Python Lecture 02

林杰 (國立台灣大學資訊管理學系)

EMILY in IM

Agenda

- 元組 Tuple
- 集合 Set
- 字典 Dictionary
- 函數 Function
- 物件 Object
- 模組與套件

課程進行

- 課程進行方式比照 Python Lecture 01
- .ipynb 請先準備好
- 開啟方式：Google Colab / Jupyter Notebook
- 課程講義、隨堂練習、歡迎先往下做題目

元組 Tuple

元組 Tuple - 動機

tuple 和 list 是非常相似的東西，不過 tuple 在建立之後就無法更改裡面的資料。

為什麼不直接用 list 就好？

1. Tuple 是不可變的，可以防止資料被意外更改。
2. Tuple 可以作為字典的鍵（key），在下一部分會說明。

Tuple 的應用

我們可能會將什麼東西存進 tuple 中呢？

1. 平面座標：`coordinate = (2, 4)`
2. RGB 顏色表示：`color = (255, 255, 0)`

另一個可以認識 tuple 的原因：
tuple 這個結構和離散數學與自動機理論中的許多理論定義與操作有關，適合表達「不需改變」的數學結構。

元組 Tuple - 宣告賦值

在 Python 中，我們使用 (小括號) 表達元組。

創建一個元組可以透過...

1. `a_tuple = tuple()`
2. `a_tuple = ()`
3. `coordinate = (0, 0)`

元組 Tuple - 取值

在 Python 中，我們使用 (小括號) 表達元組。

取得元組中索引值 `i` 的內容...

```
coordinate = (0, 0)

print(coordinate)  # 印出 (0, 0)
print(coordinate[0])  # 印出 0
```

元組 Tuple - 注意事項

tuple 和 list 是非常相似的東西，不過 tuple 在建立之後就無法更改裡面的資料。

所以我們不能對一個 tuple 新增元素、刪除元素。

元組 Tuple - 練習

小試身手 (1)

請寫出一段 Python 程式碼，來創建一個元組，並嘗試將其中的元素改為其他值。

觀察並說明程式執行的結果。

集合 Set

集合 Set - 動機

集合是數學中一個十分常用的概念，它用來描述一組明確而互不重複的對象。

在 Python 中，set 是一個裝資料的大容器：

- 同樣的值只出現一次
- 裡面裝的元素之間是沒有位置關係的
- 學習用 Python 操作 Set，順便複習集合 :)

集合 Set - `add()`

使用 `add()` 可以讓我們在集合中加入一筆資料。

```
a_set = set() # 空集合
a_set.add(10) # 添加元素 10
a_set.add(20) # 添加元素 20
a_set.add(10) # 再次添加 10(無效，集合不允許重複)
a_set # ???
```

Q&A

上面 `a_set` 這個集合中，有哪些元素?

