爬蟲基礎入門-Part 2-1

圖解功能介紹

處理型態

元素

字符串: ‘ 不論長怎樣 ’ or “ 不論長怎樣 ”

整數: 10,22,34….

浮點數:3.1415962…

籃子型態

List ： [元素, 元素, 元素 , 元素 , 元素]

元素級別　　元素/籃子

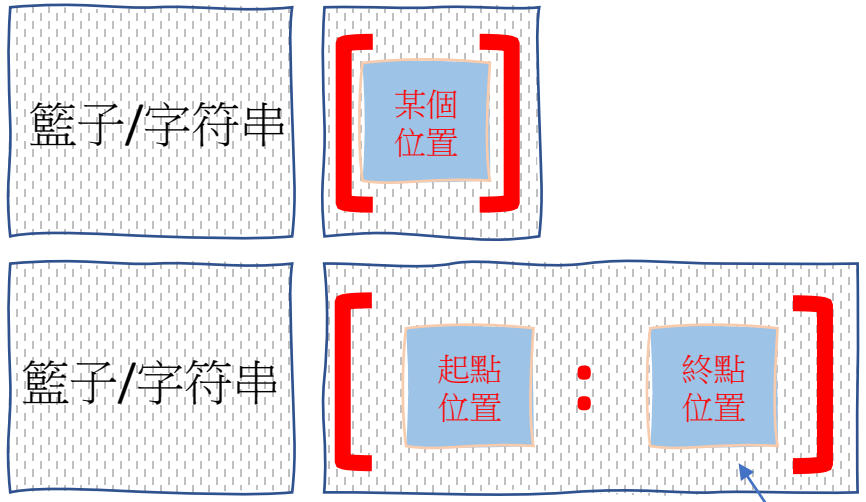
dict ： { key ： value , ： , ： }

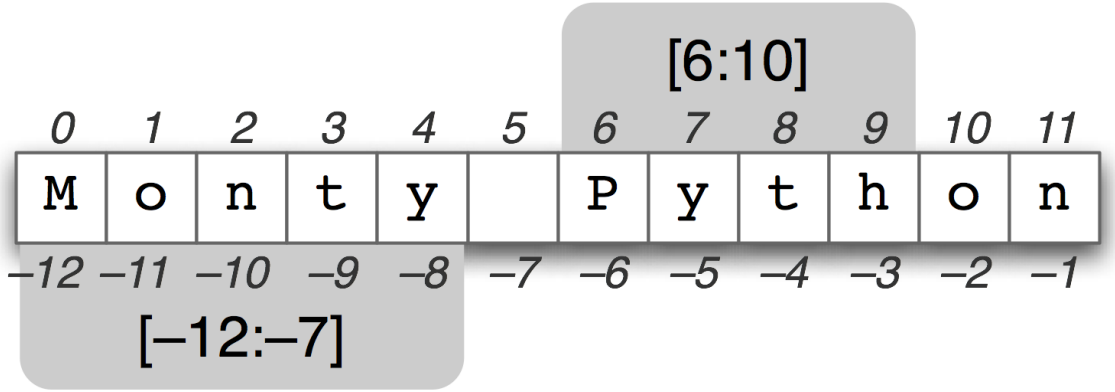
item　　item　　item

set ： { 元素 , 元素 , 元素 , 元素 }

資料總攬,不重複

位置



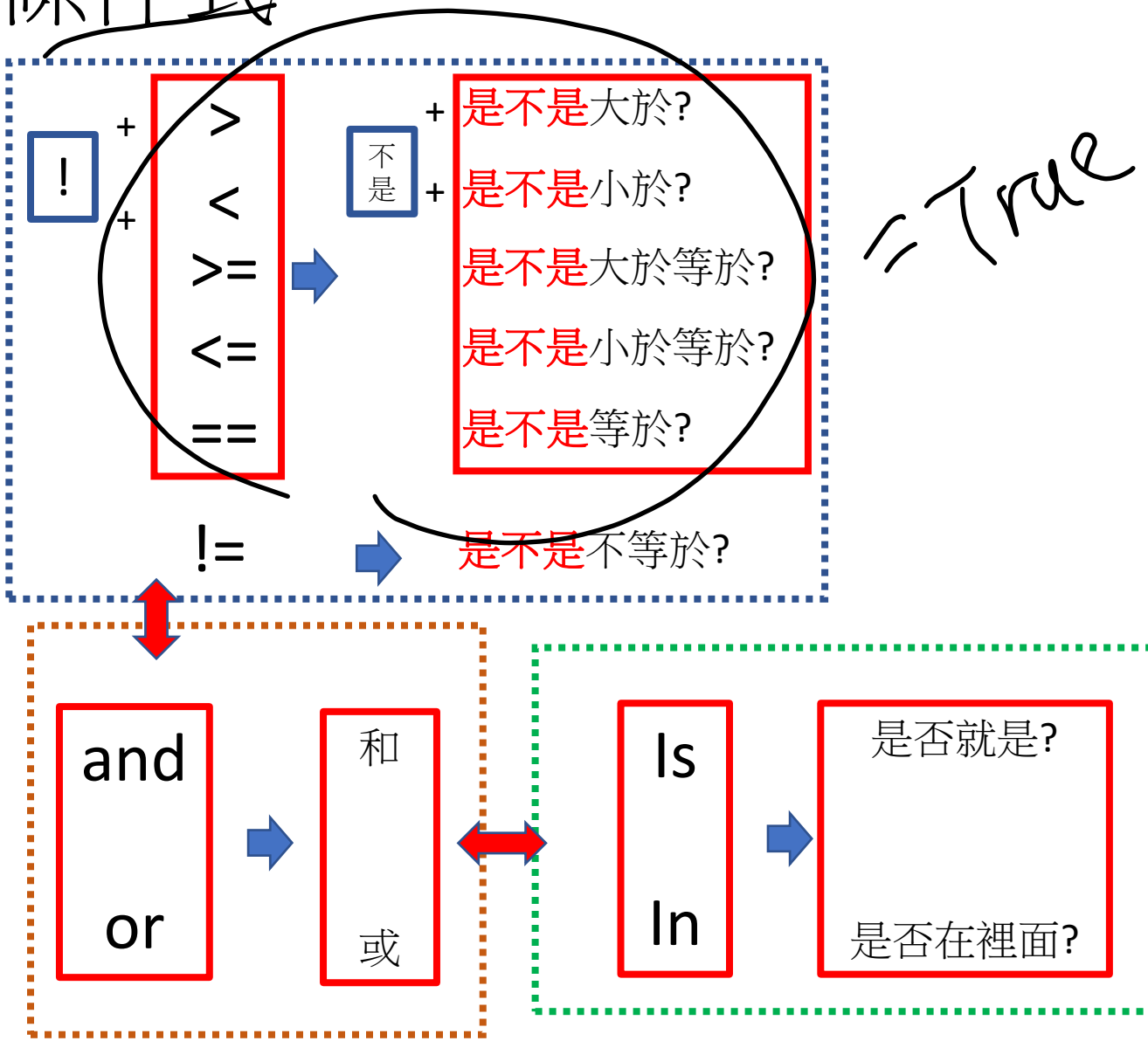


舉例:

```python
#位置
data_list = [10,20,30,40,50,60,70]
list_position = 2

print('第幾個位置:',list_position)
print('當個位置的值',data_list[list_position])
print('==================================================')
data_str1 = 'this is my data column'
data_str2 = "have other data column is '10'"
str_position = -10
print('str 選定的位置:', str_position)
print('字符串1當個位置的值:',data_str1[str_position])
print('字符串2當個位置的值:',data_str2[str_position])
print('==================================================')
#起點:終點(不含)
str_p = 0
end_p = 3
my_filter_data = data_list[str_p:end_p]
print('我設定的起點/終點:',str_p,end_p)
print('我自己設定的區間:',my_filter_data)
print('==================================================')
#字符串裡面有變數
summary = f'我的數據{data_list},\n它的型態是{type(data_list)}'
# summary = '我的數據{},\n它的型態是{}'.format(data_list,type(data_list))
print(summary)
```

```
第幾個位置: 2
當個位置的值 30
==================================================
str 選定的位置: -10
字符串1當個位置的值: a
字符串2當個位置的值: m
==================================================
我設定的起點/終點: 0 3
我自己設定的區間: [10, 20, 30]
==================================================
我的數據[10, 20, 30, 40, 50, 60, 70],
它的型態是<class 'list'>
```

