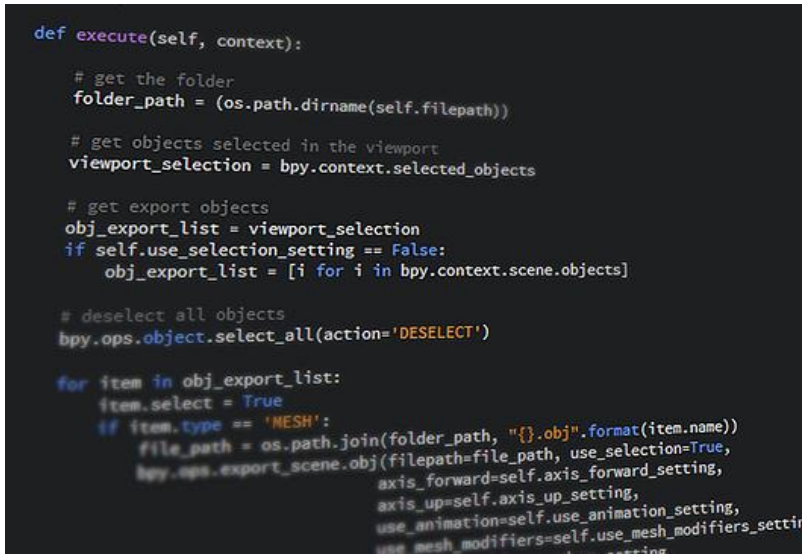
Python
函式 function

賴璉錡
lclai.t11@o365.fcu.edu.tw

DRY(Don’t Repeat Yourself)

- 寫程式時應避免同樣程式碼重複出現在很多地方
  - **可讀性低**
  - **不易維護**
- 所以要進行適當封裝，來提升程式碼的重用性(Reusable)



函式(Function)結構

- Python函式的結構包含了def關鍵字、函式名稱、參數及實作內容。
- 函式名稱的命名習慣使用小寫字母，並且以底線來分隔單字。
- 參數用來接收外部資料。
- 而實作的內容則是這個函式所要執行的任務，需注意縮排。
- 函式分成**有回傳值**與**沒有回傳值**兩種形式。

```
def cal_tax(income):
    tax = 0
    if income >= 0:
        if income <= 540000:
            tax = income * 0.05
        elif income <= 1210000:
            tax = income * 0.12
        elif income <= 2420000:
            tax = income * 0.20
        else:
            tax = income * 0.3
        return tax
    else:
        return 0
```

Practice:折扣計算函式

- 創建一個函式來計算商品的折扣後價格。折扣取決於顧客的購物金額。以下是折扣的規則：
  - **購物金額低於1000元，無折扣。**
  - **購物金額在1000至5000元之間，享受10%折扣。**
  - **購物金額在5000元以上，享受20%折扣。**

```
def calculate_discount(amount):
    [redacted]

# 使用範例
final_price = calculate_discount(4500)
print("折扣後的價格為：", final_price)
```

Practice: 錢包找零

- 你經營了一家小商店，每當顧客購物時，你需要幫他們找零。請撰寫一個函式，根據顧客應付的金額和他們給的金額，計算應該找回的硬幣數量。你需要考慮台灣的硬幣面額（50元、10元、5元、1元）。

```python
def find_change(price, paid):  1 usage
    # 在此處撰寫程式碼
    pass

# 測試範例
find_change( price: 135,  paid: 200)
```

輸入：
- 顧客應付的金額（正整數）。
- 顧客支付的金額（正整數，並且一定比應付的金額大或相等）。

輸出：
- 應該找回的每種硬幣數量，依序顯示50元、10元、5元、1元硬幣。

Practice: 餐廳點菜

- 你負責開發一個餐廳的點餐系統。每個餐點有不同的價格，顧客可以選擇他們想要的餐點，並且系統會計算他們的總金額。撰寫一個函式來計算顧客所選擇的餐點總價格。

```python
def calculate_total_price(menu, order):  1 usage
    # 在此處撰寫程式碼
    pass

# 測試範例
calculate_total_price(
    menu: {"漢堡": 50, "薯條": 30, "可樂": 20},
    order: ["漢堡", "薯條", "可樂"]
)
```

