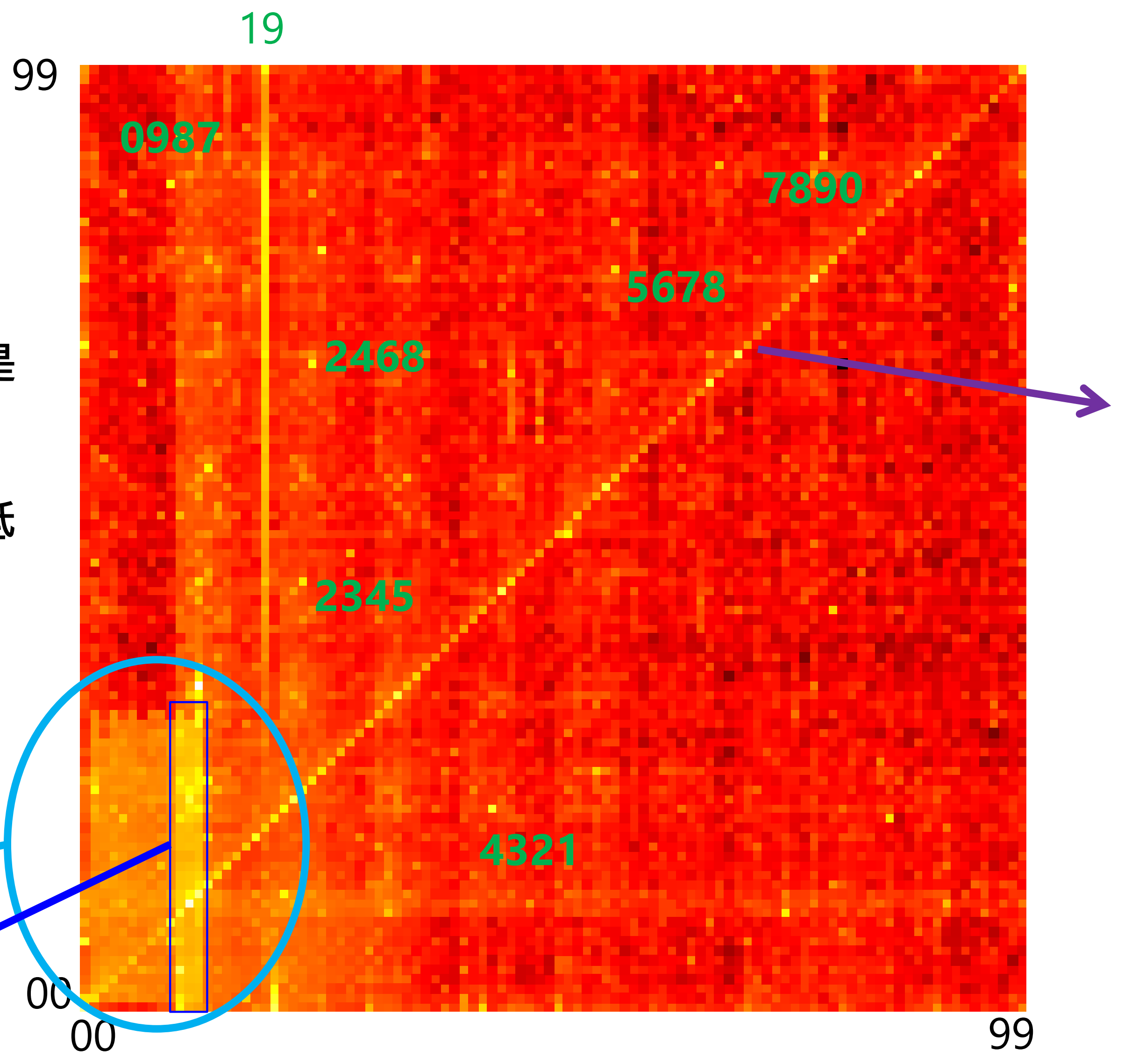
人们偏好「**19XX**」系列？

资料视觉化



「195X」一直到「198X」的出现频率比远远高过其他年份

资料视觉化-资料矩阵



越偏白黄的颜色就是频率越高的组合

偏红黑色即是频率低的组合

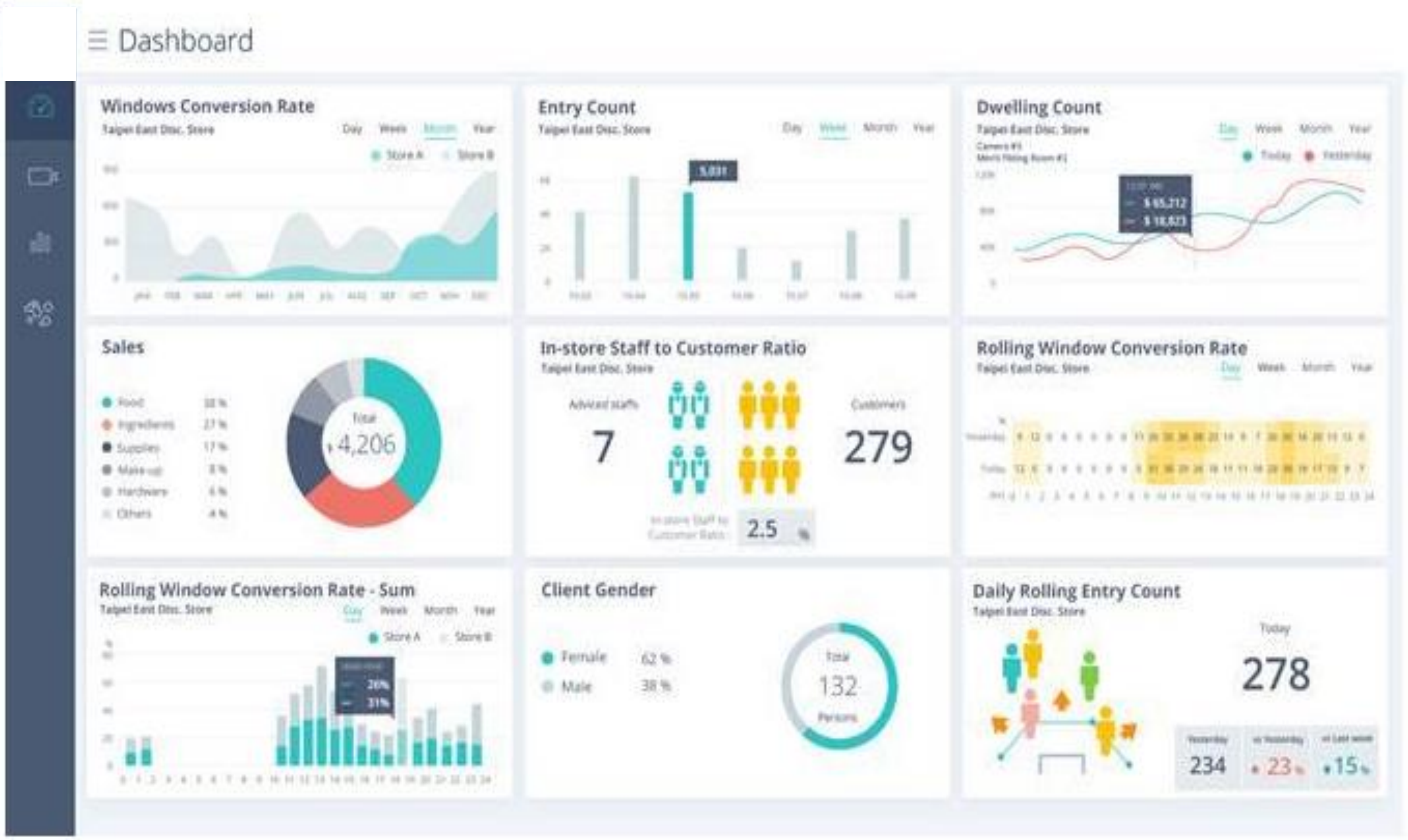
两个一组重复
(如: 1212, 5454,..)
17.8% !!

前两位数 00~20
后两位数 00~30

前两位数 10~12
后两位数 00~30

某新创公司的智慧商情分析系统

利用监控摄影机，让数字说话


Dashboard
Windows Conversion Rate
Entry Count
Dwelling Count
Sales
In-store Staff to Customer Ratio
Rolling Window Conversion Rate
Rolling Window Conversion Rate - Sum
Client Gender
Daily Rolling Entry Count
4,206
7
279
2.5
132
278
234
23
15

- 人流统计
- 橱窗转换率
- 客人停留次数与平均停留时间
- 即时反应店内人数
- 热区分析
- 动线分析
- 自动天气
- 客层分析
- 黑白名单
- 即时远端观看
- 结合POS

Source :http://iknow.stpi.narl.org.tw/Post/Read.aspx?PostID=12747

Microsoft

Manipulating and cleaning data 数据维护与清理

Section 4 第四节

Section 4 overview 第四节

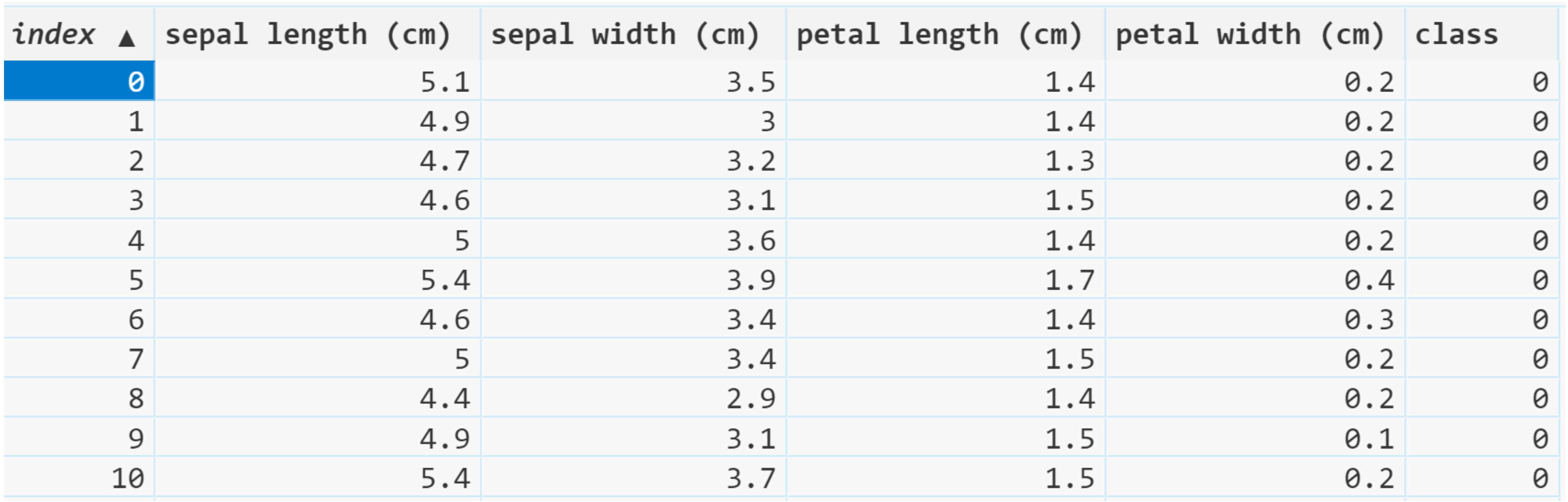
- DataFrame 资讯探索
- 遗漏值处理
  - 确认、移除、填补
- 资料集合并
  - Numpy、Pandas 序列、Pandas DataFrame

真实世界中
资料分析型的专案
八成以上的时间都花在
资料的清理与准备

Anderson's Iris data set / Iris flower data set
安德森鸢尾花卉数据集

样本数：150

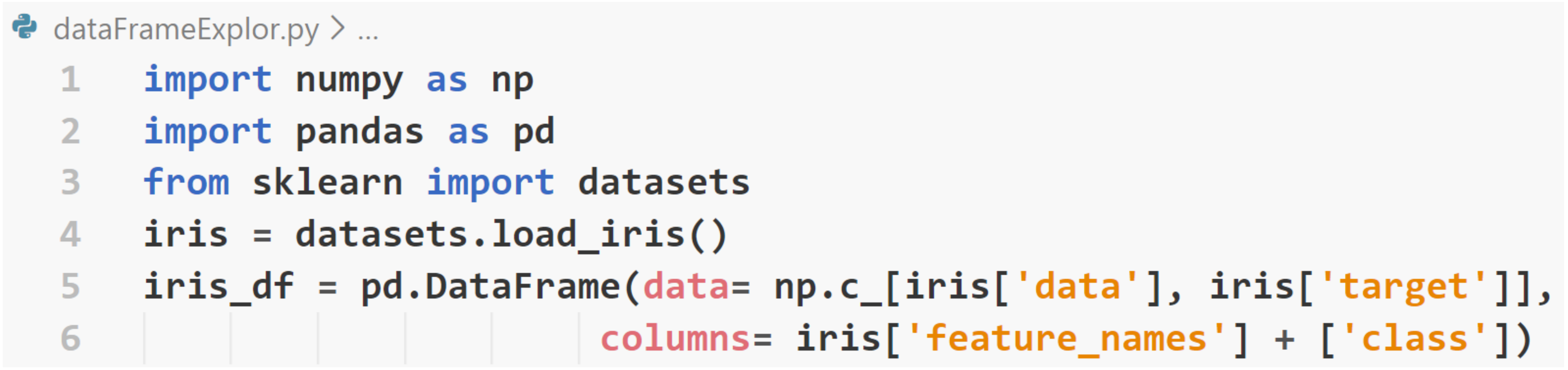
类别：0-Setosa 山鸢尾**、1-Versicolour** 变色鸢尾**、2-Virginica** 维吉尼亚鸢尾



	花萼长度	花萼宽度	花瓣长度	花萼宽度	类别
index ▲	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
0	5.1	3.5	1.4	0.2	0
1	4.9	3	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5	3.6	1.4	0.2	0
5	5.4	3.9	1.7	0.4	0
6	4.6	3.4	1.4	0.3	0
7	5	3.4	1.5	0.2	0
8	4.4	2.9	1.4	0.2	0
9	4.9	3.1	1.5	0.1	0
10	5.4	3.7	1.5	0.2	0

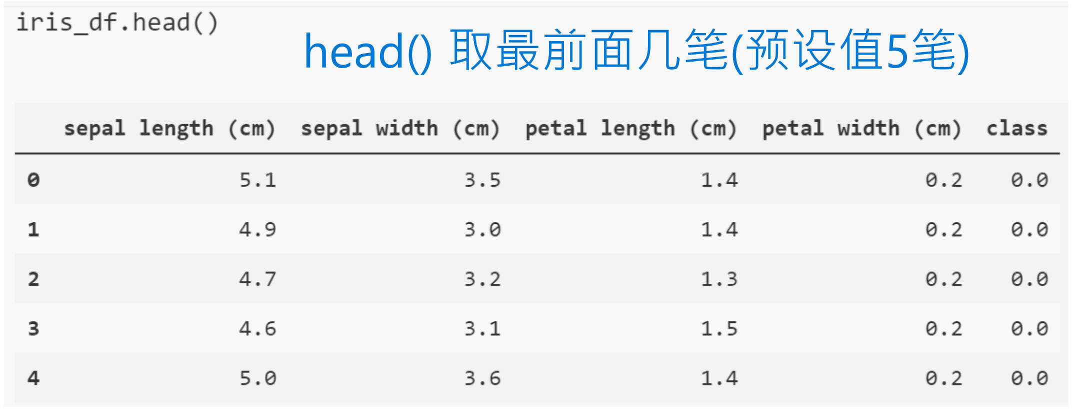
DataFrame 资讯探索

22.dataframe_explor.py



dataFrameExplor.py > ...

