Microsoft

Introduction to Python for Data Science

资料科学入门 – 使用Python程序语言 Numpy & Pandas

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

Getting started with NumPy and Pandas NumPy与Pandas入门

19:30	Welcome 开场
19:35	NumPy Arrays 数组操作(建立/索引/转型/切割/排序)
20:10	Aggregations 整体分析
20:30	10-minute break 中场休息
20:40	Pandas Data Structures Pandas 数据结构 - 序列与DataFrame
21:20	Data Operations 数据操作(转换/取出/合成/缺失值填补)
21:30	Event end 研讨会结束

Microsoft

NumPy 入门 Numerical Python

Section 2 第二节

Section 2 overview 第二节综览

- Built-in help 温馨提示
- NumPy arrays
  - Creating 建立
  - Attributes 属性
  - Indexing 索引
  - Reshaping 重塑转型
  - Splitting and joining 切割与组合
  - Fancy indexing 选择性取出
  - Sorting 排序
- Aggregations

Section 2 overview 第二节综览

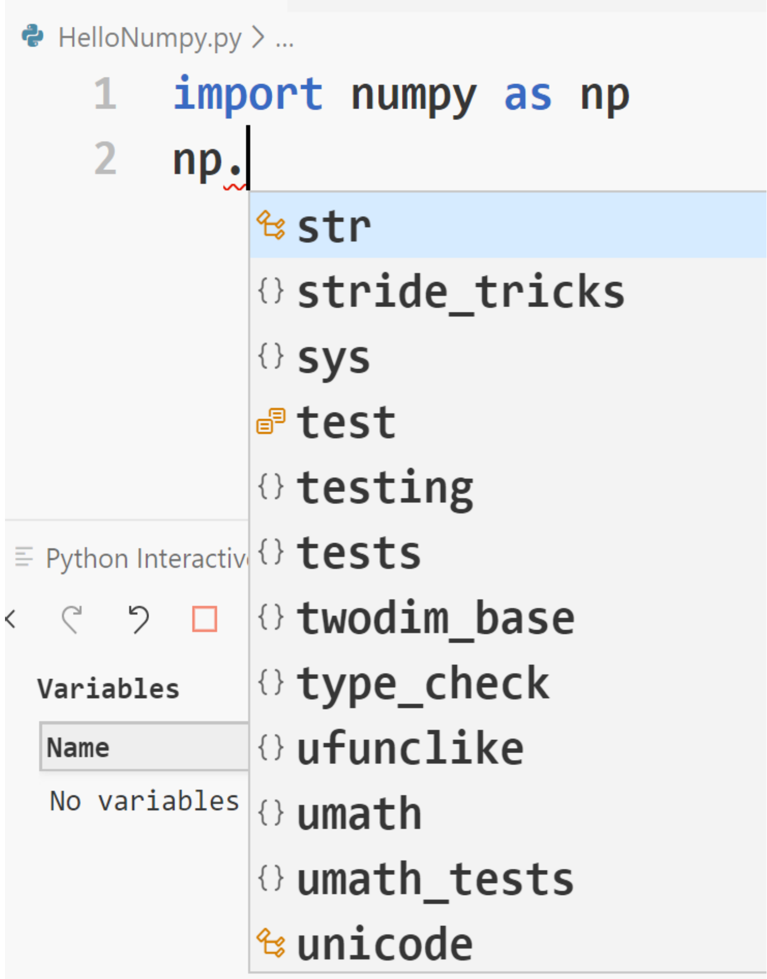
为什么要使用NumPy?

- 使用快速，函数已编译过
- 标准数学函式库，适合资料科学
- 是很多热门函式库(如pandas)的根基

**美国洛杉矶市使用NumPy来预测
房客遭遇非法驱逐的可能性**

VS Code 温馨提示

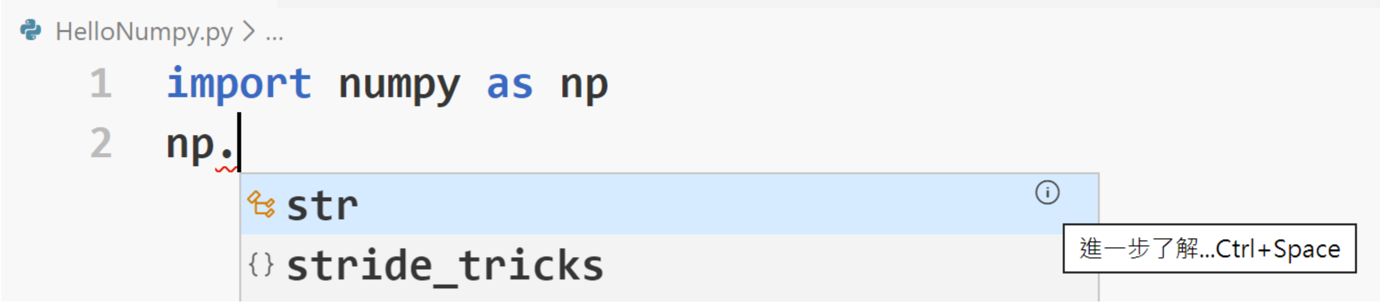
自动带出可使用的方法



HelloNumpy.py > ...

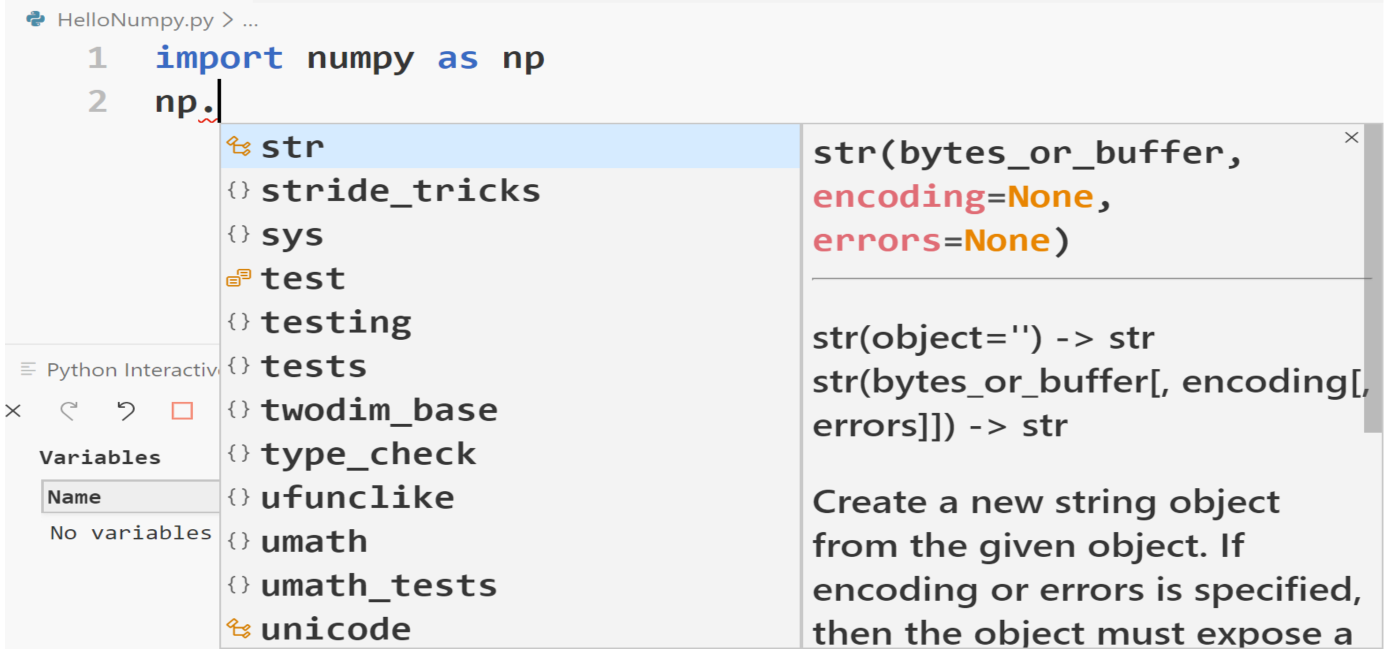
```
import numpy as np
np.
```

点击 ⓘ 进一步了解



HelloNumpy.py > ...

```
import numpy as np
np.
```



HelloNumpy.py > ...

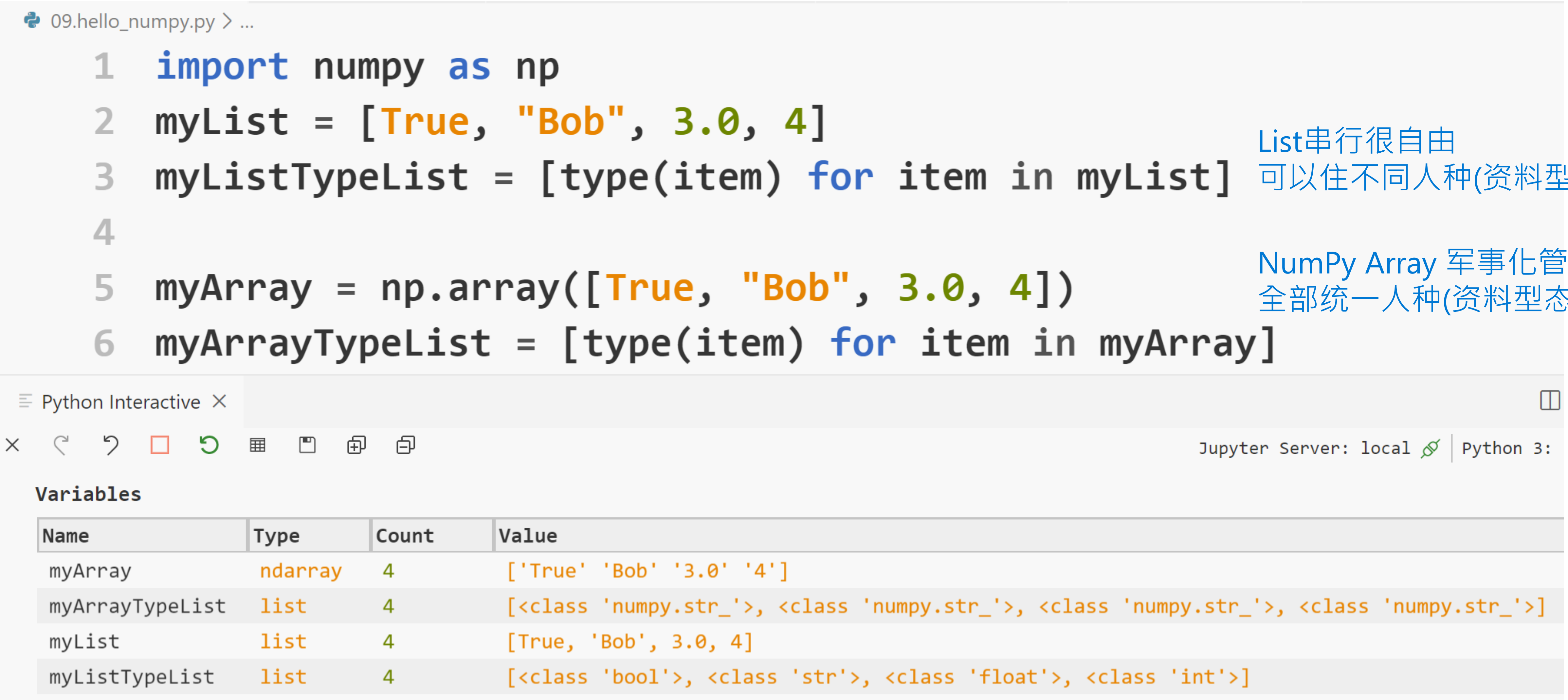
```
import numpy as np
np.
```

```
str(bytes_or_buffer,
encoding=None,
errors=None)
```

str(object='') -> str
str(bytes_or_buffer[, encoding[, errors]]) -> str

Create a new string object from the given object. If encoding or errors is specified, then the object must expose a

NumPy arrays – 给予Python List来建立



```python
import numpy as np
myList = [True, "Bob", 3.0, 4]
myListTypeList = [type(item) for item in myList]

myArray = np.array([True, "Bob", 3.0, 4])
myArrayTypeList = [type(item) for item in myArray]
```

Variables

Name	Type	Count	Value
myArray	ndarray	4	['True' 'Bob' '3.0' '4']
myArrayTypeList	list	4	[<class 'numpy.str_'>, <class 'numpy.str_'>, <class 'numpy.str_'>, <class 'numpy.str_'>]
myList	list	4	[True, 'Bob', 3.0, 4]
myListTypeList	list	4	[<class 'bool'>, <class 'str'>, <class 'float'>, <class 'int'>]

List串行很自由
可以住不同人种(资料型态)

NumPy Array 军事化管理
全部统一人种(资料型态)

09.hello_numpy.py

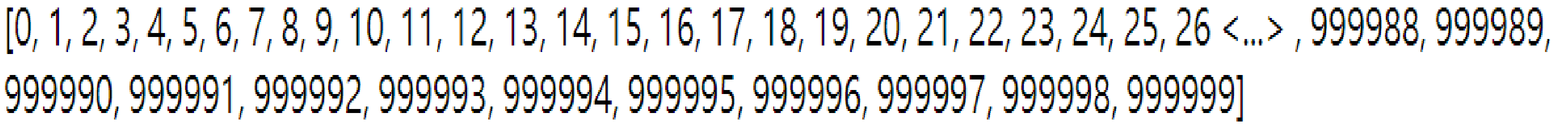
NumPy VS. Python List 运行时间比较

```
import numpy as np

my_list = list(range(1000000))
%time my_list2 = [x*2 for x in my_list]
```

```
Wall time: 86.8 ms
```

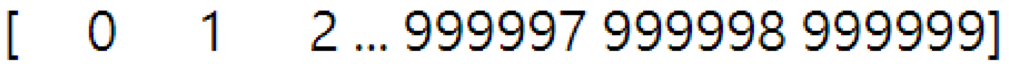
```
my_arr = np.array(my_list)
%time my_arr = my_arr*2
```

```
Wall time: 2.95 ms
```



```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 <...> , 999988, 999989,
999990, 999991, 999992, 999993, 999994, 999995, 999996, 999997, 999998, 999999]
```

my_list



```
[     0      1      2 ... 999997 999998 999999]
```

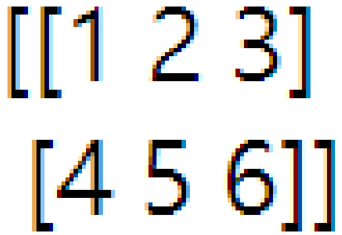
my_arr

numpy-time.py

NumPy的算术运算

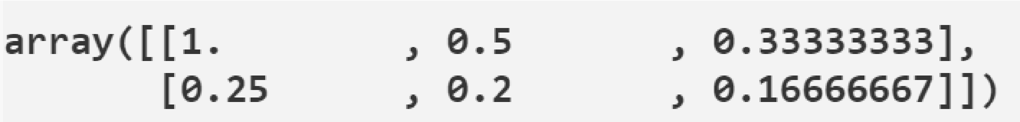
- 跟单一常数做运算

```
import numpy as np

arr = np.array([[1,2,3],[4,5,6]])
arr.dtype
```



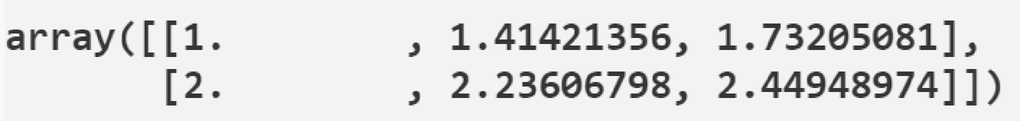
```
[[1 2 3]
 [4 5 6]]
```

```
1/arr
```



```
array([[1.        , 0.5       , 0.33333333],
       [0.25      , 0.2       , 0.16666667]])
```

```
arr**0.5
```