集合 Set - `remove()`

`remove()` 方法從集合中刪除指定的元素。如果該元素不存在，會拋出 `KeyError`。

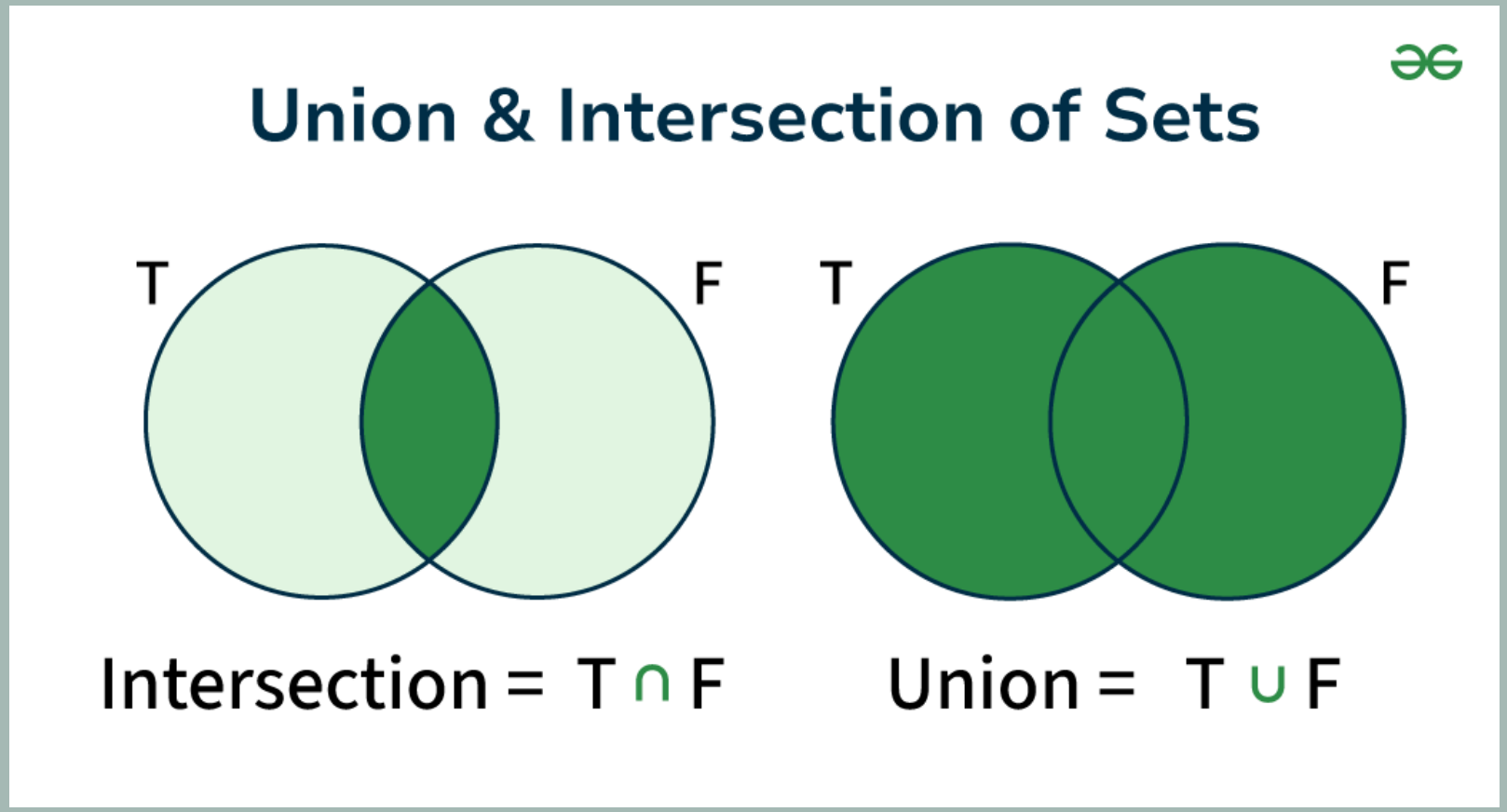
```
a_set = {1, 2, 3, 4}
a_set.remove(3) # 移除元素 3
print(a_set) # {1, 2, 4}
a_set.remove(5) # KeyError
```

集合 Set - `discard()`

`discard()` 方法從集合中刪除指定的元素。如果該元素不存在，不會拋出錯誤。

```
a_set = {1, 2, 4}
a_set.discard(5) # 沒有錯誤
```

集合 Set - 聯集與交集



集合 Set - `union()`

`union()` 方法用於返回兩個集合的聯集，包含兩個集合所有唯一元素的集合。

```
a_set = {1, 2, 3}
b_set = {3, 4, 5}
result = a_set.union(b_set) # 返回兩集合的聯集
print(result) # {1, 2, 3, 4, 5}
```

集合 Set - `intersection()`

`intersection()` 方法用於返回兩個集合的交集，即同時存在於兩個集合中的元素。

```
a_set = {1, 2, 3}
b_set = {3, 4, 5}
result = a_set.intersection(b_set) # 返回兩集合的交集
print(result) # {3}
```

集合 Set

小試身手 (2)

Set 練習 (請看 Jupyter Notebook)

字典 Dictionary

字典 Dictionary - 介紹

- 字典是由鍵值對 (key-value pairs) 組成的無序集合。
- 鍵是唯一的，而值可以是任何資料類型。
- 可以通過鍵來存取對應的值。

Mapping (keys → values)

- "Name" → "Tran"
- "Age" → 37
- "Address" → "Vietnam"

字典 Dictionary - 宣告賦值

在 Python 中，我們使用 **{大括號}** 表達元組。

- 創建一個空字典：`my_dict = {} # 空字典`
- 使用鍵值對初始化字典：

```
person = {"name": "Alice",
          "age": 25,
          "city": "New York"}
print(person) # {'name': 'Alice', 'age': 25, 'city': 'New York'}
```

