python 程式設計

第 六 講

字串：string

字串

- 使用單引號或雙引號夾住的字元：'abc'，"abc"，"Tom's"
- 多列文字使用三個引號：

```
a = '''國立中央大學
數學系'''

b = "國立中央大學\n數學系"          # a 與 b 是一樣的字串
```

- 以三個引號夾住某段程式碼不作設定，效果等同註解

```
...
"""
s = 0
for n in range(100) : s += n
    print( "sum of 0..99 is" , s )
"""
...
```

❖ 若使用此種註解方式，第一個三引號仍須遵循縮排規定

字元 (一)

- 字串為萬國碼字元序列，以 UTF-8 編碼
- len(字串):取得字串長度，即字元數
- list(字串):分解字串成字元串列

```
>>> school = '中央大學'
>>> foo = list(school)
>>> foo
['中', '央', '大', '學']
```

- for 迴圈佚代取出字元

```
math = '中央大學MATH'

# for 迴圈每次取出一個字元
for c in math : print( c , end=" - " )

# 使用下標取得字元
for i in range(len(math)) : print( math[i] , end=" - " )
```

以上兩個迴圈都輸出

中 - 央 - 大 - 學 - M - A - T - H -

字元 (二)

- 特殊字元

\' 單引號字元	\\ 反斜線字元	\ooo 8 進位 ooo 字元
\" 雙引號字元	\b 退後字元	\xhh 16 進位 hh 字元
\n 換行字元	\t 定位字元	

❖ 相等字元

'm' == '\155' (八進位數) == '\x6d' (十六進位數)

原生字串 (raw string)

- 原生字串：在引號前加 **r** ，代表引號所夾的字元就是原字元

```
>>> print( '\\' )
\
>>> print( r'\\' )
\\

>>> print( "cat\ndog" )
cat
dog
>>> print( r"cat\ndog" )
cat\ndog
```

字串合成與複製 (一)

- 多個字串合成

```
a = "中央大學"  " "  "數學系"          # a = "中央大學 數學系"
```

- 跨列字串合成:使用小括號

```
b = ( "春眠不覺曉，處處聞啼鳥。"
      "夜來風雨聲，花落知多少。" )
```

- 字串合成:使用 +

```
c = "中央大學" + " " + "數學系"        # c = "中央大學 數學系"
```

- 字串複製: 使用 *

```
d = "加油!" * 3                        # d = "加油!加油!加油!"
```

字串合成與複製 (二)

- 字元序列

```
foo = "中央大學數學系"
```

字串 foo	'中'	'央'	'大'	'學'	'數'	'學'	'系'
正向下標	0	1	2	3	4	5	6
逆向下標	-7	-6	-5	-4	-3	-2	-1

複製字串內字元 (一)

- 使用下標截取字元

`a[:]`	複製全部
`a[i:j]`	複製 `a[i]` 到 `a[j-1]` 間的字元
`a[i:]`	複製 `a[i]` 到末尾的所有字元
`a[:j]`	複製 `a[0]` 到 `a[j-1]` 的所有字元
`a[i:j:k]`	複製 `a[i]`、`a[i+k]`、`a[i+2k]`... 直到下標不超過 `j`

複製字串內字元 (二)

- 順向複製字元：

```
>>> a = "中央大學 MATH"

>>> b = a[5:]                    # b = "MATH"
>>> c = a[:]                     # c = "中央大學 MATH"
>>> d = a[0:]                    # d 同上
>>> e = a[0:3:2]                 # e = "中大"
```

- 逆向複製字元：k 為負數

```
>>> a = "NCU MATH"

>>> b = a[-1:-5:-1]              # b = "HTAM"
>>> c = a[-1::-1]                # c = "HTAM UCN" 逆轉字串
>>> d = a[::-1]                  # d = 同上
```

數字與字串 (一)

- **float(foo):**將字串 **foo** 轉型為浮點數
- **int(foo):**將字串 **foo** 轉型為 10 進位整數
- **int(foo,x):**將 **x** 進位的 **foo** 數字字串轉型為 10 進位整數

```
>>> a = float("3.14")          # a = 3.14
>>> b = int("24")              # b = 24
>>> c = int("24",5)            # c = 14
>>> d = int("ff",16)           # d = 255
```

數字與字串 (二)

- **bin(n):將整數 n 轉成 2 進位數字字串**
- **hex(n):將整數 n 轉成 16 進位數字字串**

```
>>> bin(13)
'0b1101'                    # 0b 代表 2 進位(binary)
>>> hex(60)
'0x3c'                      # 0x 代表 16 進位(hexadecimal)

>>> int(bin(13),2)          # 13 先轉為 2 進位數字字串再轉回整數
13
>>> int(hex(60),16)         # 60 先轉為 16 進位數字字串再轉回整數
60
```

