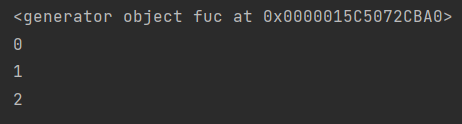
yield：生成器，類似涵式的 return，不同在於，使用next()可以繼續執行yield後的涵式內容，直到再遇到yield結束，next輸出的結果為前一個結果

ex： def fun(x)

while True:

結果：



yield(x)

x+=1

n=fun(0)

print(n)

print(next(n))

print(next(n))

(lambda 參數：計算式/表達式)(引數)s

ex: ans = (lambda n : n\*0.8)(100) 🡺 n = 80.0

**以下是lambda可搭配的三種函數：**

map(,)：可逐一帶入多個引數( 串列or疊代器iter() )值給lambda計算，輸出list，

第一個一定是放func，後面可放多個疊代器

ex： num=[100,90,80,70]

ans=map (lambda n : n\*0.8)(num) 🡺ans=[80,72,64,56]

reduce(,)：用於兩個參數的lambda，可將多個引數共同處理後回傳結果

ex： num=[100,90,80]

ans=reduce(lambda x,y: x\*y,num) 🡺 ans=720000 #他把num內全部參數都相乘了

filter(,)：用於篩選多筆資料

ex：num=[100,90,80,70,60,50,40,30]

ans=filter(lambda x: 60<x<90, num) 🡺 ans=[70,80]

**class !! note在class.py中**

class Drink:

def \_ \_init\_ \_(self,name,price):

Drink = 此class name

\_ \_init\_ \_=初始化屬性

name&price=屬性

get\_name & get\_prict

=提取nameor get\_price的方法

self.name=name

self.price=price

def get\_name(self):

return self.name

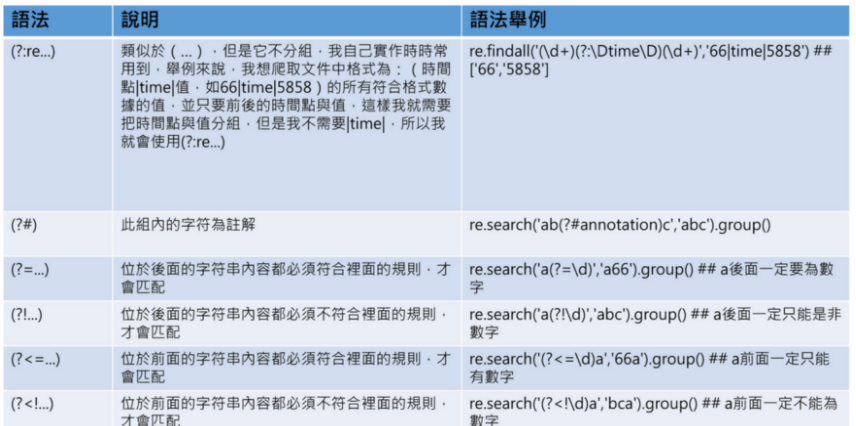
def get\_price(self):

return self.price

**正規表示式(寫的不是很好)**

meta字元規則：import re #模組 (皆需雙引號) (還有更多 上網找)

|  |  |  |  |
| --- | --- | --- | --- |
| ^str | 字串開頭為str | A\* | A重複0次以上 |
| str$ | 字串結尾為str | A+ | A重複1次以上 |
| . | 任意一個字元 | A? | 重複1次或0次 (沒以上) |
| ( ) | 群組 | A{n} | 重複n次 |
| AO|OB | AO或者OB，滿足其一即可 | A{n,} | 重複n次以上 |
| \s | 空格=[ \t\n\r\f] | A{a,b} | 重複a~b次 |
| \S | 非空格 | A\*? | 重複0次以上最短可能答案 |
| \d | 數字[0-9] | A+? | 重複1次以上最短可能答案 |
| \D{3} | 3個非數字 | [012] | 包含0或1或2其一字元 |
| \w | 數字或英文字 | [^012] | 不包含0或1或2其一字元 |
| \W | 非數字或英文字 | [A-Za-z] | 包含英文字母任意字元 |
| \A | 字串開頭 | [0-9] | 包含0到9任意字元 |
| \Z | 字串結尾，匹配字串的結尾。如果有任何換行符，它將在換行符之前匹配 | | |
| str\_a(?=str\_b) | 取str\_a，如果他後面有str\_b | (?<=str\_a)str\_b | 取str\_b，如果他前面有str\_a |
| str\_a(?!str\_b) | 取str\_b，如果他後面沒有str\_b | (?<!str\_a)str\_b | 取str\_b，如果他前面沒有str\_a |
| [\_] | 匹配含有底線的 | @ | 找到匹配 @ 的 |



特殊情況：當需要匹配一些在正則表達式中有特殊意義的字符時，像是\d、\w、\s等，此時就要多加一個””

搜尋方法：

正則模式變數=re.complie(“正則模式”)