字典 Dictionary - 取值

```
person = {'name': 'Alice', 'age': 25, 'city': 'New York'}
```

當我們想要取出某個 key 所對應到的 value，我們可以用 `dict[key]`。

例如，如果想要求出 name 這個 key 所對應的 value，我們可以這樣寫：

```
person['name'] # 程式就會回傳 'Alice'
```

字典 Dictionary - 更新

```
person = {'name': 'Alice', 'age': 25, 'city': 'New York'}
```

字典更改和增加的語法一模一樣！

如果我們想要將 `age` 的值改為 `30` ，可以這樣寫：

```
person["age"] = 30 # 更改 age 的值
```

如果我們想要新增鍵值對，可以這樣寫：

```
person["school"] = "NTU"
```

字典 Dictionary - 更新

`person` 就變成了

```
{'name': 'Alice', 'age': 30, 'city': 'New York', 'school': 'NTU'}
```

字典 Dictionary - 刪除元素

我們有兩種方法可以刪除字典中的元素：

1. 使用 `del` 刪除指定鍵:

```
del person["city"] # 刪除 "city" 這個鍵及其對應的值
print(person) # {'name': 'Alice', 'age': 30, 'school': 'NTU'}
```

字典 Dictionary - 刪除元素

2. 使用 `pop()` 刪除指定鍵：

```
age = person.pop("age") # 刪除並返回 "age" 的值
print(person) # {'name': 'Alice', 'school': 'NTU'}
print("Age:", age) # 30
```

現在 `person` 變成 `{'name': 'Alice', 'school': 'NTU'}`。

字典 Dictionary - 取出鍵或值

在某些應用中，我們只想取出字典中的鍵或值：

使用 `keys()` 取出所有鍵：

```
keys = person.keys()
print(keys) # dict_keys(['name', 'school'])
```

在這裡，`keys` 的資料型態是 `dict_keys`。

字典 Dictionary - 取出鍵或值

使用 `values()` 取出所有值：

```
values = person.values()
print(values) # dict_values(['Alice', ‘NTU'])
```

字典 Dictionary - 取出鍵與值

在某些應用中，我們只想取出字典中的鍵或值：

使用 `items()` 取出所有鍵值對

```
items = person.items()
print(items) # dict_items([('name', 'Alice'), (‘school', ‘NTU')])
```

備註

資料型態 dict_keys、dict_values 和 dict_items 都是可以迭代的物件，可以用 `for` 迴圈取出所有元素。

字典 Dictionary

小試身手 (3)

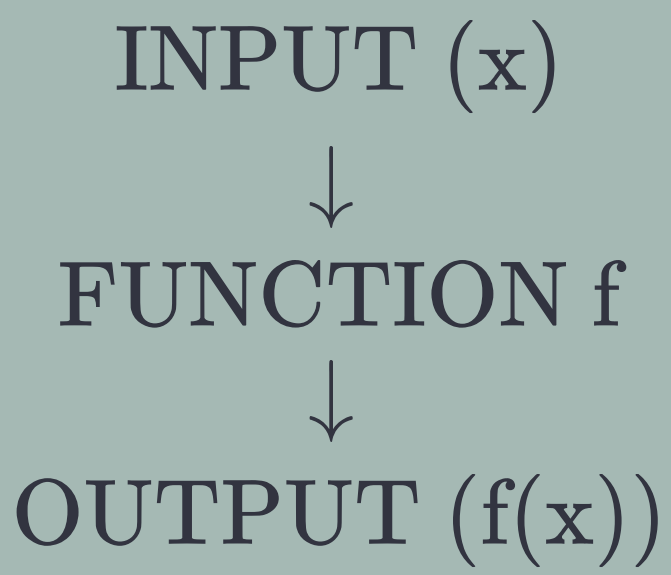
（改編自 ZeroJudge b523）

題目：有多個輸入被存在 `input_lines` 中，每一個元素都是可能包含大小寫英文字母、數字、空白的字串。

對於每一個字串，判斷其是否為第一次出現，若該字串是第一次出現，就印出 NO。若該字串曾經出現過，則印出 YES。

函數 Function

函數 Function - 圖解



函數 (functions) 是一段組織好的、可以重複使用的程式碼，用於執行特定的任務。

函數 Function - 介紹

- 在數學中，函數 (function) 是什麼？
- 假設 $f(x) = 3x$，這裡的 $f(x)$ 就是一個函數。
- 當 $x = 3$ 時，$f(x)$ 回傳 9。