- 數字逆轉

```
>>> e = int( str(12345)[::-1] )      # e = 54321
```

字串字元不能更動

- 字串字元無法更改（**immutable**）

```
>>> a = "數學系"
>>> a[2] = "人"                    # 錯誤，字串字元無法更動
>>> a[:2] = "物理"                 # 錯誤，字串字元無法更動
```

- 重新組合字串

```
>>> b = "理學院數學系"
>>> b = b[:3] + "物理" + b[-1]    # b = "理學院物理系"
```

字串分解 (一)

- list(foo):分解 foo 字串為字元串列

```
>>> a = "MATH"
>>> b = list(a)                    # b = ['M', 'A', 'T', 'H']
```

- foo.split(sep,n):分解 foo 字串, 前 n 個字串依 sep 分隔字串分解

```
>>> a = "M-A-T-H".split('-')       # a = ['M', 'A', 'T', 'H']
>>> b = "M-A-T-H".split('-',1)     # b = ['M', 'A-T-H']
>>> c = "M-A-T-H".split('-',2)     # c = ['M', 'A', 'T-H']

>>> d = "M--A--T--H".split('--')   # d = ['M', 'A', 'T', 'H']
>>> e = "M--A--T--H".split('-')    # e = ['M', '', 'A', '', 'T', '', 'H']
```

以下 sep 為一個或兩個空格

```
>>> c = "中 大  MATH".split(" ")    # c = ['中', '大', '', '', 'MATH']
>>> d = "中 大  MATH".split("  ")   # d = ['中 大', ' MATH']
```

❖ sep 不得為空字串

字串分解 (二)

- **foo.split()**:分解 **foo** 字串，取出非空格間的字元

```
>>> a = "中 大  MATH".split()          # a = ['中', '大', 'MATH']
>>> b = " 中 大  MATH ".split()        # b = ['中', '大', 'MATH']
>>> c = "中 大\n\t  MATH".split()      # c = ['中', '大', 'MATH']
```

❖ python 的空格包含(' ')、換行('\n')、定位('\t')、回行首字元('\r')

- 字串分解常與輸入合用，藉以一次設定許多變數

```
# 輸入三個整數以空格分開
>>> a , b , c = list( map( int , input("> ").split() ) )
> 2 4 9
>>> a + b + c
15
```

字串合併 (一)

- **sep.join(foo)**：將字串或字串串列 **foo** 合併起來，字串間有 **sep** 分隔字串

```
>>> a = "--".join(['中', '央', '大', '學'])   # a = "中--央--大--學"

>>> b = "".join(['MA', 'TH'])                 # b = "MATH"

>>> c = "-".join( input("-> ") )              # c = '1-2-3'
-> 123

>>> d = " ".join( "ncu" )                     # d = 'n c u'
```

- 對調 年/月/日 成為 月/日/年

```
>>> "/".join("105/3/26".split('/',1)[::-1])
3/26/105
```

字串合併 (二)

- 將字串中的分隔字元換成其他字串

```
>>> a = "中---央-大--學"
>>> b = "".join( a.split("-") )    # b = "中央大學"
>>> c = "**".join( b )             # c = "中**央**大**學"
```

以上末兩式可合併成一列

- 分離、合併、逆轉混用

```
>>> a = "123.45.6789"
>>> b = ("".join(a.split(".")))[::-1]    # b = 987654321
```

- 與 **map** 配合即可不用迴圈列印整個串列資料

```
>>> print( " ".join( map( str , [12,34,56] ) ) )
12 34 56
```

字串合併 (三)

- 依換行字元分解為串列
  - **foo.splitlines()**：將 **foo** 字串依換行字元分解成字串串列

```
>>> foo = "abc\n123\n\nsss"

>>> foo.splitlines()
['abc', '123', '', 'sss']

>>> foo.splitlines(1)
['abc\n', '123\n', '\n', 'sss\n']
```

❖ 若 splitlines 內有任何非零參數，則會顯示換行字元

萬國碼字元編號 (一)

- **ord(c)**：字元 **c** 在萬國碼編號
- **chr(n)**：萬國碼編號 **n** 的對應字元

```
>>> n = ord('a')                # n = 97
>>> c = chr(97)                 # c = 'a'

>>> a = ord('中')               # a = 20013
>>> b = chr(20013)              # b = '中'
>>> chr(ord('中'))
'中'
```

萬國碼字元編號 (二)

- 字串比大小：依次比較各字元萬國碼編號

```
>>> "abc" < "ade"
True

>>> "abc" < "ab"
False

>>> "中大" > "中央"          # ord("大"):22823   ord("央"):22830
False
```