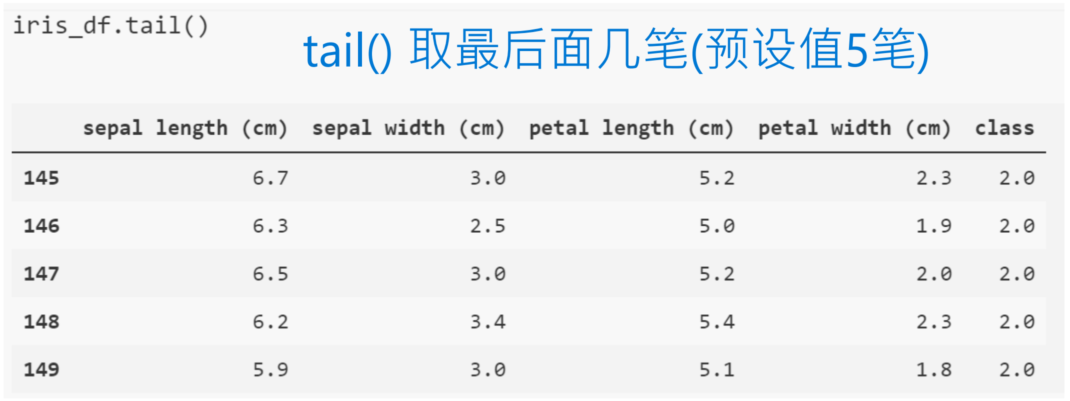
```python
import numpy as np
import pandas as pd
from sklearn import datasets
iris = datasets.load_iris()
iris_df = pd.DataFrame(data= np.c_[iris['data'], iris['target']],
                       columns= iris['feature_names'] + ['class'])
```



```
iris_df.head()
```

head() 取最前面几笔(预设值5笔)

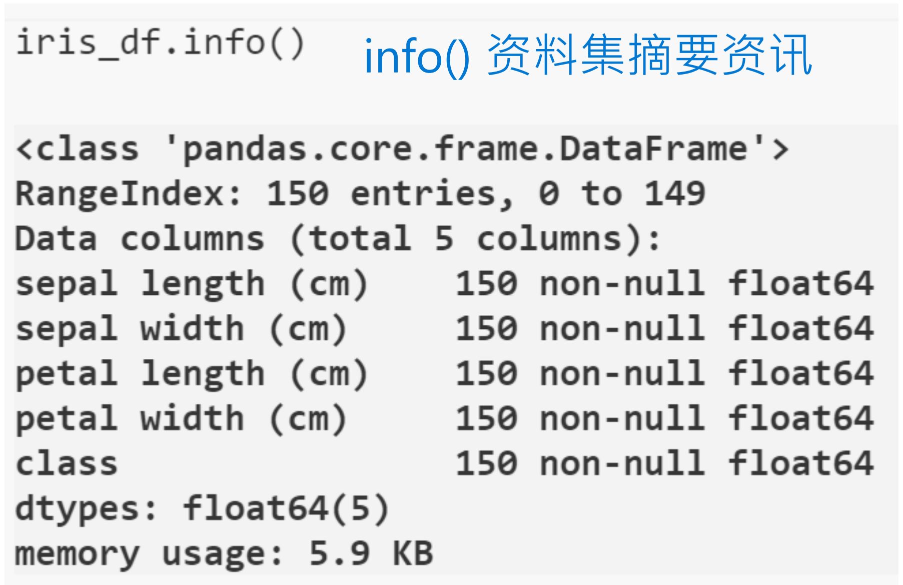
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0



```
iris_df.tail()
```

tail() 取最后面几笔(预设值5笔)

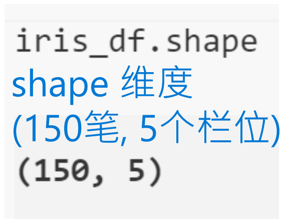
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
145	6.7	3.0	5.2	2.3	2.0
146	6.3	2.5	5.0	1.9	2.0
147	6.5	3.0	5.2	2.0	2.0
148	6.2	3.4	5.4	2.3	2.0
149	5.9	3.0	5.1	1.8	2.0



```
iris_df.info()
```

info() 资料集摘要资讯

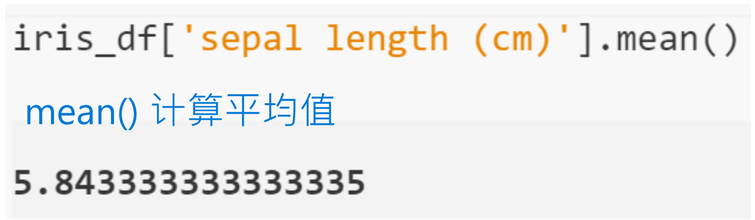
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
sepal length (cm)    150 non-null float64
sepal width (cm)     150 non-null float64
petal length (cm)    150 non-null float64
petal width (cm)     150 non-null float64
class                150 non-null float64
dtypes: float64(5)
memory usage: 5.9 KB
```



```
iris_df.shape
```

shape 维度
(150笔, 5个栏位)

```
(150, 5)
```



```
iris_df['sepal length (cm)'].mean()
```

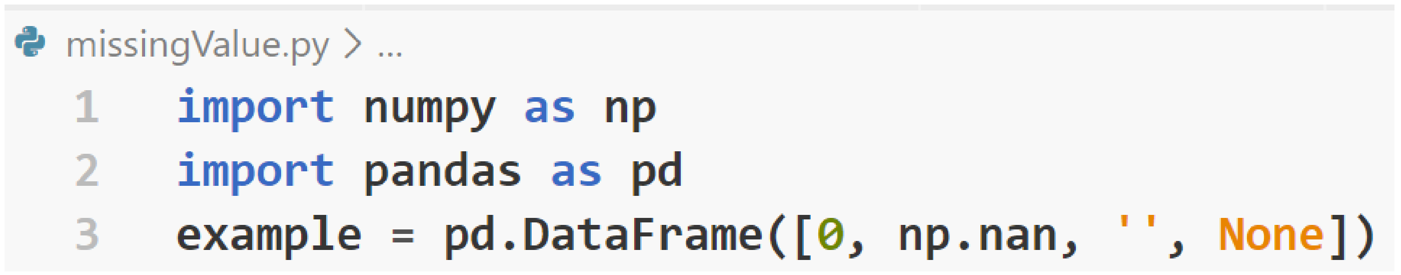
mean() 计算平均值

```
5.843333333333335
```

Microsoft Reactor

遺漏值处理

- Python 的空值: None
- NumPy/pandas 的空值: NaN



missingValue.py > ...

```python
import numpy as np
import pandas as pd
example = pd.DataFrame([0, np.nan, '', None])
```

原始内容

```
example
```

	0
0	0
1	NaN
2	
3	None

判断是否为空值

```
example.isnull()
```

	0
0	False
1	True
2	False
3	True

把空值列丢掉

```
example.dropna()
```

	0
0	0
2	

把空值列都补上0

```
example.fillna(0)
```

	0
0	0
1	0
2	
3	0

23.missing_value.py

Reactor

遗失值处理方式

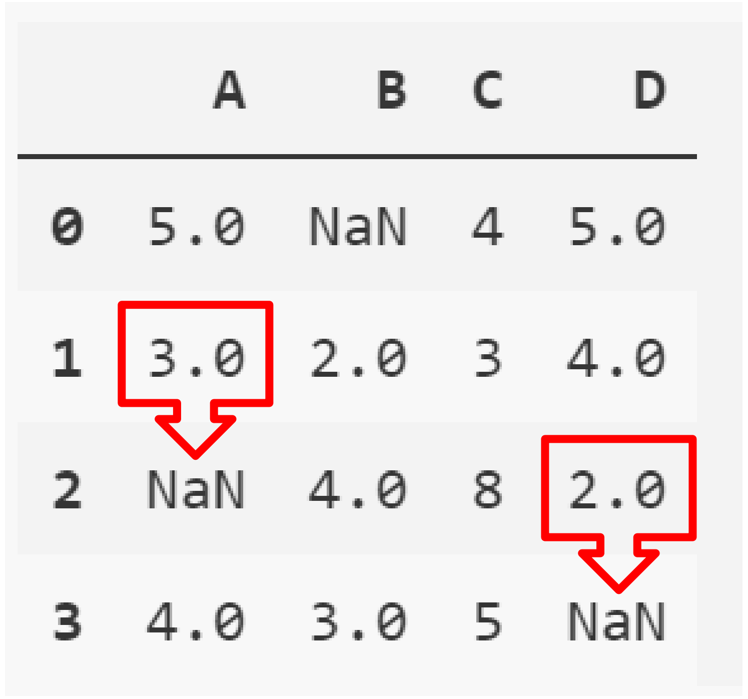
方法	用途	备注
dropna()	把遗失值该列删去	资料量够大
fillna(x)	以指定值x来替代遗失值	可指定参数决定替代值
ffill()	用前面一个的值来替替代	全名：forward fill
bfill()	用后面一个的值来替代	全名：backward fill
isnull()	判断是否为空值	在原本每个值的位置回传布尔值
notnull()	判断是否不是空值	与isnull()反义

Microsoft Reactor

ffill() : forward fill

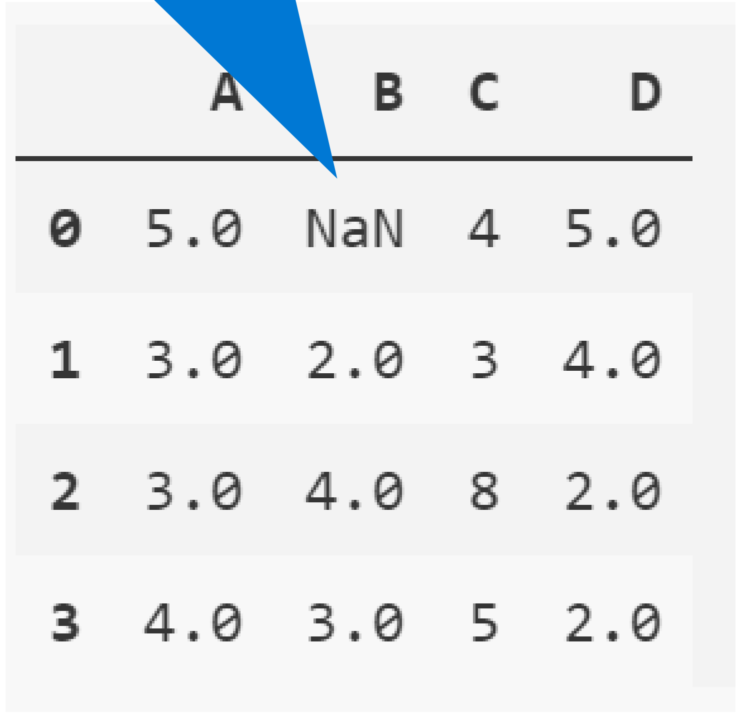
```python
import numpy as np
import pandas as pd

df = pd.DataFrame(
    {
        "A":[5,3,None,4],
        "B":[None,2,4,3],
        "C":[4,3,8,5],
        "D":[5,4,2,None]
    }
)