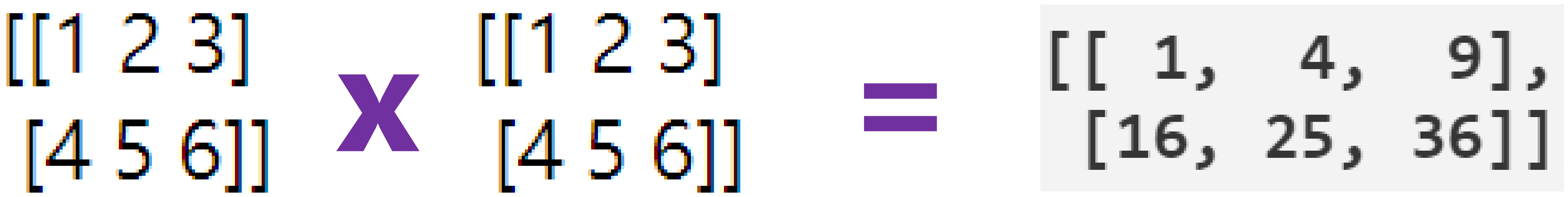


```
array([[1.        , 1.41421356, 1.73205081],
       [2.        , 2.23606798, 2.44948974]])
```

NumPy的算术运算

- 强项：整批运算，无须for loop

```
import numpy as np

arr = np.array([[1,2,3],[4,5,6]])
arr.dtype

arr*arr
```



```
[[1 2 3]
 [4 5 6]]
```

x

```
[[1 2 3]
 [4 5 6]]
```

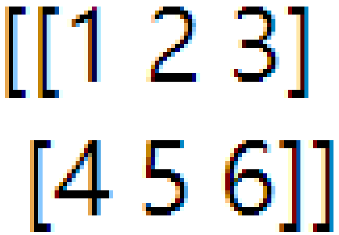
=

```
[[ 1,  4,  9],
 [16, 25, 36]]
```

NumPy的算术运算

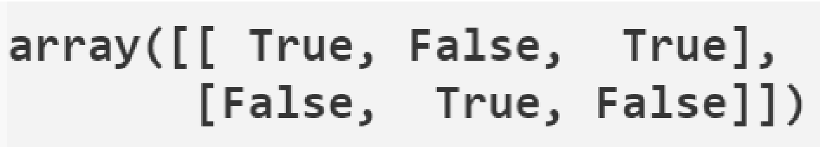
- 两个数组进行比较，产出布尔数组

```python
import numpy as np

arr  = np.array([[1,2,3],[4,5,6]])
arr2 = np.array([[0,4,1],[7,2,12]])

arr > arr2
```

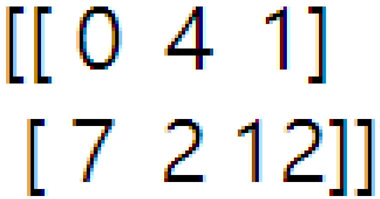


```
[[1 2 3]
 [4 5 6]]
```

arr



```
array([[ True, False,  True],
       [False,  True, False]])
```

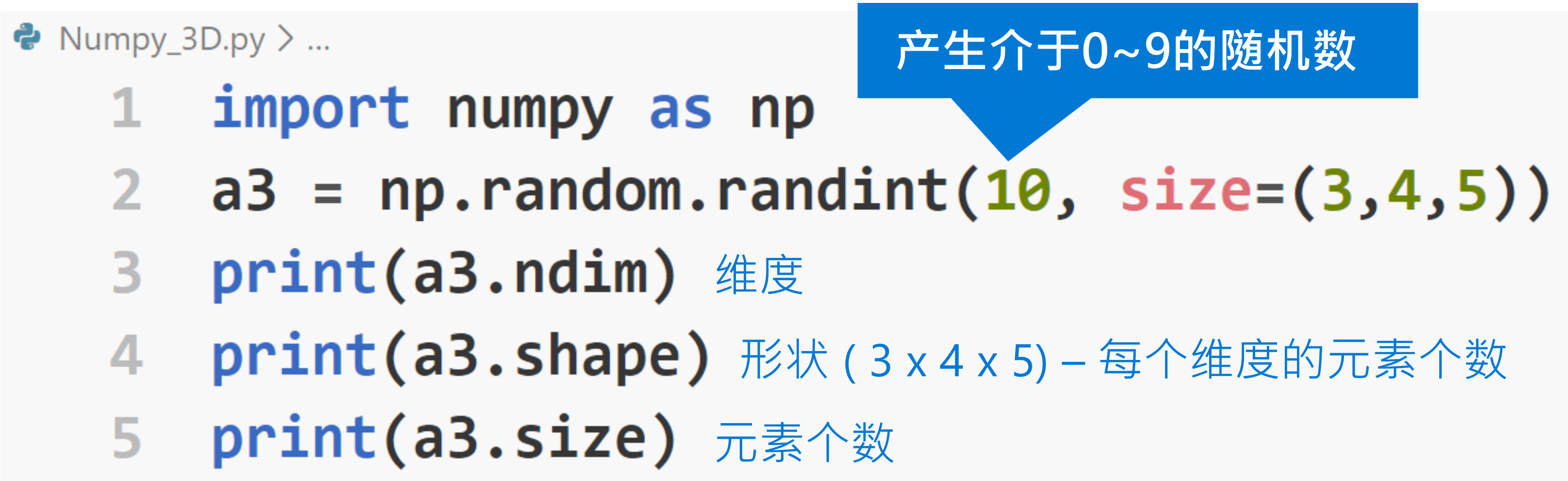


```
[[ 0  4  1]
 [ 7  2 12]]
```

arr2

NumPy array 阵列建立与常用属性

产生介于0~9的随机数



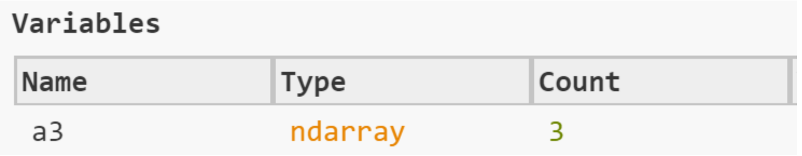
Numpy_3D.py > ...

```
import numpy as np
a3 = np.random.randint(10, size=(3,4,5))
print(a3.ndim)
print(a3.shape)
print(a3.size)
```

维度

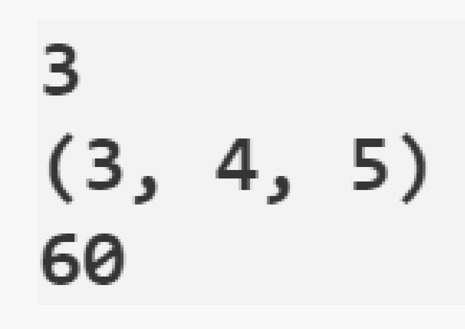
形状 (3 x 4 x 5) – 每个维度的元素个数

元素个数

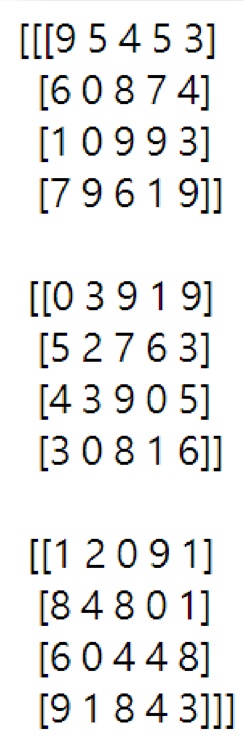


Variables

Name	Type	Count
a3	ndarray	3



```
3
(3, 4, 5)
60
```



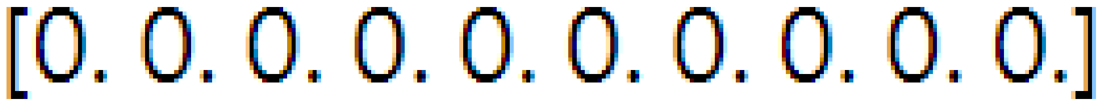
```
[[[9 5 4 5 3]
  [6 0 8 7 4]
  [1 0 9 9 3]
  [7 9 6 1 9]]

 [[0 3 9 1 9]
  [5 2 7 6 3]
  [4 3 9 0 5]
  [3 0 8 1 6]]

 [[1 2 0 9 1]
  [8 4 8 0 1]
  [6 0 4 4 8]
  [9 1 8 4 3]]]
```

```
#限定随机数起讫范围的写法(20~30)
a4 = np.random.randint(20,31, size=(3,4,5))
```

10.numpy_3d.py

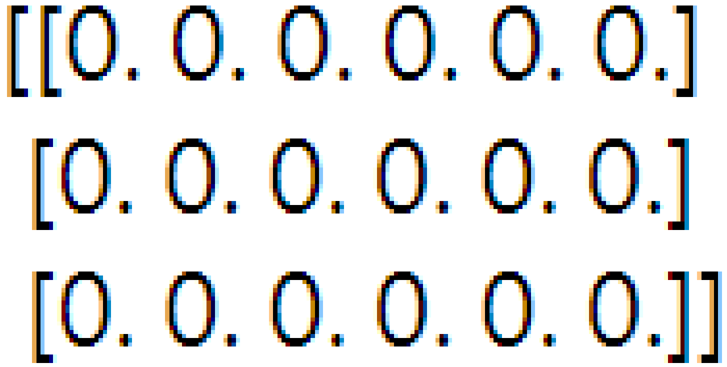
NumPy 建立内容全为0的数组

```python
import numpy as np
zero_array = np.zeros(10)
zero_2d_array = np.zeros((3,6))
```



```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

zero_array



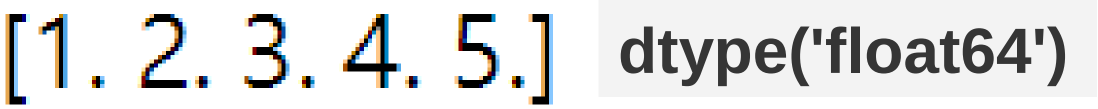
```
[[0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]]
```

zero_2d_array

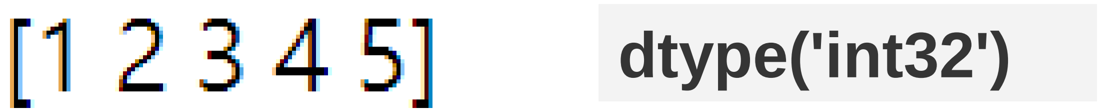
NumPy 数据型态观察

- dtype：数据型态
- astype：数据型态转换

```
import numpy as np

float_array = np.array([1.0,2.0,3.0,4.0,5.0])
float_array.dtype

int_array = float_array.astype(np.int32)
int_array.dtype
```



```
[1. 2. 3. 4. 5.]    dtype('float64')
```

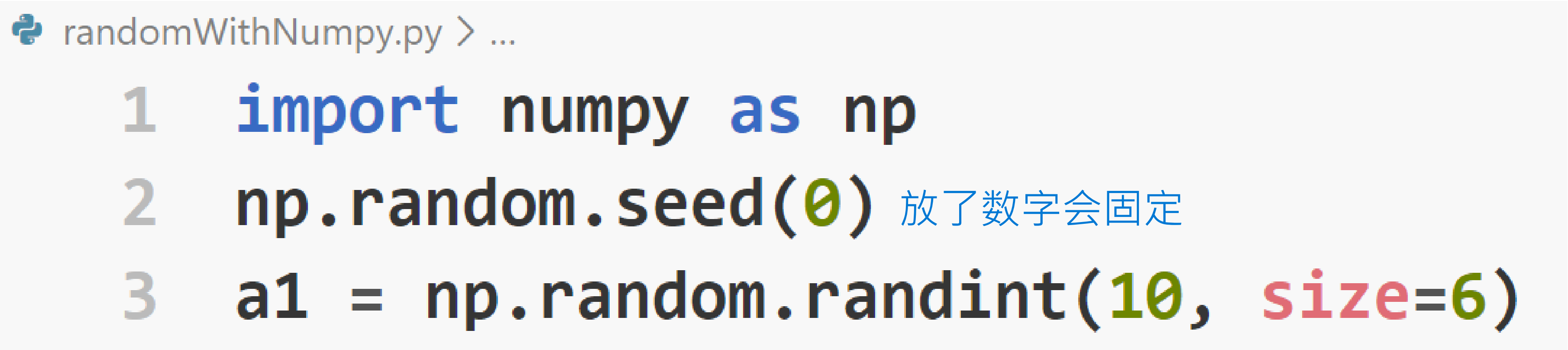


```
[1 2 3 4 5]    dtype('int32')
```

NumPy 常用数据型态 / 代码			
int32 / i4	int64 / i8	bool / ?	object / O
float32 / f4	float64 / f8	string_ / S10	unicode_ / U

自定义长度

NumPy array 索引取值

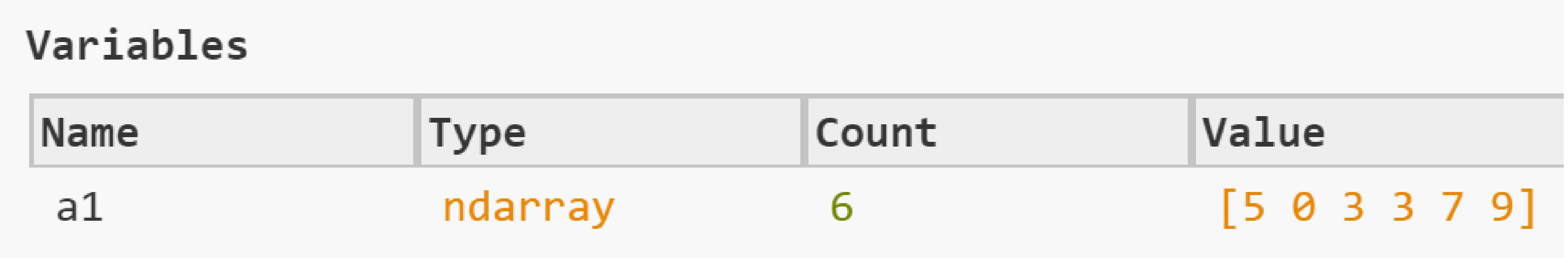


randomWithNumpy.py > ...