條件式



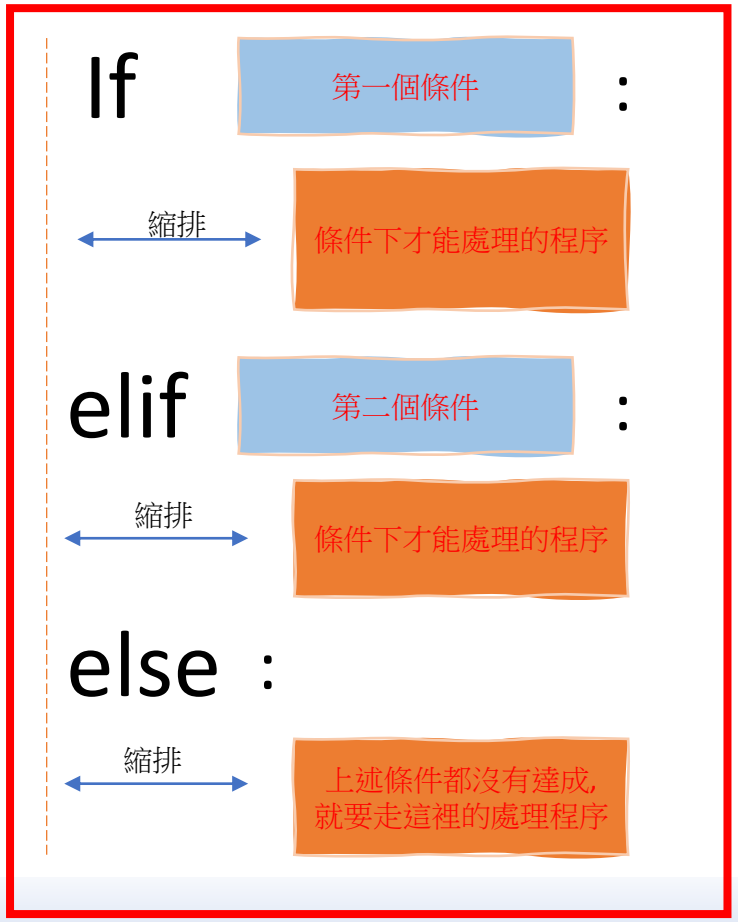
舉例:

```python
#條件式-比大小
data_number1 = 100
data_number2 = 80
print('數據1是否大於數據2',data_number1>data_number2)
print('數據1是否小於數據2',data_number1<data_number2)
print('數據1是否等於數據2',data_number1==data_number2)
print('數據1是否大於數據2',data_number1!=data_number2)
#條件式- in is
data_list1 = [1,2,3,4,5]
check_num = 1
print( f'確認是否{check_num}在{data_list1}裡面?', check_num in data_list1 )
str1 = 'hello'
str2 = 'hello!!!'
print(f'確認字符串1{str1}是否就是字符串2{str2}?', str1 is str2)

#條件式  bool 取交集與聯集 TT=T TF = F  FF=F
print(False and False)
print(True and False)
print(True and True)

print(False or False)
print(True or False)
print(True or True)

print(data_number1 == data_number2 and check_num <data_list1[-1])
```

```
數據1是否大於數據2 True
數據1是否小於數據2 False
數據1是否等於數據2 False
數據1是否大於數據2 True
確認是否1在[1, 2, 3, 4, 5]裡面? True
