Python
串列 List 操作

臺北科技大學資訊工程系

List 操作 - append()

```
def myList():
    list = ['a', 'b', 'c']
    for x in list:
        print(x)

    my_list = []
    for i in range(0,10):   # for(i=0; i<10; i++)
        my_list.append( i+1 )

    if my_list[0] == 1 and len(my_list) < 10:
        my_list[0] += 1
        print ('1 state')
    elif (10 in my_list) or not(len(my_list)==10):
        print ('2 state')
        print ('range(i,j) is i~j-1')
    else:
        print ('3 state')
        print ('none of above')

    for i in my_list:
        print (i, end=' ')
def main():
    myList()
```

```
a
b
c
2 state
range(i,j) is i~j-1
1 2 3 4 5 6 7 8 9 10
```

List 產生器 comprehension

- 結合迴圈與條件測試式，減少繁瑣語法。

```
def listTest():
    inputData = input("Enter a list of number: ").split(" ")
    number_list = []
    for number in inputData:
        number_list.append(int(number))
    print(number_list)
```

```
# 1 2 3 4 5
# ['1', '2', '3', '4', '5']


# [1, 2, 3, 4, 5]
```

```
Enter a list of number: 1 2 3 4 5
[1, 2, 3, 4, 5]
```

[運算式 for 項目 in 可疊代項目]

```
inputData = input("Enter a list of number: ").split(" ")
number_list = [int(number)    for number in inputData]
print(number_list)
number_list = [int(number)*2 for number in inputData]
print(number_list)
```

```
Enter a list of number: 1 2 3 4 5
[1, 2, 3, 4, 5]
[2, 4, 6, 8, 10]
```

List 產生器

- 運算式 for 項目 in 可疊代項 if 條件式。

```
number_list = [number for number in range(1,6) if number % 2 == 1]
print(number_list)
```

```
[1, 3, 5]
```

List 產生器

- 運算式 for 項目 in 可疊代項 if 條件式。

```
def listTest():
    inputData = input("Enter a list of number: ").split(" ")
    print(inputData)
    number_list = [int(number) for number in inputData if int(number)%2==1]
    print(number_list)

listTest()
```

```
Enter a list of number: 11 22 33 44 55 66
['11', '22', '33', '44', '55', '66']
[11, 33, 55]
```

List 產生器

- 運算式 for 項目 in 可疊代項 for 迴圈。

```
for row in range(1,4):
    for col in range(1,3):
        print([row, col])
```

```
[1, 1]
[1, 2]
[2, 1]
[2, 2]
[3, 1]
[3, 2]
```



```
data = [[row, col] for row in range(1,4) for col in range(1,3)]
print(data)
```

```
[[1, 1], [1, 2], [2, 1], [2, 2], [3, 1], [3, 2]]
```

Exercise 1

- a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100].
- 輸出所有偶數元數的 list

```
a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
b = [number for number in a if number % 2 == 0]

print(b)
```

```
[4, 16, 36, 64, 100]
```

Exercise 1

- a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10].
- 寫 isPrime(x)，用 list generator 產生某串列 a 中是質數者

```
import math

def isPrime(n):
    if n <= 1:
        return False
    for i in range(2, int(math.sqrt(n)) + 1):
        if n % i == 0:
            return False
    return True

a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
b = [number for number in a if isPrime(number)]
print(b)
```

```
[2, 3, 5, 7]
```

Exercise

- 使用 random.randint() 產生 1~30 整數亂數10個，放進 list，
- 將最前面和最後面放進新 list 印出。

```
def list_ends(a_list):
    return [a_list[0], a_list[len(a_list) - 1]]

# generate a list using random integers
import random
a = []
for i in range(1,10):
    a.append(random.randint(1, 30)) # 產生 1~30 整數亂數10個
print(a)

print(list_ends(a))
```

```
[23, 24, 22, 21, 17, 16, 5, 24, 14]
[23, 14]
```

Exercise

- 輸入 list，回傳所有不重複的元素 list

```
# this one uses a for loop
def dedupe_v1(x):
  y = []
  for i in x:
    if i not in y:
      y.append(i)
  return y
```

```
a = [1,2,3,4,3,2,1]
print(dedupe_v1(a))
```

```
[1, 2, 3, 4]
```

```
#this one uses sets
def dedupe_v2(x):
    return list( set(x) )
```

{1, 2, 3, 4}

```
a = [1,2,3,4,3,2,1]
print(dedupe_v2(a))
```

```
[1, 2, 3, 4]
```