```python
import numpy as np
np.random.seed(0)
a1 = np.random.randint(10, size=6)
```

放了数字会固定

产生0~9的乱数共6个



Variables

Name	Type	Count	Value
a1	ndarray	6	[5 0 3 3 7 9]

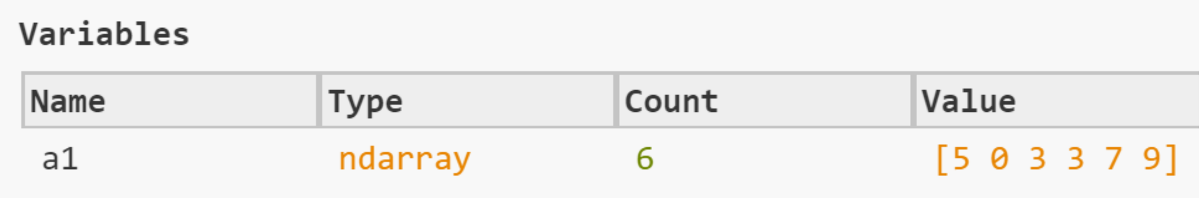
0	1	2	3	4	5
5	0	3	3	7	9

```python
print(a1[0])
print(a1[1])
print(a1[2])
print(a1[3])
print(a1[4])
print(a1[5])
```

11.random_with_numpy.py

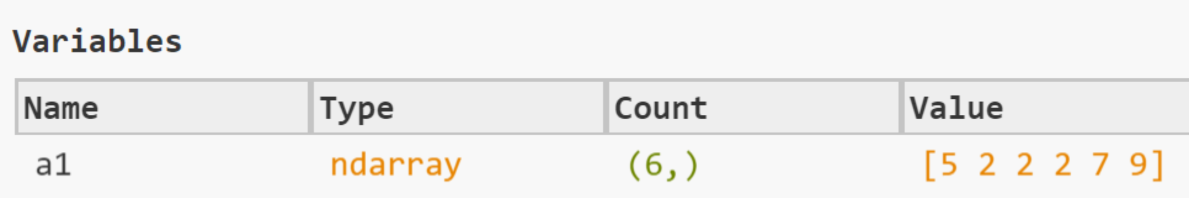
NumPy array 索引赋值

```
import numpy as np
np.random.seed(0)
a1 = np.random.randint(10, size=6)
a1[1:4]=2 #影响第1,2,3个元素
```



Variables

Name	Type	Count	Value
a1	ndarray	6	[5 0 3 3 7 9]



Variables

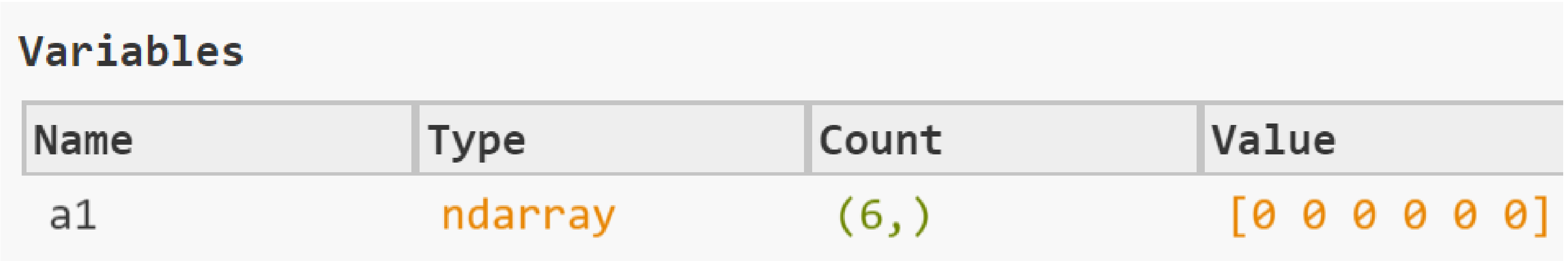
Name	Type	Count	Value
a1	ndarray	(6,)	[5 2 2 2 7 9]

0	1	2	3	4	5
5	0	3	3	7	9

0	1	2	3	4	5
5	2	2	2	7	9

NumPy array 索引赋值

```
import numpy as np
np.random.seed(0)
a1 = np.random.randint(10, size=6)
a1[1:4]=2 #影响第1,2,3个元素
a1[:] = 0 #影响所有元素
```

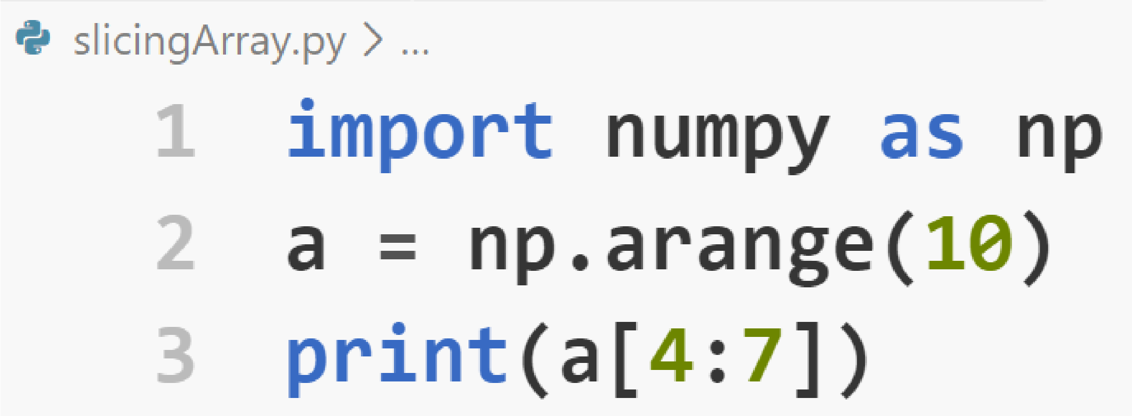


Variables

Name	Type	Count	Value
a1	ndarray	(6,)	[0 0 0 0 0 0]

0	1	2	3	4	5
0	0	0	0	0	0

NumPy array: slicing array 阵列切割



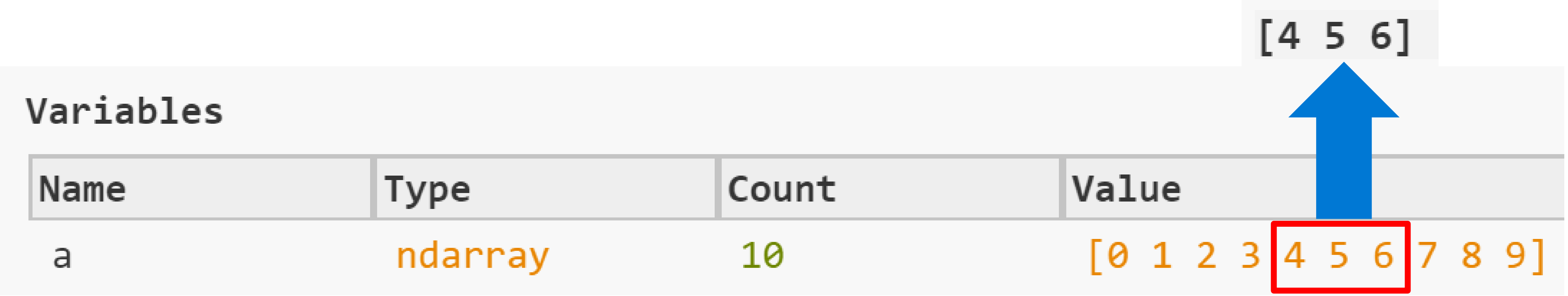
slicingArray.py > ...

```python
import numpy as np
a = np.arange(10)
print(a[4:7])
```

12.slicing_array.py

回传从 0(预设起始值)~9(结束值-1)的数字

取出阵列部分元素变成另一个阵列

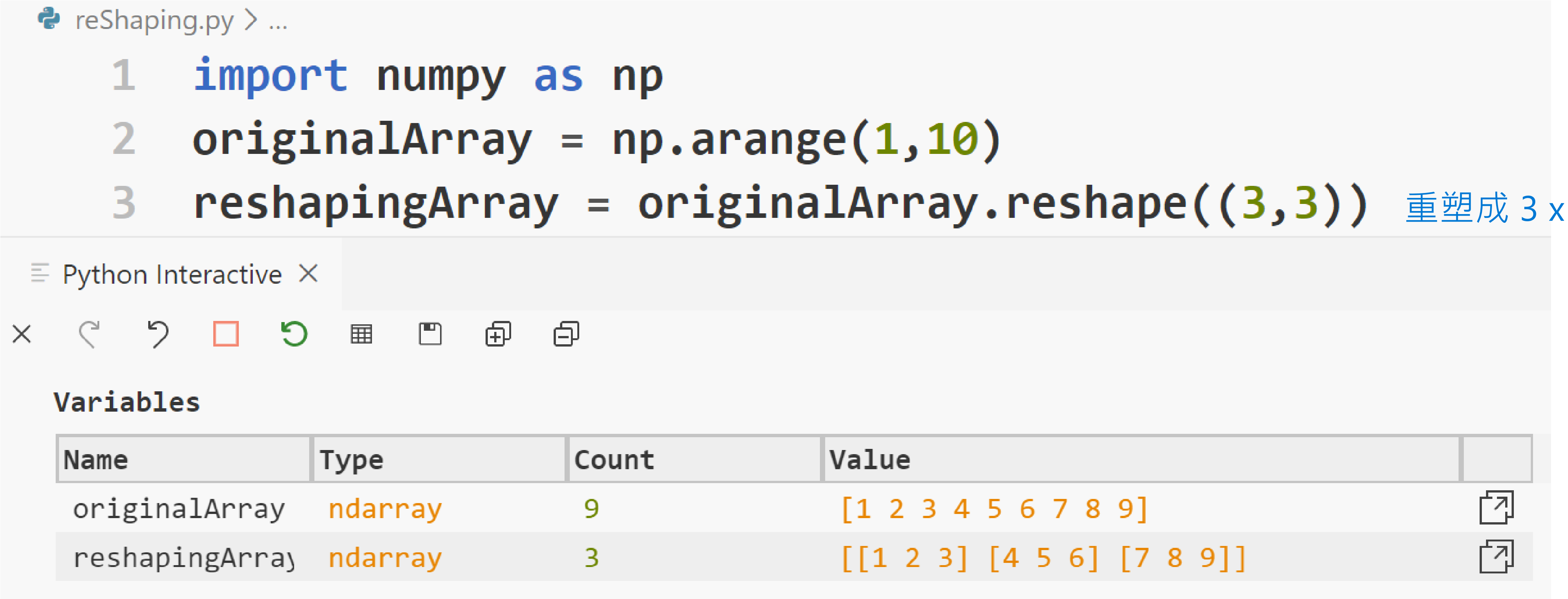


```
[4 5 6]
```

Variables

Name	Type	Count	Value
a	ndarray	10	[0 1 2 3 4 5 6 7 8 9]

NumPy array: reshaping 重塑转型

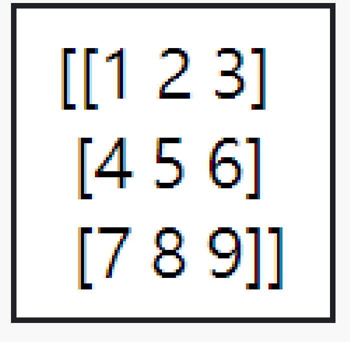


```python
import numpy as np
originalArray = np.arange(1,10)
reshapingArray = originalArray.reshape((3,3))
```

重塑成 3 x 3

Variables

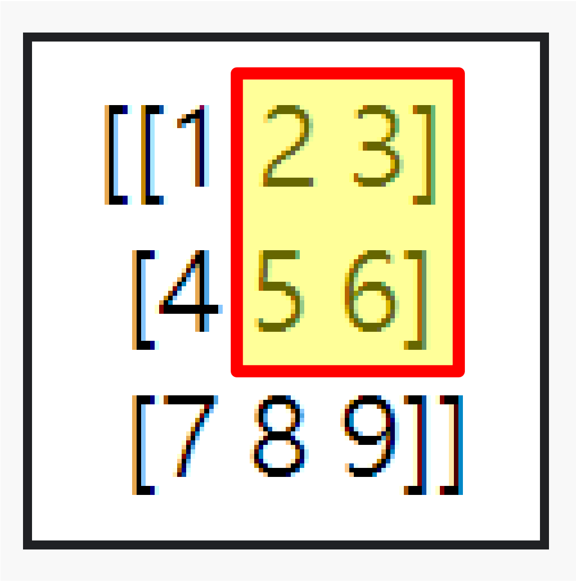
Name	Type	Count	Value
originalArray	ndarray	9	[1 2 3 4 5 6 7 8 9]
reshapingArray	ndarray	3	[[1 2 3] [4 5 6] [7 8 9]]



```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

13.reshaping.py

NumPy array 九宫格取值练习

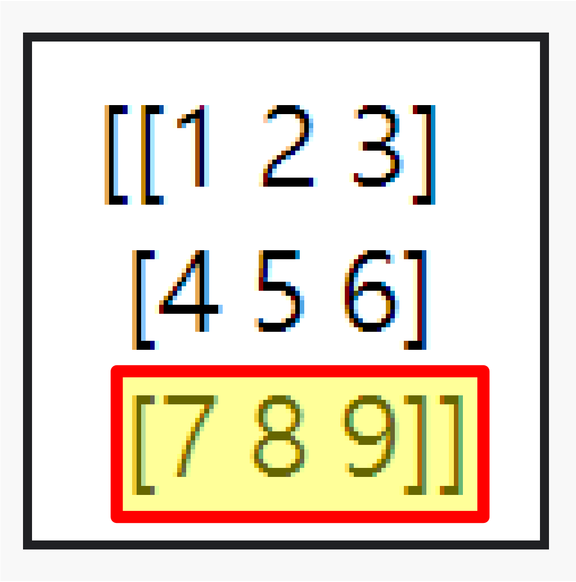


```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

第0列跟第1列的第1个元素跟第2个元素

reshapingArray[:2, 1:]

NumPy array 九宫格取值练习

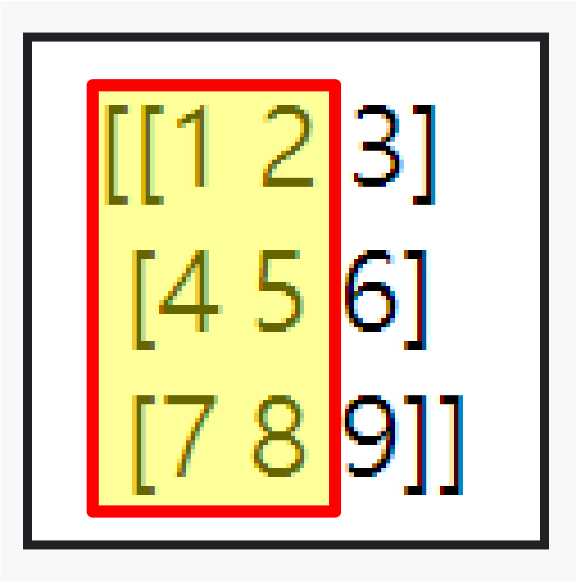


```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

第2列的所有元素

```
reshapingArray[2]
reshapingArray[2, : ]
```

NumPy array 九宫格取值练习

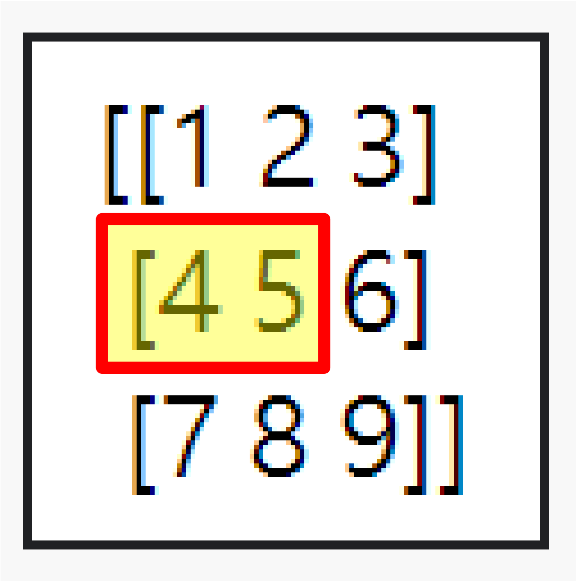


```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

每1列的第0个元素跟第1个元素

reshapingArray[: , :2]

NumPy array 九宫格取值练习



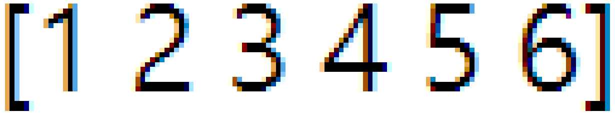
```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

第1列的第0个元素跟第1个元素

```
reshapingArray[1, :2]
```

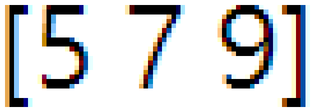
NumPy array: joining 组合

```
import numpy as np
array_one = np.array([1,2,3])
array_two = np.array([4,5,6])
array_combine = np.concatenate([array_one, array_two])
arrry_add = array_one + array_two
array_2d = np.array([array_one,array_two])
```