df.ffill(axis=0)
```

代表以字段轴来填补



	A	B	C	D
0	5.0	NaN	4	5.0
1	3.0	2.0	3	4.0
2	NaN	4.0	8	2.0
3	4.0	3.0	5	NaN

我前面没坐人没得抄 😫😫

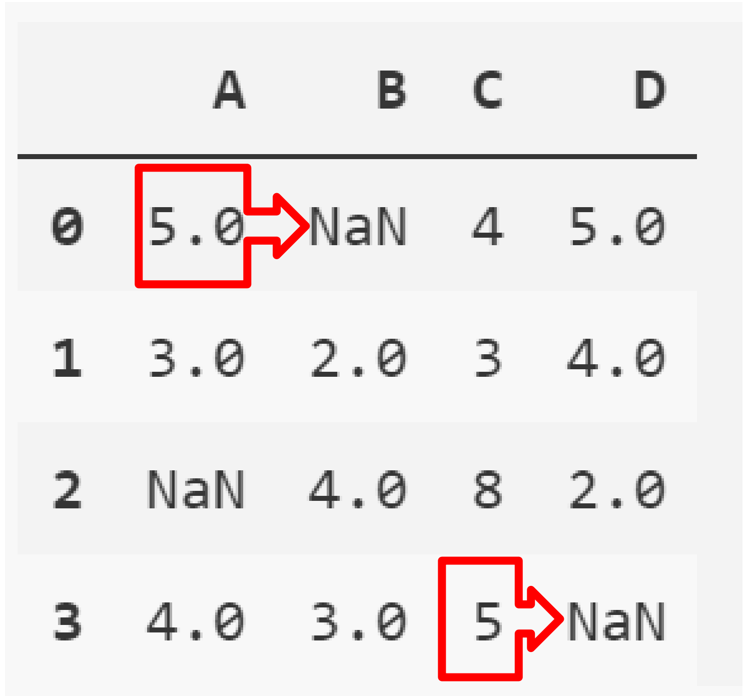


	A	B	C	D
0	5.0	NaN	4	5.0
1	3.0	2.0	3	4.0
2	3.0	4.0	8	2.0
3	4.0	3.0	5	2.0

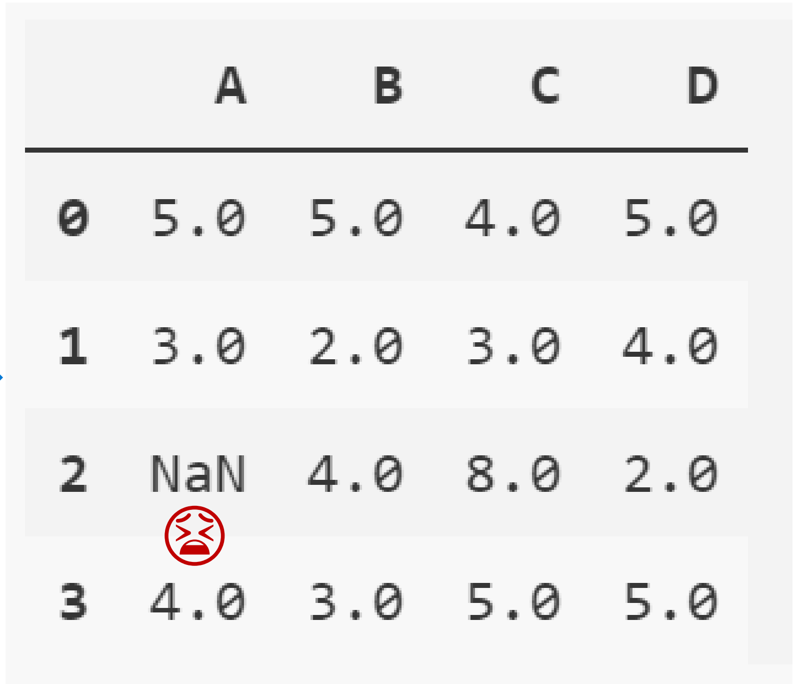
ffill() : forward fill

```python
import numpy as np
import pandas as pd

df = pd.DataFrame(
    {
        "A":[5,3,None,4],
        "B":[None,2,4,3],
        "C":[4,3,8,5],
        "D":[5,4,2,None]
    }
)

df.ffill(axis=1)
```

代表以列轴来填补



	A	B	C	D
0	5.0	NaN	4	5.0
1	3.0	2.0	3	4.0
2	NaN	4.0	8	2.0
3	4.0	3.0	5	NaN

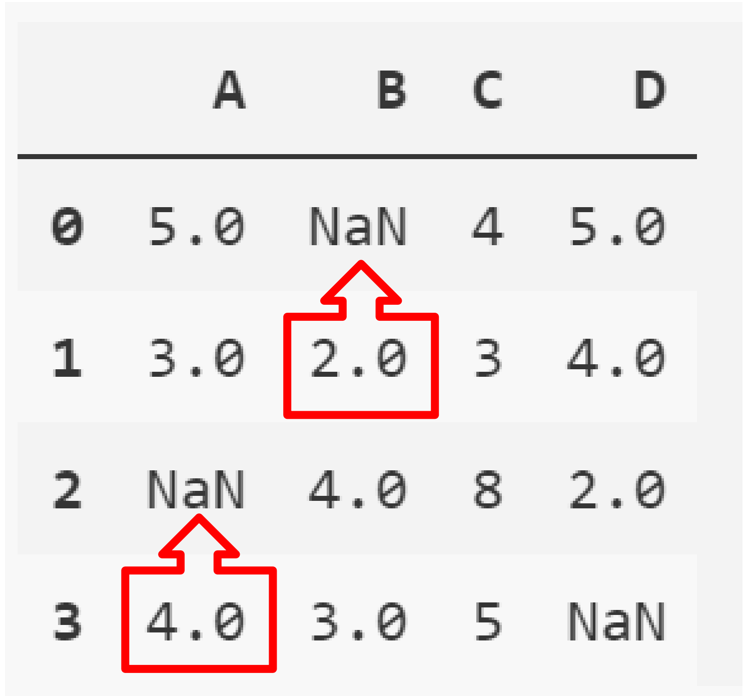


	A	B	C	D
0	5.0	5.0	4.0	5.0
1	3.0	2.0	3.0	4.0
2	NaN	4.0	8.0	2.0
3	4.0	3.0	5.0	5.0

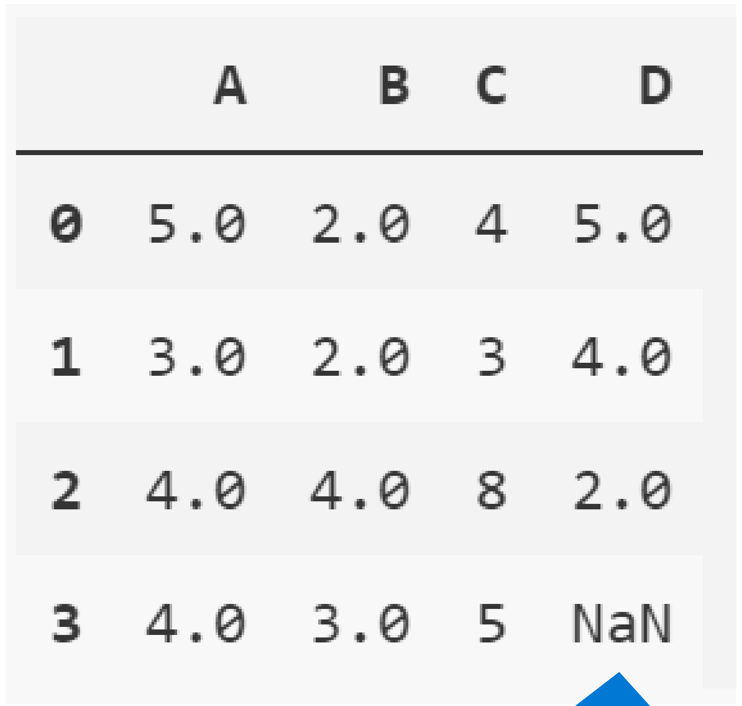
bfill() : backward fill

```python
import numpy as np
import pandas as pd

df = pd.DataFrame(
    {
        "A":[5,3,None,4],
        "B":[None,2,4,3],
        "C":[4,3,8,5],
        "D":[5,4,2,None]
    }
)

df.bfill(axis=0)
```

代表以字段轴来填补



	A	B	C	D
0	5.0	NaN	4	5.0
1	3.0	2.0	3	4.0
2	NaN	4.0	8	2.0
3	4.0	3.0	5	NaN



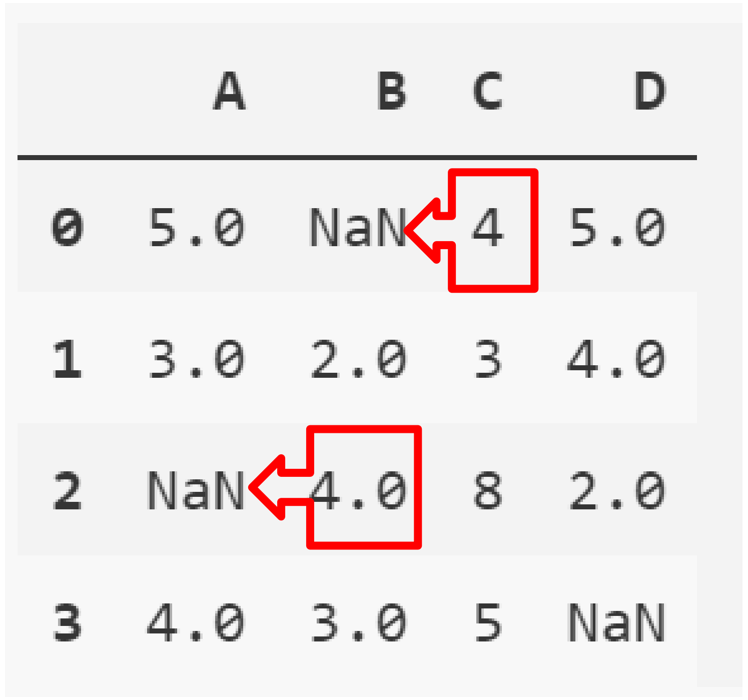
	A	B	C	D
0	5.0	2.0	4	5.0
1	3.0	2.0	3	4.0
2	4.0	4.0	8	2.0
3	4.0	3.0	5	NaN

我后面没坐人没得抄 😫😫

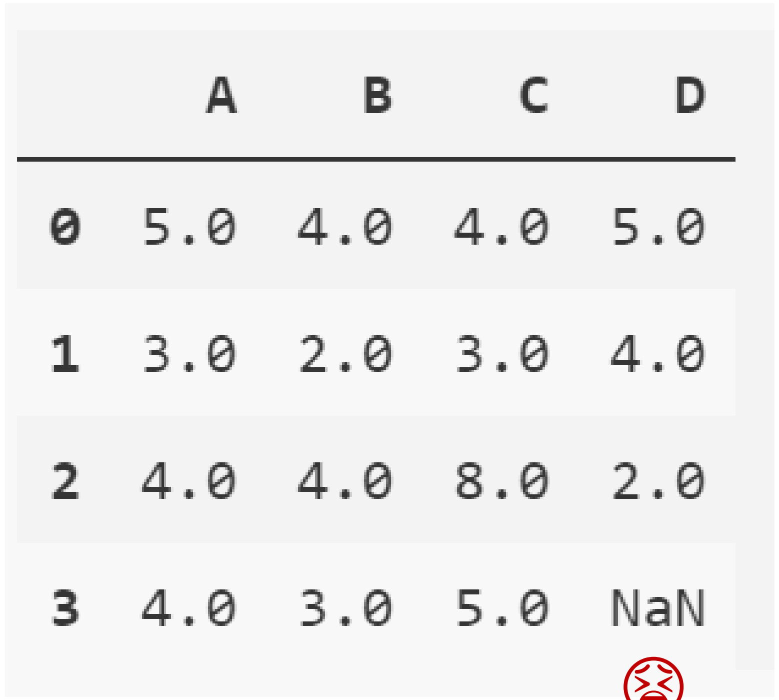
bfill() : backward fill

```python
import numpy as np
import pandas as pd

df = pd.DataFrame(
    {
        "A":[5,3,None,4],
        "B":[None,2,4,3],
        "C":[4,3,8,5],
        "D":[5,4,2,None]
    }
)

df.bfill(axis=1)
```

代表以列轴来填补



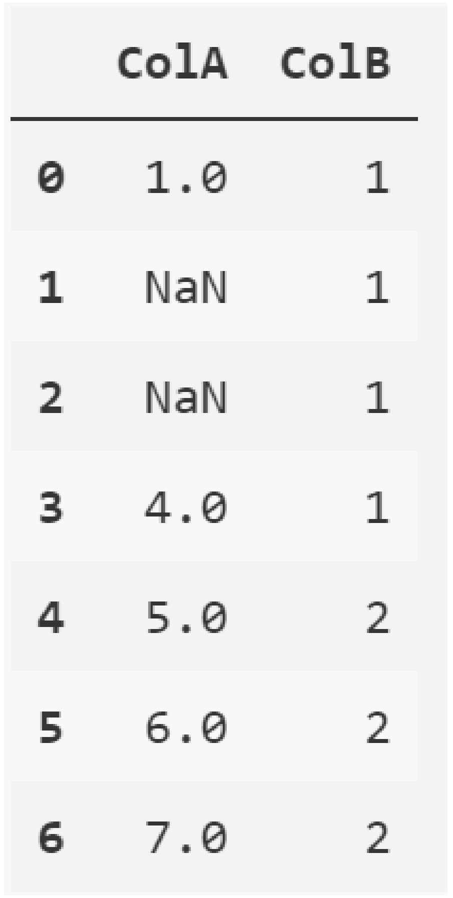
	A	B	C	D
0	5.0	NaN	4	5.0
1	3.0	2.0	3	4.0
2	NaN	4.0	8	2.0
3	4.0	3.0	5	NaN



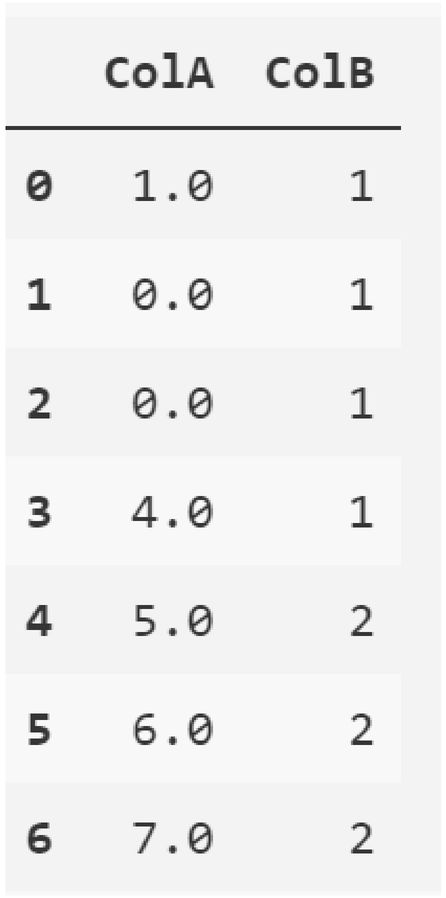
	A	B	C	D
0	5.0	4.0	4.0	5.0
1	3.0	2.0	3.0	4.0
2	4.0	4.0	8.0	2.0
3	4.0	3.0	5.0	NaN

fillna() 带参数用法

