



# Week03 - 2023-1008

---

## 上週回顧

-  `float()` 取位數補充
-  練習 "d065"

## `float()` 取位數補充

### 透過型別轉換

另外我們也可以透過先前學習到的型別轉換先轉換成 `str`，再透過對 `str` 的處理轉換成 `float` 因而得到我們想要的數值。

```
probability = 0.123456789
probability_str = str(probability)
probability_str = probability_str[:4]
probability = float(probability_str)
print(probability)      # 0.12
```

### 透過運算的方式截斷

將一個浮點數 `probability` 乘以 100，然後再將結果轉換為整數。接著，它會再將該整數轉換回浮點數，並保留兩位小數。這樣的處理方式通常稱為「四捨五入到兩位小數」。

這樣的處理方式可能有助於限制浮點數的小數部分，以確保它只保留兩位小數，而不進行四捨五入

```
probability = 0.123456789
print(int(probability * 100) / 100)
```

## Zerojudge "d065"

變數, 一行, map, input, split, int, max, if

文文和兩個同學最近喜歡在 ZeroJudge 上解題。有一天他們看到了孔子說的：「三人行必有我師焉。」就吵了起來，因為他們每個人都認為自己是三個人之中的「老師」。後來他們決定要比比看誰在 ZeroJudge 上的 AC 題數最多。

Sample	Input	Output
說明	輸入只有一行，含有三個由空白所隔開的非負整數。	輸出這三個整數中最大的那一個。
# 1	35 26 48	48
# 2	37 59 59	59

### 解法一：搭配先前課程解法

透過變數先將使用者的輸入保存，並且以及邏輯判斷以找出最大值。

- 使用者的概念：我們是撰寫程式的角色，而使用我們程式的角色就是使用者，我們為了滿足使用者的使用，因此要設計能夠運行的程式，因此以此概念便有了 `input()` 的搭配使用，以此達到與使用者互動。

### [Python 解]

```
a, b, c = map(int, input().split())

max_value = a

if b > max_value:
    max_value = b
if c > max_value:
    max_value = c

print(max_value)
```

解法二：將程式行數縮短 因為 Python 本身有內建取最大值的方法

```
a, b, c = map(int, input().split())
print(max(a, b, c))
```

---

## string 字串操作

在 Python 中如果我們想要輸入 `string` 我們會透過 `'''` 一個引號的方式，其中在 Python 中 `"` 跟 `'` 的效力是相同的

### [!IMPORTANT]

左右兩邊的引號要一模一樣

1. 合格：`'Hello World'`
2. 合格：`"Hello World"`
3. 不合格：`'Hello World"`
4. 不合格：`"Hello World'`

## 字串的索引

字串的索引是從 0 開始，也就是說第一個字的索引是 0，第二個字的索引是 1，以此類推。

```
word = "Hello Python!"

print(word)
```

## 字串其實就是 **list**

```
word = "Hello Python!"

print(word[0])      # H
print(word[1])      # e

for i in word:
    print(i)
```

## 字串的長度

```
word = "Hello Python!"
print(len(word))
```

## 字串檢查

我們如果想檢查一個單詞又或是字母是否在字串之中，我們會有以下的方法

```
txt = "The best things in life are free!"
print("free" in txt)
```

```
txt = "The best things in life are free!"
if "free" in txt:
    print("Yes, 'free' is present.")
```

## 字串的切片 (在 **list** 細講)

```
word = "Hello Python!"
# 取得 word 中第 2~5 個字元
print(word[1:5])      # ello
# 取得 word 中第 2~最後一個字元
print(word[1:])        # ello Python!
# 取得 word 中第 1~5 個字元
print(word[:5])        # Hello
# 取得 word 中第 1~最後一個字元
print(word[:])         # Hello Python!
# 取得 word 中第 1~倒數第 2 個字元
print(word[:-1])       # Hello Python
# 取得 word 中第 1~倒數第 2 個字元
print(word[:-2])       # Hello Pytho
```

## 多行 `string`

當我們今天想輸入多行的 `string` 過去我們會用 `'''` 一個引號的方式，為了程式碼的可讀性，我們會改用 `"""` 三個引號的方式。

```
a = """Lorem ipsum dolor sit amet,  
consectetur adipiscing elit,  
sed do eiusmod tempor incididunt  
ut labore et dolore magna aliqua."""  
print(a)
```

## `print()` 函式的使用

在 Python 中我們如果想要輸出一個 `string` 我們會透過 `print()` 這個函式來達成，其中在 Python 中 `print()` 會自動換行，如果我們不想要換行，我們可以透過 `end` 這個參數來達成。

- `print(*objects, end='\n')`
- 可以傳入零或多個參數，用逗點分隔
- 預設 `print()` 會換行，可用 `end` 參數來改變

```
print("Hello")  
print("World")
```

```
print("Hello", end=" ")  
print("World")  
  
# 同理  
print("Hello", "World")
```

## `print` 補充 `sep` 間隔

其實 `print` 函式預設的間隔是一個空格，如果我們想要自行調整我們可以去更改 `sep` 的參數

```
print("Hello", "World", sep="----")
```