確認字符串1hello是否就是字符串2hello!!!? False
False
False
True
False
True
True
False
```

判別式

If ------>如果
Elif----->再不然
Else---->沒有如果,一定要做



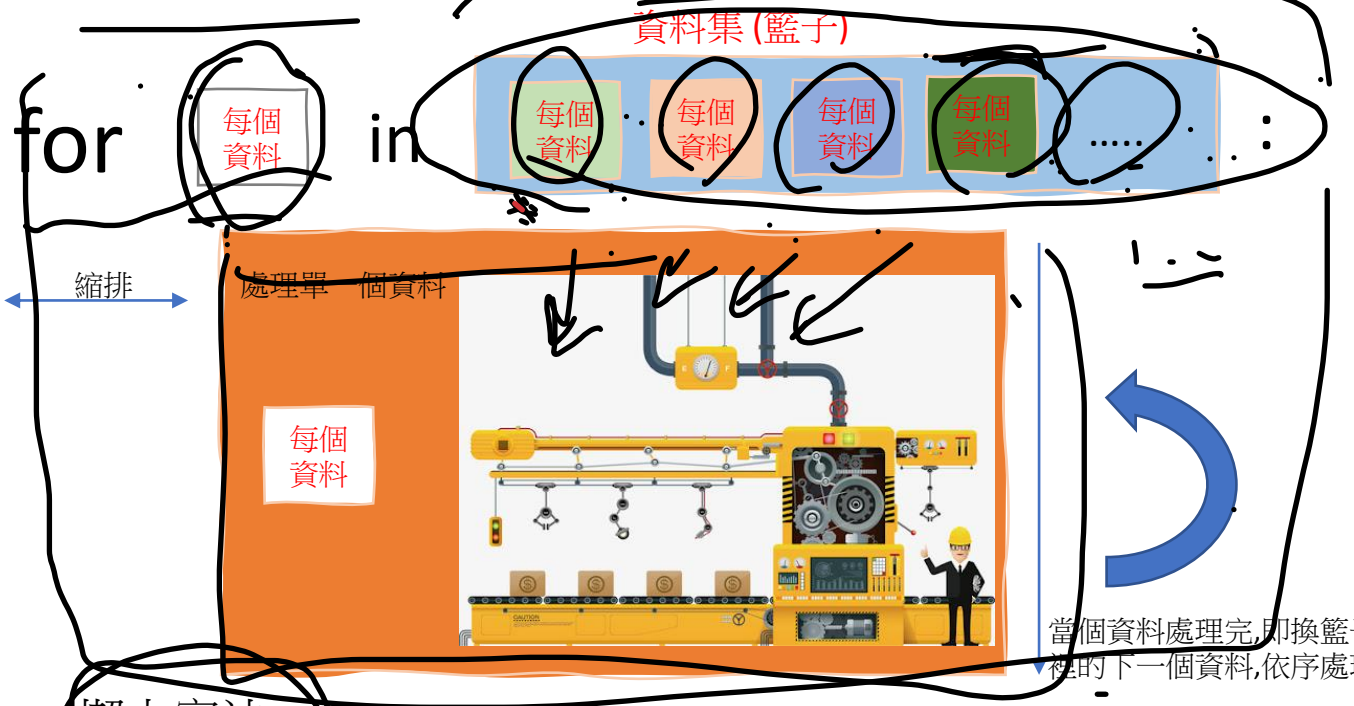
舉例:

```
1  #判別式 if elif else
2  data_number =10
3  Threshold_value = 5
4  if data_number >= Threshold_value:
5      print(f'第一層判斷,{data_number} 比 {Threshold_value}大')
6  else:
7      print(f'第一層判斷,{data_number} 比 {Threshold_value}小')
8
9  print('==========================================================')
10 data_number =7
11 Threshold_value1 =5
12 Threshold_value2 = 15
13 if data_number>Threshold_value1 and data_number<Threshold_value2:
14     print(f'你選的數字在於{Threshold_value1}到{Threshold_value2}之間')
15 elif data_number<Threshold_value1 or data_number>Threshold_value2:
16     print(f'你的數字要嘛比{Threshold_value1}小,要嘛比{Threshold_value2}大')
17 else:
18     print('你的數字剛好等於筏值')
19
```