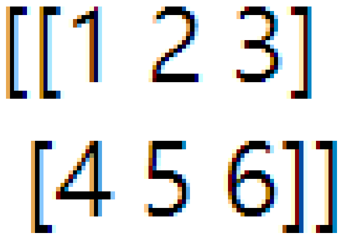
```
[1 2 3 4 5 6]
```

array_combine



```
[5 7 9]
```

array_add



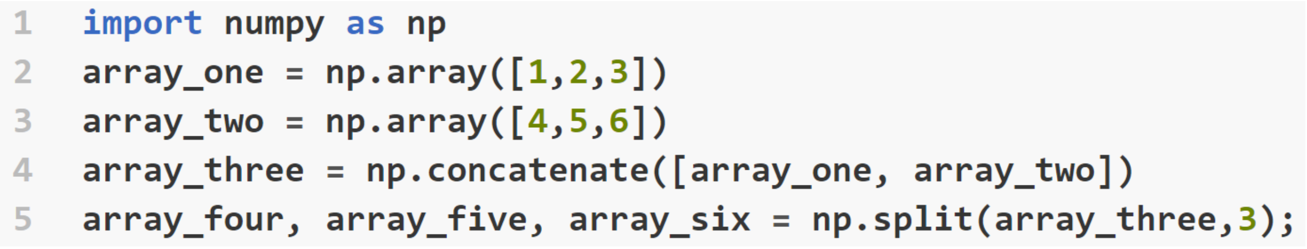
```
[[1 2 3]
 [4 5 6]]
```

array_2d

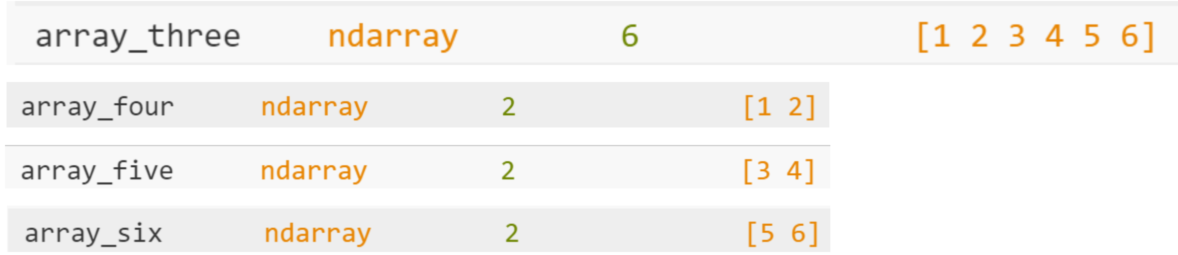
join-numpy.py

NumPy array: splitting 切割

均分成3份

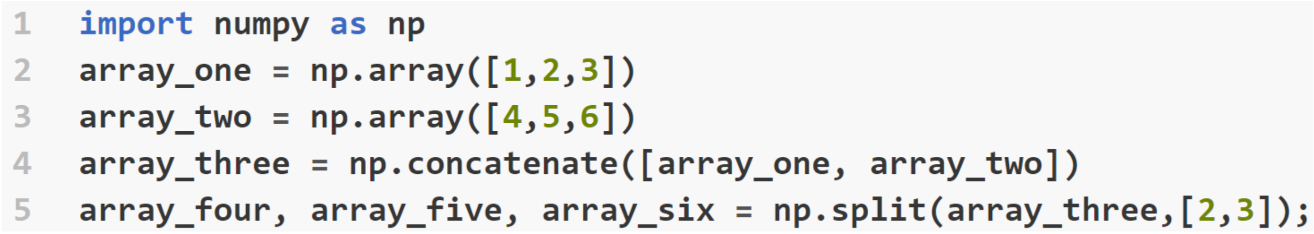


```python
import numpy as np
array_one = np.array([1,2,3])
array_two = np.array([4,5,6])
array_three = np.concatenate([array_one, array_two])
array_four, array_five, array_six = np.split(array_three,3);
```

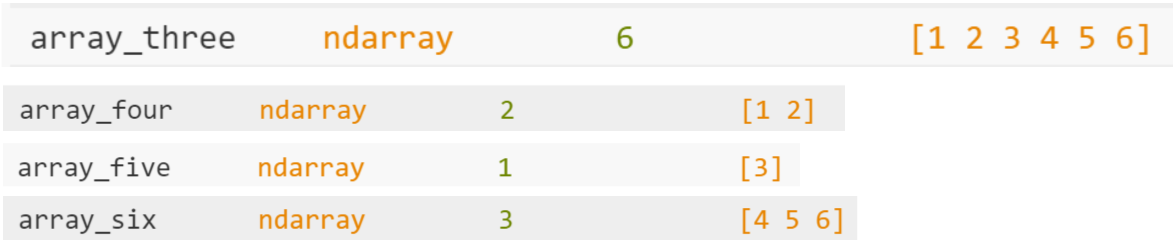


array_three	ndarray	6	[1 2 3 4 5 6]
array_four	ndarray	2	[1 2]
array_five	ndarray	2	[3 4]
array_six	ndarray	2	[5 6]

[x,y] -> [:2] ,[2,3], [3:]



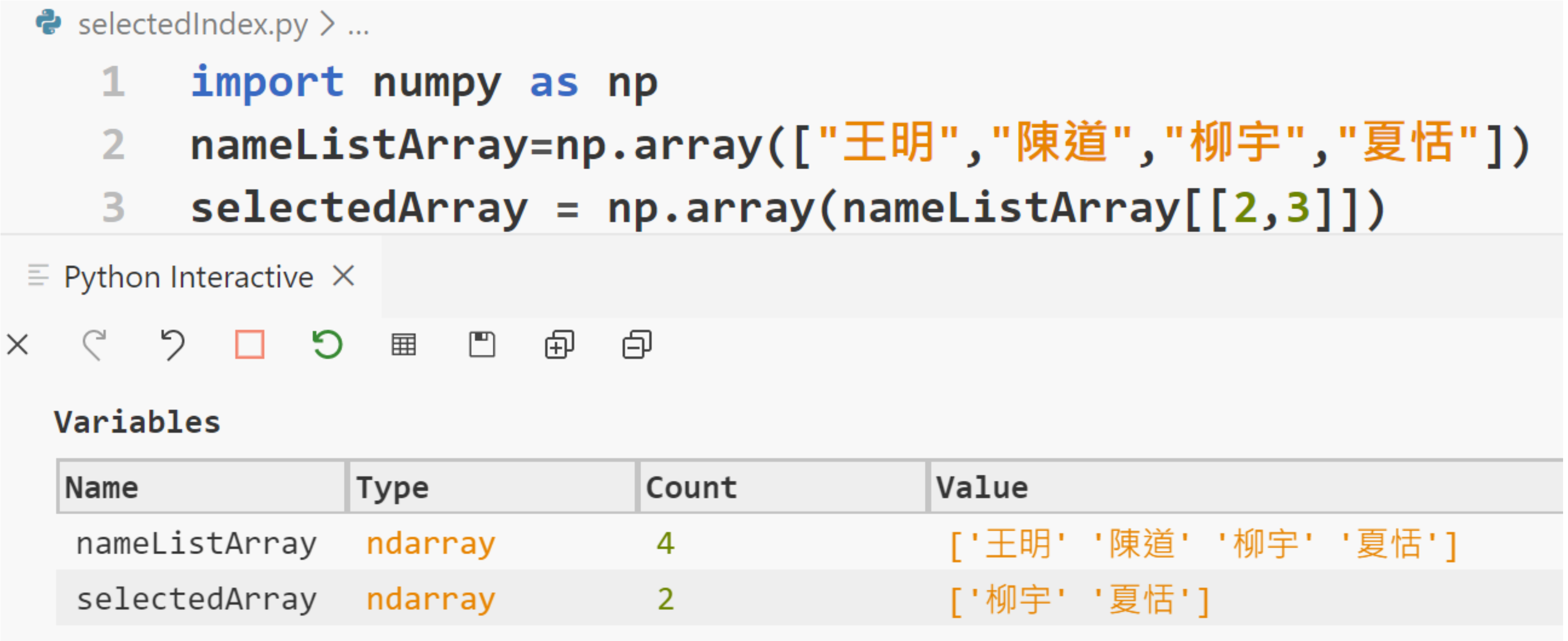
```python
import numpy as np
array_one = np.array([1,2,3])
array_two = np.array([4,5,6])
array_three = np.concatenate([array_one, array_two])
array_four, array_five, array_six = np.split(array_three,[2,3]);
```



array_three	ndarray	6	[1 2 3 4 5 6]
array_four	ndarray	2	[1 2]
array_five	ndarray	1	[3]
array_six	ndarray	3	[4 5 6]

14.join_and_split.py

NumPy array: fancy indexing 选择性取出



selectedIndex.py > ...

```python
import numpy as np
nameListArray=np.array(["王明","陳道","柳宇","夏恬"])
selectedArray = np.array(nameListArray[[2,3]])
```

Python Interactive

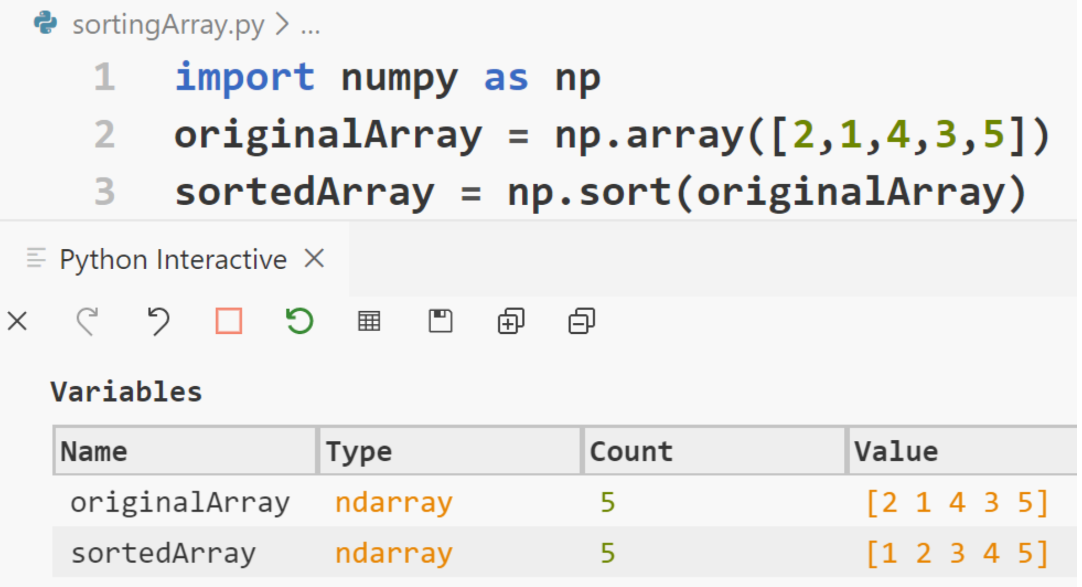
Variables

Name	Type	Count	Value
nameListArray	ndarray	4	['王明' '陳道' '柳宇' '夏恬']
selectedArray	ndarray	2	['柳宇' '夏恬']

也可以写成
np.array(nameListArray[2:])

NumPy array: sorting 排序

由小到大



sortingArray.py > ...

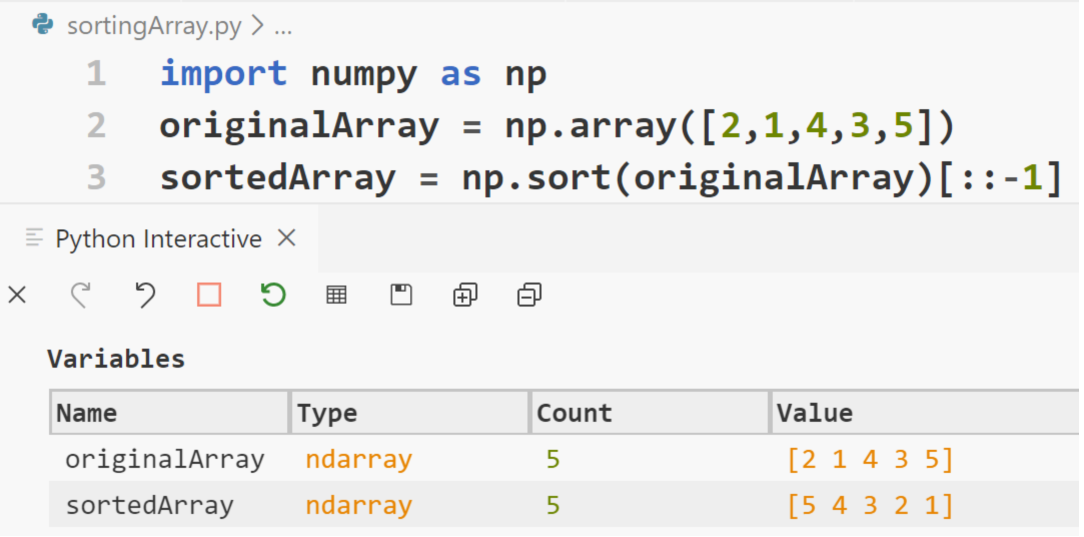
```python
import numpy as np
originalArray = np.array([2,1,4,3,5])
sortedArray = np.sort(originalArray)
```

Python Interactive

Variables

Name	Type	Count	Value
originalArray	ndarray	5	[2 1 4 3 5]
sortedArray	ndarray	5	[1 2 3 4 5]

由大到小



sortingArray.py > ...

