

- Ex2-4Add1: 請從數據可視化機器的實務角度來看，python 很常用，但是 javascript 卻很少使用的技術？反之是否有 javascript 比較常用，而 python 卻很少用的技術呢？
-
- Ex2-4Add2: 若有一個像陣列的容器，請問要如何印出全部的項？請就 pyhton 與 JavaScript 分別寫出這個簡單又重要的程式。
-
- Ex2-4Add3: Filter, Map, Reduce, 是在函數式程式設計中常用的指令。請分別用 pyhton 與 javascript，解決相同的問題：問題是先有一個容器物件內容是 1~5，然後 filter(過濾)出奇數項，再使用 map(對應函數)將每項值都平方，最後再 Reduce(減少)成只有一項的總和。
-

Ex2-4Add1: 請從數據可視化機器的實務角度來看，python 很常用，但是 javascript 卻很少使用的技術？反之是否有 javascript 比較常用，而 python 卻很少用的技術呢？

Ans:

例如：JavaScript 常見的習慣用法是方法鏈接(method chaining)：方法鏈接是涉及使用「點」表示法，從其自己的方法返回一個對象，以便在結果上調用另一個方法，例如 D3.select 就有很高的使用率。

同時，在 python 則很少有這種用法。

反之。python 很常見到元組的反打包(Tuple Unpacking)。

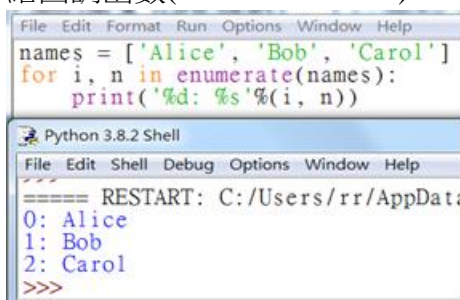
例如，兩數的交換或多參數的傳遞...等。這些，JavaScript 都不可這樣使用。

Ex2-4Add2: 若有一個像陣列的容器，請問要如何印出全部的項？請就 pyhton 與 JavaScript 分別寫出這個簡單又重要的程式。

Ans:

通常，在追蹤列表(list)的同時，追蹤項目的索引(item's index)，常常會很有用。因此，Python 具有非常方便的內置列舉函數：(左圖程式)

JavaScript 的列表方法（例如 forEach 和函數的 map, reduce, filter），並將疊代的項目及其索引提供給回調函數(callback function)：(右圖程式)



```
File Edit Format Run Options Window Help
names = ['Alice', 'Bob', 'Carol']
for i, n in enumerate(names):
    print('%d: %s'%(i, n))

Python 3.8.2 Shell
File Edit Shell Debug Options Window Help
==== RESTART: C:/Users/rr/AppData
0: Alice
1: Bob
2: Carol
>>>
```



```
JavaScript + No-Library (pure JS)
1 var names = ['Alice', 'Bob', 'Carol'];
2 names.forEach(function(n, i){
3     console.log(i + ': ' + n);
4 });
5
6
```

Running fiddle

```
"0: Alice"
"1: Bob"
"2: Carol"
>_
```

Ex2-4Add3: Filter, Map, Reduce, 是在函數式程式設計中常用的指令。請分別用 pyhton 與 javascript，解決相同的問題：問題是先有一個容器物件內容是 1~5，然後 filter(過濾)出奇數項，再使用 map(對應函數)將每項值都平方，最後再 Reduce(減少)成只有一項的總和。

Ans:

JS:

```
var nums = [1, 2, 3, 4, 5];
var sum = nums.filter(function(o){ return o%2 })
```

```
.map(function(o){ return o * o})  
.reduce(function(a, b){return a+b});  
console.log('Sum of the odd squares is ' + sum);
```

或

```
var l=[1,2,3,4,5]  
var isOdd = function(x){ return x%2; };  
var sq     = function(x){ return x*x; };  
var total  = function(a,b){ return a+b; };  
z=l.filter(isOdd).map(sq).reduce(total);  
console.log(z);
```

#####

在 python:

#Python' s list comprehensions

```
nums = [1,2,3,4,5]  
odd_squares = [x * x for x in nums if x%2]  
sum(odd_squares)
```

或

```
from functools import reduce  
nums = [ 1, 2, 3, 4, 5]  
odds = filter(lambda x: x % 2, nums)  
odds_sq = map(lambda x: x * x, odds)  
reduce(lambda x, y: x + y, odds_sq)
```