萬國碼字元編號 (三)

- 基本操作

```
>>> [ ord(x) for x in "中央大學" ]
[20013, 22830, 22823, 23416]

>>> "".join( [ chr(x) for x in [20013, 22830, 22823, 23416] ] )
'中央大學'
```

- 'abcd … z' 字串

```
>>> "".join( [ chr(ord('a')+x) for x in range(26) ] )
'abcdefghijklmnopqrstuvwxyz'
```

移除字串兩側空格

- foo.strip() :去除 foo 字串兩側的空格，回傳剩餘的字串
- foo.lstrip():去除 foo 字串左側的空格，回傳剩餘的字串
- foo.rstrip():去除 foo 字串右側的空格，回傳剩餘的字串

```
>>> foo = "  中央大學  "
>>> a = foo.strip()                    # a = "中央大學"
>>> b = foo.lstrip()                   # b = "中央大學  "
>>> c = foo.rstrip()                   # c = "  中央大學"
>>> foo                                # foo 保持不變
"  中央大學  "
```

移除字串兩側指定字元

- **foo.strip (b)** :去除 **foo** 字串兩側在 **b** 字串內的字元，回傳剩餘的字串
- **foo.lstrip(b)**:去除 **foo** 字串左側在 **b** 字串內的字元，回傳剩餘的字串
- **foo.rstrip(b)**:去除 **foo** 字串右側在 **b** 字串內的字元，回傳剩餘的字串

```
>>> foo = "--132-45—6783--"
>>> a = foo.strip("-")              # a = "132-45--6783"
>>> b = foo.strip("-13")            # b = "2-45--678"
>>> c = foo.lstrip("-123")          # c = "45--6783--"
>>> d = foo.rstrip("-123")          # d = "--132-45--678"
>>> foo                             # foo 保持不變
"--132-45--6783--"
```

搜尋字串

- foo.find(s):在 foo 字串搜尋 s 字串出現的下標位置
- foo.find(s,a,b):在 foo[a:b] 字串搜尋 s 字串出現的下標位置
- foo.rfind(s):在 foo 字串搜尋 s 字串出現的最高下標位置
- foo.rfind(s,a,b):在 foo[a:b] 字串搜尋 s 字串出現的最高下標位置

以上第二與第四種形式也可不設定 b ,則搜尋範圍變成 foo[a:]

```
>>> foo = "math.123.math.32123"
>>> a = foo.find("math")          # a = 0
>>> b = foo.find("math",1,10)     # b = -1 , -1 代表沒有找到
>>> c = foo.find("math",1)        # c = 9
>>> d = foo.find("math",1,13)     # d = 9
>>> e = foo.rfind("123")          # e = 16
```

❖ 若搜尋不到字串，則回傳 -1

搜尋字串：in 與 not in

- **foo in a**：檢查 **a** 字串是否包含 **foo** 字串，回傳真假值
- **foo not in a**：檢查 **a** 字串是否不包含 **foo** 字串，回傳真假值

```
>>> '8' in "13579"
False
>>> '13' in "123456789"
False
>>> '654' in "987654321"
True
>>> '456' not in "987654321"
True
```

搜尋字串出現次數

- foo.count(s):在 foo 字串搜尋 s 字串出現次數
- foo.count(s,a):在 foo[a:] 字串搜尋 s 字串出現次數
- foo.count(s,a,b):在 foo[a:b] 字串搜尋 s 字串出現次數

```
>>> foo = "math.123.math.32123"

>>> a = foo.count("math")                  # a = 2
>>> b = foo.count("math",1,10)             # b = 0
>>> c = foo.count("math",5)                # c = 1
```

檢查字串的起始或末尾有某字串

- **foo.startswith(s)**：檢查 **foo** 字串是否以 **s** 字串起始
- **foo.startswith(s,a,b)**：檢查 **foo[a:b]** 字串是否以 **s** 字串起始
- **foo.endswith(s)**：檢查 **foo** 字串是否以 **s** 字串終結
- **foo.endswith(s,a,b)**：檢查 **foo[a:b]** 字串是否以 **s** 字串終結

以上的 **s** 也可為包含字串的常串列(**tuple**)，第二與第四種形式的 **b** 也可省略，代表搜尋範圍變成 **foo[a:]**

```
>>> foo = "www.math.ncu.edu.tw"

>>> a = foo.startswith("www")                  # a = True
>>> b = foo.startswith("ncu",3)                # b = False
>>> c = foo.startswith(("phy","math"),4)       # c = True

>>> d = foo.endswith(("kr","tw","cn"))         # d = True
>>> e = foo.endswith(("tw","jp"),4)            # e = True
```