輸入：

- 一個餐點的字典，每個餐點對應其價格（整數或浮點數）。
- 一個顧客所選的餐點清單。

輸出：

- 所有選擇餐點的總價格。

```python
calculate_total_price(
    {"漢堡": 50, "薯條": 30, "可樂": 20},
    ["漢堡", "薯條", "可樂"]
)
```

函式(Function)參數

- 參數簡單來說就是接收外部所傳來的資料，進而執行相關的邏輯運算。
- 參數個數取決於函式內部運算時所需的資料，所以在一般情況下，呼叫函式時一定要傳入相對的參數個數資料，否則就會出現例外錯誤。

```
def cal_quantity(money, item_name, price):
    quantity = int(money / price)
    print("我有", money, "元，我可以購買", quantity, "個", item_name)
```

```
cal_quantity(100, "籃球", 20)
我有 100 元，我可以購買 5 個 籃球
```

函式(Function)參數

- **關鍵字參數(Keyword Argument)**：呼叫函式時，在傳入參數值的前面加上函式所定義的參數名稱。

```
def cal_quantity(money, item_name, price):
    quantity = int(money / price)
    print("我有", money, "元，我可以購買", quantity, "個", item_name)


cal_quantity(money=100, item_name="籃球", price=20)
cal_quantity(money=100, price=20, item_name="籃球")
```

函式(Function)參數

- **預設值參數(Default Argument)**：在函式定義的參數中，將可以選擇性傳入的參數設定一個預設值，當來源端有傳入該資料時，使用來源端的資料，沒有傳入時，則依照設定的預設值來進行運算。

```
def calculate_total(price, discount=1.0, tax=0.05)
    total_price = price * discount * (1 + tax)
    return total_price


print(calculate_total(100))                105.0
print(calculate_total(100, 0.8))           84.0
print(calculate_total(100, 0.8, 0.03))     82.4
```

函式(Function)參數

- 關鍵字參數在函式宣告時，必定在預設值參數之前。

```
def calculate_total(discount=1.0, price, tax=0.05):
    total_price = price * discount * (1 + tax)
    return total_price
```

```
    def calculate_total(discount=1.0, price, tax=0.05):
                                      ^
SyntaxError: non-default argument follows default argument
```

Practice:溫度設定

- 透過右方程式碼來檢視何種帶入的參數在何種狀況下能正確執行。

```
def set_temperature(name, temp=28):
    print(name, "將冷氣設定在", temp, "度。")


set_temperature("Mike")
set_temperature(temp=25, name="Andy")
set_temperature(temp=100)
```

Practice: 網路購物運費計算

- 設計一個函式 calculate_shipping，計算網路購物的運費。預設運費為 100 元，購物金額超過 1000 元則免運。使用者可以選擇是否使用優惠券，來進一步減少運費。
- **函式說明：**
  - **預設運費為 100 元。**
  - **購物金額超過 1000 元免運。**
  - **如果使用優惠券，可額外減少 20 元運費。**

輸入：
- 購物金額（正整數）
- 是否使用優惠券（選填，布林值，預設為 False）

輸出：
- 計算並顯示最終運費。

```
def calculate_shipping(total_amount, use_coupon=False):  3 usages
    [redacted]
    print(f"最終運費: {shipping_fee} 元")

# 測試範例
calculate_shipping(1200)
calculate_shipping( total_amount: 800, use_coupon=True)
calculate_shipping(600)
```

函式(Function)種類

- **有回傳值**：在函式完成運算後，會在最後加上 `return` 關鍵字將結果回傳給來源端，進而做其它的運用

```
def return_add_value(number):
    total = 0
    for i in range(number + 1):
        total = total + i
    return total


total_value = return_add_value(10)
print(total_value)
```

```
55
```

函式(Function)種類

- **無回傳值**：在函式運算的最後，沒有加上 `return` 關鍵字，單純執行完某一項任務

```
def print_add_value(number):
    total = 0
    for i in range(number + 1):
        total = total + i
    print(total)


total_value = print_add_value(10)
print(total_value)
```

```
55
None
```