```python
import numpy as np
import pandas as pd

df = pd.DataFrame({
     'ColA':[1, np.nan, np.nan, 4,5,6,7],
     'ColB':[1,1,1,1,2,2,2]
})


df.fillna(value=0)
```



	ColA	ColB
0	1.0	1
1	NaN	1
2	NaN	1
3	4.0	1
4	5.0	2
5	6.0	2
6	7.0	2

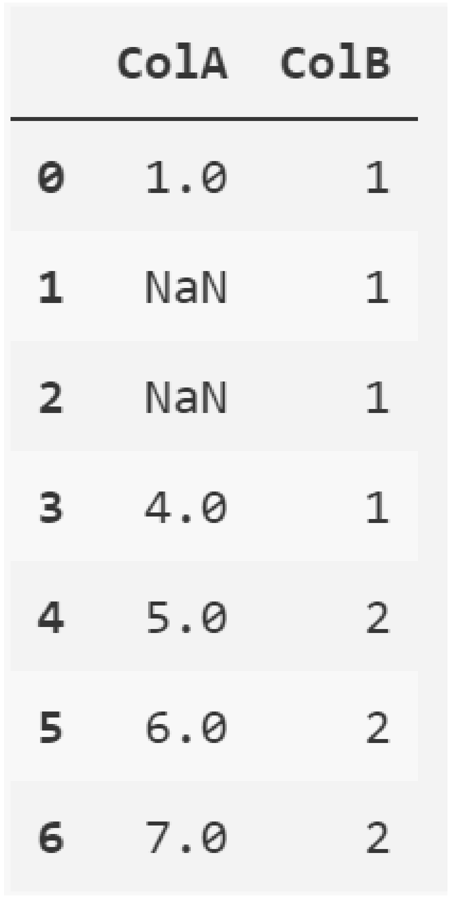


	ColA	ColB
0	1.0	1
1	0.0	1
2	0.0	1
3	4.0	1
4	5.0	2
5	6.0	2
6	7.0	2

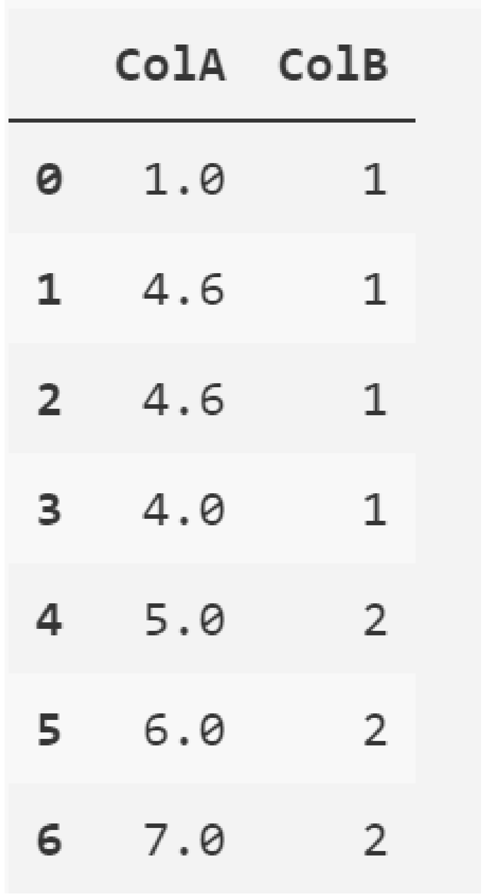
fillna() 带参数用法

```python
import numpy as np
import pandas as pd

df = pd.DataFrame({
     'ColA':[1, np.nan, np.nan, 4,5,6,7],
     'ColB':[1,1,1,1,2,2,2]
})


df.fillna(value=df.mean())
```

(1+4+5+6+7)/5 = 4.6



	ColA	ColB
0	1.0	1
1	NaN	1
2	NaN	1
3	4.0	1
4	5.0	2
5	6.0	2
6	7.0	2

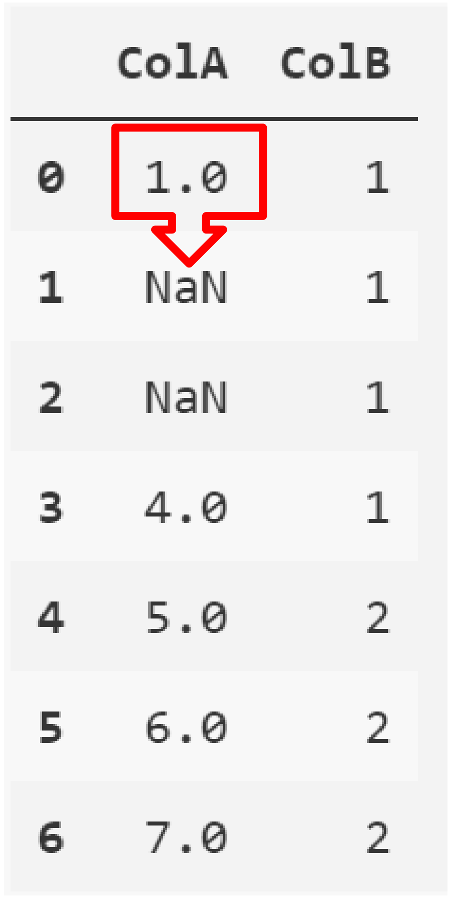


	ColA	ColB
0	1.0	1
1	4.6	1
2	4.6	1
3	4.0	1
4	5.0	2
5	6.0	2
6	7.0	2

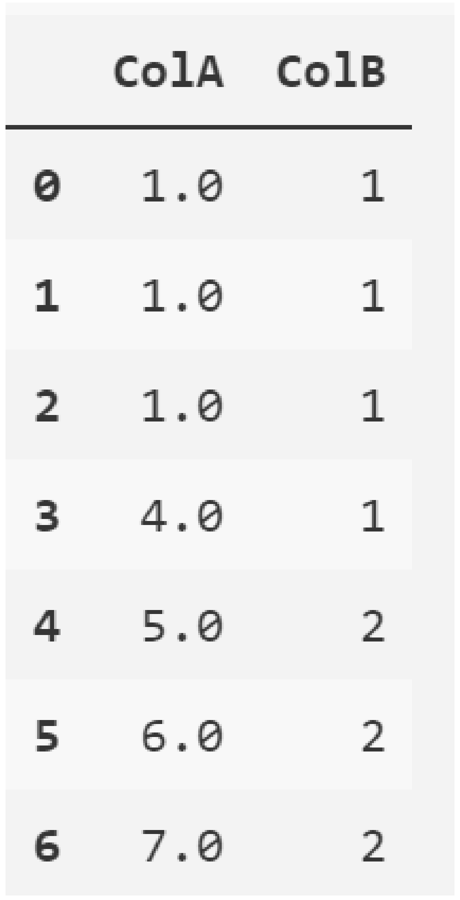
fillna() 带参数用法

```python
import numpy as np
import pandas as pd

df = pd.DataFrame({
    'ColA':[1, np.nan, np.nan, 4,5,6,7],
    'ColB':[1,1,1,1,2,2,2]
})


df.fillna(method='ffill', axis=0)
```

同一字段，抄前面的



	ColA	ColB
0	1.0	1
1	NaN	1
2	NaN	1
3	4.0	1
4	5.0	2
5	6.0	2
6	7.0	2

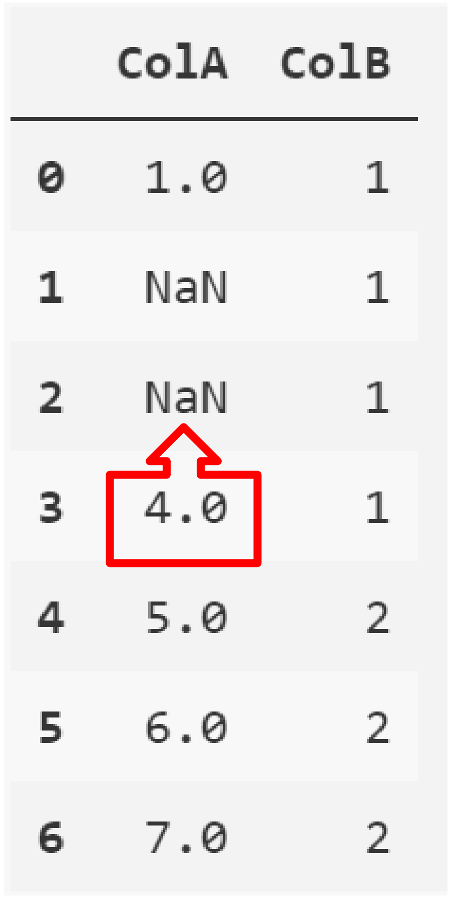


	ColA	ColB
0	1.0	1
1	1.0	1
2	1.0	1
3	4.0	1
4	5.0	2
5	6.0	2
6	7.0	2

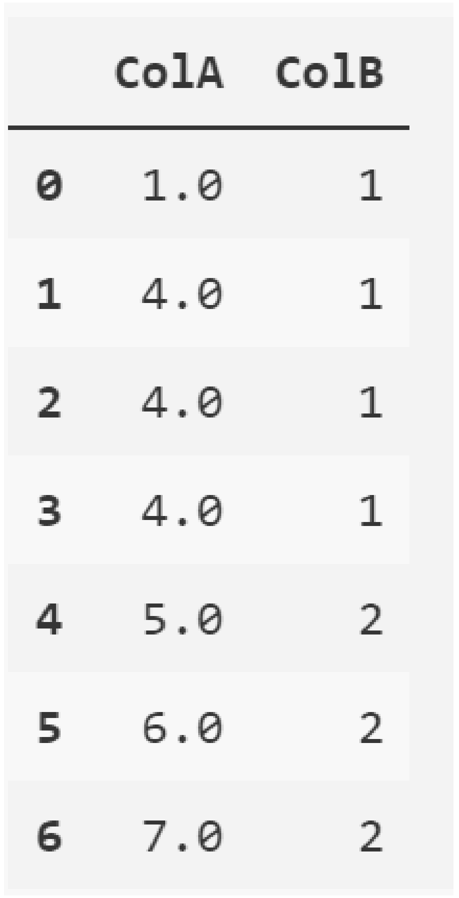
fillna() 带参数用法

```python
import numpy as np
import pandas as pd

df = pd.DataFrame({
     'ColA':[1, np.nan, np.nan, 4,5,6,7],
     'ColB':[1,1,1,1,2,2,2]
})


df.fillna(method='bfill', axis=0)
```

同一字段，抄后面的



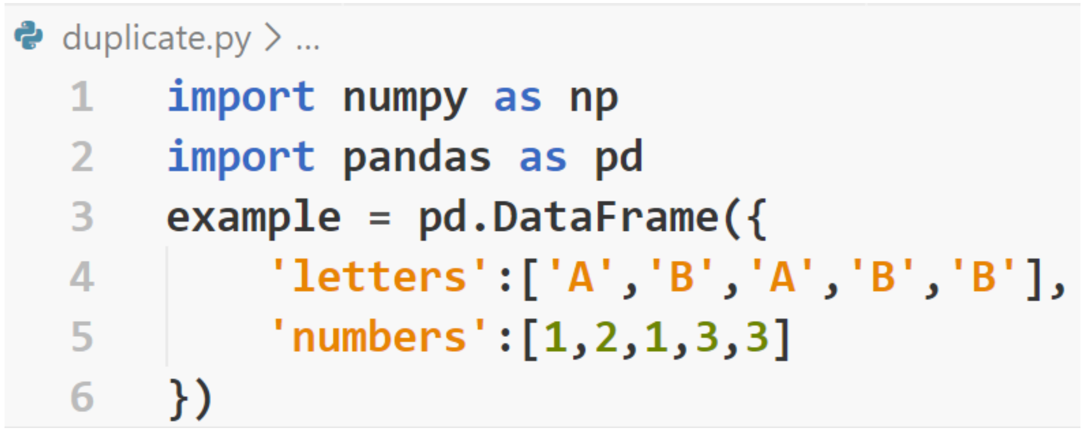
	ColA	ColB
0	1.0	1
1	NaN	1
2	NaN	1
3	4.0	1
4	5.0	2
5	6.0	2
6	7.0	2



	ColA	ColB
0	1.0	1
1	4.0	1
2	4.0	1
3	4.0	1
4	5.0	2
5	6.0	2
6	7.0	2

重复值

- 判断是否有重复值
- 删去重复值



duplicate.py > ...

