Pandas 迭代與重複操作 (3/19更新)

重要知識點 ›
DataFrame 當中的 For Loop ›
橫向的資料迭代 ›

重要知識點

- 知道 DataFrame 中迴圈的運作規則
- 了解 DataFrame 中 Map、 Apply、Applymap 差異
- 知道不建議在 DataFrame 進行迭代操作的原因

DataFrame 當中的 For Loop

當你對一個 DataFrame 進行迴圈 (Loop) 操作，你預期結果會是什麼嗎？

```python
import pandas as pd

df = pd.DataFrame({
  'name': ['Alice', 'Bob'],
  'age': [20, 32]
})

for c in df:
  print(c)
```

橫向的資料迭代

如果我們想要對以「筆」為單位的資料迭代的話，最暴力的方法可以這樣做：

```python
import pandas as pd

df = pd.DataFrame({
  'name': ['Alice', 'Bob'],
  'age': [20, 32]
})

for i in range(len(df)):
  print(df.iloc[i])
```

iteritems()、iterrows()、itertuples()

第二種方法可以直接用 DataFrame 內建的 iterative 方法：

```python
for d in df.iteritems():
  print(d)

for d in df.iterrows():
  print(d)

for d in df.itertuples():
  print(d)
```

1.
('name', 0 Alice 1 Bob Name: name, dtype: object)
('age', 0 20 1 32 Name: age, dtype: int64)

2.
(0, name Alice age 20 Name: 0, dtype: object)
(1, name Bob age 32 Name: 1, dtype: object)

3.
Pandas(Index=0, name='Alice', age=20)
Pandas(Index=1, name='Bob', age=32)

apply

第三種方法是使用 Pandas 當中的 apply 方法，apply 是一種用於逐行或逐列的循環處理方法，常搭配 lambda 匿名函式一起使用：

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

df = pd.DataFrame({
  'score': [98, 67, 85],
  'age': [20, 32, 28]
})
```

```python
df.apply(np.max)
# score    98
# age      32
# dtype: int64

df.apply(np.min)
# score    67
# age      20
# dtype: int64
```

```python
df.apply(lambda x: x.max() - x.min())
# score    31
# age      12
# dtype: int64
```

map

另外一種跟 apply 很像的方法叫做 map：

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

df = pd.DataFrame({
  'score': [98, 67, 85],
  'age': [20, 32, 28]
})

df['age'].map(lambda x: -x)
```

0	-20
1	-32
2	-28

applymap

在 Pandas 當中，有一種同時 apply 和 map 方法稱為 applymap：

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

df = pd.DataFrame({
  'score': [98, 67, 85],
  'age': [20, 32, 28]
})

df.applymap(lambda x: -x)
```

	score	age
0	-98	-20
1	-67	-32
2	-85	-28

Map、 Apply、Applymap

接著，我們總結一下這三種類似的方法本質上有何差異：

- map：對 series 所有元素作一樣的操作
- apply：對 series 或 dataframe 逐行或逐列做一樣的操作
- applymap：對 dataframe 所有元素作一樣的操作

補充：lambda 匿名函式

Map、 Apply、Applymap 很常搭配 lambda 匿名函式 一起使用，但其實裡面也可以放函式名稱，我們來比較看看：

```python
df.apply(lambda x: x.max() - x.min())
# score    31
# age      12
# dtype: int64
```

