Problem-Based Learning Python 3

Lecture: 11-function

Online Python

- https://www.programiz.com/python-programming/online-compiler/

Lecture Notes

- https://web.phy.ntnu.edu.tw/~hongyi/?url=notes

矩陣(Numpy)、串列(List)、表格(Pandas)

- 串列List

```
[[96, 65, 73], [88, 76, 82], [92, 84, 89]]
```

- 矩陣Numpy array

```
[[96 65 73]
 [88 76 82]
 [92 84 89]]
```

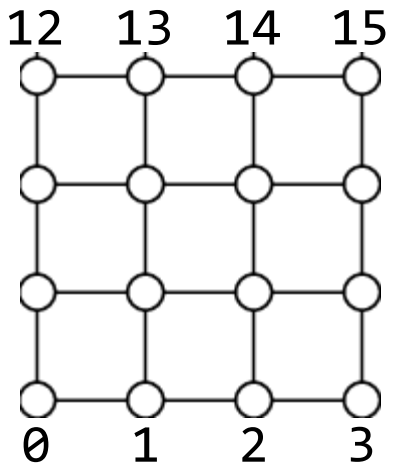
- 表格Pandas DataFrame

	A	B	C ← columns
a	96	65	73
b	88	76	82
c	92	84	89

↑ index

Problem

- 使用for、if、print



- 結果：

```
(x,y)  i  +x  +y
(0,0)  0   1   4
(1,0)  1   2   5
(2,0)  2   3   6
```

Solution

```
import numpy as np
nx = 4
ny = 4
size = nx * ny
print(u'(x,y)  i  +x  +y')
for iy in range(ny):
    for ix in range(nx):
        i = iy * nx + ix
        ipx = ix + 1
        if ix == nx - 1:
            ipx = 0
        jpx = (iy)*nx + ipx
        ipy = iy + 1
        if iy == ny - 1:
            ipy = 0
        jpy = (ipy)*nx + ix
        print(f"({ix:1d},{iy:1d}) {i:2d} {jpx:2d}  {jpy:2d}" )
```

Problem

- 使用Numpy產生一個新的矩陣
- `H = np.zeros((size, size), dtype=float)`

$$H[i,j] = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ \vdots \\ 15 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & \cdots & 15 \end{array} \\ \begin{pmatrix} & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \end{pmatrix} \end{array}$$

- 令H[i,j]=-1.0
- 其中i和j的關係為：

(x,y)	i	j=+x	j=+y
(0,0)	0	1	4
(1,0)	1	2	5
(2,0)	2	3	6

```
[[ 0. -1.  0.  0. -1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0. -1.  0.  0. -1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1.  0.  0. -1.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [-1.  0.  0.  0.  0.  0.  0. -1.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0. -1.  0.  0. -1.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0. -1.  0.  0. -1.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0. -1.  0.  0. -1.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0. -1.  0.  0.  0.  0.  0.  0. -1.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0. -1.  0.  0. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -1.  0.  0. -1.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0. -1.  0.  0.  0.  0.  0.  0. -1.]
 [-1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -1.  0.  0.]
 [ 0. -1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -1.  0.]
 [ 0.  0. -1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. -1.]
 [ 0.  0.  0. -1.  0.  0.  0.  0.  0.  0.  0.  0. -1.  0.  0.  0.]]
```

Solution

```
H = np.zeros((size, size), dtype=float)
for i in range(size):
    iy = int(i/nx)
    ix = i - iy * nx
    ipx = ix + 1
    if ix == nx - 1:
        ipx = 0
    jpx = (iy)*nx + ipx
    ipy = iy + 1
    if iy == ny - 1:
        ipy = 0
    jpy = (ipy)*nx + ix
    H[i,jpx] = -1.0
    H[i,jpy] = -1.0
```

對稱矩陣

- `H = H + H.T`
- `print(H)`

```
[[ 0. -1.  0. -1. -1.  0.  0.  0.  0.  0.  0.  0. -1.  0.  0.  0.]
 [-1.  0. -1.  0.  0. -1.  0.  0.  0.  0.  0.  0.  0. -1.  0.  0.]
 [ 0. -1.  0. -1.  0.  0. -1.  0.  0.  0.  0.  0.  0.  0. -1.  0.]
 [-1.  0. -1.  0.  0.  0.  0. -1.  0.  0.  0.  0.  0.  0.  0. -1.]
 [-1.  0.  0.  0.  0. -1.  0. -1. -1.  0.  0.  0.  0.  0.  0.  0.]
 [ 0. -1.  0.  0. -1.  0. -1.  0.  0. -1.  0.  0.  0.  0.  0.  0.]
 [ 0.  0. -1.  0.  0. -1.  0. -1.  0.  0. -1.  0.  0.  0.  0.  0.]
 [ 0.  0.  0. -1. -1.  0. -1.  0.  0.  0.  0. -1.  0.  0.  0.  0.]
 [ 0.  0.  0.  0. -1.  0.  0.  0.  0. -1.  0. -1. -1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0. -1.  0.  0. -1.  0. -1.  0.  0. -1.  0.  0.]
 [ 0.  0.  0.  0.  0.  0. -1.  0.  0. -1.  0. -1.  0.  0. -1.  0.]
 [ 0.  0.  0.  0.  0.  0.  0. -1. -1.  0. -1.  0.  0.  0.  0. -1.]
 [-1.  0.  0.  0.  0.  0.  0.  0. -1.  0.  0.  0.  0. -1.  0. -1.]
 [ 0. -1.  0.  0.  0.  0.  0.  0.  0. -1.  0.  0. -1.  0. -1.  0.]
 [ 0.  0. -1.  0.  0.  0.  0.  0.  0.  0. -1.  0.  0. -1.  0. -1.]
 [ 0.  0.  0. -1.  0.  0.  0.  0.  0.  0.  0. -1. -1.  0. -1.  0.]]
```

矩陣對角化

- `E, V = np.linalg.eigh(H)`
- `print(E)`

```
[-4.00000000e+00 -2.00000000e+00 -2.00000000e+00 -2.00000000e+00
 -2.00000000e+00 -1.32775367e-15 -6.75099748e-16 -6.65698176e-16
 -4.40966188e-16  6.66430211e-16  9.13556330e-16  2.00000000e+00
  2.00000000e+00  2.00000000e+00  2.00000000e+00  4.00000000e+00]
```

如何讓程式看起來簡潔(不一定明瞭)，和如何讓程式跑得快是不一樣的概念

- 如何讓程式看起來簡潔(不一定明瞭)
- 第一段程式中出現的程式碼

```
        ipx = ix + 1
        if ix == nx - 1:
            ipx = 0
        jpx = (iy)*nx + ipx
        ipy = iy + 1
        if iy == ny - 1:
            ipy = 0
        jpy = (ipy)*nx + ix
```

- 寫程式不要重覆寫，容易出錯
- 第二段程式中出現的程式碼

```
        ipx = ix + 1
        if ix == nx - 1:
            ipx = 0
        jpx = (iy)*nx + ipx
        ipy = iy + 1
        if iy == ny - 1:
            ipy = 0
        jpy = (ipy)*nx + ix
```

定義函式function及原來程式改寫

```
def hopping(nx, ny, i):
    iy = int(i/nx)
    ix = i - iy * nx
    ipx = ix + 1
    if ix == nx - 1:
        ipx = 0
    jpx = (iy)*nx + ipx
    ipy = iy + 1
    if iy == ny - 1:
        ipy = 0
    jpy = (ipy)*nx + ix
    return jpx, jpy
```

- 如何使用
- `jpx, jpy = hopping(nx, ny, i)`