```python
import numpy as np
import pandas as pd
example = pd.DataFrame({
    'letters':['A','B','A','B','B'],
    'numbers':[1,2,1,3,3]
})
```

```
example
```

	letters	numbers
0	A	1
1	B	2
2	A	1
3	B	3
4	B	3

```
example.duplicated()
```

```
0    False
1    False
2     True
3    False
4     True
dtype: bool
```

```
example.drop_duplicates()
```

	letters	numbers
0	A	1
1	B	2
3	B	3

24.duplicate.py

肉品种类标示

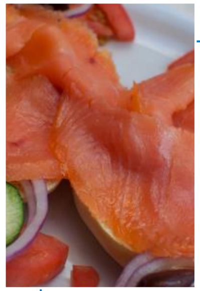
Bacon
培根

Pulled Pork
手撕猪肉

Pastrami
烟熏牛肉

Corned Beef
粗盐腌牛肉

Honey Ham
蜜汁火腿

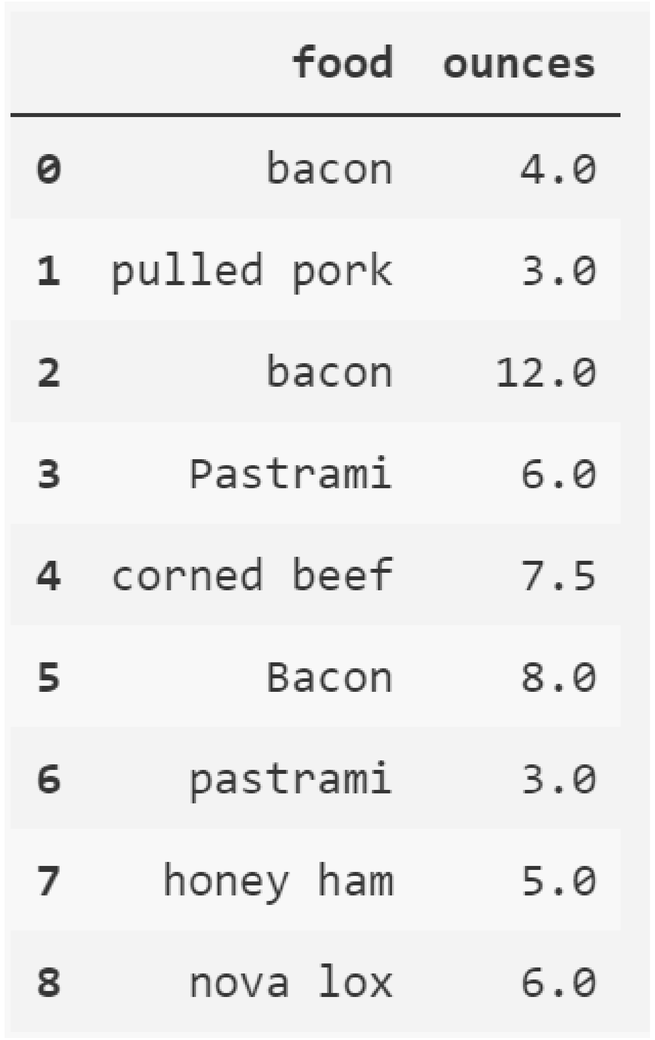


Nova Lox
盐渍鲑鱼

肉品种类标示

- 食物名称、重量

```python
import numpy as np
import pandas as pd

df = pd.DataFrame(
    {
        'food':['bacon','pulled pork','bacon',
                'Pastrami','corned beef','Bacon',
                'pastrami','honey ham', 'nova lox'],
        'ounces':[4,3,12,6,7.5,8,3,5,6]
    }
)
```



	food	ounces
0	bacon	4.0
1	pulled pork	3.0
2	bacon	12.0
3	Pastrami	6.0
4	corned beef	7.5
5	Bacon	8.0
6	pastrami	3.0
7	honey ham	5.0
8	nova lox	6.0

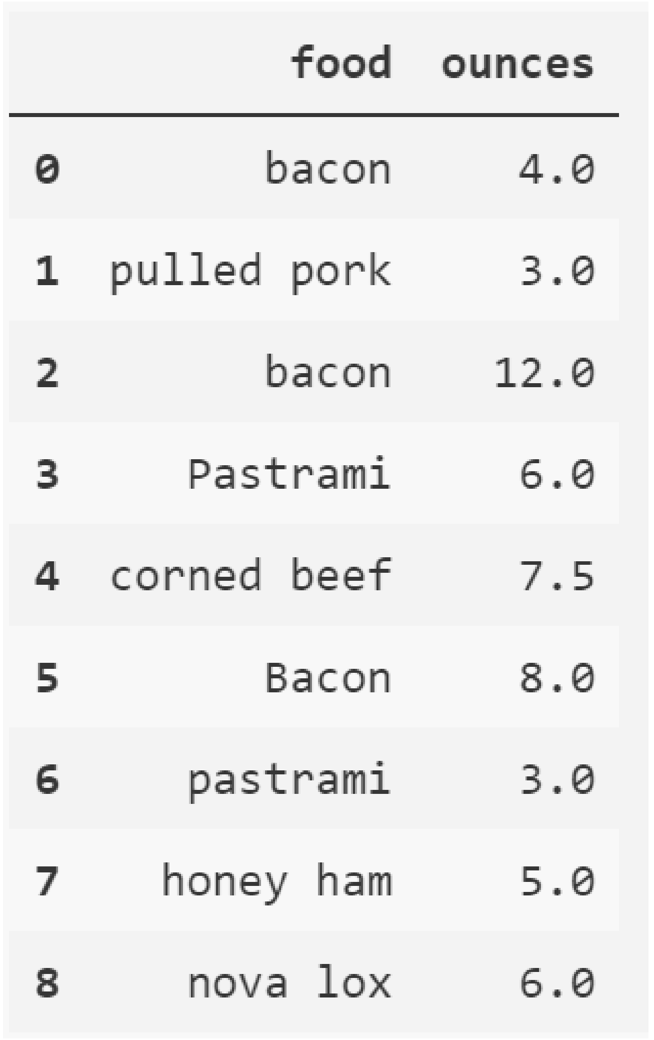
肉品种类标示

- 食物名称的大小写修正一致

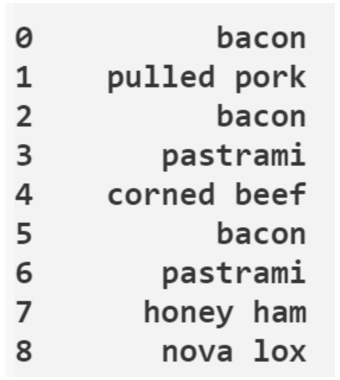
```
import numpy as np
import pandas as pd

df = pd.DataFrame(
    {
        'food':['bacon','pulled pork','bacon',
                'Pastrami','corned beef','Bacon',
                'pastrami','honey ham', 'nova lox'],
        'ounces':[4,3,12,6,7.5,8,3,5,6]
    }
)

lowercased = df['food'].str.lower()
lowercased
```

若要直接更新数据，可写成：

```
df['food'] = df['food'].str.lower()
df
```



	food	ounces
0	bacon	4.0
1	pulled pork	3.0
2	bacon	12.0
3	Pastrami	6.0
4	corned beef	7.5
5	Bacon	8.0
6	pastrami	3.0
7	honey ham	5.0
8	nova lox	6.0



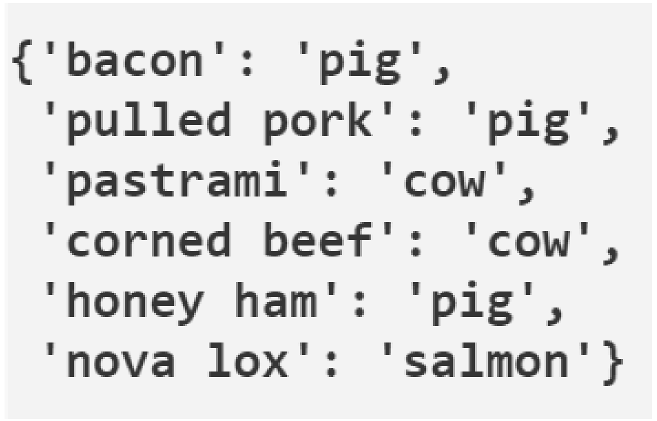
```
0          bacon
1    pulled pork
2          bacon
3       pastrami
4    corned beef
5          bacon
6       pastrami
7      honey ham
8       nova lox
```

lowercased

肉品种类标示

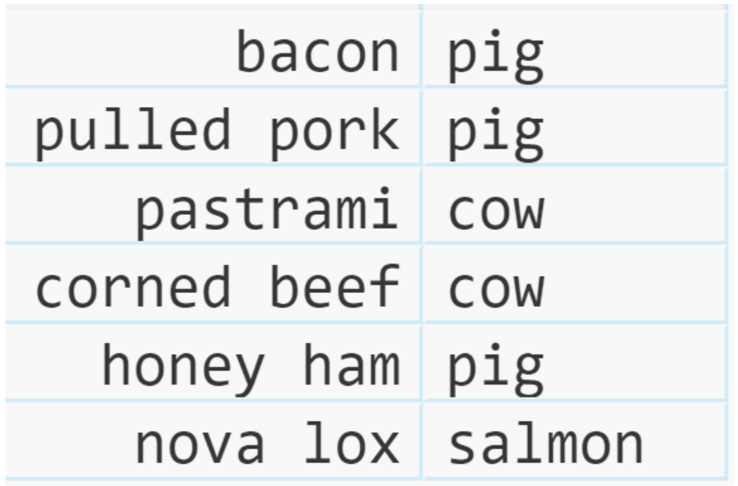
- 建立一个食物名称与肉品种类的dictionary

```python
meat_to_animal = {
     'bacon':'pig',
     'pulled pork':'pig',
     'pastrami':'cow',
     'corned beef':'cow',
     'honey ham':'pig',
     'nova lox':'salmon'
}

meat_to_animal
type(meat_to_animal)
```



```
{'bacon': 'pig',
 'pulled pork': 'pig',
 'pastrami': 'cow',
 'corned beef': 'cow',
 'honey ham': 'pig',
 'nova lox': 'salmon'}
```

dict



bacon	pig
pulled pork	pig
pastrami	cow
corned beef	cow
honey ham	pig
nova lox	salmon

dict in VS Code Data Viewer

肉品种类标示

- 新增一个字段标示肉品种类，并利用已转成小写名称的dict与map方法来进行查找

```
df['animal'] = lowercased.map(meat_to_animal)
df
```

	food	ounces
0	bacon	4.0
1	pulled pork	3.0
2	bacon	12.0
3	Pastrami	6.0
4	corned beef	7.5
5	Bacon	8.0
6	pastrami	3.0
7	honey ham	5.0
8	nova lox	6.0

lowercased(food)

```
0          bacon
1    pulled pork
2          bacon
3       pastrami
4    corned beef
5          bacon
6       pastrami
7      honey ham
8       nova lox
```

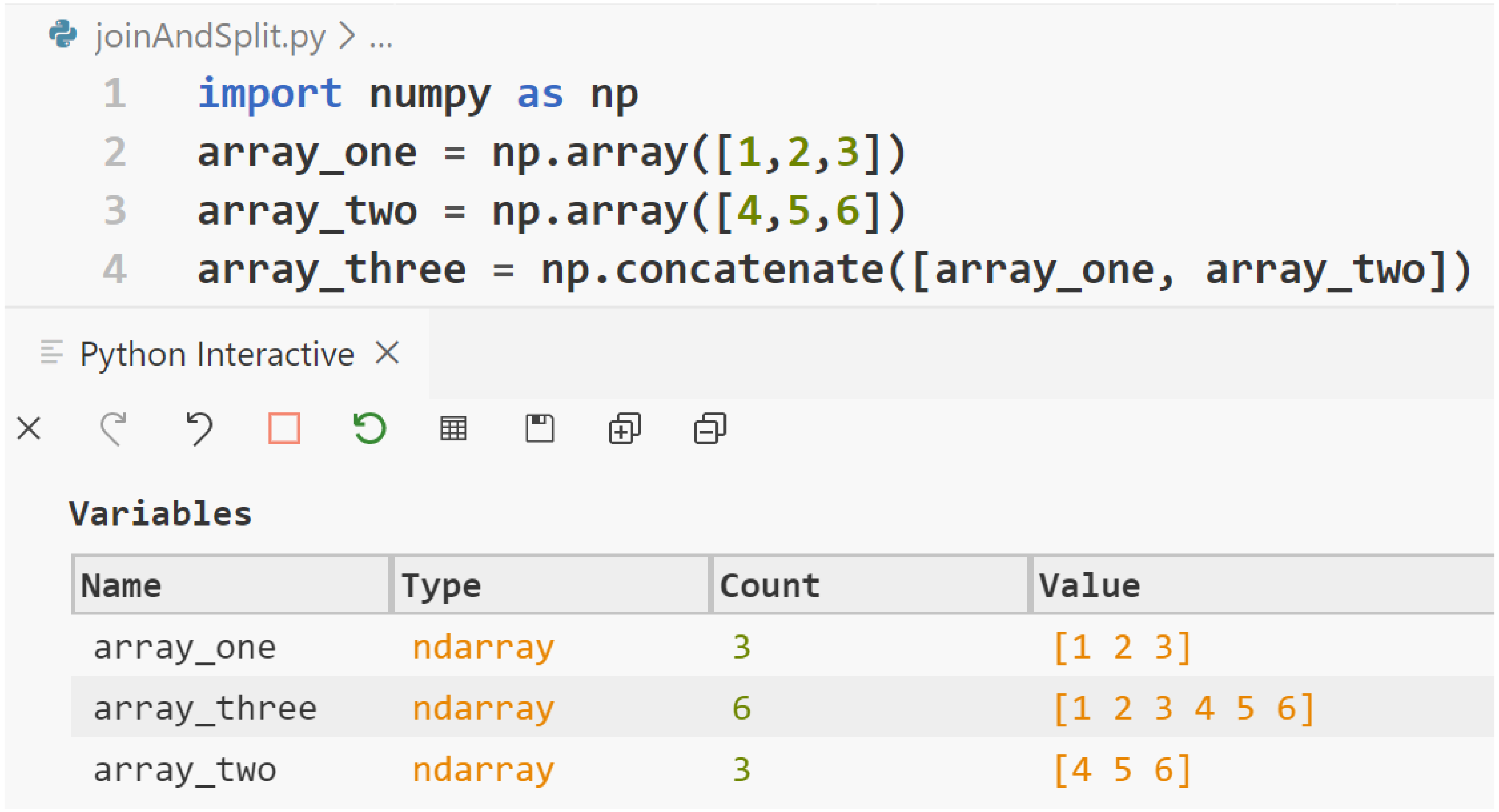
meat_to_animal

bacon	pig
pulled pork	pig
pastrami	cow
corned beef	cow
honey ham	pig
nova lox	salmon

	food	ounces	animal
0	bacon	4.0	pig
1	pulled pork	3.0	pig
2	bacon	12.0	pig
3	Pastrami	6.0	cow
4	corned beef	7.5	cow
5	Bacon	8.0	pig
6	pastrami	3.0	cow
7	honey ham	5.0	pig
8	nova lox	6.0	salmon

资料集合并(一) Numpy

concatenate()



joinAndSplit.py > ...