```python
def f(x):
  return x.max() - x.min()
print(df.apply(f))
# score    31
# age      12
# dtype: int64
```

補充：lambda 匿名函式是對於 Python 入門者來說比較難的一個坎，很多人會覺得這是一種「炫技」的手法。

不建議在 DataFrame 進行迭代操作

如同我們在 NumPy 提過的，不建議在 DataFrame 進行迭代操作。原因是這樣就回到了 Python 列表的操作，無法享受矩陣特性帶來的優勢。

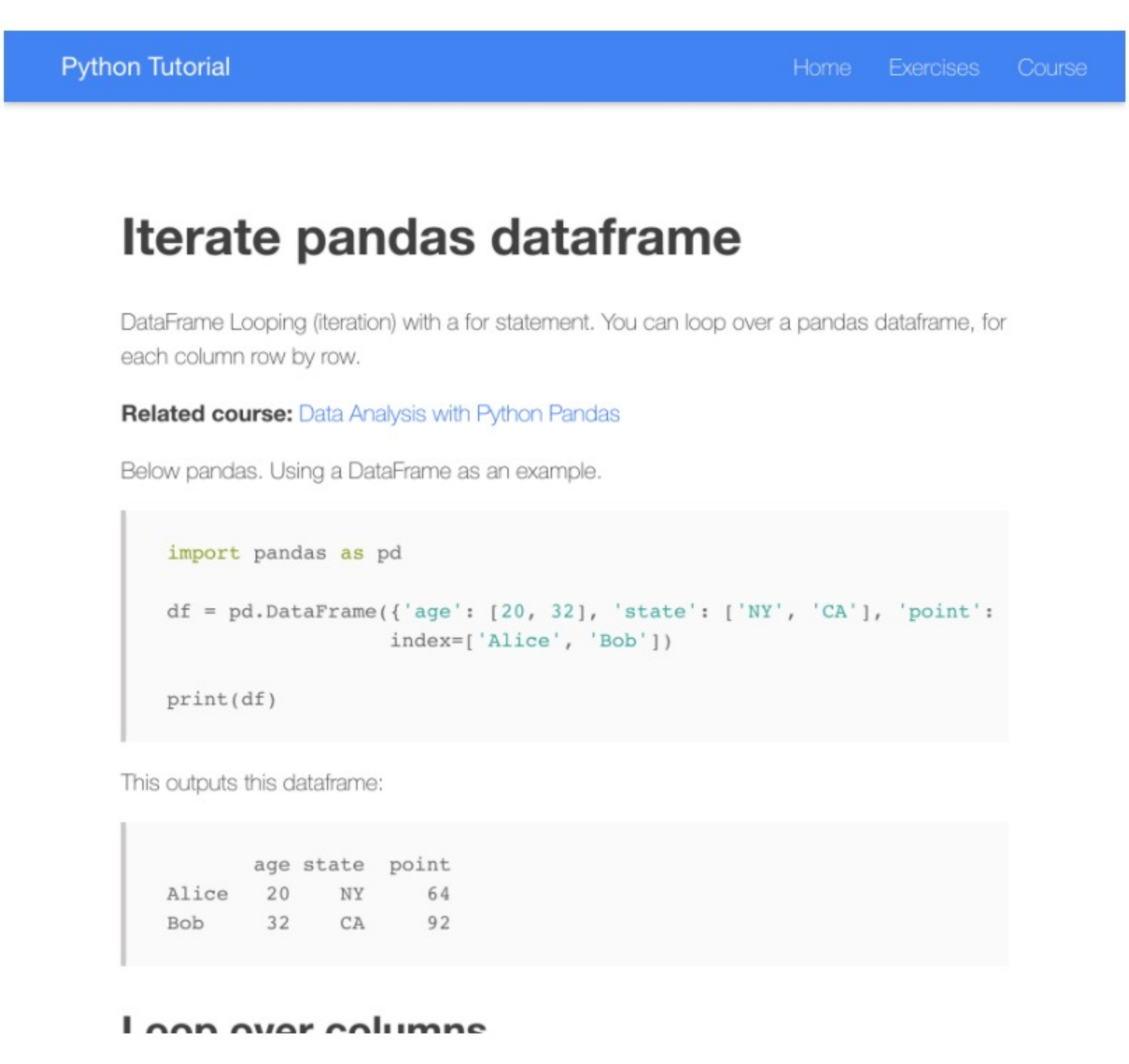
知識點回顧

- 知道 DataFrame 中迴圈的運作規則
- 了解 DataFrame 中 Map、 Apply、Applymap 差異
- 知道不建議在 DataFrame 進行迭代操作的原因

參考資料

Iterate pandas dataframe
網站：pythontutorial

本篇文章示範了基本的 dataframe 循環操作，包含迴圈與 Iterate 方法。



Iterate pandas dataframe

DataFrame Looping (iteration) with a for statement. You can loop over a pandas dataframe, for each column row by row.

Related course: Data Analysis with Python Pandas

Below pandas. Using a DataFrame as an example.

```python
import pandas as pd

df = pd.DataFrame({'age': [20, 32], 'state': ['NY', 'CA'], 'point':
                  index=['Alice', 'Bob'])

print(df)
```

This outputs this dataframe:

```
       age state  point
Alice   20    NY     64
Bob     32    CA     92
```

Loop over columns

Introduction to Pandas apply, applymap and map
網站：towardsdatascience

apply, applymap and map 是很多新手容易混亂的地方，本文做了一系列的範例操作，適合新手完整的學習。

lambda 運算式
網站：openhome

在 Python 中可以用 lambda 運算式來定義函式，執行運算時將直接採用函式作為最小的單位。

在Python中缺少其它語言中的switch陳述句，以下結合字典物件與lambda模擬switch的示範：