取代舊字串成新字串

- **foo.replace(old,new)**:將 **foo** 字串內 **old** 全部改為 **new**
- **foo.replace(old,new,n)**:將 **foo** 字串前 **n** 個 **old** 改為 **new**

以上 **old** 與 **new** 都是字串，同時取代動作僅會產生新字串，原字串不變

```
>>> foo = "數學系 物理系 化學系"

>>> a = foo.replace("數學","中文")     # a = '中文系 物理系 化學系'
>>> b = foo.replace("系","人",2)       # b = '數學人 物理人 化學系'
```

大小寫轉換

- foo.upper():將 foo 字串內的英文字母轉為大寫
- foo.lower():將 foo 字串內的英文字母轉為小寫
- foo.title():將 foo 字串內每個英文字的第一個字母轉為大寫

```
>>> foo = "My name is Tom."

>>> a = foo.upper()          # a = "MY NAME IS TOM."
>>> b = foo.lower()          # b = "my name is tom."
>>> c = foo.title()          # c = "My Name Is Tom."
```

❖ title 定義為至少有個大寫字母，大寫字母在非大小寫字元之後，小寫字母則緊鄰在大寫字母之後。

判斷字串字元類型

- foo.isdigit():判斷 foo 字串是否全為數字
- foo.isalpha():判斷 foo 字串是否全為英文字母
- foo.isalnum():判斷 foo 字串是否全為英文字母或數字
- foo.isupper():判斷 foo 字串是否全為英文字母全為大寫
- foo.islower():判斷 foo 字串是否全為英文字母全為小寫
- foo.istitle():判斷 foo 字串的所有英文字第一個字元是否全為大寫
- foo.isspace():判斷 foo 字串是否全為空格字元

```
>>> a = "123".isdigit()                    # a = True
>>> b = "123abc".isalpha()                 # b = False
>>> c = "123abc".isalnum()                 # c = True
>>> d = "We Are In 30s.".istitle()         # d = False
>>> e = "My Age Is 30.".istitle()          # e = True
>>> f = " \n\t\r".isspace()                # f = True
```

format 格式輸出 (一)

- 使用 **format** 設定輸出格式:

```
>>> "{}/{}/{}".format("1977",8,10)
'1977/8/10'
```

- **format** 輸出字串

```
>>> a = "{}有 {} 公斤".format("香蕉",148) # a = '香蕉有 148 公斤'
```

- 設定輸出位置:{0},{1},…,{n}

```
>>> a = "{1}月-{2}日-{0}年".format(2017,3,13)  # a = '3月-13日-2017年'
>>> b = "{2}/{1}/{0}".format(2017,3,13)        # b = '13/3/2017'
```

❖ {n} 的 n 可以重複

- 設定輸出寬度、填補字元、精度、對齊方式

```
>>> a = "{0}:{1:5} kg".format('香蕉',234)          # a = '香蕉:  234 kg'
>>> b = "{0:>4}:{1:#>5} kg".format('鳳梨',234)     # b = '  鳳梨:##234 kg'
>>> c = '{0}:{1:#>7.2f} kg'.format('芭樂',234.5)   # c = '芭樂:#234.50 kg'
```

format 格式輸出 (二)

- 整數格式輸出
  - 填補字元、對齊、寬度

```
>>> "{0:#<5}{1:@>5}{2:*^6}".format(123,45,67)
'123##@@@45**67**'
```

❖ <、>、^ 分別為向左、向右、置中對齊符號，填補字元於其前，寬度於其後

  - 進位方式

```
>>> "{0:#<4}-{0:010b}-{0:#>4x}".format(234)
'234#-0011101010-##ea'
```

❖ 進位字母置於寬度之後，b 二進位、o 八進位、x/X 小寫/大寫十六進位

format 格式輸出 (三)

- 整數格式輸出
  - 正負號與其位置

```
>>> a = "{0:#=+5}".format(12)      # a = '+##12'
>>> b = "{0:#>+5}".format(12)      # b = '##+12'
>>> c = "{0:#<+5}".format(12)      # c = '+12##'
>>> d = "{0:#^+5}".format(12)      # d = '#+12#'
>>> e = "{0:#=5}".format(-12)      # e = '-##12'
```

❖ 寬度數字前的 + 號代表當整數為正數時則輸出正號

  - 逗點

```
>>> a = "{:#>12,}".format(9834567)  # a = '###9,834,567'
```

format 格式輸出 (四)

- 浮點數格式輸出
  - 小數點輸出

```
>>> "{:>f}".format(12.239013533)    # 預設小數點精度為 6
'12.239014'

>>> "{:#>7.2f}".format(12.2390)     # 7 格列印，小數佔用 2 格
'##12.24'
```