重複數據刪除（英語：data deduplication）

Exercise

- 輸入兩個list
  - a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
  - b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
  - 輸出兩個都有的元素【每個元素不重複/交集】
  - 以下程式可能有錯(例如 b 有重複的元素)，如何修正?

```
def ex01():
  a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
  b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
  c = []
  for i in b:
    if i in a:
        c.append(i)
  print(c)
ex01()
```

```
[1, 2, 3, 5, 8, 13]
```

修正

```
if (i in a) and (i not in c):
```

Exercise

- 以下程式輸出?
  - 聯集
  - 交集

```
import random

def ex02(a, b):
  print(a)
  print(b)
  result_overlap = a + b
  result = []
  for element in result_overlap:
    if element not in result:
      result.append(element) # 聯集
  print(result)

a = random.sample(range(1,30), 12)
b = random.sample(range(1,30), 16)
ex02(a,b)
result_overlap = [i for i in set(a) if i in b] # 交集
print(result_overlap)
```

```
[16, 25, 19, 17, 11, 26, 27, 21, 13, 2, 6, 24]
[12, 13, 7, 19, 18, 23, 3, 27, 21, 20, 24, 10, 1, 2, 8, 5]
[16, 25, 19, 17, 11, 26, 27, 21, 13, 2, 6, 24, 12, 7, 18, 23, 3, 20, 10, 1, 8, 5] # 聯集
[2, 13, 19, 21, 24, 27] # 交集
```

Exercise

- 輸入兩個 list，
- 回傳任一個有的元素一次【每個元素不重複/聯集】

```
def ex01():
  a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
  b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
  c = []
  for i in a:              #加入不重複的 a 元素
    if i not in c:
      c.append(i)
  for i in b:              #加入不重複的b元素
    if i not in c:
      c.append(i)
  print(c)
```

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]

c = set(a) & set(b)      # 使用 set & 交集
print(c)

c = set(a) | set(b)      # 使用 set | 聯集
print(c)
```

```
[1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 4, 6, 7, 9, 10, 11, 12]
```

```
{1, 2, 3, 5, 8, 13} # 交集
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 21, 89, 34, 55} # 聯集
```

Exercise

- 程式
  - 給定一串數字，將這一連串數字分為奇數在左，偶數在右，
  - 奇數按照由小到大排，
  - 偶數按照由大到小排。
  - 輸入說明:輸入整數 n 代表有 n 個數，接著輸入 n 行整數
  - 輸出說明:將奇數放左(小到大排)，偶數在放右(大到小排)

```
numList = [int(input()) for i in range(int(input()))]

right = sorted([x for x in numList if x%2 == 0], reverse = True)

left  = sorted([x for x in numList if x%2  != 0])

print(left + right)
```

輸入:
7
1
2
3
4
5
6
7
輸出:
[1, 3, 5, 7, 6, 4, 2]

Exercise

- 程式
  - 輸入一行 n 個整數，將這一連串數字分為
  - 被 3 整除，在右，
  - 被 3 除餘 1，在左，
  - 被 3 除餘 2，在中間，
  - 中間按照由小到大排，左右按照由大到小排。

```
data = input('輸入: ').split()
numList = [int(x) for x in data]

right  = sorted([x for x in numList if x%3  ==0], reverse=True)
left   = sorted([x for x in numList if x%3  ==1], reverse=True)
middle = sorted([x for x in numList if x%3  ==2])
print(left + middle + right)
```

```
輸入: 1 2 3 4 5 6 7 8 9 10
[10, 7, 4, 1, 2, 5, 8, 9, 6, 3]
```

Exercise - 資料分析

- COVID-18 病毒流行，請分析計算A城市流行狀況
  - A城市某一天開始從邊境移入 people 位確診者。
  - 每一位確診者會傳染給 infectPeople 個人。
  - 確診者經過 recoveryDay 天後康復，就不會再傳染給其他人。
  - 該城市人口 protectedRate
    - 在分析期間，A城市打過疫苗＋已確診康復者，假設為固定人數。
  - 每日新增確診
    - 打過疫苗與已確診康復者不會被傳染，其他人則會被傳染確診。
    - (infectPeople/recoveryDay)*(1-protectedRate)，去掉小數取整數
  - 請輸出 periodDay 期間日中，
    - 每日確診人數、
    - 每日新增確診人數、
    - 每日康復人數。

```
( 0,   100,  100,    0),
( 1,   105,    5,    0),
( 2,   110,    5,    0),
( 3,   115,    5,    0),
( 4,   120,    5,    0),
( 5,   126,    6,    0),
( 6,   132,    6,    0),
( 7,    38,    6,  100),
( 8,    34,    1,    5),
( 9,    30,    1,    5),
(10,    26,    1,    5),
(11,    22,    1,    5),
(12,    17,    1,    6),
(13,    11,    0,    6),
(14,     5,    0,    6),
(15,     4,    0,    1),
(16,     3,    0,    1),
(17,     2,    0,    1),
(18,     1,    0,    1),
(19,     0,    0,    1),
```

Exercise