```python
score = int(input('請輸入分數：'))
level = score // 10
{
    10 : lambda: print('Perfect'),
    9  : lambda: print('A'),
    8  : lambda: print('B'),
    7  : lambda: print('C'),
    6  : lambda: print('D')
}.get(level, lambda: print('E'))()
```

在上例中，字典物件中的值的部份是lambda所建立的函式物件，你使用get()方法指定鍵，如果有符合的鍵，就傳回對應的函式物件並執行，否則就傳回get()第二個引數所指定的函式並執行，這模擬了switch中default的陳述。

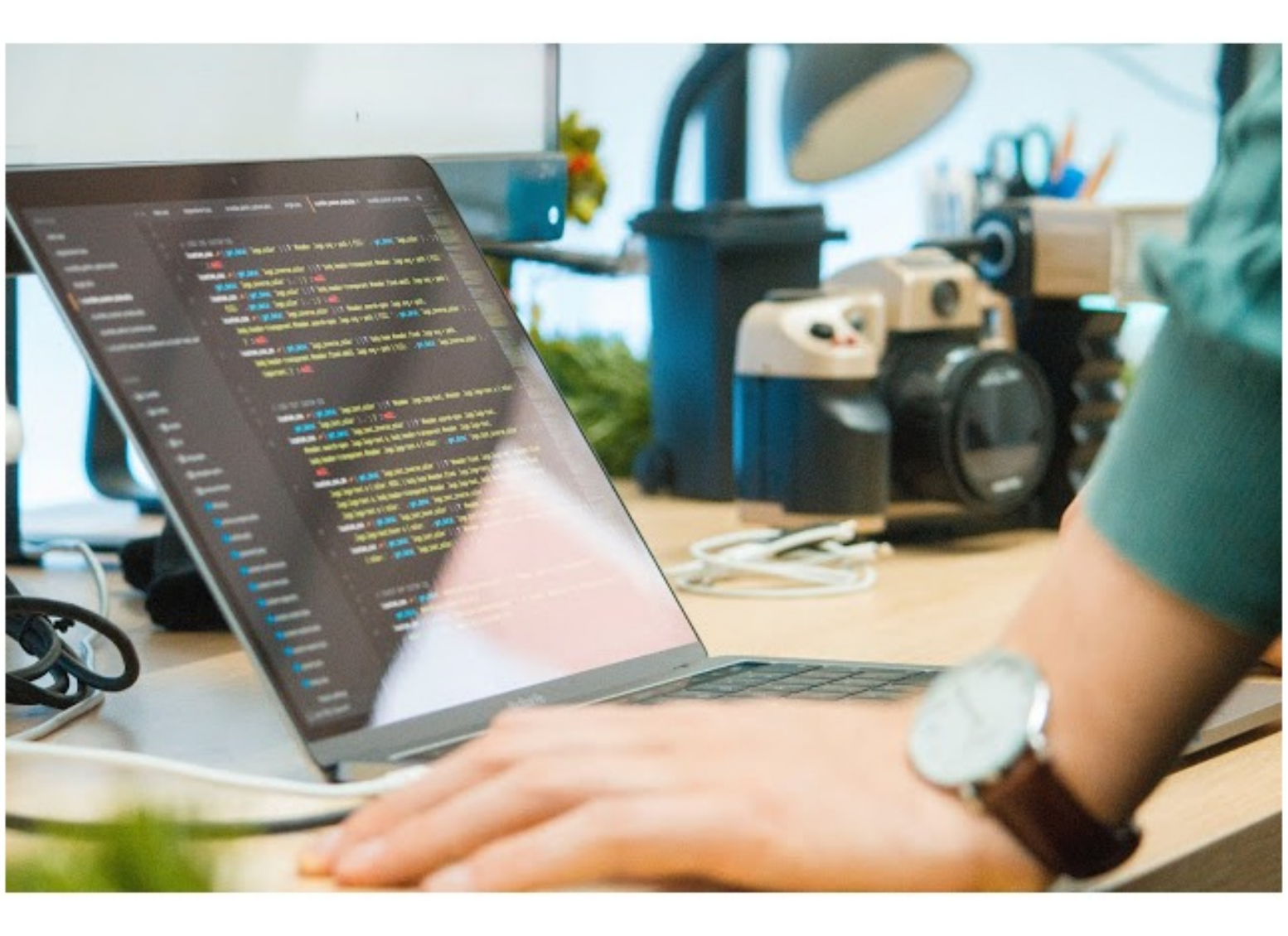
在Python中，函式是function的實例，所以你可以自由傳遞，將一個函式中所定義的函式傳回也是常見的應用。例如：