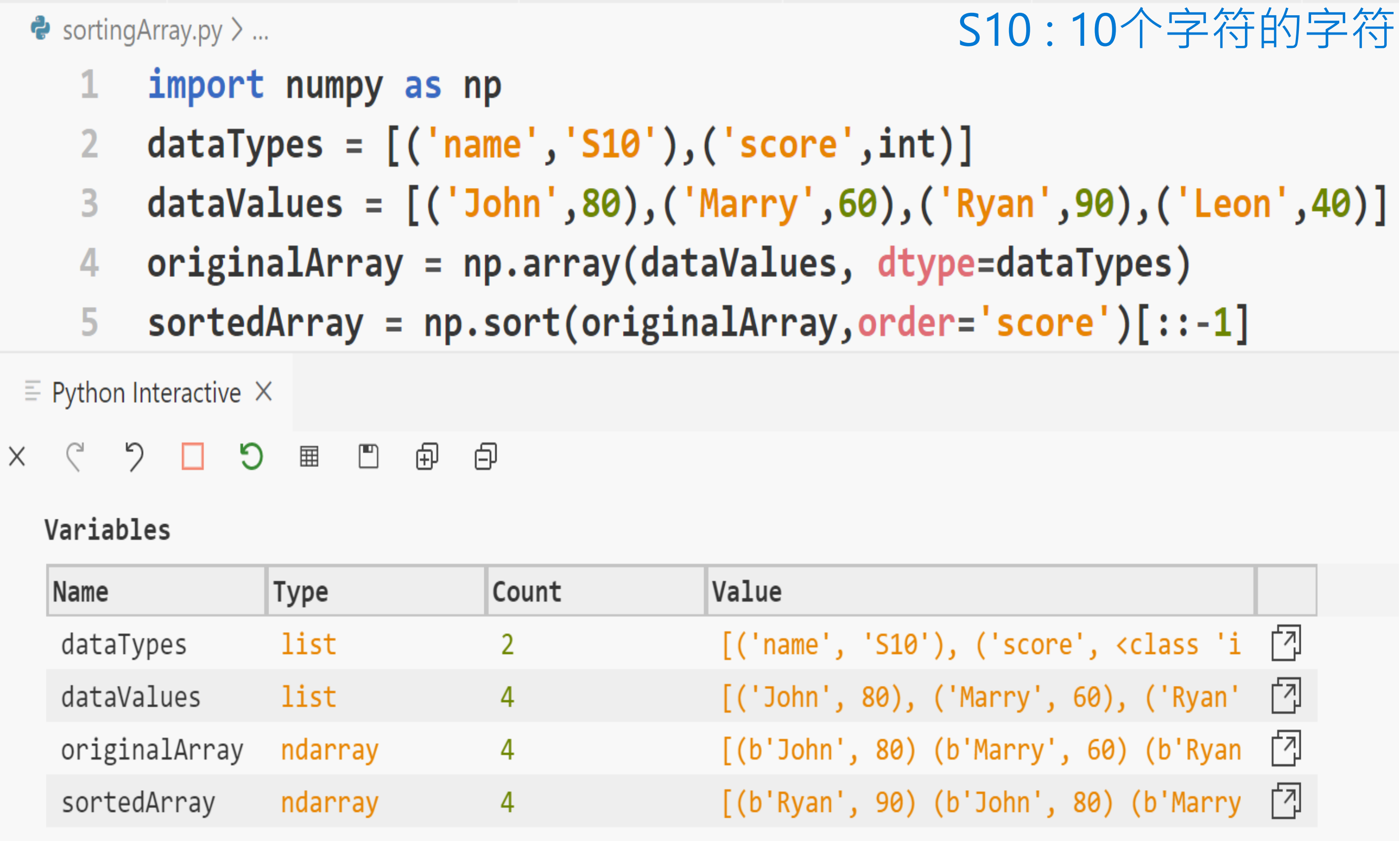
```python
import numpy as np
originalArray = np.array([2,1,4,3,5])
sortedArray = np.sort(originalArray)[::-1]
```

Python Interactive

Variables

Name	Type	Count	Value
originalArray	ndarray	5	[2 1 4 3 5]
sortedArray	ndarray	5	[5 4 3 2 1]

依特定项目进行排序



S10：10个字符的字符串

sortingArray.py > ...

```python
import numpy as np
dataTypes = [('name','S10'),('score',int)]
dataValues = [('John',80),('Marry',60),('Ryan',90),('Leon',40)]
originalArray = np.array(dataValues, dtype=dataTypes)
sortedArray = np.sort(originalArray,order='score')[::-1]
```

Python Interactive

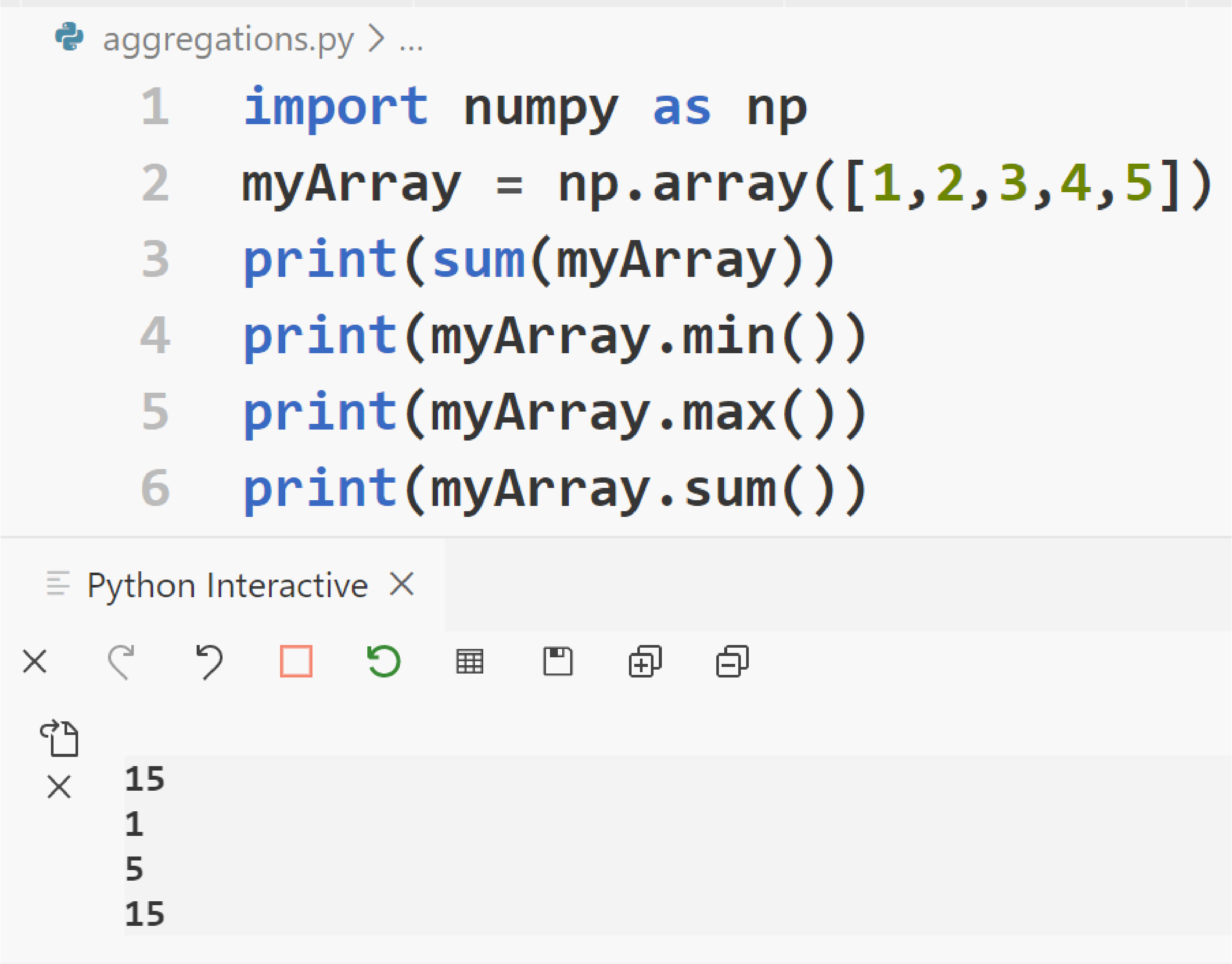
Variables

Name	Type	Count	Value
dataTypes	list	2	[('name', 'S10'), ('score', <class 'i
dataValues	list	4	[('John', 80), ('Marry', 60), ('Ryan'
originalArray	ndarray	4	[(b'John', 80) (b'Marry', 60) (b'Ryan
sortedArray	ndarray	4	[(b'Ryan', 90) (b'John', 80) (b'Marry

16.sorting_array.py

Microsoft Reactor

Aggregations 整体分析



```python
import numpy as np
myArray = np.array([1,2,3,4,5])
print(sum(myArray))
print(myArray.min())
print(myArray.max())
print(myArray.sum())
```

```
15
1
5
15
```

加总

最小值

最大值

加总

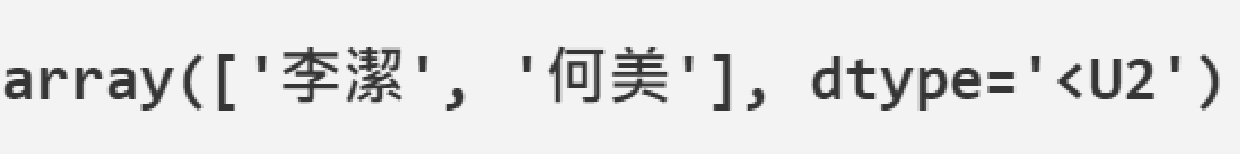
17.aggregations.py

NumPy 综合取值练习

```
import numpy as np
name = np.array(["张山","李洁","王武","陈至","何美"])
gender = np.array(["male","female","male","male","female"])
height = np.array([172,155,183,153,168,160])
weight = np.array([75.2,45.6,84.3,72.1,51.0])
score_math = np.array([45,74,62,89,32,55])
score_english = np.array([92,85,28,61,78])
score_chinese = np.array([88,55,70,61,98])
```

班上的女孩儿们的名子是?

```
name[gender=="female"]
```



```
array(['李潔', '何美'], dtype='<U2')
```

multiArray.py

NumPy 综合取值练习

```python
import numpy as np
name = np.array(["张山","李洁","王武","陈至","何美"])
gender = np.array(["male","female","male","male","female"])
height = np.array([172,155,183,153,168,160])
weight = np.array([75.2,45.6,84.3,72.1,51.0])
score_math = np.array([45,74,62,89,32,55])
score_english = np.array([92,85,28,61,78])
score_chinese = np.array([88,55,70,61,98])
```

班上男生的国文平均?

(88 + 70 + 61) / 3 = 73

```python
score_chinese[gender!="female"].mean()
```

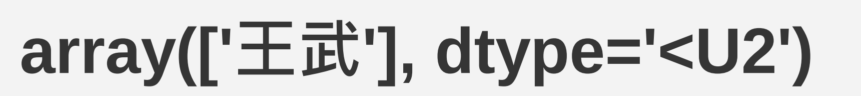
```
73.0
```

NumPy 综合取值练习

```
import numpy as np
name = np.array(["张山","李洁","王武","陈至","何美"])
gender = np.array(["male","female","male","male","female"])
height = np.array([172,155,183,153,168,160])
weight = np.array([75.2,45.6,84.3,72.1,51.0])
score_math = np.array([45,74,62,89,32,55])
score_english = np.array([92,85,28,61,78])
score_chinese = np.array([88,55,70,61,98])
```

班上谁英文不及格?

```
name[score_english<60]
```



```
array(['王武'], dtype='<U2')
```

Microsoft

pandas 简介

Section 3 第三节

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

Section 3 overview 第三节综览

- Pandas 的资料结构
  - Series 序列
  - DataFrame
- 资料操作
  - 串行、序列转换
  - 取出部分
  - 序列合成
  - 缺失值填补

Section 3 overview 第三节综览

为什么要使用pandas?

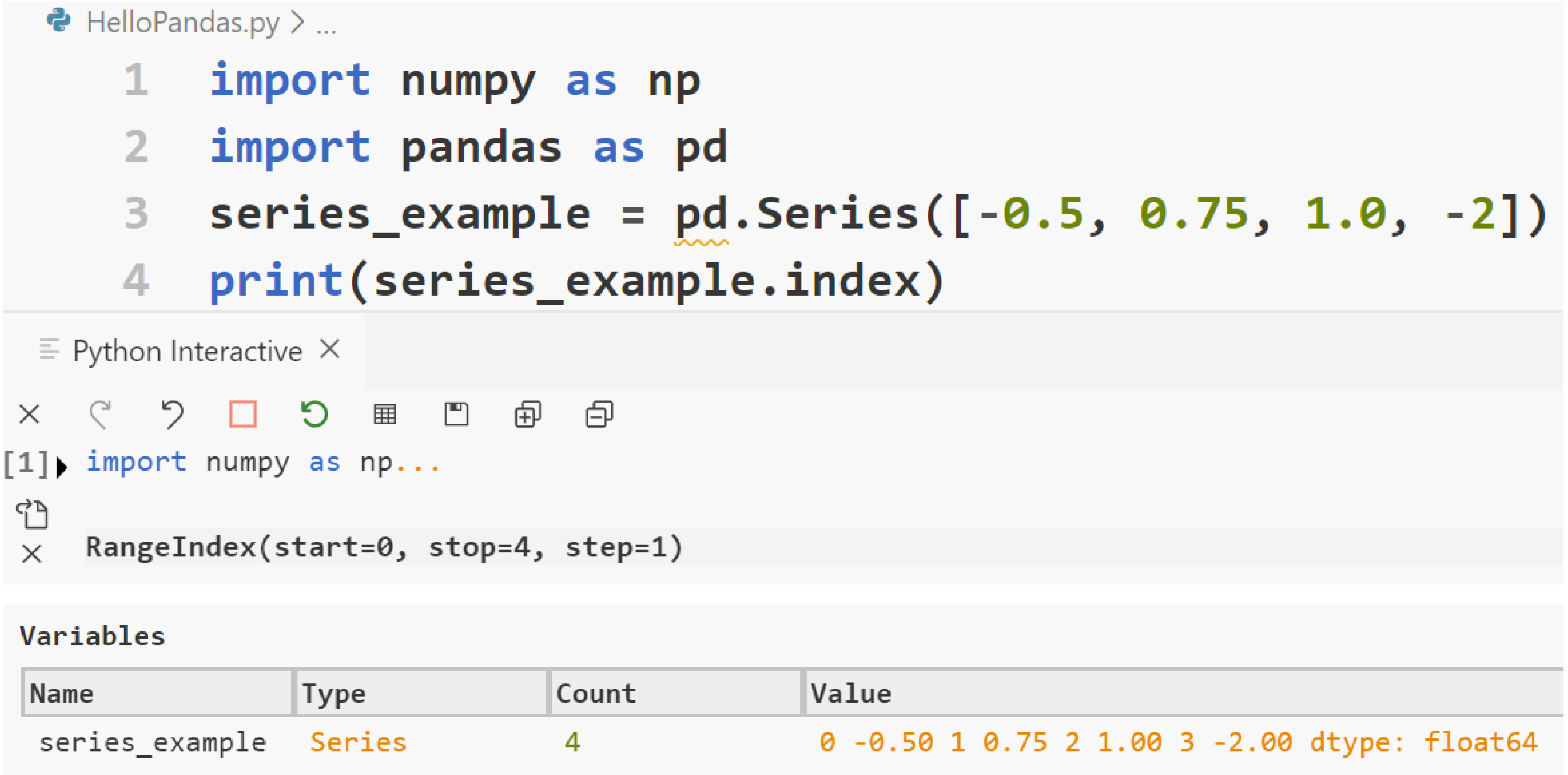
- 是Python的标准函式库，专门用来处理与操作资料
- 有直觉的、表格型的资料结构(DataFrame)
- 有可以快速进行资料计算与转型的方式(NumPy)
- 资料探索便利，与许多视觉化函式库紧密结合(matplotlib, seaborn)

	NumPy	Pandas
数值处理	相同数据型态	不同数据型态
常见数据形式	数组	表格式

知名线上串流服务商使用pandas与Jupyter notebook来做资料探索、准备与验证

Pandas 资料结构: Series 序列

预设index值：
0 ~ N-1 (N为资料笔数)



HelloPandas.py > ...