  - 科學記號輸出

```
>>> "{0:e}{0:#>10.2e}{0:#>10.2E}".format(12.2390)
'1.223900e+01##1.22e+01##1.22E+01'
```

  - 百分號輸出

```
>>> '{0:%}||{0:#>.1%}||{0:#>10.2%}'.format(12.239)
'1223.900000%||1223.9%||##1223.90%'
```

format 格式輸出 (五)

- 使用名稱參數:使用名稱代替數字代號

```
>>> "{b} has {a:#>3} dollars.".format(a=99,b="Tom")
'Tom has #99 dollars.'

>>> {a:{f}{g}{x}.{y}f}".format(a=2.343,f="#",g=">",x=5,y=2)
'#2.34'
```

format 格式輸出 (六)

- 串列格式輸出
  - **format** 左側字串內大括號數量需與串列元素數量相同
  - 先組合 **format** 左側字串，再拆解輸出的串列

```
>>> a = [ 2 , 8 , 24 , 7 ]
>>> b = "{:0>3} "*len(a)
>>> b.format(*a)
'002 008 024 007 '
```

❖ 使用星號於串列前可拆解串列成一個個元素

```
>>> a = "1 1 2 3 5 8 13 21 34 55"
>>> b = a.split()
>>> ("{:->3}"*len(b)).format(*b)
'--1--1--2--3--5--8-13-21-34-55'
```

❖ 使用小括號先組合格式字串

位元組型別(一)：bytes與bytearray型別

- **string** 與 **bytes** 為同筆資料的不同儲存方式的物件，**string** 儲存萬國碼字元，**bytes** 儲存字元的編碼實作序列。
- **bytes** 物件一旦設定後即不可更動，**bytearray** 型別則為 **bytes** 型別的可變更版本 **(mutable)**

```
>>> a = 'c'                          # hex(ord(a)) = 0x63
>>> b = bytes(a,'utf-8')             # utf-8 編碼為 63
>>> b                                # b 佔用一個位元組
b'c'                                 # 等同 b'\x63'

>>> c = '中'                         # hex(ord(c)) = 0x4e2d
>>> d = bytes(c,'utf-8')             # utf-8 編碼為 e4 b8 ad
>>> d                                # d 佔用三個位元組
b'\xe4\xb8\xad'
```

位元組型別 (二)

以下為 e 字串轉為 utf-8 編碼後的 bytearray 物件與 big5 編碼後的 bytes 物件

```
>>> e = '中央'                          # e 為兩個萬國碼字元
>>> bytearray(e,'utf-8')               # 轉為 utf-8 編碼, 共六個位元組
bytearray(b'\xe4\xb8\xad\xe5\xa4\xae')

>>> bytes(e,'big5')                    # 轉為 big5 編碼, 共四個位元組
b'\xa4\xa4\xa5\xa1'
```

位元組型別 (三)

	string	bytes
型別名稱	str	bytes , bytearray
儲存內容	萬國碼字元	編碼後字元組
序列	字元	介於 [0,255] 的位元組
型式	'cat' '中'	bytearray(b'cat') (bytearray 物件) b'\xe4\xb8\xad' (bytes 物件)
str → bytes	x = '中'	y = x.encode() , y 為 bytes 物件 y = bytes(x,'utf-8') y = bytearray(x,'utf-8')
str ← bytes	x = y.decode() x = str(y,'utf-8')	y = b'\xe4\xb8\xad' y = bytearray(b'\xe4\xb8\xad')

❖ 上表內的 encode() 與 decode() 也可設定編碼方式，預設編碼方式為 utf-8

位元組型別 (四)

- 範例
  - 字母編碼後序號

```
for x in bytearray('cat','utf-8') :
    print( x , end=" " )
輸出:
  99 97 116
```

  - 編碼後的位元組

```
>>> "中央".encode()
b'\xe4\xb8\xad\xe5\xa4\xae'

>>> list( "中央".encode() )
[228, 184, 173, 229, 164, 174]
```

位元組型別 (五)