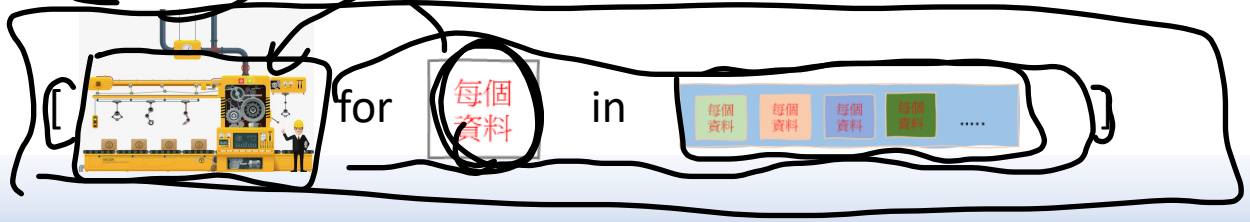
```
第一層判斷,10 比 5大
==========================================================
你選的數字在於5到15之間
```

有限迴圈（函數）

當*每一個資料*在*資料集裡*,每一個資料要依序*處理固定程序*



懶人寫法:

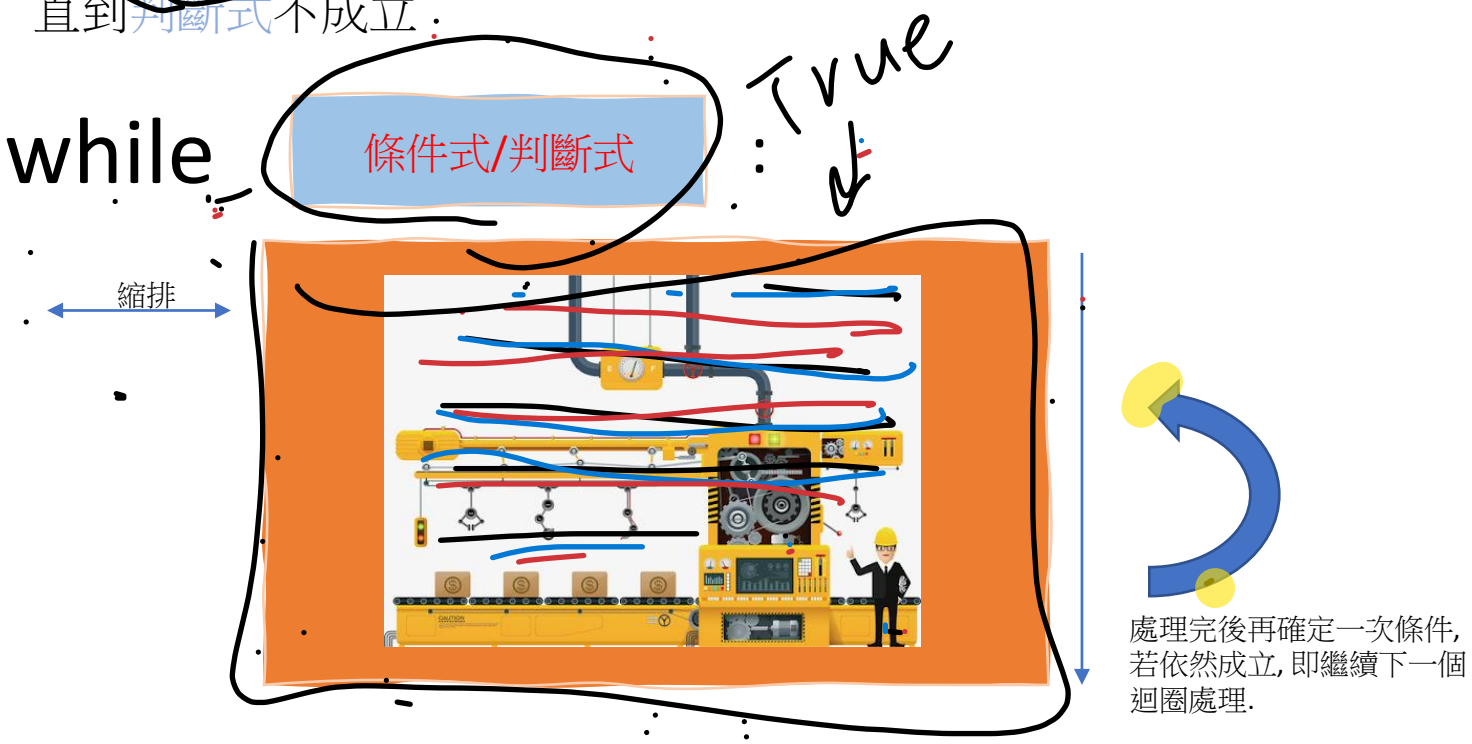


舉例:

```python
#有限迴圈
data_list = [10,22,34,55,66,80,10]
data_treated = []
for i in data_list:
    print('正在處理的數據:',i)
    y =i+1
    print('處理後的數據:',y)
    data_treated.append(y)
    print('==========每次循環結束==========')
print('正常循環處理後:',data_treated)
print('=========================我是分隔線=====================================')
#懶人寫法, 上述也可以表示成:
data_treated2 = [i+1 for i in data_list]
print('用懶人方法寫的處理:',data_treated2)


print('=========================我是分隔線=====================================')
data_sum = 0
processing = 1
for i in data_list:
    print('正在處理的數據:',i)
    data_sum = data_sum + i
    print('當下循環的總和:',data_sum)
    print(f'==========第{processing}次循環結束==========')
    processing = processing+1



#學習重點:
#1.學習到 for in 有限循環, 每次都是只處理籃子裡面的每一項元素
#2.學習到 list 這個tpye 有一個功能叫做append, 可以在每次循環添加一些我處理完的資料
#3.學習到 計數器的概念, 可以對數據做累加的寫法, 以及可以把處理完的值量代給自己本身,供下一次循環使用
#4. 請懂懶人寫法
# 自我練習:
#嘗試 data_list 改為 range(起點, 終點, 間隔)

```

無限迴圈

當條件成立時,直接處理固定程序,處理完後,重新繼續判斷,繼續迴圈,直到判斷式不成立.