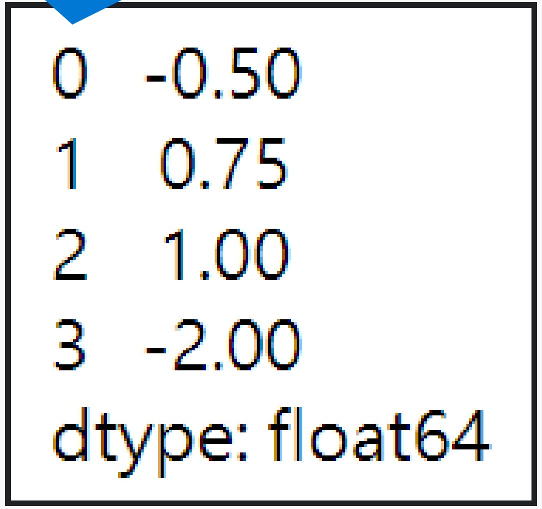
```python
import numpy as np
import pandas as pd
series_example = pd.Series([-0.5, 0.75, 1.0, -2])
print(series_example.index)
```

Python Interactive

```
[1] import numpy as np...

RangeIndex(start=0, stop=4, step=1)
```

Variables

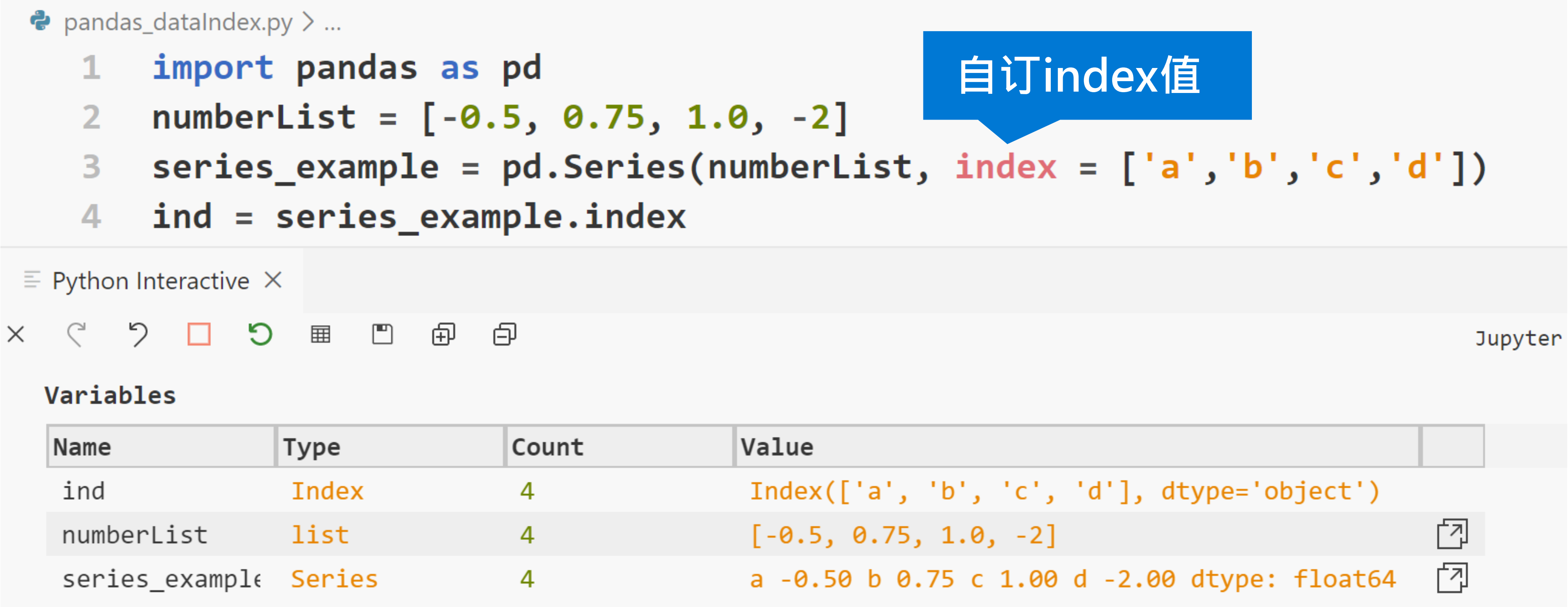
Name	Type	Count	Value
series_example	Series	4	0 -0.50 1 0.75 2 1.00 3 -2.00 dtype: float64



```
0   -0.50
1    0.75
2    1.00
3   -2.00
dtype: float64
```

18.hello_pandas.py

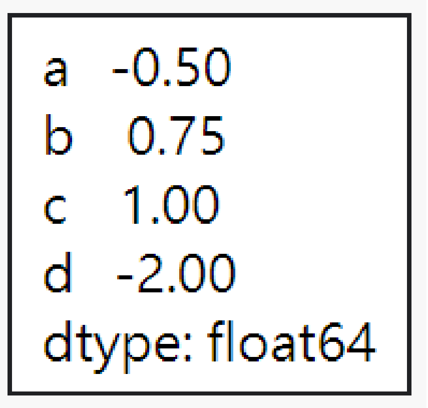
Pandas 资料操作: list 串行 + Index索引 -> Series 序列



```python
import pandas as pd
numberList = [-0.5, 0.75, 1.0, -2]
series_example = pd.Series(numberList, index = ['a','b','c','d'])
ind = series_example.index
```

Variables

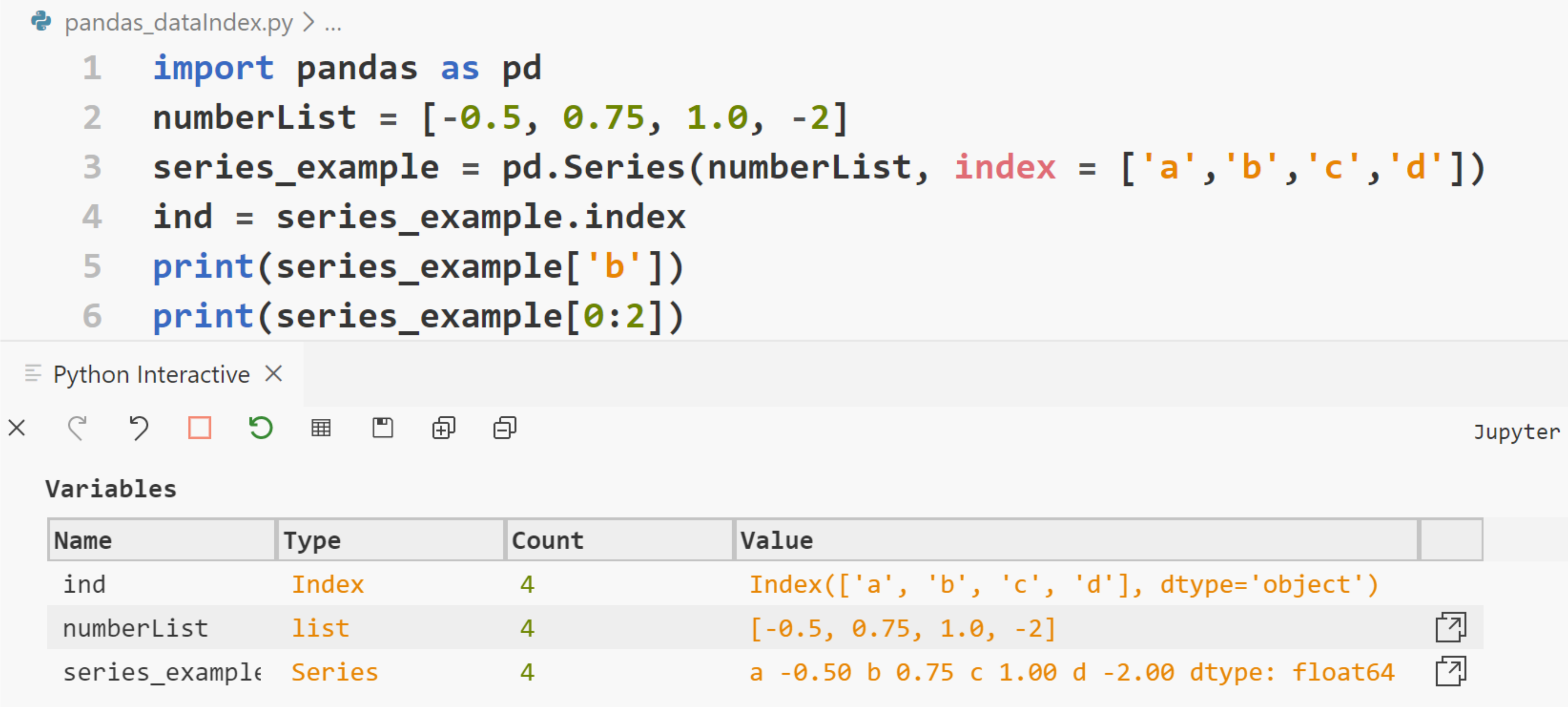
Name	Type	Count	Value
ind	Index	4	Index(['a', 'b', 'c', 'd'], dtype='object')
numberList	list	4	[-0.5, 0.75, 1.0, -2]
series_example	Series	4	a -0.50 b 0.75 c 1.00 d -2.00 dtype: float64



```
a   -0.50
b    0.75
c    1.00
d   -2.00
dtype: float64
```

20.pandas_dataIndex.py

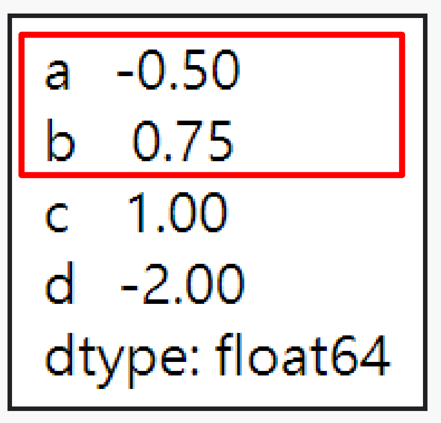
Pandas 资料操作: 取出部分序列



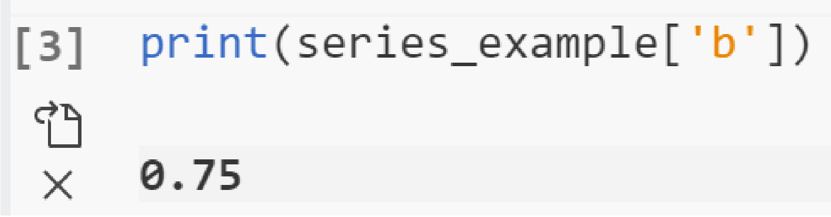
```python
import pandas as pd
numberList = [-0.5, 0.75, 1.0, -2]
series_example = pd.Series(numberList, index = ['a','b','c','d'])
ind = series_example.index
print(series_example['b'])
print(series_example[0:2])
```

Variables

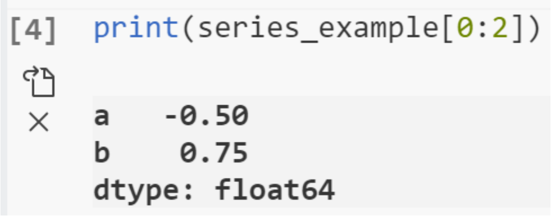
Name	Type	Count	Value
ind	Index	4	Index(['a', 'b', 'c', 'd'], dtype='object')
numberList	list	4	[-0.5, 0.75, 1.0, -2]
series_example	Series	4	a -0.50 b 0.75 c 1.00 d -2.00 dtype: float64



```
a   -0.50
b    0.75
c    1.00
d   -2.00
dtype: float64
```



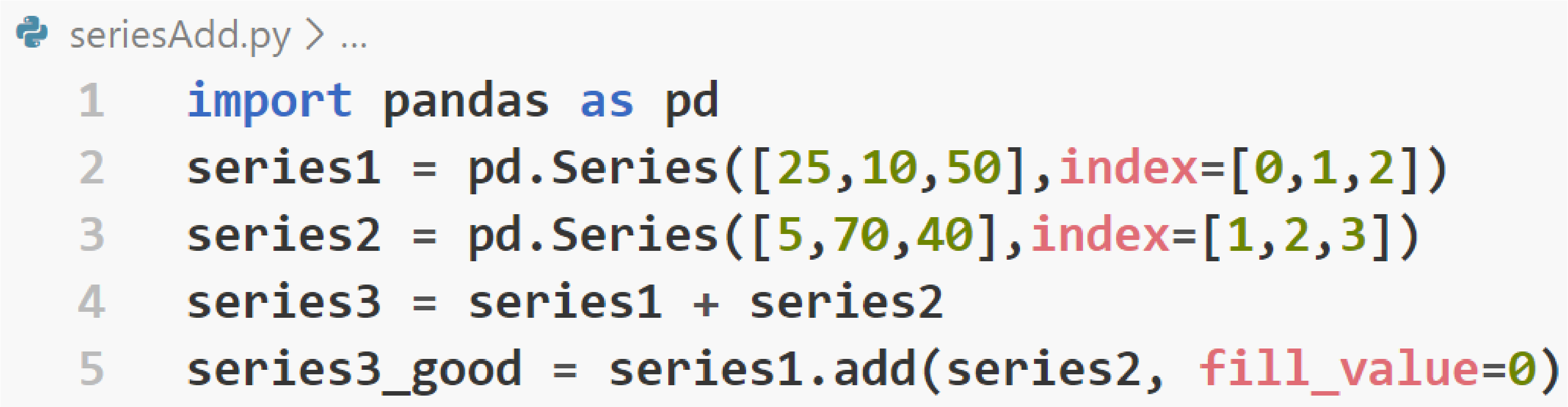
```
[3]  print(series_example['b'])

0.75
```



```
[4]  print(series_example[0:2])

a   -0.50
b    0.75
dtype: float64
```

20.pandas_dataIndex.py

Pandas 资料操作: 序列合成、缺失值填补



seriesAdd.py > ...

```python
import pandas as pd
series1 = pd.Series([25,10,50],index=[0,1,2])
series2 = pd.Series([5,70,40],index=[1,2,3])
series3 = series1 + series2
series3_good = series1.add(series2, fill_value=0)
```

两个序列不对等，形成缺失值

预设将缺失值设定为0

```
0   25
1   10
2   50
dtype: int64
```

Series1

```
1    5
2   70
3   40
dtype: int64
```

Series2

```
0     NaN
1    15.0
2   120.0
3     NaN
dtype: float64
```

Series3

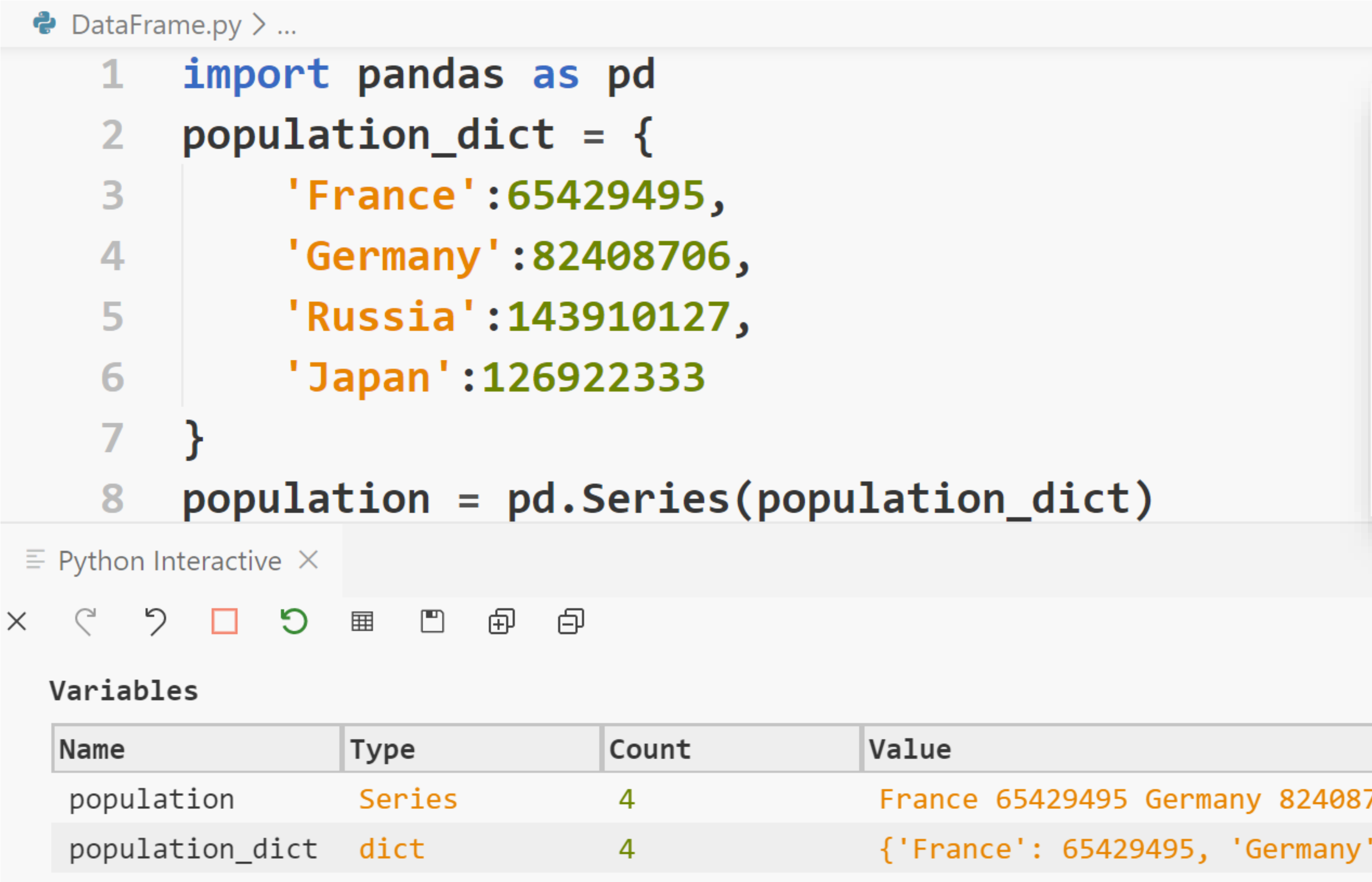
```
0    25.0
1    15.0
2   120.0
3    40.0
dtype: float64
```