函式(Function)變數範圍(Scope)

- 在任何程式語言中，變數都有它的有效範圍，也就是變數所在的程式碼位置，會影響到是否可以進行存取。
- **區域變數(Local Variable)**：在函式中所定義的變數就是區域變數。只有在函式的範圍中都可以進行存取，而函式以外的其它地方，則無法進行存取。
- **全域變數(Global Variable)**：只要在同一個Python檔案中，皆可進行存取。

函式(Function)變數範圍(Scope)

```python
grade = [70, 80, 75]


def cal_average(grade_list):
    total = 0
    count = len(grade_list)
    for i in grade_list:
        total += i
    average = total / count

    print("grade in function:", grade)
    print("grade_list in function:", grade_list)
    print("average in function:", average)
    return average

grade_average = cal_average(grade)
print("average", average)
print("grade_average", grade_average)
```

```
Traceback (most recent call last):
  File "C:/Users/selab/PycharmProjects/pythonProject/0322prac6.py", line 17,
    print("average", average)
NameError: name 'average' is not defined
grade in function: [70, 80, 75]
grade_list in function: [70, 80, 75]
average in function: 75.0
```

遞迴(Recursion)

- 遞迴(Recursion)的概念是將一個大的問題，分割成許多小問題去解決。而從程式設計角度來看，函式不單只能被其他函式呼叫，也能被它自己呼叫，也就是在一個函式當中呼叫它自己，即為遞迴函式(Recursive Function)。

```
def factorial(n):
    if n == 1:
        return 1
    else:
        return n * factorial(n - 1)


print(factorial(5))
```

```
Factorial(5)
= 5 * Factorial(4)
= 5 * 4 * Factorial(3)
= 5 * 4 * 3 * Factorial(2)
= 5 * 4 * 3 * 2 * Factorial(1)
= 5 * 4 * 3 * 2 * 1
= 120
```

Practice:費氏數列

月份	1	2	3	4	5	6	7
兔子總數(對)	1	1	2	3	5	8	13
可生育的兔子數(對)	0	1	1	2	3	5	8

- 義大利人費波那契(Leonardo Fibonacci) 他描述兔子生長的數目時用上了這數列。
  - **第一個月初有一對剛誕生的兔子**
  - **年齡大於等於兩個月的兔子可以生育**
  - **每月每對可生育的兔子會生下一對新兔子**
  - **兔子永不死去**

已知 Fib(1) = 1, Fib(2) = 1

Fib(6)

= Fib(4) + Fib(5)

= [Fib(3) + Fib(2)] + [(Fib(4) + Fib(3))]

= [(Fib(2) + Fib(1)) + 1] + [(Fib(3) + Fib(2)) + (Fib(2) + fib(1))]

= [(1+1)+1] + [(Fib(2)+Fib(1) + 1) + (1 + 1)]

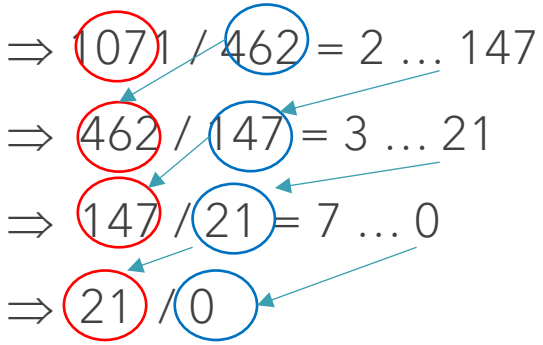
= [3] + [(1 + 1 + 1) + 2]

= 3 + 5 = 8

HW:最大公因數(GCD)

- 請示使用者輸入兩個數，然後用遞迴方式找到這兩個數的最大公因數
- 能整除輸入兩數的最大數字
- 輾轉相除法

a = 1071, b = 462



此時餘數是0，所以1071和462的最大公因數是21

檢查: ◯為0，則 ◯為最大公因數