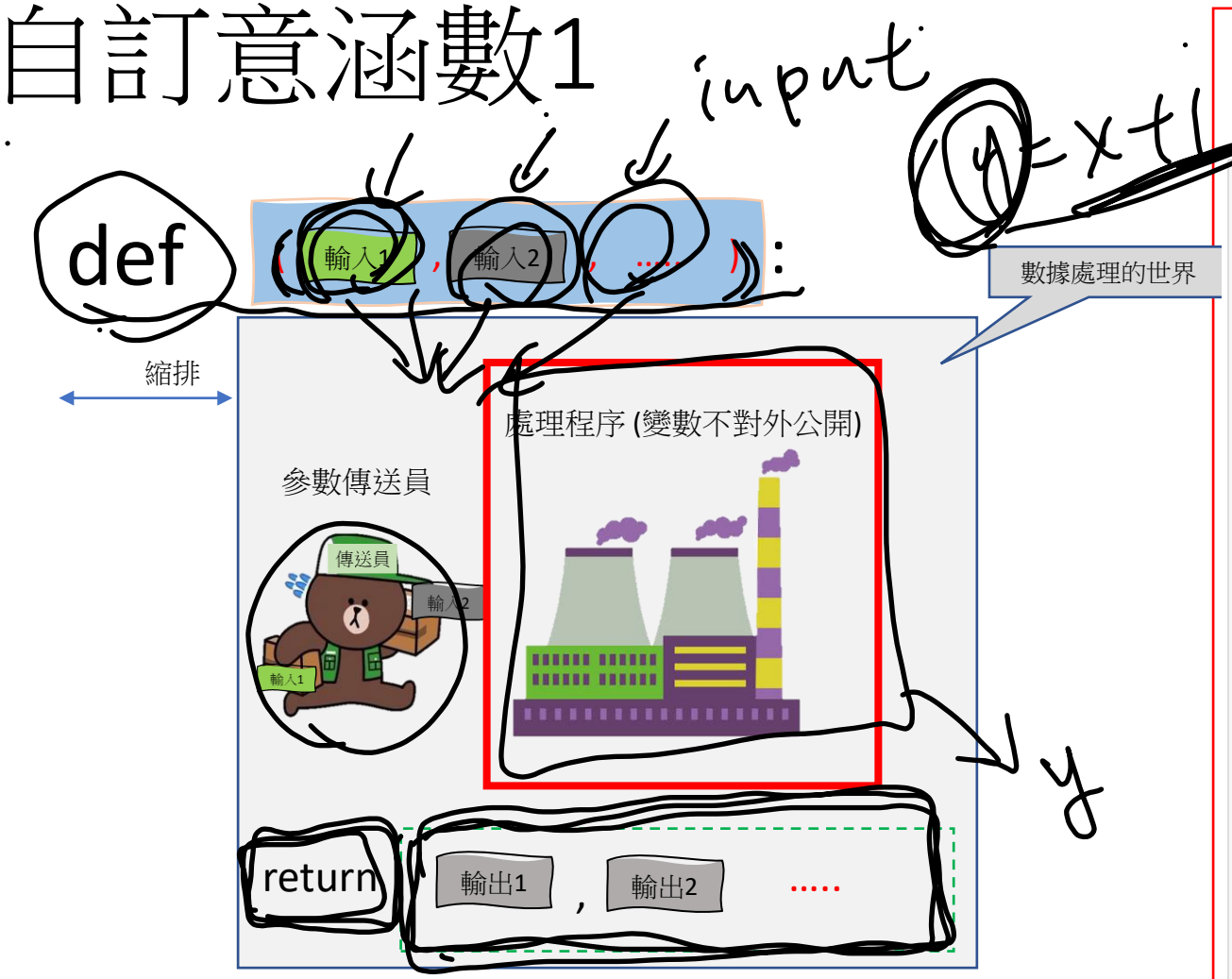
處理完後再確定一次條件,若依然成立,即繼續下一個迴圈處理.

舉例:

```
1  #while 迴圈  無限迴圈 (條件迴圈)
2  my_data = 20
3  while True:
4      print('準備處理:',my_data)
5      my_data = my_data+1
6      print('處理後:',my_data)
7
8  print('=========================我是分隔線============================')
9  my_data = 20
10 processing = True
11 while processing:
12     print('準備處理:',my_data)
13     my_data = my_data+1
14     print('處理後:',my_data)
15
16
17     if my_data>25:
18         processing = False
19         print('=======跳出循環=======')
20     else:
21         print('還不用跳出循環')
22
23 #學習重點:
24 # 1. 學習到了while 跟for in 迴圈很像, 只是針對條件是否成立,作為迴圈依據
25 # 2. 學習到了 可以在迴圈中埋入switch 中斷的開關, 並結合判斷式做數據處理,防止無限迴圈
26 # 3. while 通常判斷式 ,不寫即默認是True
```

```
=========================我是分隔線============================
準備處理: 20
處理後: 21
還不用跳出循環
準備處理: 21
處理後: 22
還不用跳出循環
準備處理: 22
處理後: 23
還不用跳出循環
準備處理: 23
處理後: 24
還不用跳出循環
準備處理: 24
處理後: 25
還不用跳出循環
準備處理: 25
處理後: 26
=======跳出循環=======
```