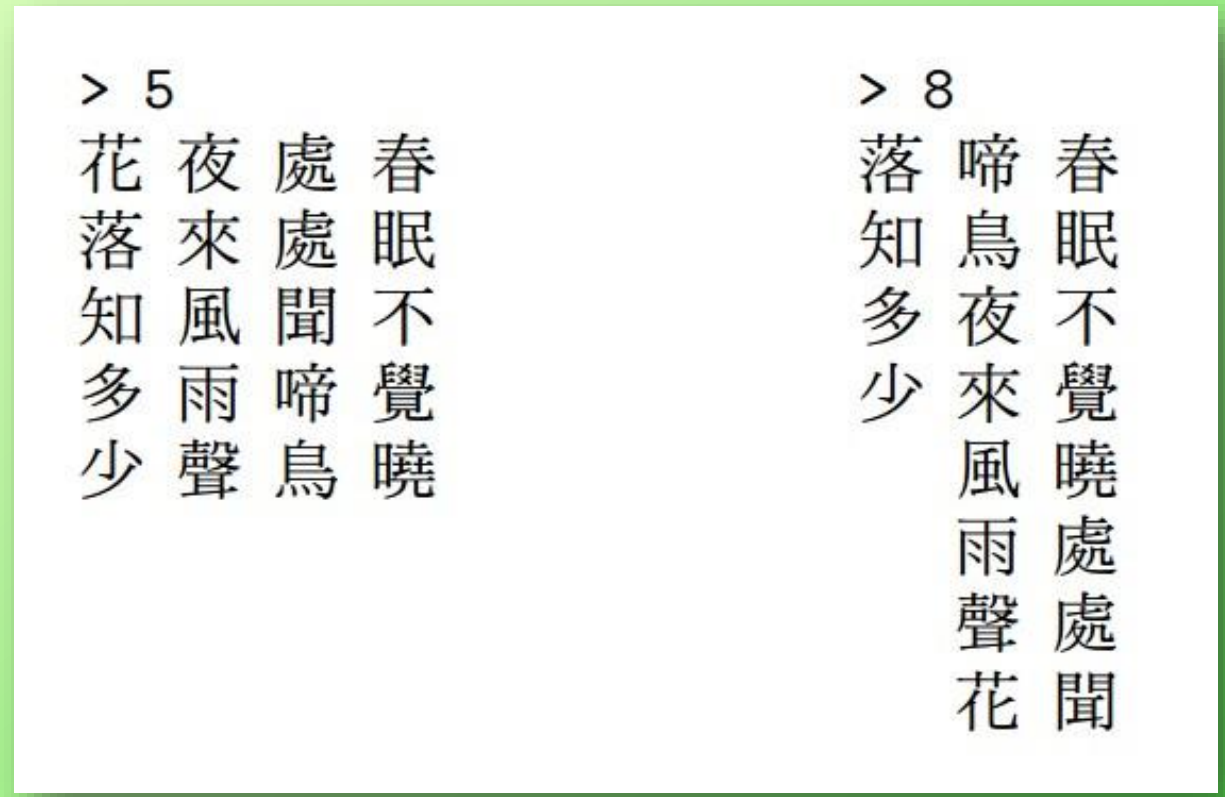
- 取得部份編碼後字元

```
>>> a = "中央".encode()                    # a 為 bytes 物件
>>> a[3:6]
b'\xe5\xa4\xae'
>>> a[3:6].decode()
'央'
```

- 更改 **bytearray**： 直接取代儲存的位元組內容

```
>>> b = bytearray( '中央' , 'utf-8' )  # b 為 bytearray 物件
>>> b[3:6] = '大'.encode()
>>> b.decode()
'中大'

>>> c = bytearray( 'cat' , 'utf-8' )
>>> c[1] = ord('u')                    # 將 c[1] 改為 'u' 字元的序號
>>> c
bytearray(b'cut')
```

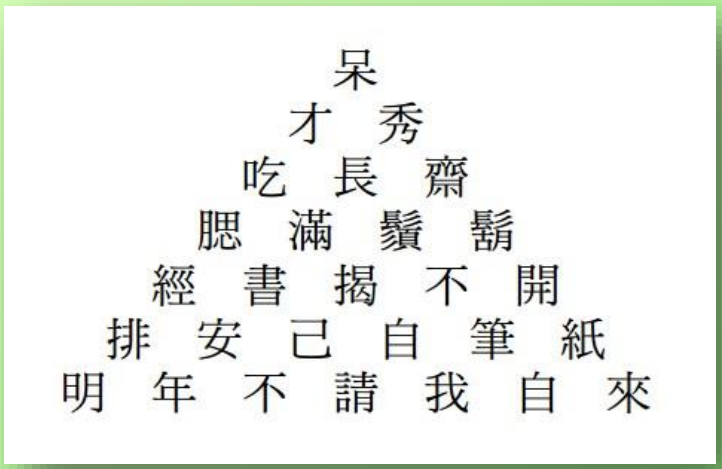
五言詩在書法中的排列 (一)