```
def show(periodDay, recoveryDay, infHist, addHist):
    for d in range(periodDay):
        if d >= recoveryDay:
            s = "(%2d, %5d, %4d, %4d), " %(d, infHist[d], addHist[d],addHist[d-recoveryDay])
        else:
            s = "(%2d, %5d, %4d, %4d), " %(d, infHist[d], addHist[d],0)
        print(s, end='')
        # if (d+1)%5==0:
        print()

def main():
    people =100
    recoveryDay = 7
    periodDay = 20
    infectPeople = 1.2
    protectedRate = 0.7
    addHist, infHist=g(people, recoveryDay, periodDay, infectPeople, protectedRate)
    show(periodDay, recoveryDay, infHist, addHist)
    return 0
```

Exercise

```
def g(people:int, recoveryDay:int, periodDay:int, infectPeople:float, protectedRate:float):
    addHist = [0 for i in range(periodDay+1)]
    infHist = [0 for i in range(periodDay+1)]
    day =0;
    addHist[day] = people
    infHist[day] = people
    day = day + 1
    while people>0 and day <periodDay:
        addHist[day] = int(infHist[day-1]*(infectPeople/recoveryDay)*(1-protectedRate))
        infHist[day] = infHist[day-1] + addHist[day]
        if day >=recoveryDay:
            infHist[day] = infHist[day] - addHist[day-recoveryDay]
        day = day + 1
    return addHist, infHist

main()
```

```
( 0,   100,  100,    0),
( 1,   105,    5,    0),
( 2,   110,    5,    0),
( 3,   115,    5,    0),
( 4,   120,    5,    0),
( 5,   126,    6,    0),
( 6,   132,    6,    0),
( 7,    38,    6,  100),
( 8,    34,    1,    5),
( 9,    30,    1,    5),
(10,    26,    1,    5),
(11,    22,    1,    5),
(12,    17,    1,    6),
(13,    11,    0,    6),
(14,     5,    0,    6),
(15,     4,    0,    1),
(16,     3,    0,    1),
(17,     2,    0,    1),
(18,     1,    0,    1),
(19,     0,    0,    1),
```

Exercise – 括號檢查

- 輸入一串字元，檢查括號是否成對出現
  - {({()()})} – True
  - {a({(f)(g)}h)} – True
  - {a[{(f+c)(g+s)}*h+d]} – True
  - ([x+y]*{(a+b)*((a+c)*d)}) – True
  - {({)(})} – False
  - ([x+y]*{(a+b)*[(a+c]*d)}) – False
  - ())((x+y*(a+b)*((a+c)*d)) – False
  - {(h+{a+b)(x*y}*z)} – False
- 解題
  - 遇到左括號 {, [, (，push 進入 stack
  - 遇到右括號 },],)，從 stack 中 pop 同類型左括號
  - 最後 stack 空 - True

push {
push (
push {
push (
pop (
push (
pop (
pop {
pop (
pop {
True

push {
push (
push {
False

Exercise – 括號檢查

- 輸入一串字元，檢查括號是否成對出現

```
def push(stack, data, top):
   stack[top+1] = data;
   return (top+1)

def pop(stack, top):
   return top - 1;

def main():
    print(M('{({()()})}'))
    print(M('{a({(f)(g)}h)}'))
    print(M('{a[{(f+c)(g+s)}*h+d]}'))
    print(M('([x+y]*{(a+b)*((a+c)*d)})'))
    print(M('{(){()}}'))
    print(M('([x+y]*{(a+b)*[(a+c]*d)})'))
    print(M('()][((x+y*(a+b])*[(a+c)*d))'))
    print(M('{(h+{a+b)(x*y}*z)}'))

main()
```

```
True
True
True
True
False
False
False
False
```

Exercise – 括號檢查

- 輸入一串字元，檢查括號是否成對出現

```
def isMatch(x, y, left, right):
    if left.index(x) == right.index(y): return True
    return False

def M(s):
    stack = ['' for i in range(20)]
    left, right  = ['{','[','('], ['}',']',')']
    top = -1;
    for i in range(len(s)):
        if s[i] in left:
            top = push(stack, s[i], top)
        elif s[i] in right:
            if (top == -1): return False
            #print('%d, %c, %c\n' %(top, stack[top], s[i]));
            if isMatch(stack[top], s[i], left, right)==False: return False
            top = pop(stack, top)
    if (top == -1): return True
    else: return False
```

Built-in methods for List

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list
list()	It is also possible to use the list() constructor to make a new list

END