Series3_good

series_add.py

Microsoft Reactor

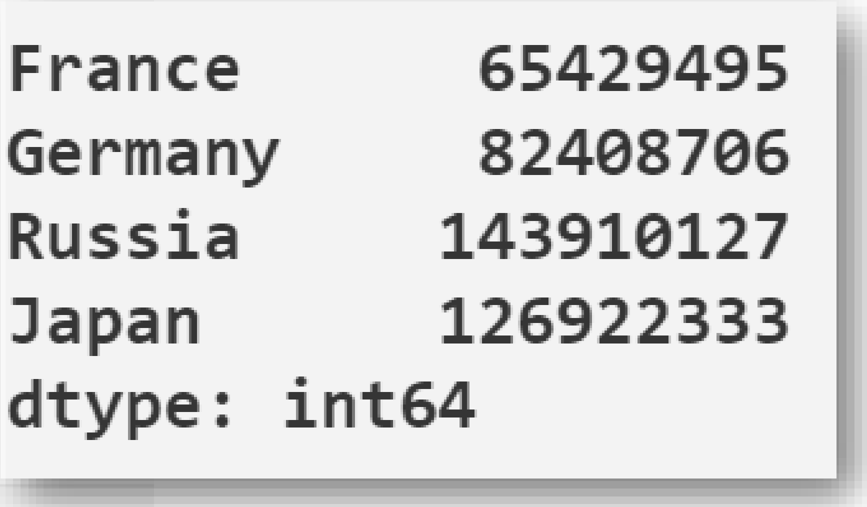
Pandas 资料操作: 用Dictionary字典来建立序列



```python
import pandas as pd
population_dict = {
        'France':65429495,
        'Germany':82408706,
        'Russia':143910127,
        'Japan':126922333
}
population = pd.Series(population_dict)
```

Variables

Name	Type	Count	Value
population	Series	4	France 65429495 Germany 8240870(
population_dict	dict	4	{'France': 65429495, 'Germany':



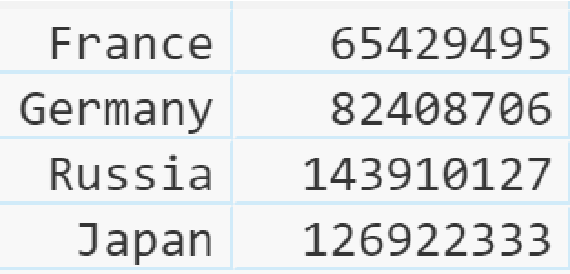
```
France        65429495
Germany       82408706
Russia       143910127
Japan        126922333
dtype: int64
```

population.py

Pandas 资料操作: 用Dictionary字典来建立序列

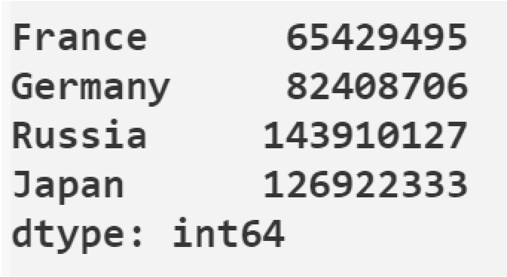
- 依国家名称首字字母进行排序

```
import numpy as np
import pandas as pd
population_dict = {
    'France':65429495,
    'Germany':82408706,
    'Russia':143910127,
    'Japan':126922333
}

population = pd.Series(population_dict)
population

#取出国家名称index进行排序
country_name_sorted = np.sort(population.index)
#搭配已经排序好的国家名称来建立Series
population_sorted = pd.Series(population_dict,country_name_sorted)
```



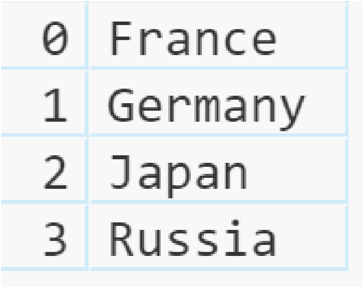
France	65429495
Germany	82408706
Russia	143910127
Japan	126922333

population_dict



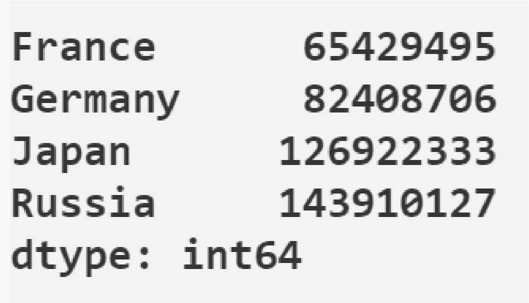
```
France       65429495
Germany      82408706
Russia      143910127
Japan       126922333
dtype: int64
```

population



0	France
1	Germany
2	Japan
3	Russia

country_name_sorted



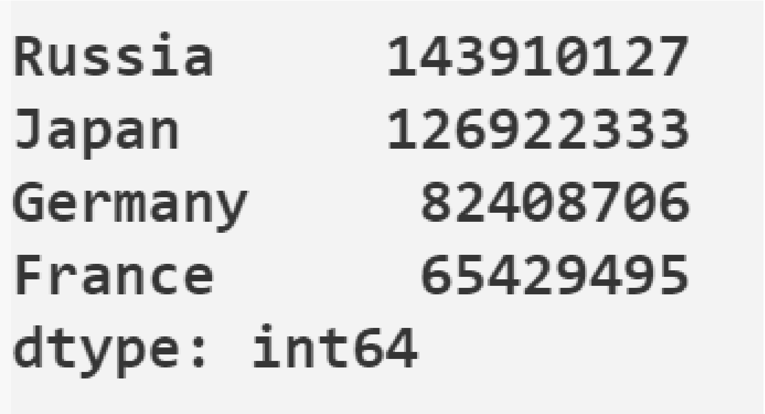
```
France       65429495
Germany      82408706
Japan       126922333
Russia      143910127
dtype: int64
```

population_sorted

Pandas 资料操作: 用Dictionary字典来建立序列

- 依人口数进行排序

```
import numpy as np
import pandas as pd
population_dict = {
    'France':65429495,
    'Germany':82408706,
    'Russia':143910127,
    'Japan':126922333
}

population = pd.Series(population_dict)

#取出国家名称index进行排序
country_name_sorted = np.sort(population.index)

#搭配已经排序好的国家名称来建立Series
population_sorted = pd.Series(population_dict,country_name_sorted)

#照人口数来进行排序
population_sorted.sort_values(ascending=False)
```

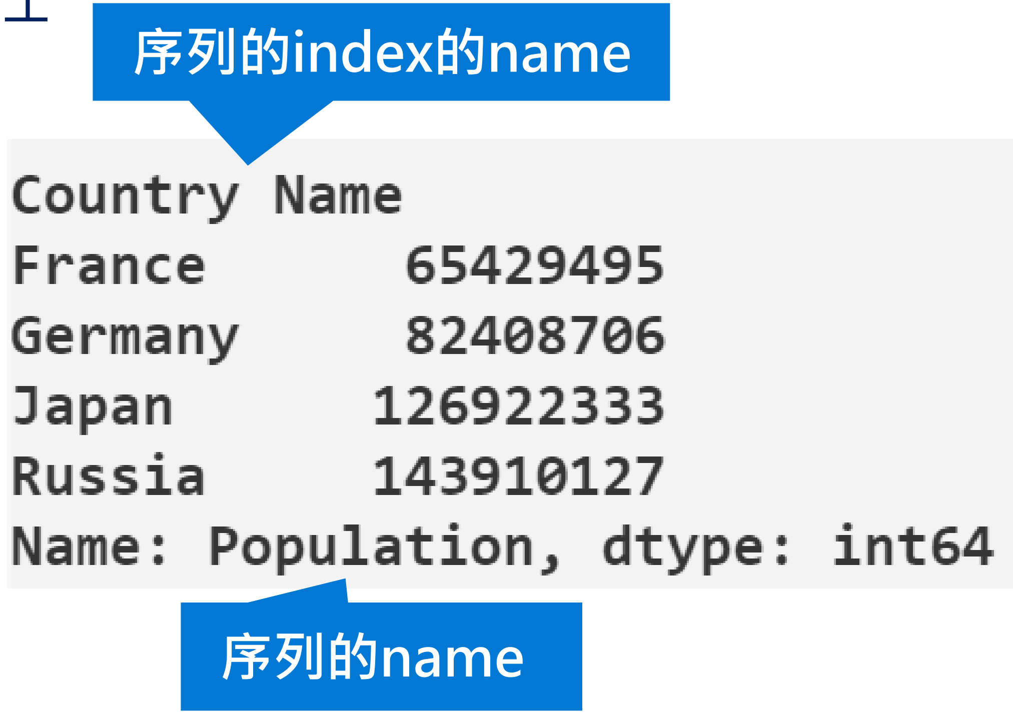


```
Russia     143910127
Japan      126922333
Germany     82408706
France      65429495
dtype: int64
```

Pandas 资料操作: 用Dictionary字典来建立序列

- 序列的name属性以及index的name属性

```python
import numpy as np
import pandas as pd
population_dict = {
    'France':65429495,
    'Germany':82408706,
    'Russia':143910127,
    'Japan':126922333
}
population = pd.Series(population_dict)
#取出国家名称index进行排序
country_name_sorted = np.sort(population.index)
#搭配已经排序好的国家名称来建立Series
population_sorted = pd.Series(population_dict,country_name_sorted)
#照人口数来进行排序
population_sorted.sort_values(ascending=False)
#帮Series加上name属性
population_sorted.name = 'Population'
#帮Series的index加上name属性
population_sorted.index.name = 'Country Name'
```



序列的index的name

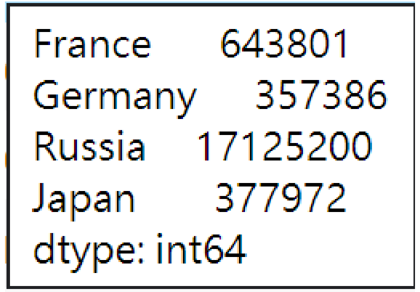
```
Country Name
France        65429495
Germany       82408706
Japan        126922333
Russia       143910127
Name: Population, dtype: int64
```

序列的name

Pandas 资料结构: DataFrame

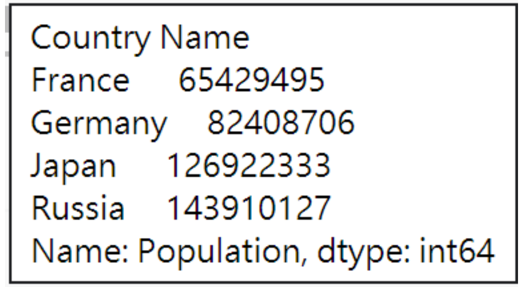
加上其他序列，组合成DataFrame

```
area_dict = {
    'France':643801,
    'Germany':357386,
    'Russia':17125200,
    'Japan':377972
}
area = pd.Series(area_dict)

countries = pd.DataFrame({'Area':area,'Population':population_sorted})
countries
```



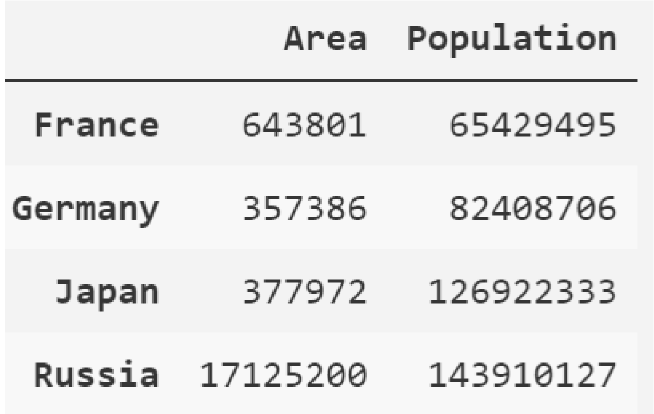
```
France      643801
Germany     357386
Russia    17125200
Japan       377972
dtype: int64
```

area



```
Country Name
France      65429495
Germany     82408706
Japan      126922333
Russia     143910127
Name: Population, dtype: int64
```

population_sorted



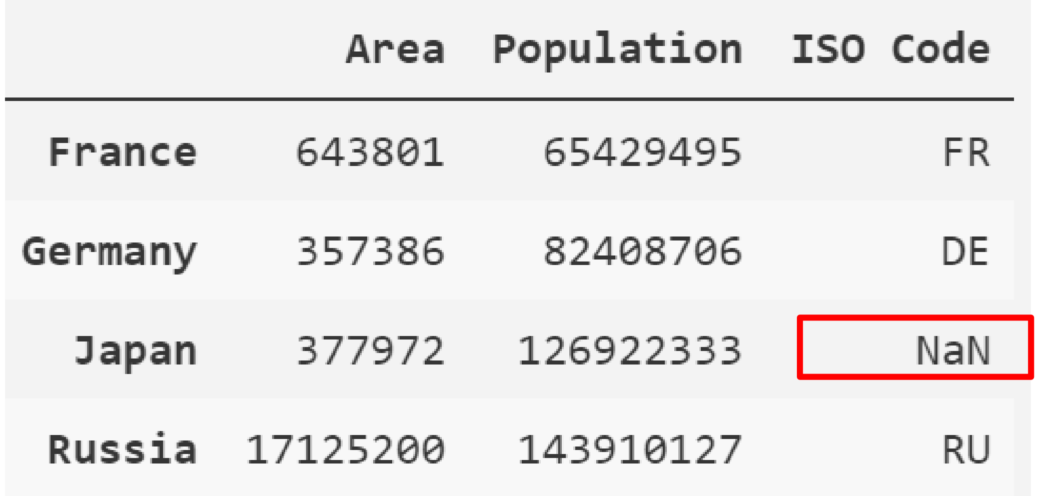
	Area	Population
France	643801	65429495
Germany	357386	82408706
Japan	377972	126922333
Russia	17125200	143910127

countries

Pandas 资料结构: DataFrame

若序列不对等，会形成缺失值

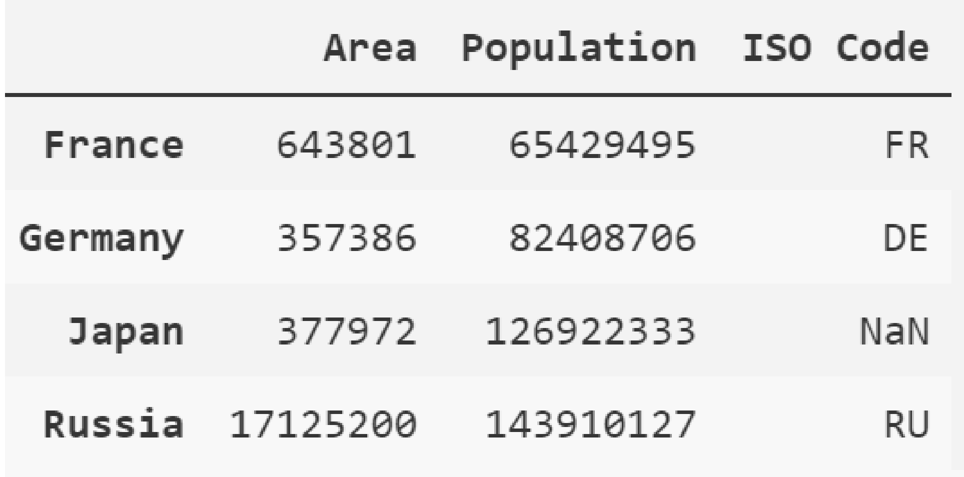
```
iso_code_dict = {
    'France':'FR',
    'Germany':'DE',
    'Russia':'RU',
}
iso_code = pd.Series(iso_code_dict)

countries['ISO Code']=iso_code
countries
```