```
import numpy as np
array_one = np.array([1,2,3])
array_two = np.array([4,5,6])
array_three = np.concatenate([array_one, array_two])
```

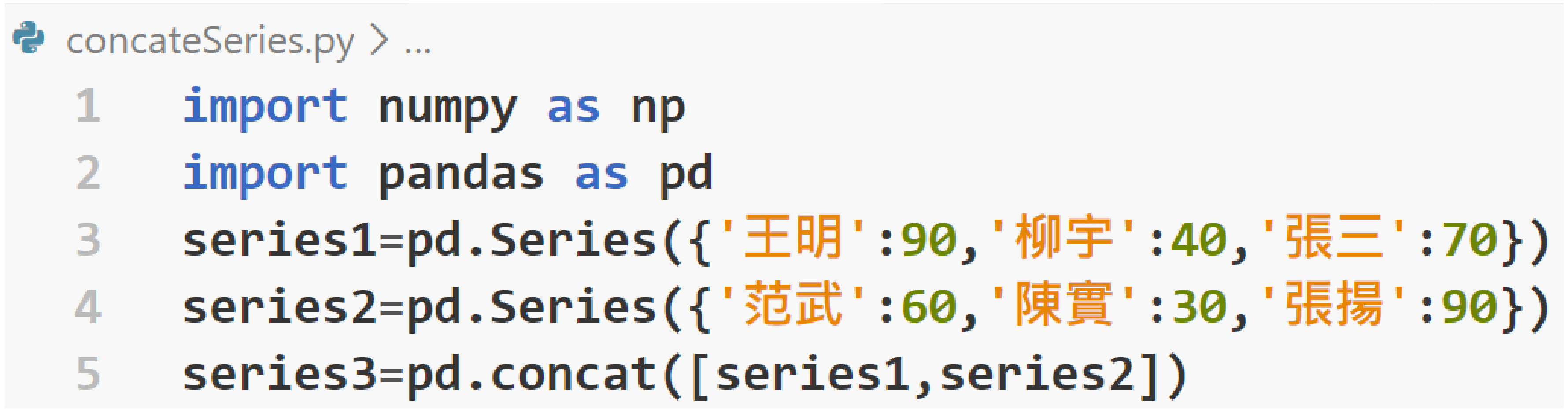
Python Interactive

Variables

Name	Type	Count	Value
array_one	ndarray	3	[1 2 3]
array_three	ndarray	6	[1 2 3 4 5 6]
array_two	ndarray	3	[4 5 6]

14.join_and_split.py

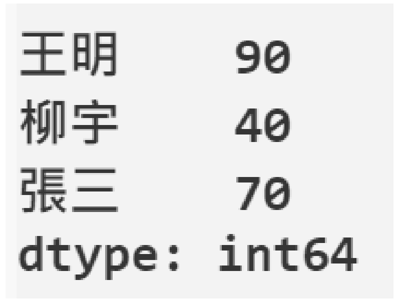
资料集合并(二) Pandas 序列合并



concateSeries.py > ...

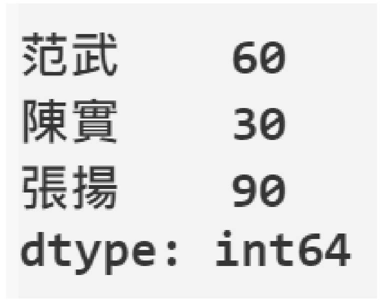
```python
import numpy as np
import pandas as pd
series1=pd.Series({'王明':90,'柳宇':40,'張三':70})
series2=pd.Series({'范武':60,'陳實':30,'張揚':90})
series3=pd.concat([series1,series2])
```

series1



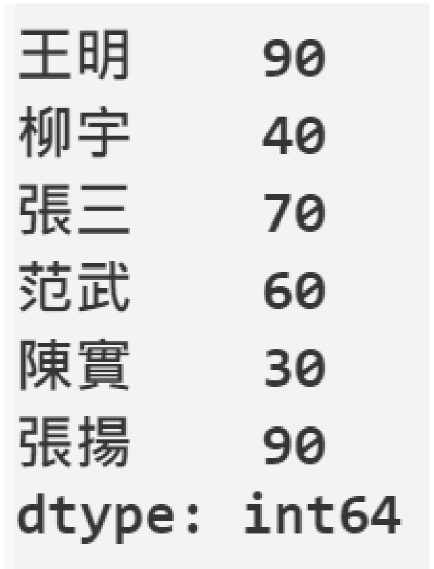
```
王明      90
柳宇      40
張三      70
dtype: int64
```

series2



```
范武      60
陳實      30
張揚      90
dtype: int64
```

series3

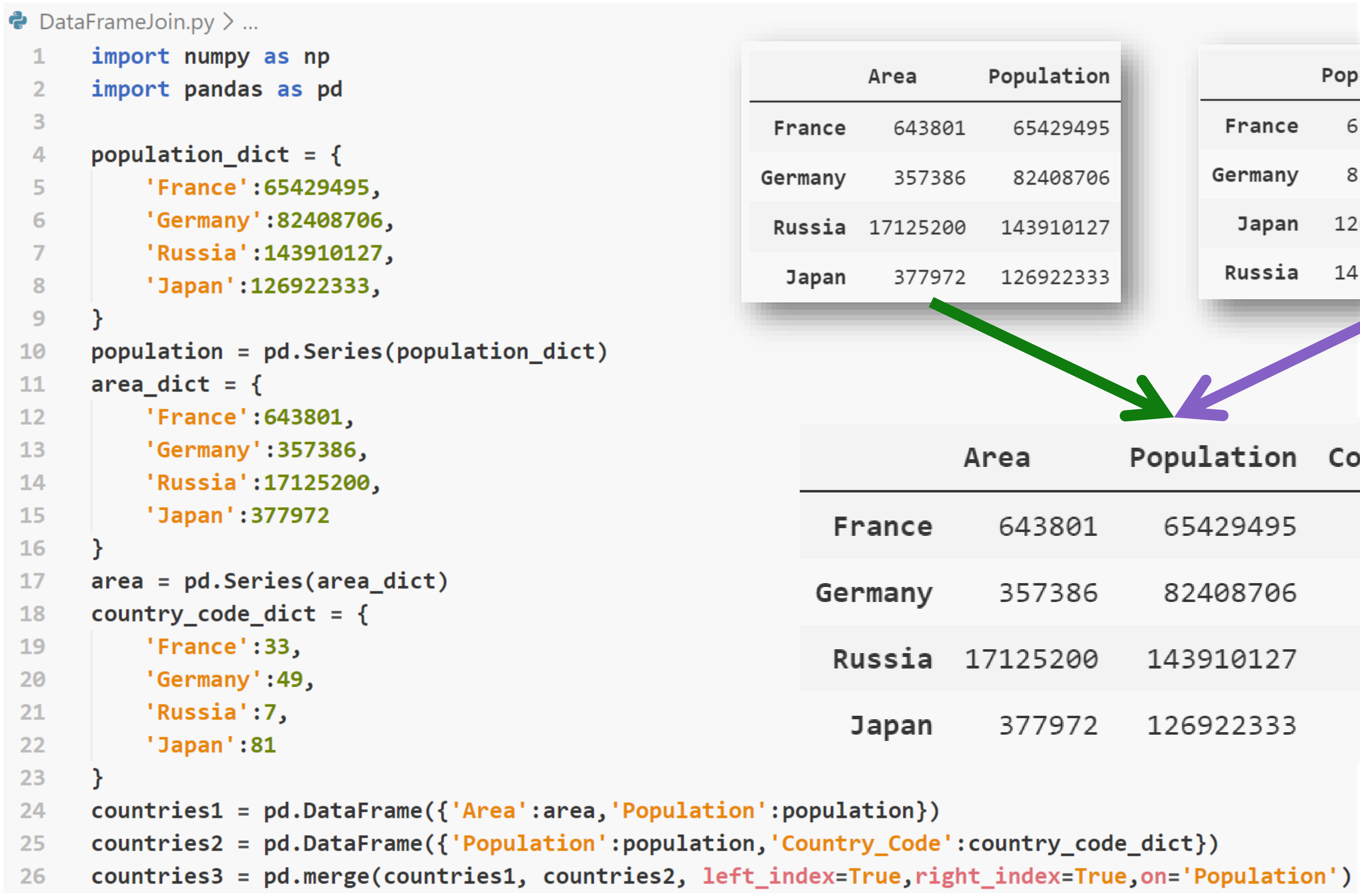


```
王明      90
柳宇      40
張三      70
范武      60
陳實      30
張揚      90
dtype: int64
```

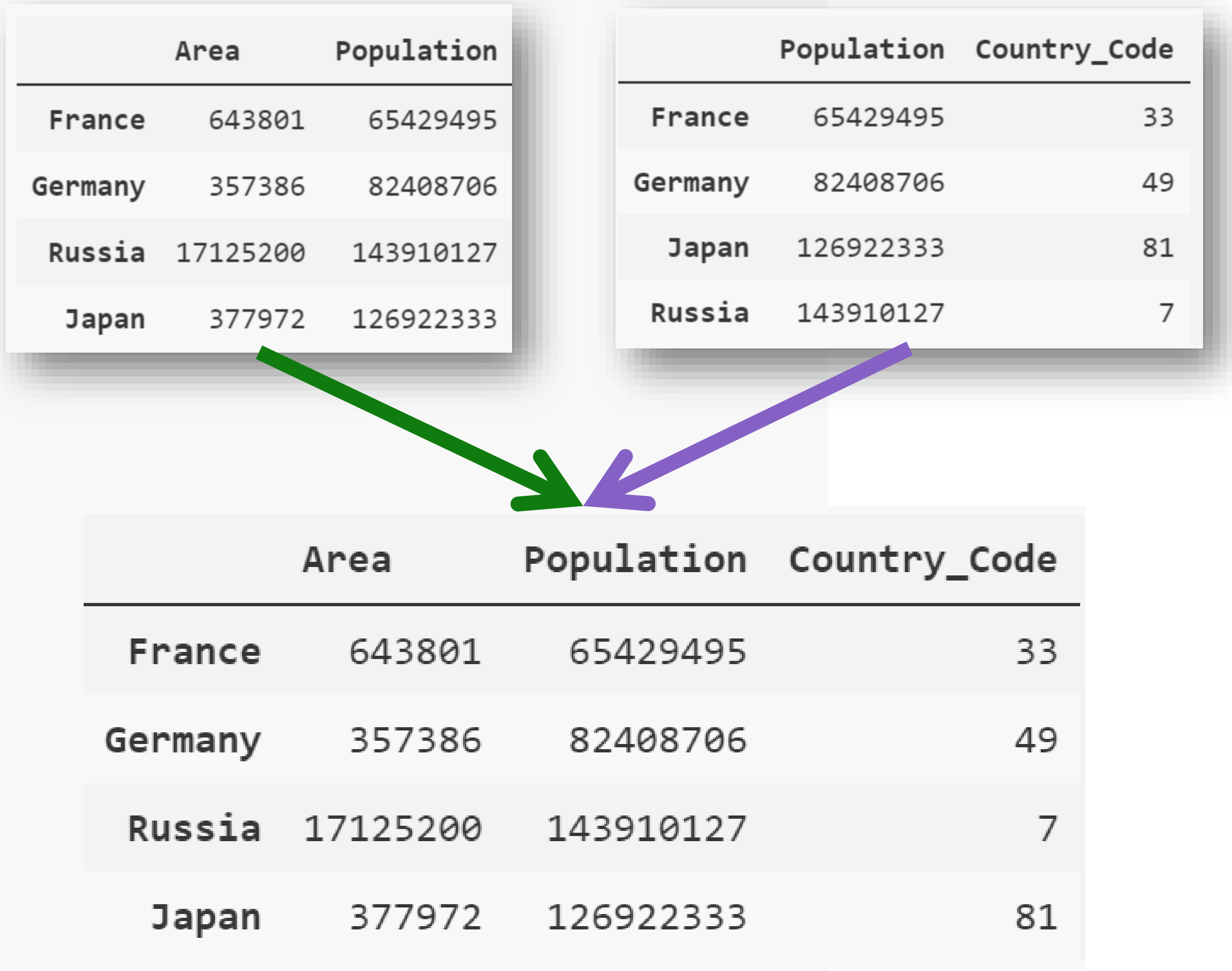
25.concate_series.py

资料集合并(三) Pandas DataFrame合并

26.dataframe_join.py


DataFrameJoin.py > ...