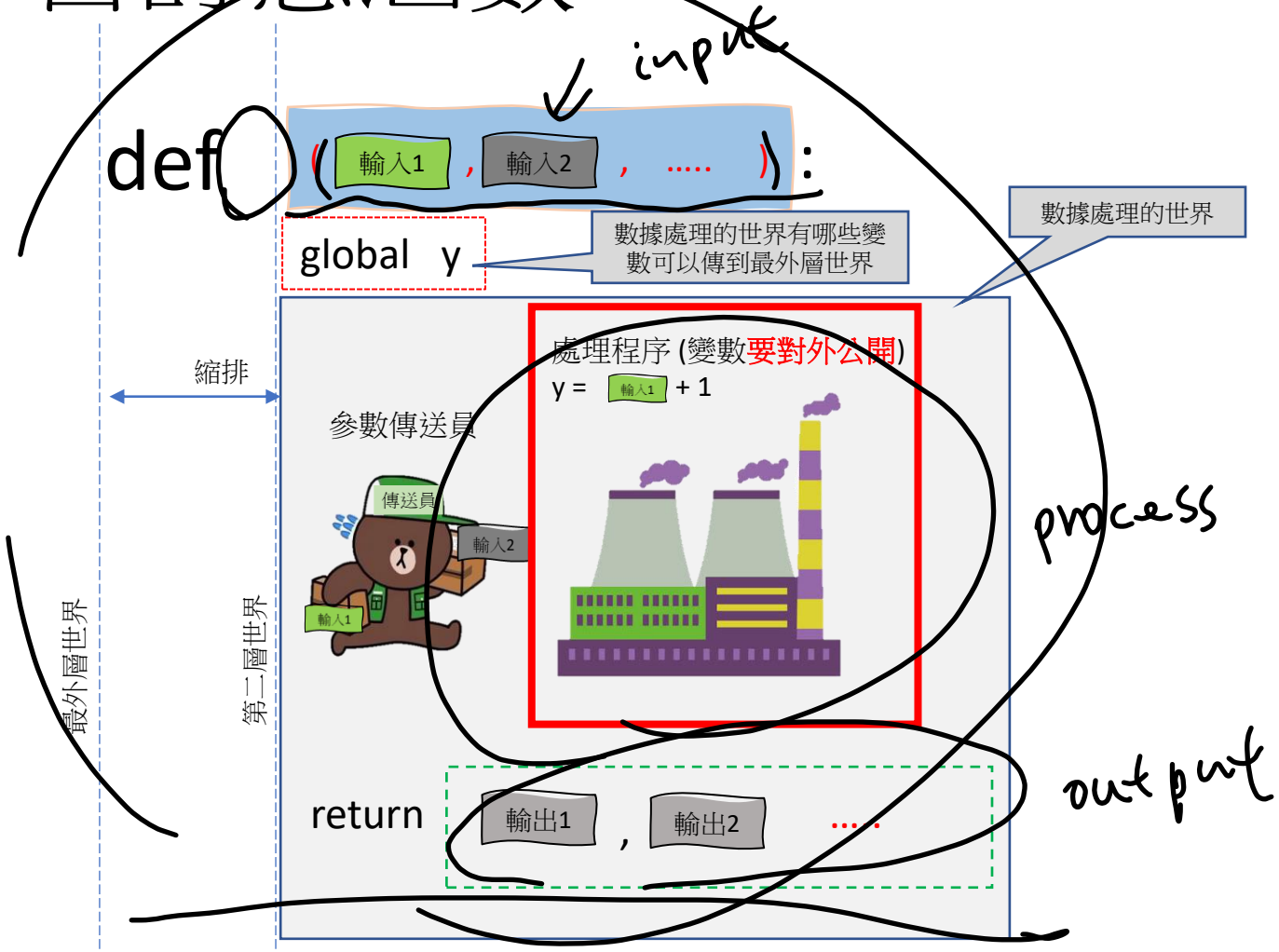
自訂意涵數1



舉例:

```python
#自訂義函數
def my_add_fun(num1,num2):
    result = num1+num1
    return result
#調用
my_result = my_add_fun(1,3)
print('調用我自訂義的加法函數結果:',my_result)
print('=========================我是分隔線==========================')
#(把比大小賽的判斷寫成def)
def my_judge( data_1 , threshold ):     #偷偷註記: data_1 , threshold 都是可被計算,被比較的數字
    if data_1>threshold:
        print(f'您的數字{data_1}大於{threshold}')
    elif data_1<threshold:
        print(f'您的數字{data_1}小於{threshold}')
    else:
        print(f'您的數字等於{threshold}')

#調用
print(my_judge(100,20))

print('=========================我是分隔線==========================')
#(加總算平均)
def my_cal_mean(data):
    sum_acc = 0
    for i in data:
        sum_acc = sum_acc + i

    mean = sum_acc/len(data)

    return mean
#調用
data_list = [1,2,3,4,5,6,7,8,9,10]
result = my_cal_mean(data_list)
print(result)
print('=========================我是分隔線==========================')
# 學習重點
#學會自訂一函數, 函數一開始的輸入type 要設定好, 方便以當下的type做該功能的調用.
#學習自訂一函數,最後return 是一個回傳的動作, 代表之後調用,會代後的值就是這個return的值
#學習到有些定義函數,著重在處理數據而已, 不一定要return 回傳值出去