正則模式變數.search(string) 🡪搜尋整個字串，沒找到回傳none，如果成功會傳回匹配對象

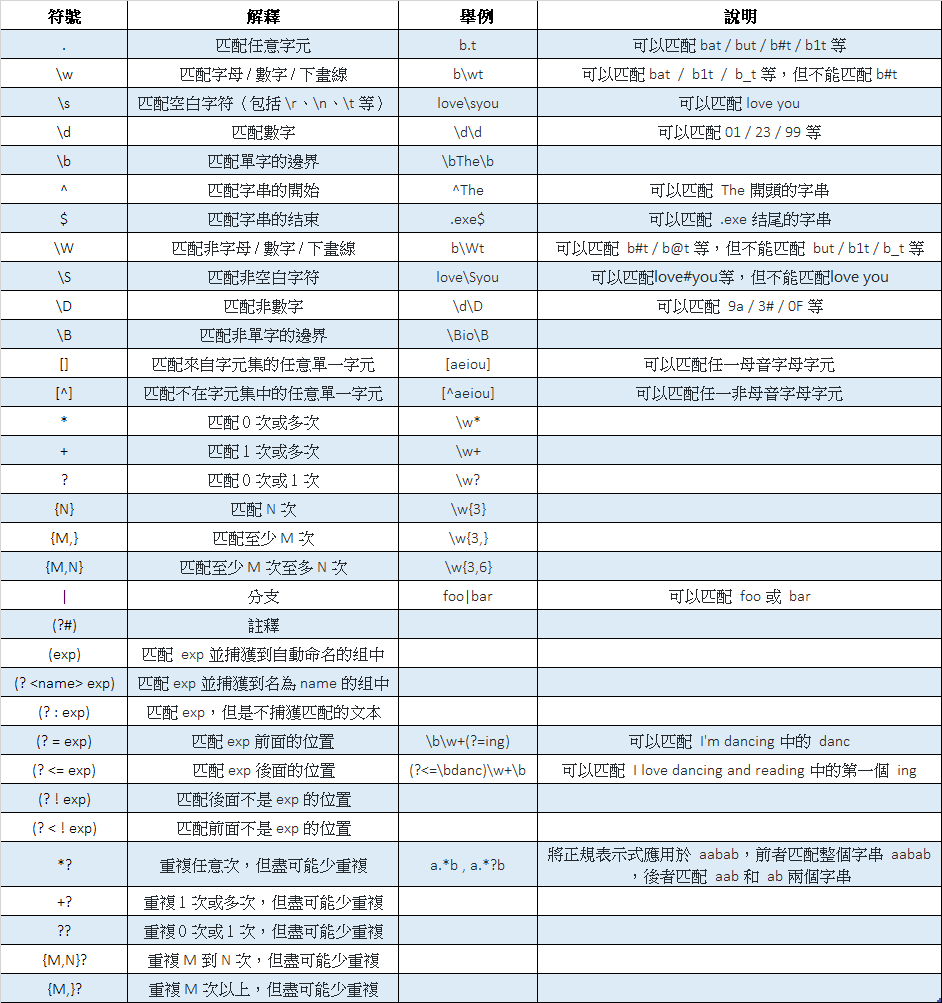
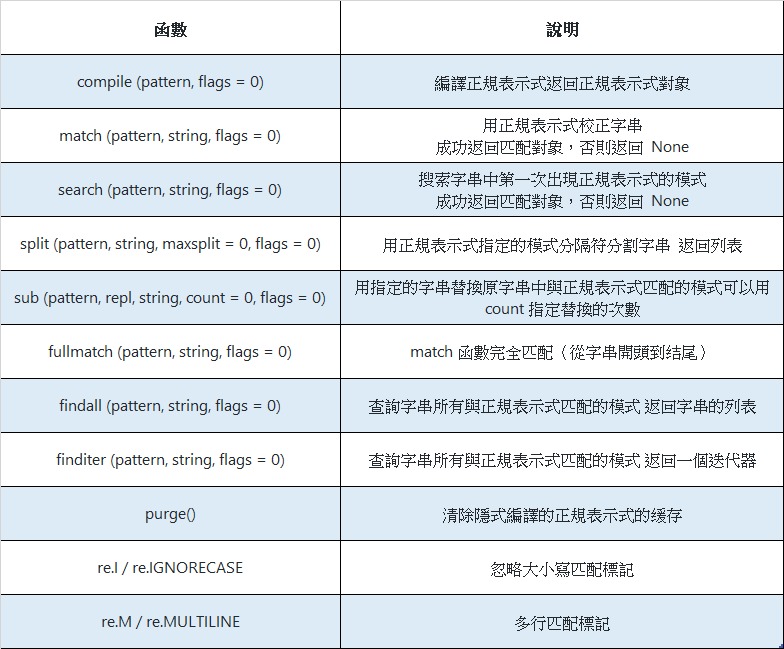
re.match(re rule,string,flags) 🡪只會找string開投是否規則符合的

.group 🡪把正則找到的所有結果結合成一tuple

re.findall(re rule,string,pos,endpos) 🡪找尋所有匹配字符，裝進list傳回，若無找到回傳空list

re.finditer() 🡪功能類似findall，但傳回的格是為迭代器

re.sub() 🡪匹配字符後，替換成想要的字符

re.compile(re.rule) 🡪定義正則歸責於一變數(pattern)  

import時就會執行一次，可用 if \_\_name\_\_ == \_\_main\_\_ : 判斷是否為主要執行的main，

不是主要的話則 \_\_name\_\_ 的值為該執行的.py檔名

Multiprocessing & Multithreading：都是多執行序，

差異在：Multithreading沒有 terminale，Multithreading之間可以共享記憶體。

import os

import mulitiprocessinf as mp

os.getpid() #取得當前執行的PID

a=mp.Process(target=func,args=(放入func的參數))

#可指定用多少PID執行 以下為3(process=3)

from mulitiprocessinf import Pool

with Pool(process=3) as process\_1:

result = process\_1.map(func,迭戴器)

a.start() 開始執行a程序

a.join() 等待a程序執行結束

共用記憶體部分需要有鎖的保護機制

lock.acquire()

code…

lock.release()