```python
def add(n1):
    def func(n2):
        return n1 + n2
    return func

print(add(1)(2))  # 顯示 3
```

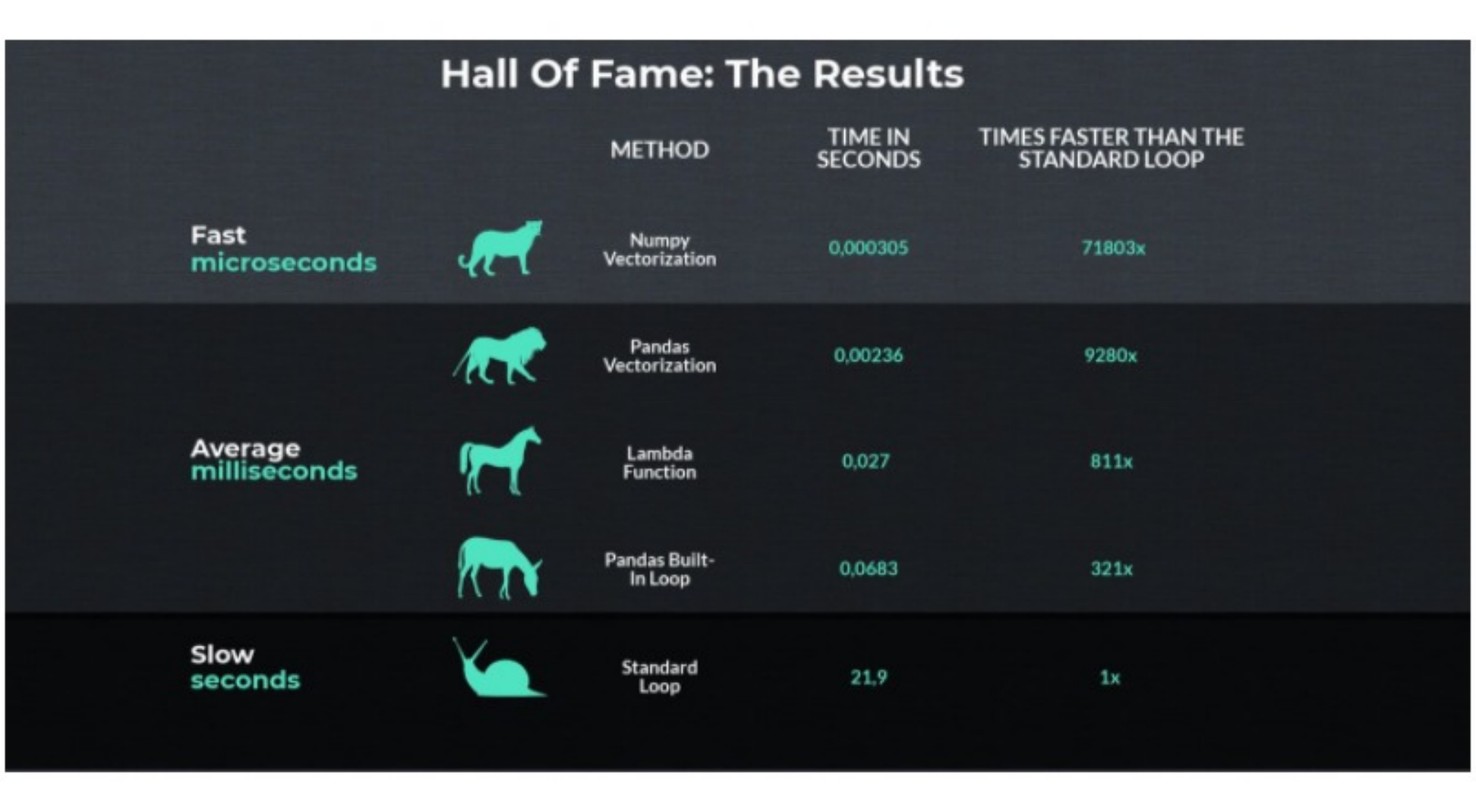
[Python教學]Python Lambda Function應用技巧分享
網站：learncodewithmike

Lambda函式，也就是匿名函式，不需要定義名稱，只有一行運算式，語法非常簡潔，功能強大，所以現代程式語言如Java、C#及Python等都支援Lambda函式，適用於小型的運算，Python的一些內建函式甚至使用它作為參數值的運算。



【Python效率】五種Pandas循環方法效率對比
網站：zhuanlan

絕大部分 Python 新手使用的下標循環語句實際上是非常之慢的，甚至在小數據集上也會消耗大量的運行時間。 文章中文進一步對五種不同的 for Loop 方法進行橫向對比，使大家更加明晰自己平時寫的各種對於循環到底效率幾何。



Hall Of Fame: The Results

	METHOD	TIME IN SECONDS	TIMES FASTER THAN THE STANDARD LOOP
Fast microseconds	Numpy Vectorization	0,000305	71803x
	Pandas Vectorization	0,00236	9280x
Average milliseconds	Lambda Function	0,027	811x
	Pandas Built-In Loop	0,0683	321x
Slow seconds	Standard Loop	21,9	1x

下一步：閱讀範例與完成作業