```

自訂義函數2

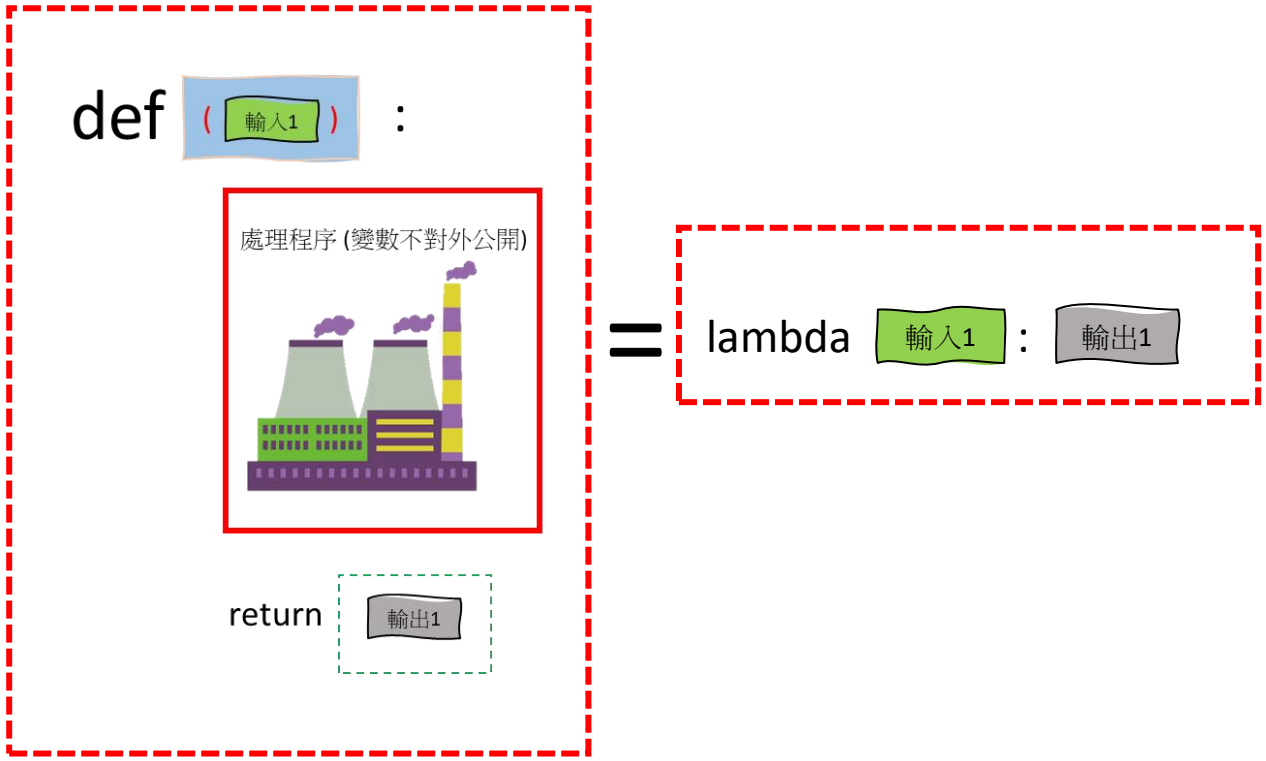


舉例:

```python
#自訂義函數  將函數處理過程中的變數,也可以傳到最外面,當作全域變數
def my_cal_mean(data):
    global test
    sum_acc = 0
    for i in data:
        sum_acc = sum_acc + i
    test = 1000
    mean = sum_acc/len(data)

    return mean
result = my_cal_mean([1,2,3,4,5])
print(result)
#嘗試把第三行註解掉,跑跑看是否能得到內部參數
print(test)
```

```
3.0
1000
```

自訂意涵數3



舉例:

```python
#自訂義函數懶人寫法

#原本正規寫法:
def my_add_fun(num1,num2):
    result = num1+num2
    return result
add_result = my_add_fun(1,2)
print('正規寫法的結果:',add_result)


def my_multiply_fun(num1,num2):
    result = num1*num2
    return result
multiply_result = my_multiply_fun(3,2)
print('正規寫法的結果:',multiply_result)
print('==================================================')
#懶人寫法 ex1:
result = (lambda num1,num2 :num1+num2)(1,2)
print('懶人寫法的結果:',result)
#懶人寫法 ex2:
multiply = (lambda x1,x2:x1*x2)(3,2)
print('懶人寫法的結果:',multiply)

```

```
正規寫法的結果: 3
正規寫法的結果: 6
==================================================
懶人寫法的結果: 3
懶人寫法的結果: 6
```

什麼是套件包?

你用conda創的環境

你的電腦環境

My_new_env

pip install 別人寫好的功能包 == 1.3.0

(不加就默認最新版)

Ex: pip install jupyter

pip install pandas

pip install 別人寫好的功能包

有些功能會基於別人寫的功能做疊加, 因此版本匹配很重要

My_new_env2

pip install 別人寫好的功能包 == 2.1.0

(不加就默認最新版)

重點1:
我們在安裝這些套件,通常會用pip install 或者conda install 做安裝

重點2: 開發時,通常會針對專案需求,創立一個虛擬環境 ,做版本管控

重點3:我們在開發功能時,不免俗會直接拿別人已經寫好的功能包做功能疊加的應用 (指令集參照官網)

Coding time