五言詩在書法中的排列 (二)

```
p = "春眠不覺曉處處聞啼鳥夜來風雨聲花落知多少"

while True :

    # 讀入詩句列數
    h = int(input("> "))

    # 計算詩句行數
    w = len(p)//h + ( 1 if len(p)%h else 0 )

    for i in range(h) :
        for j in range(w-1,-1,-1) :

            k = j*h+i
            print( p[k] if k < len(p) else "  " , end=" ")

        print()

    print()
```

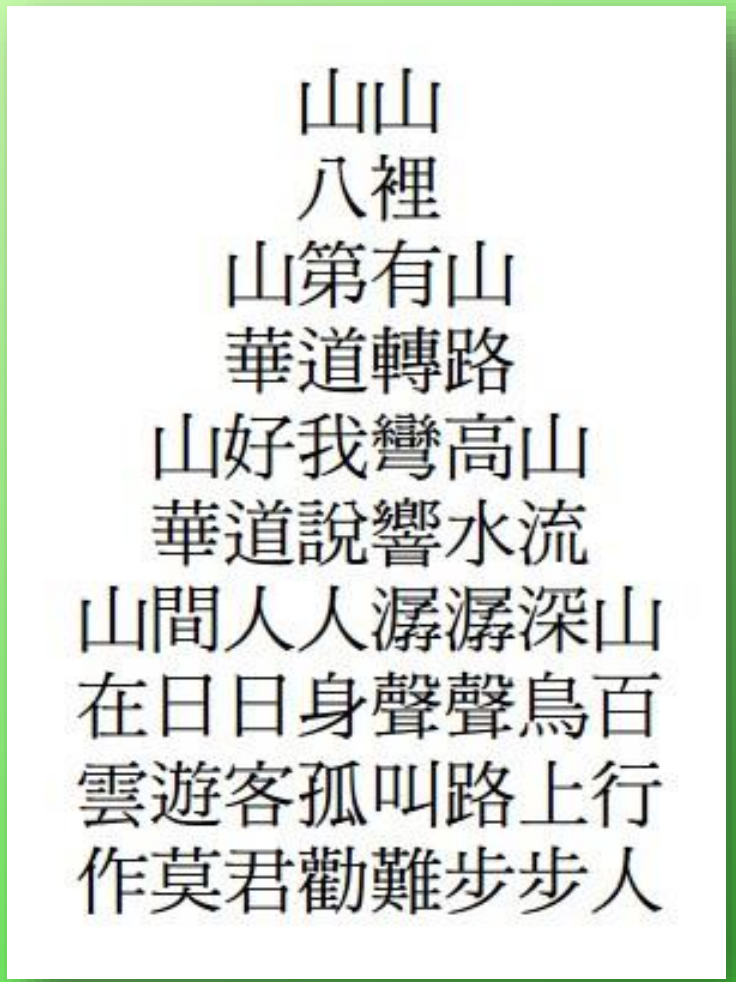
一至七字詩




```
p = "呆秀才吃長齋鬍鬚滿腮經書揭不開紙筆自己安排明年不請我自來"

k = 0
for s in range(1,8) :
    if s%2 :
       print( ' '*(2*(7-s)) + '  '.join(p[k:k+s]) )
    else :
       print( ' '*(2*(7-s)) + '  '.join(p[k:k+s][::-1]) )

    k += s
```

八山疊翠詩 (一)