	Area	Population	ISO Code
France	643801	65429495	FR
Germany	357386	82408706	DE
Japan	377972	126922333	NaN
Russia	17125200	143910127	RU

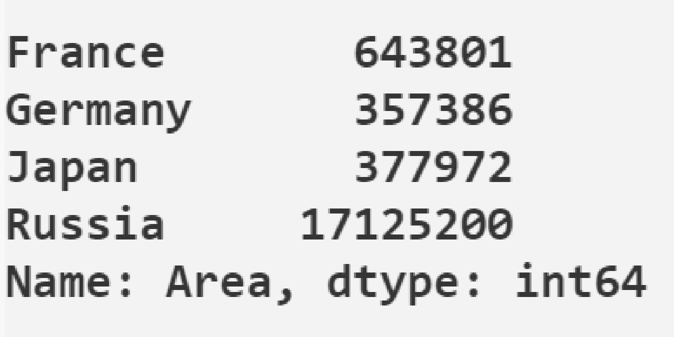
Pandas 资料操作: 取出部分DataFrame

```
countries['Area']

countries.iloc[:3,:2]

countries.loc[:'Russia',:'Population']
```



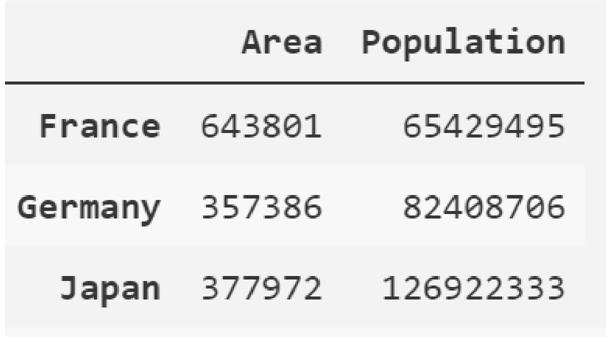
	Area	Population	ISO Code
France	643801	65429495	FR
Germany	357386	82408706	DE
Japan	377972	126922333	NaN
Russia	17125200	143910127	RU

countries



```
France          643801
Germany         357386
Japan           377972
Russia        17125200
Name: Area, dtype: int64
```

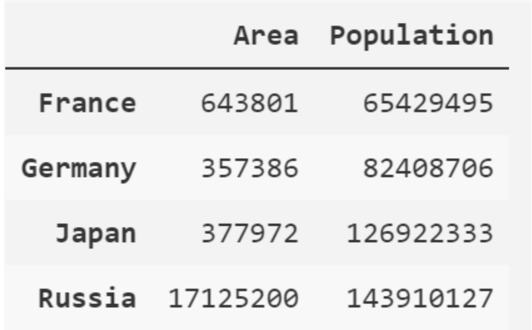
countries['Area']



	Area	Population
France	643801	65429495
Germany	357386	82408706
Japan	377972	126922333

countries.iloc[:3,:2]

第0,1,2列、第0,1栏



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

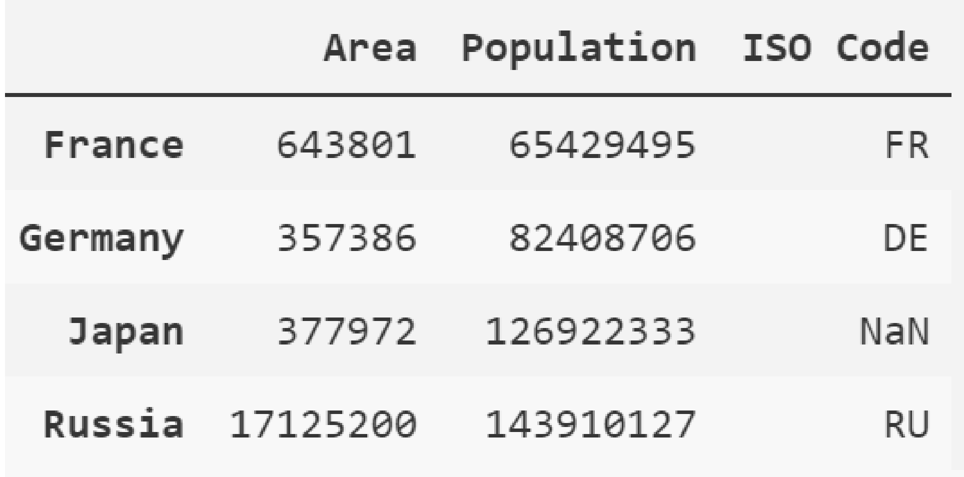
countries.loc[:'Russia',:'Population']

注意：有包含到结束条件

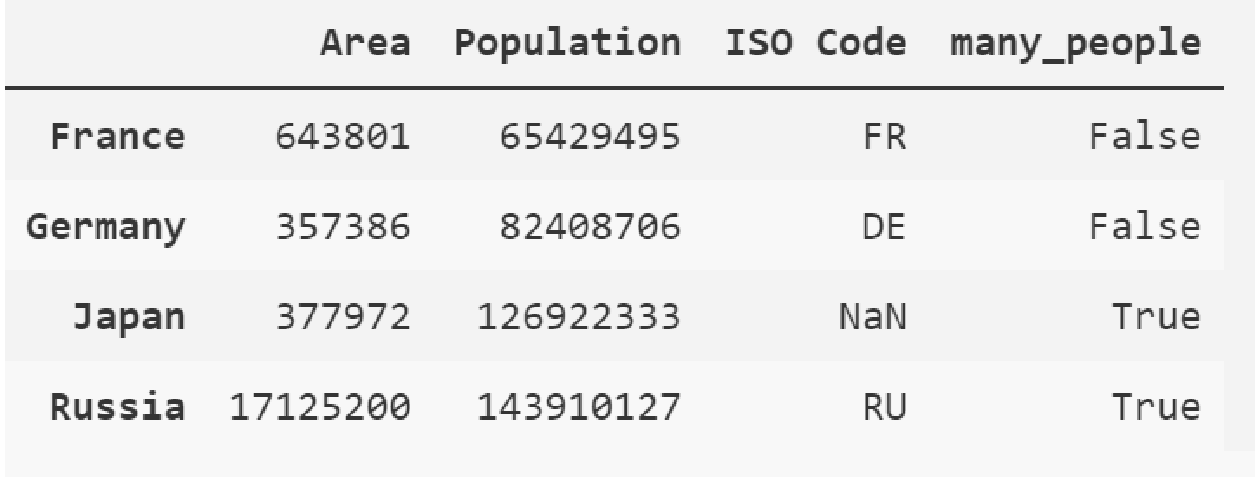
Pandas 资料操作: 依据条件新增字段值

- 如果人口数大于1亿人则为真(True)

```
countries['many_people'] = countries.Population >= 100000000
```



	Area	Population	ISO Code
France	643801	65429495	FR
Germany	357386	82408706	DE
Japan	377972	126922333	NaN
Russia	17125200	143910127	RU



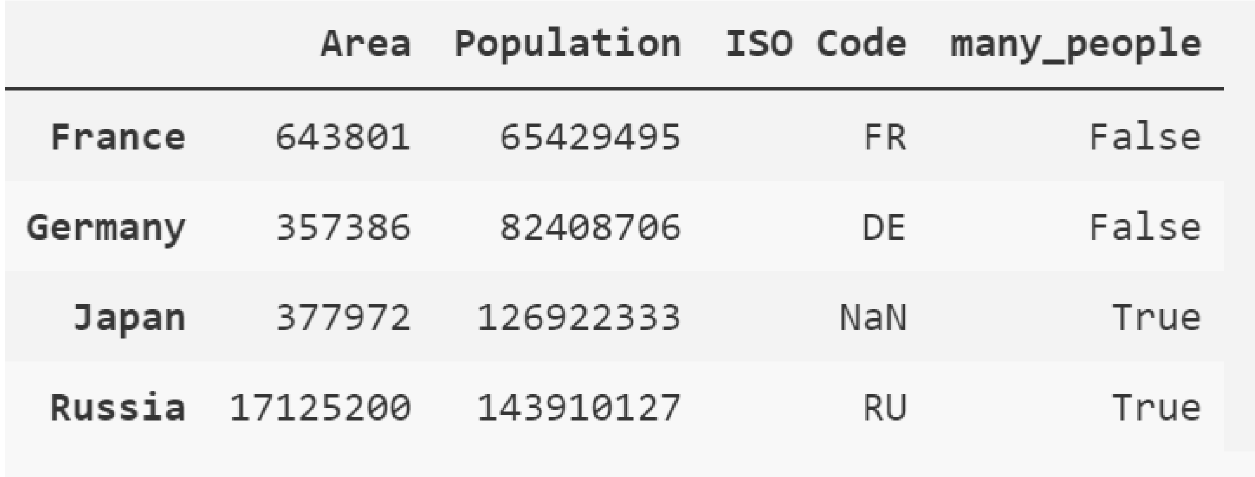
	Area	Population	ISO Code	many_people
France	643801	65429495	FR	False
Germany	357386	82408706	DE	False
Japan	377972	126922333	NaN	True
Russia	17125200	143910127	RU	True

countries

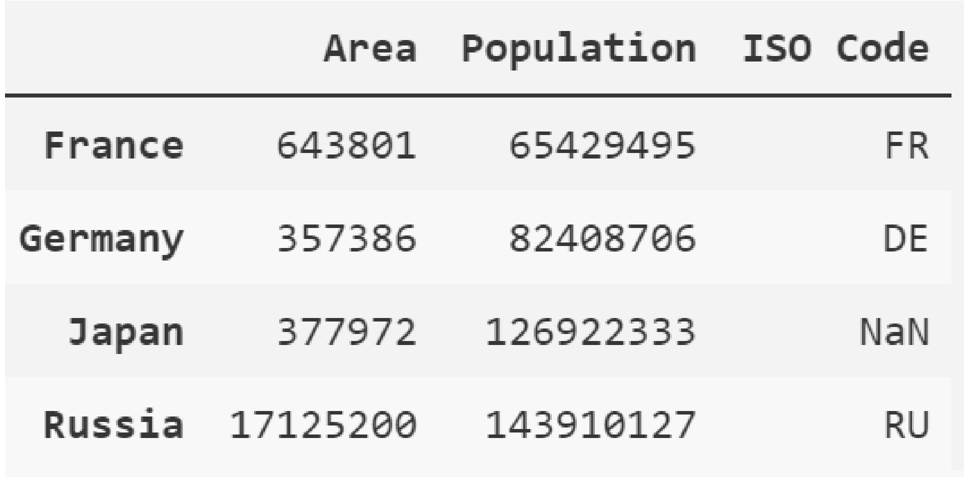
Pandas 资料操作: 删除整栏

- 把整个字段删除

```
del countries['many_people']
```



	Area	Population	ISO Code	many_people
France	643801	65429495	FR	False
Germany	357386	82408706	DE	False
Japan	377972	126922333	NaN	True
Russia	17125200	143910127	RU	True

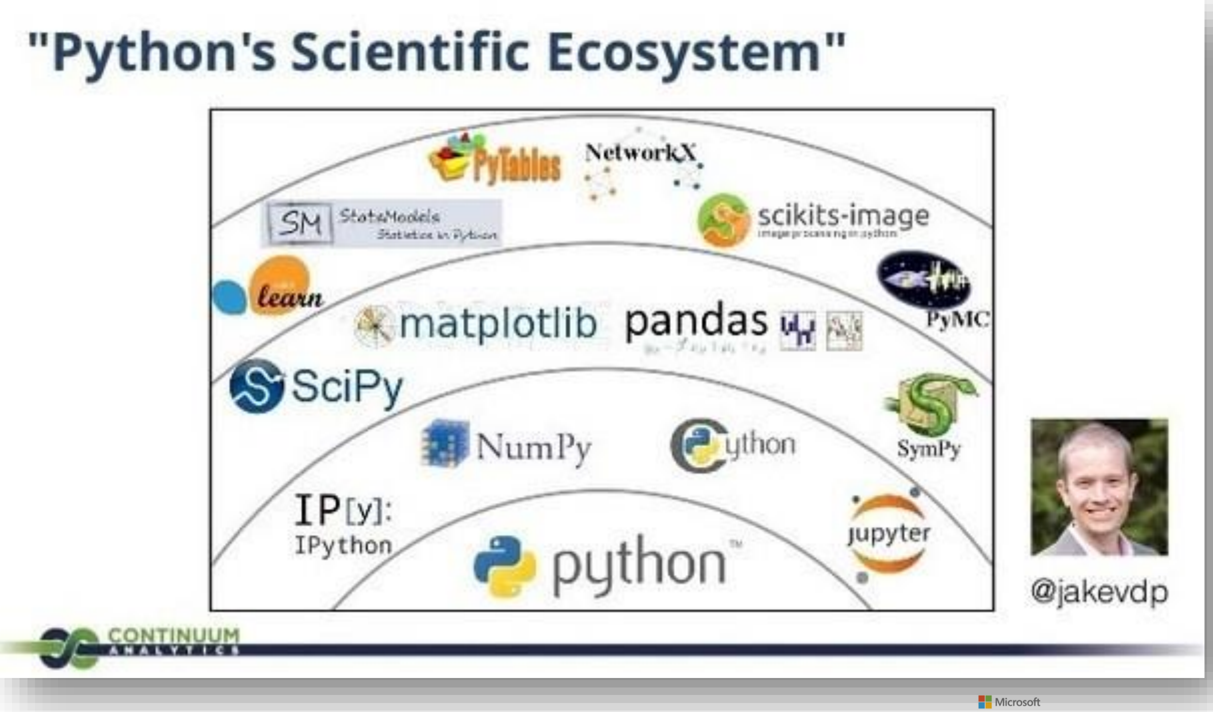


	Area	Population	ISO Code
France	643801	65429495	FR
Germany	357386	82408706	DE
Japan	377972	126922333	NaN
Russia	17125200	143910127	RU

countries

各种函式库综览

函式库	主要特性	用途
NumPy	数值运算	几乎是所有分析的基础
SciPy	科学计算	讯号处理、线性代数、统计
Pandas	表格式分析	资料探索、统计、视觉化
Matplotlib	绘图	资料视觉化
Scikit-learn	机器学习	机器学习算法


"Python's Scientific Ecosystem"

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

This document is provided "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it. Some examples are for illustration only and are fictitious. No real association is intended or inferred. Microsoft makes no warranties, express or implied, with respect to the information provided here.