```python
import numpy as np
import pandas as pd

population_dict = {
    'France':65429495,
    'Germany':82408706,
    'Russia':143910127,
    'Japan':126922333,
}
population = pd.Series(population_dict)
area_dict = {
    'France':643801,
    'Germany':357386,
    'Russia':17125200,
    'Japan':377972
}
area = pd.Series(area_dict)
country_code_dict = {
    'France':33,
    'Germany':49,
    'Russia':7,
    'Japan':81
}
countries1 = pd.DataFrame({'Area':area,'Population':population})
countries2 = pd.DataFrame({'Population':population,'Country_Code':country_code_dict})
countries3 = pd.merge(countries1, countries2, left_index=True,right_index=True,on='Population')
```



	Area	Population
France	643801	65429495
Germany	357386	82408706
Russia	17125200	143910127
Japan	377972	126922333

	Population	Country_Code
France	65429495	33
Germany	82408706	49
Japan	126922333	81
Russia	143910127	7

	Area	Population	Country_Code
France	643801	65429495	33
Germany	357386	82408706	49
Russia	17125200	143910127	7
Japan	377972	126922333	81

Microsoft Reactor

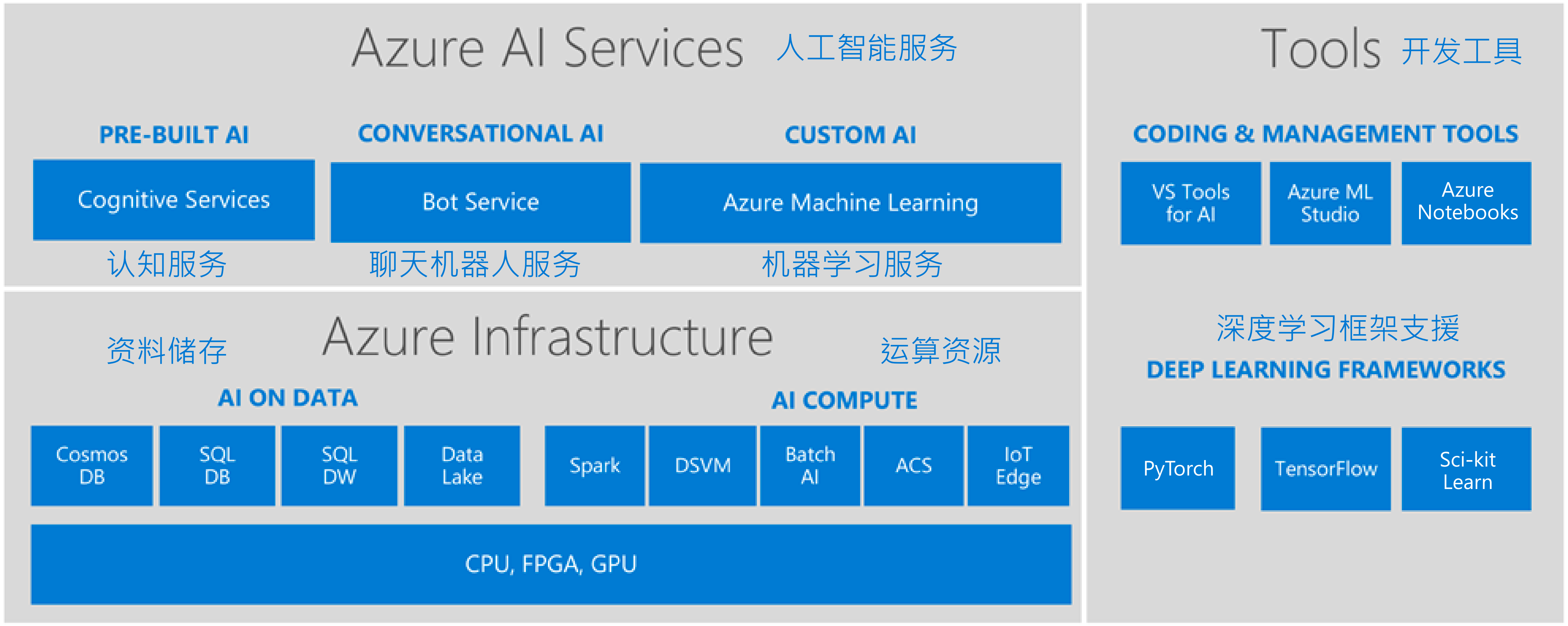
数据清理与维护

- 数据内容确认
  - 看个几笔资料 head()、tail()
  - 摘要信息 info()
  - 维度 shape
- 遗漏值处理
  - 确认遗漏值 isnull()
  - 移除 – 数据量允许情况最合适处理方式 dropna()
  - 填补 – 前后左右、平均值、频率最高、其他模型产生 (均为替代折衷方案) fillna()、ffill()、bfill()
- 资料集合并
  - Numpy concatenate()
  - Pandas 序列 concat()
  - Pandas DataFrame merge()

Microsoft

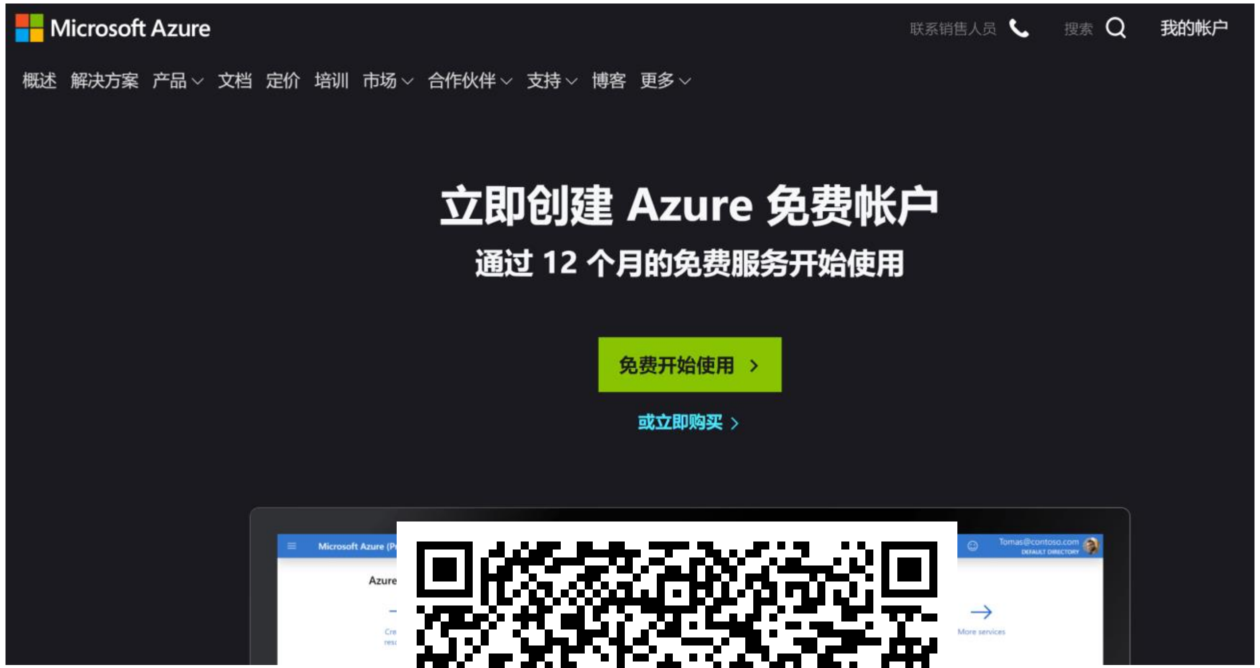
其他学习资源介绍

Microsoft AI Platform 微软人工智能平台



https://docs.microsoft.com/learn/modules/choose-data-science-option-in-azure/2-ml-options-on-azure

如何免费使用微软Azure服务



Microsoft Azure 联系销售人员 搜索 我的帐户

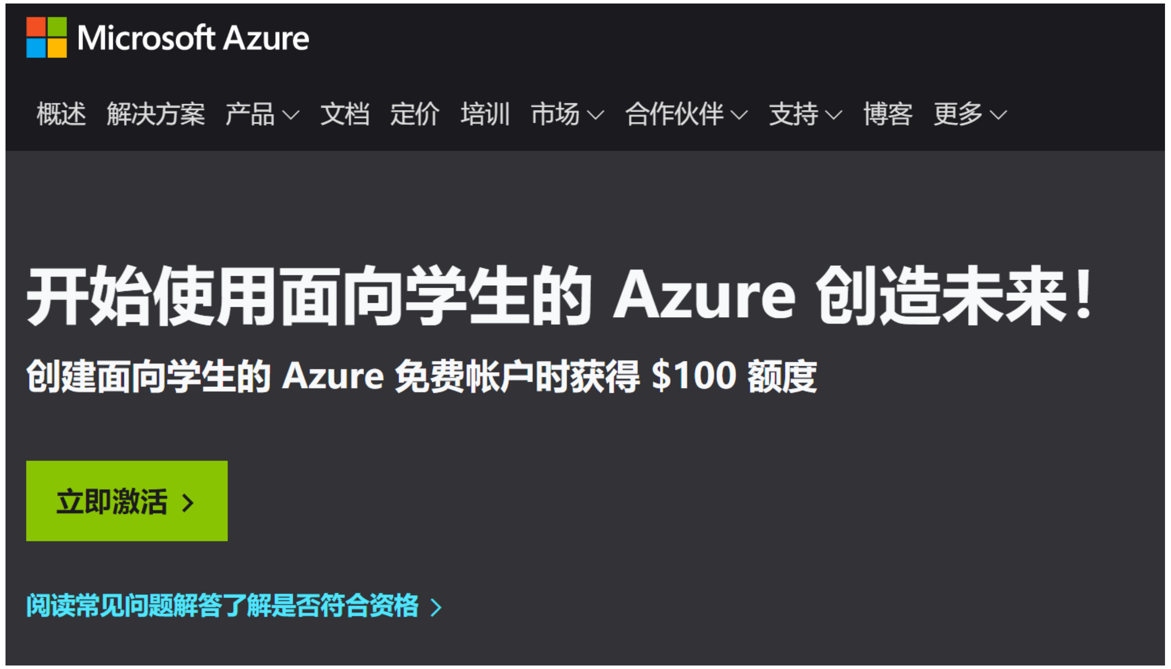
概述 解决方案 产品 文档 定价 培训 市场 合作伙伴 支持 博客 更多

立即创建 Azure 免费帐户

通过 12 个月的免费服务开始使用

免费开始使用 >

或立即购买 >



Microsoft Azure

概述 解决方案 产品 文档 定价 培训 市场 合作伙伴 支持 博客 更多

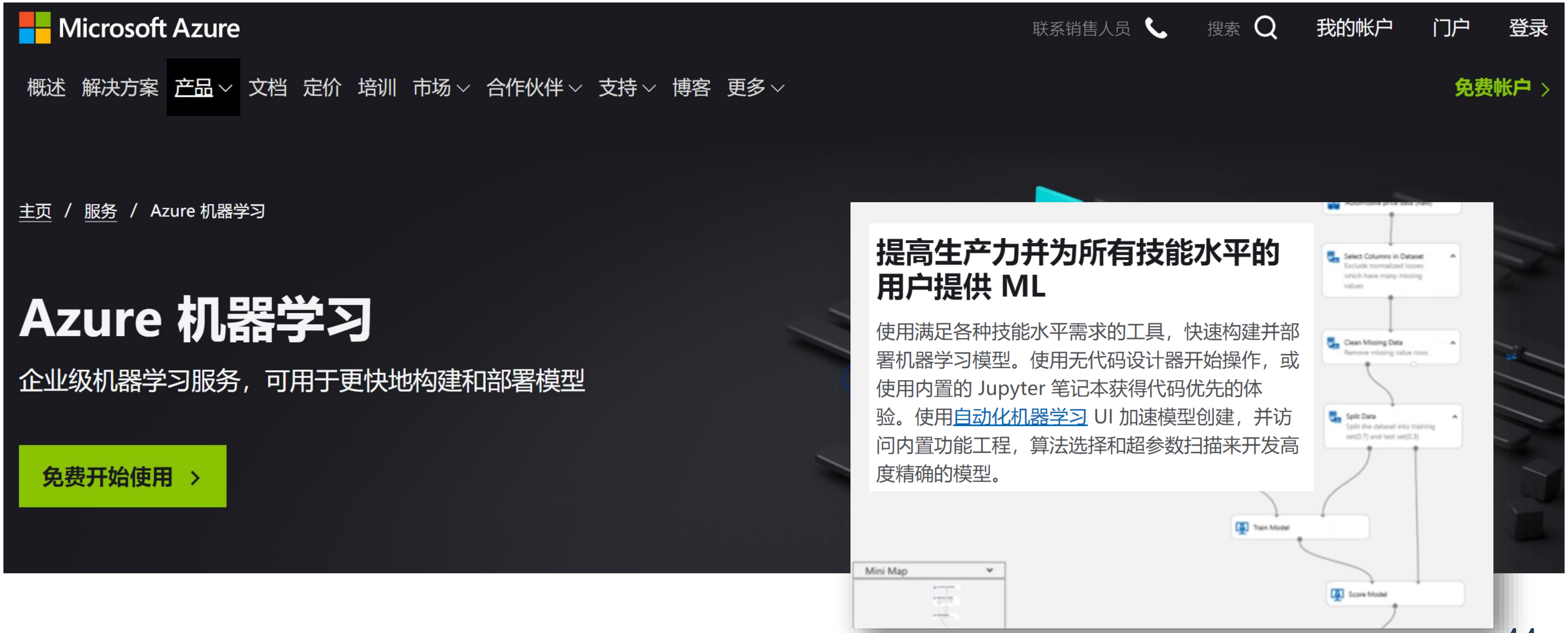