## 格式化

當我們今天想要讓程式的輸出變得更為美觀，我們常會使用格式化的技巧

## f-字串 (f-Strings)

```
name = "Hugo"  
age = 18  
print(f"Hello, my name is {name} and I am {age} years old.")
```

### 文字格式化方法 (`str.format()`)

```
name = "Hugo"  
age = 18  
print("Hello, my name is {} and I am {} years old.".format(name, age))
```

### 文字格式化運算子 (`% Operator`)

```
name = "Hugo"  
age = 18  
print("Hello, my name is %s and I am %d years old." % (name, age))
```

## list 串列操作

在 Python 中如果我們想要輸入 `list` 我們會透過 `[]` 一個中括號的方式，其中在 Python 中 `[]` 跟 `()` 的效力是相同的

[!IMPORTANT] 左右兩邊的中括號要一模一樣

1. 合格：`[1, 2, 3]`
2. 合格：`(1, 2, 3)`
3. 不合格：`[1, 2, 3)`
4. 不合格：`(1, 2, 3]`

### 新增元素到 `list`

```
fruits = ["apple", "banana", "cherry"]  
fruits.append("orange")  
print(fruits)          # ['apple', 'banana', 'cherry', 'orange']
```

--- [本週上課內容分隔線] ---

---

### 移除元素到 `list`

```
fruits = ["apple", "banana", "cherry"]  
fruits.remove("banana")  
print(fruits)          # ['apple', 'cherry']
```

## 插入元素到 **list**

```
fruits = ["apple", "banana", "cherry"]
fruits.insert(1, "orange")
print(fruits)          # ['apple', 'orange', 'banana', 'cherry']
```

## 修改元素值

```
fruits = ["apple", "banana", "cherry"]
fruits[1] = "orange"
print(fruits)          # ['apple', 'orange', 'cherry']
```

## 組合及替換多個 **list**

```
fruits = ["apple", "banana", "cherry"]
cars = ["Ford", "BMW", "Volvo"]
fruits.extend(cars)
print(fruits)          # ['apple', 'banana', 'cherry', 'Ford', 'BMW', 'Volvo']
```

## 刪除元素

```
fruits = ["apple", "banana", "cherry"]
fruits.pop(1)
print(fruits)          # ['apple', 'cherry']
```

## 刪除 **list**

```
fruits = ["apple", "banana", "cherry"]
del fruits
print(fruits)          # NameError: name 'fruits' is not defined
```

## 清空 **list**

```
fruits = ["apple", "banana", "cherry"]
fruits.clear()
print(fruits)          # []
```

## 複製 `list`

```
fruits = ["apple", "banana", "cherry"]
x = fruits.copy()
print(x)           # ['apple', 'banana', 'cherry']
```

## 反轉 `list`

```
fruits = ["apple", "banana", "cherry"]
fruits.reverse()
print(fruits)      # ['cherry', 'banana', 'apple']
```

## 排序 `list`

```
fruits = ["apple", "banana", "cherry"]
fruits.sort()
print(fruits)      # ['apple', 'banana', 'cherry']
```

## 串列的長度

```
fruits = ["apple", "banana", "cherry"]
print(len(fruits)) # 3
```

## 串列的索引

```
fruits = ["apple", "banana", "cherry"]
print(fruits[0])   # apple
```

## 串列的切片

```
fruits = ["apple", "banana", "cherry"]
print(fruits[1:3]) # ['banana', 'cherry']
```

## 串列的檢查

```
fruits = ["apple", "banana", "cherry"]
if "apple" in fruits:
    print("Yes, 'apple' is in the fruits list")
```

## 串列的迴圈

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    print(x)
```

## 串列的迴圈 (索引)

```
fruits = ["apple", "banana", "cherry"]
for i in range(len(fruits)):
    print(fruits[i])
```

## 串列的迴圈 (索引)

```
fruits = ["apple", "banana", "cherry"]
i = 0
while i < len(fruits):
    print(fruits[i])
    i += 1
```

## 課堂小練習

練習操作 `string` 並且搭配 `list` 的概念，給定一個字串 `Hello World!`，請試著做輸出 `World Hello` 以及 `Hello`

[!Note]

利用 `split` 的概念以及串列的切片實作

```
word = "Hello World"
# 開始你的努力吧！
# ...
# ...
```

## 參考答案



```
word = "Hello World"

# 解法一
word = "Hello World"
word = word.split(" ")

print("解法一: ")
print(word[1], word[0])
print(word[0])

# 解法二
word = "Hello World"
print("解法二: ")
print(word[6: 11], word[0: 6])
print(word[0: 6])
```

 [第二週課程](#) |  [第四週課程](#)

## License

All of these teaching materials are owned by [Hugo ChunHo Lin](#).

These materials are intended for tutoring purposes. They are open-source to foster a more vibrant Python learning community. We warmly welcome fellow enthusiasts interested in Python to use them. If you use a substantial portion of the source code, please include a link back to this repository.