山裡有山路轉彎
高山流水響潺潺
深山百鳥聲聲叫
路上行人步步難
勸君莫作雲遊客
孤身日日在山間
人人說道華山好
我道華山第八山

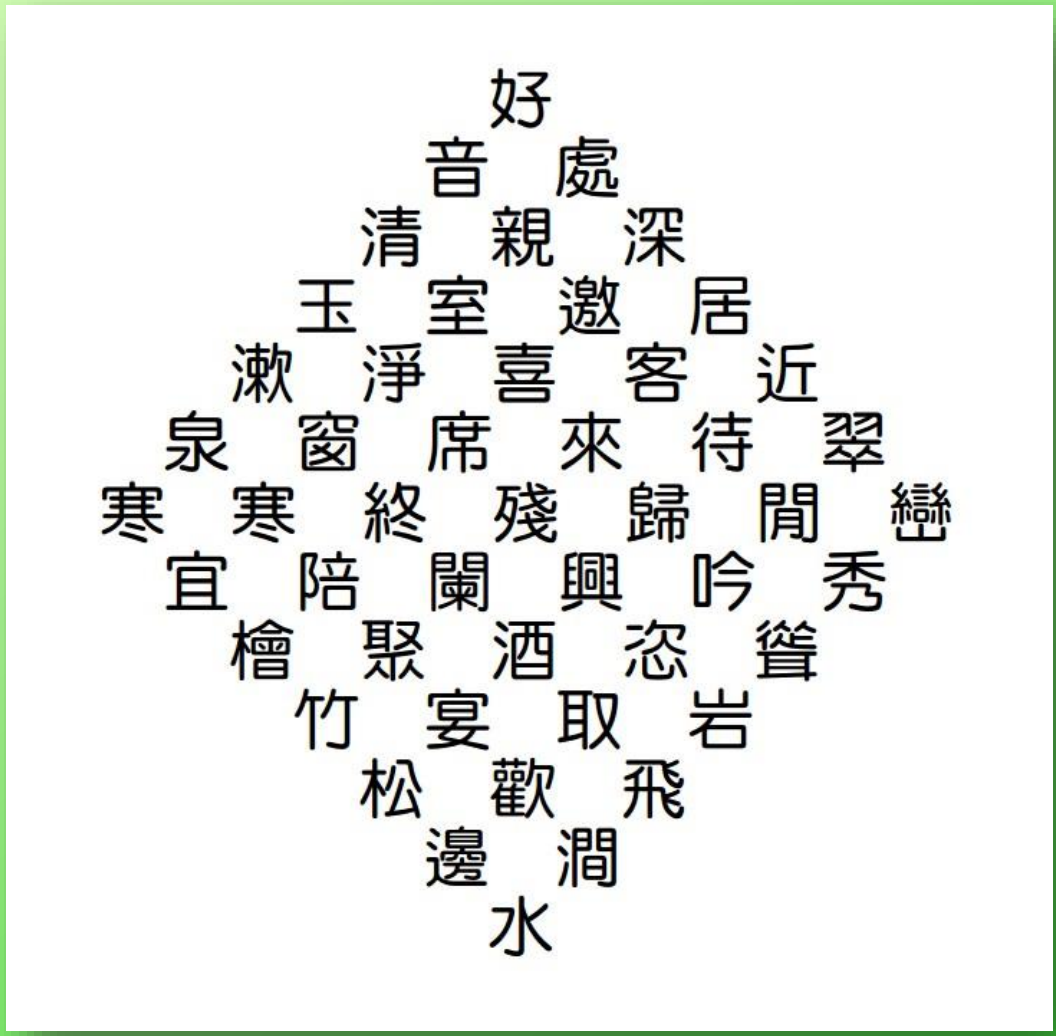


八山疊翠詩 (二)

```
p = ( "山裡有山路轉彎高山流水響潺潺"
      "深山百鳥聲聲叫路上行人步步難"
      "勸君莫作雲遊客孤身日日在山間"
      "人人說道華山好我道華山第八山" )

# a:左半部詩句 b:右半部詩句
a , b = p[28:][::-1] , p[:28]

# 由上而下, 半部詩句的橫列字數
w = [1,1,2,2,3,3,4,4,4,4]

k = 0
for i in range(len(w)) :
    if i%2 :
        print( "  "*(4-w[i]) + a[k:k+w[i]] + b[k:k+w[i]][::-1] )
    else :
        print( "  "*(4-w[i]) + a[k:k+w[i]][::-1] + b[k:k+w[i]] )

    k += w[i]
```

連環錦纏枝迴文詩 (一)

寒泉漱玉清音好
好處深居近翠巒
巒秀聳岩飛澗水
水邊松竹檜宜寒
寒窗淨室親邀客
客待閒吟恣取歡
歡宴聚陪終席喜
喜來歸興酒闌殘



連環錦纏枝迴文詩 (二)

```
poem = ( "寒泉漱玉清音好" "好處深居近翠巒"
         "巒秀聳岩飛澗水" "水邊松竹檜宜寒"
         "寒窗淨室親邀客" "客待閒吟恣取歡"
         "歡宴聚陪終席喜" "喜來歸興酒闌殘" )

n = 7
ds = [ -1 , 1 , 1 , -1 ]
dt = [ 1 , 1 , -1 , -1 ]

p = [None] * (2*n-1)

for s in range(len(p)) :
    p[s] = [ "  " for t in range(2*n-1) ]

# 以上 p 的設定也可改用一列表示
# p = [ [ "  " for t in range(2*n-1) ]
#       for s in range(2*n-1) ]
```

```
s , t , k = n-1 , 0 , 0
for i in range(len(poem)) :

    # 去除句尾句首重複字
    if i and i%7 == 0 : continue

    p[s][t] = poem[i]

    s2 = s + ds[k]
    t2 = t + dt[k]

    # 轉彎條件
    if t2 == n-1 or
            ( t2 < n-1 and s2 == n ) or
            ( t2 > n-1 and s2 == n-1 ) :
        k += 1
        if k == len(ds) : k = 0

    s , t = s2 , t2

# 列印
for i in range(len(p)) :
    print( "".join(p[i]) )
```