开始使用面向学生的 Azure 创造未来！

创建面向学生的 Azure 免费帐户时获得 $100 额度

立即激活 >

阅读常见问题解答了解是否符合资格 >

微软 Azure 机器学习服务



Azure 机器学习

企业级机器学习服务，可用于更快地构建和部署模型

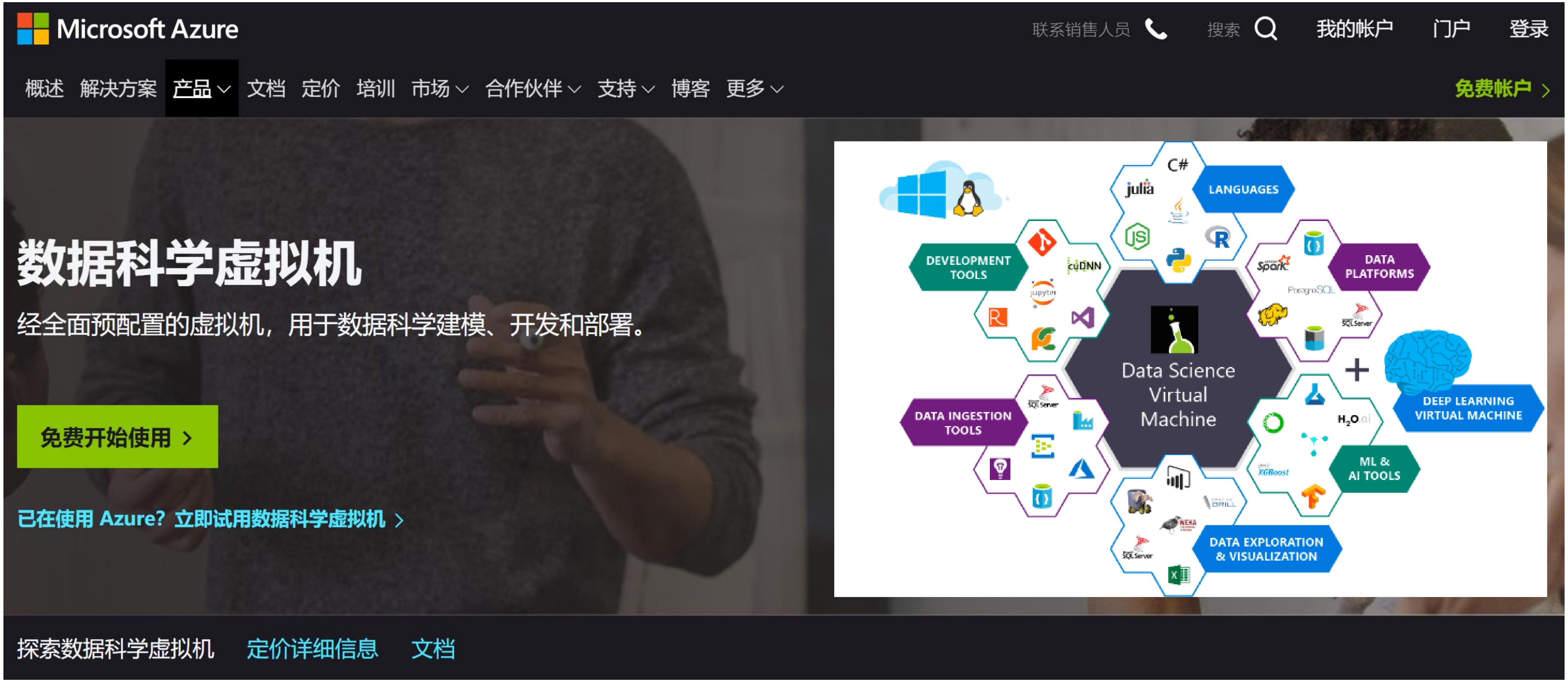
免费开始使用 >

提高生产力并为所有技能水平的用户提供 ML

使用满足各种技能水平需求的工具，快速构建并部署机器学习模型。使用无代码设计器开始操作，或使用内置的 Jupyter 笔记本获得代码优先的体验。使用自动化机器学习 UI 加速模型创建，并访问内置功能工程，算法选择和超参数扫描来开发高度精确的模型。

https://azure.microsoft.com/zh-cn/services/machine-learning/

适用于各种情境之资料科学虚拟机器服务



数据科学虚拟机

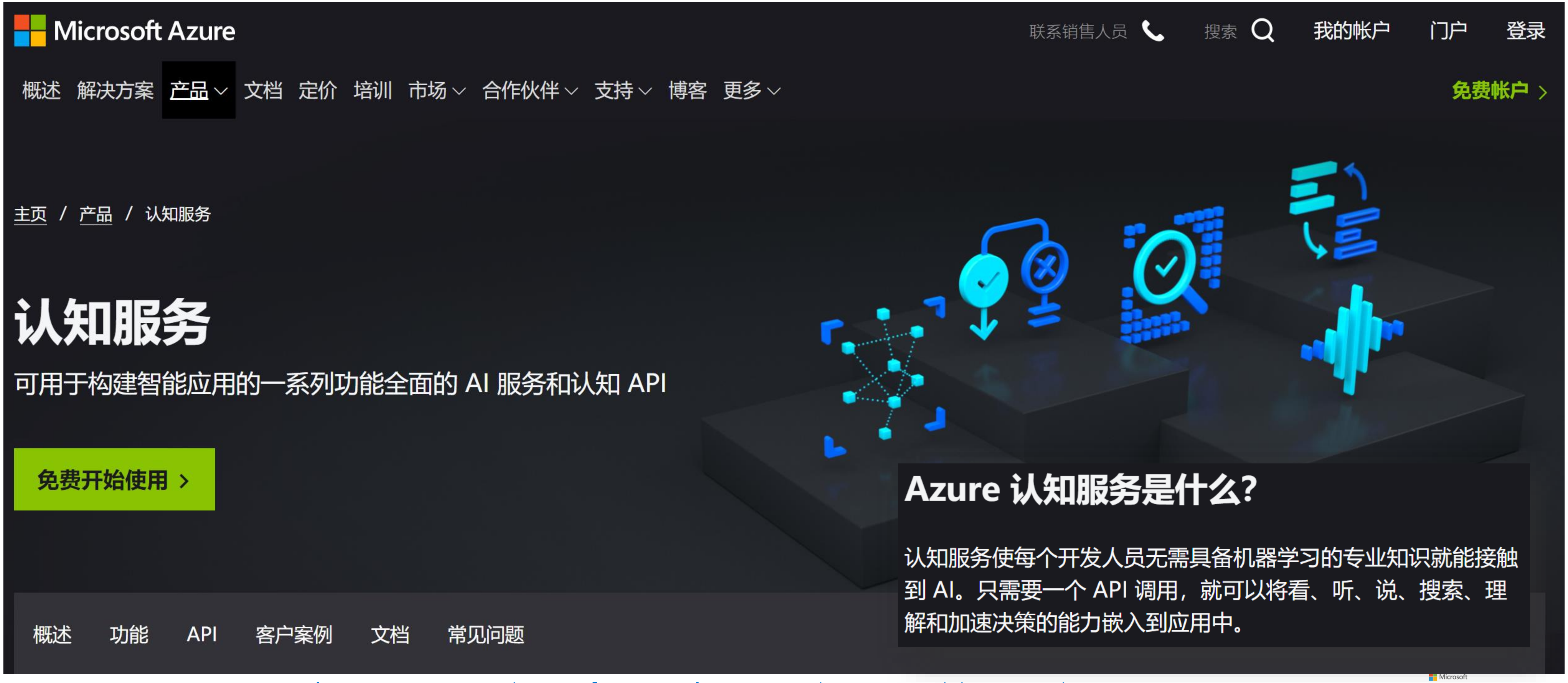
经全面预配置的虚拟机，用于数据科学建模、开发和部署。

已在使用 Azure？立即试用数据科学虚拟机 >

https://azure.microsoft.com/zh-cn/services/virtual-machines/data-science-virtual-machines/

微软 Azure 认知服务

马上可用、预先训练的AI模型



Microsoft Azure

联系销售人员 搜索 我的帐户 门户 登录

概述 解决方案 产品 文档 定价 培训 市场 合作伙伴 支持 博客 更多 免费帐户

主页 / 产品 / 认知服务

认知服务

可用于构建智能应用的一系列功能全面的 AI 服务和认知 API

免费开始使用 >

Azure 认知服务是什么?

认知服务使每个开发人员无需具备机器学习的专业知识就能接触到 AI。只需要一个 API 调用，就可以将看、听、说、搜索、理解和加速决策的能力嵌入到应用中。

概述 功能 API 客户案例 文档 常见问题

https://azure.microsoft.com/zh-cn/services/cognitive-services/

Microsoft Reactor

更多延伸学习资源

- 微软 - 线上学习
  - 在Azure Notebooks中使用Python实作机器学习
    docs.microsoft.com/learn/paths/intro-to-ml-with-python/
  - 在Azure中体验资料科学服务如何为人工智能解决方案加值
    docs.microsoft.com/learn/paths/explore-data-science-tools-in-azure/
  - 使用Azure机器学习服务打造人工智能解决方案
    docs.microsoft.com/learn/paths/build-ai-solutions-with-azure-ml-service/
  - 使用Azure资料科学虚拟机器实作机器学习
    docs.microsoft.com/learn/paths/get-started-with-azure-dsvm/
  - Azure入门
    docs.microsoft.com/learn/paths/azure-fundamentals/

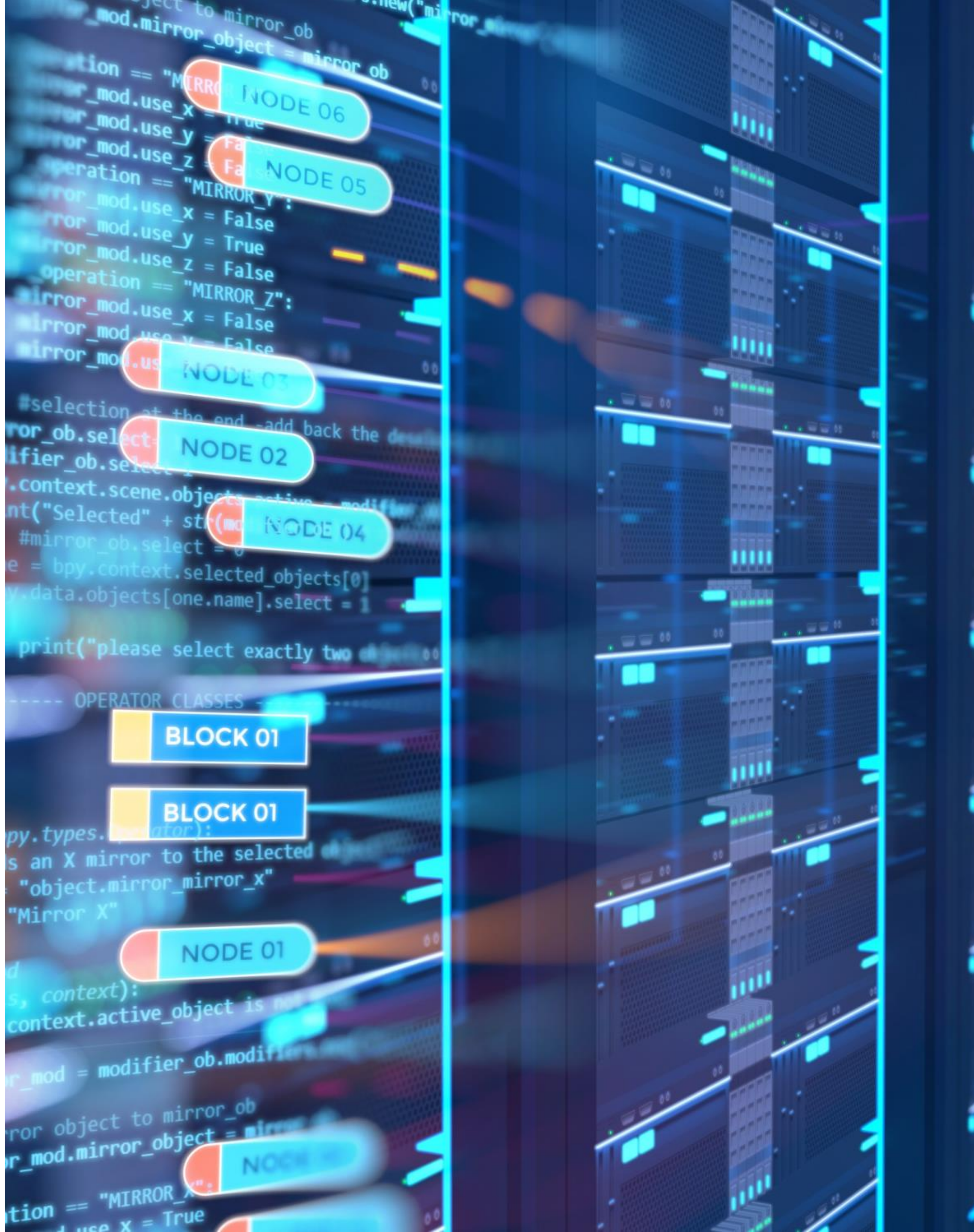
微软认证

资料科学系列：

- Azure Data Engineer Associate
- Azure Data Scientist Associate
- Azure Developer Associate

还有：

- Azure Fundamentals
- Azure AI Engineer Associate



Microsoft

Reactor

developer.microsoft.com/reactor/
@MSFTReactor on Twitter

Microsoft

议程结束
感谢聆听

请记得填写课程回馈问卷
https://aka.ms/ReactorFeedback