函數 (functions) 是一段組織好的、可以重複使用的程式碼，用於執行特定的任務。

函數 Function - 動機

我們把

- 「提供給函數處理」的資料稱作參數。
- 函數經過操作後回傳的結果稱為回傳值。

函數 Function - 動機

主要功能：

- 重用性:避免重複程式碼。
- 可讀性:將代碼結構化，便於理解和維護。
- 模組化:將程式劃分為獨立的部分。

備註

(為了教學方便，在這個講義中，參數和引數視為同義詞，不特別區分其差異。)

函數 Function - 函數定義與呼叫

定義函數的基礎結構

```
def 函數名稱(參數1, 參數2, ...):
    # 函數的說明(可選)
    函數的邏輯
    return 返回值(可選)
```

函數 Function - 函數定義與呼叫

定義函數

```
def greet(name):
    # 打招呼函數
    return f"Hello, {name}!"
```

呼叫函數

```
message = greet("Alice")
print(message) # Hello, Alice!
```

函數 Function - 函數定義與呼叫

運行步驟：

1. 定義函數:使用 `def` 關鍵字。
2. 呼叫函數:將參數傳入函數。
3. 函數執行邏輯，返回結果。

函數 Function - 帶有預設參數的函數

```python
def greet(name="Guest"):
    # 打招呼函預設參數
    return f"Hello, {name}!"

# 不傳參數，使用預設值
print(greet()) # Hello, Guest!

# 傳入參數，覆蓋預設值
print(greet("Alice")) # Hello, Alice!
```

函數 Function - 帶有預設參數的函數

補充

- 預設參數在未提供實際值時會被使用。
- 預設參數可用於設置函數預期行為。

函數 Function - 帶多個參數的函數

計算 a 和 b 的加權平均數

```python
def weighted_average(a, b, weight_a=0.5, weight_b=0.5):
    # 計算兩數的加權平均數
    return (a * weight_a) + (b * weight_b)

# 呼叫函數，使用預設權重
print(weighted_average(80, 90)) # 85.0

# 傳入自定義權重
print(weighted_average(80, 90, weight_a=0.7, weight_b=0.3)) # 83.0
```

函數 Function - 帶多個參數的函數

補充

- 可以使用多個參數打造更靈活的函數。
- 不一定每個函數都要有預設值！

函數 Function - 如何將參數傳入函數？

印出一串自我介紹的文字

```
def intro(name, country, age):
    print("My name is " + name + ". I'm " + str(age) + " years old. I'm from " + country + ".")
```

函數 Function - 如何將參數傳入函數？

若沒有特別指定，傳入參數的順序要照函數宣告時所用的順序：

```
intro("John", "Taiwan (R.O.C.)", 18)
# My name is John. I'm 18 years old. I'm from Taiwan (R.O.C.).
```

我們也可以用關鍵字傳參數，在傳入的值前面「註明」其所對應的參數名稱！

```
intro(country = "Taiwan (R.O.C.)", age = 18, name ="John")
# My name is John. I'm 18 years old. I'm from Taiwan (R.O.C.).
```

函數 Function - 變數範疇與命名空間

命名空間（namespace）是一個「儲存區」，用來對應變數名稱與其值：

全域命名空間 (global namespace)

- 儲存全域變數。

區域命名空間 (local namespace)

- 儲存函數內的區域變數。

函數 Function - 變數範疇與命名空間

變數範疇 (Scope) 定義了變數的生命週期與可見範疇。

全域範疇 (Global Scope)：

- 變數定義在函數外部，可被整個程式式訪問，通常在全域命名空間中管理。

區域範疇 (Local Scope)：

- 變數定義在函數內部，僅限該函數內部訪問，屬於區域命名空間的一部分。

例子 - 全域範疇 (Global Scope)

```
# 全域變數
x = 10  # x 在全域範疇內

def example_function():
    print(x)  # 訪問全域變數 x

example_function()  # 輸出 10
```

在這個例子中，變數 `x` 被定義在函數外部，並且在 `example_function()` 中成功訪問。

例子 - 區域範疇 (Local Scope)

```
def example_function():
    # 區域變數
    x = 5  # x 在區域範疇內
    print(x)  # 訪問區域變數 x

example_function()  # 輸出 5
# print(x) 會引發錯誤，因為 x 是區域變數，無法在函數外部訪問
```

在這個例子中，變數 `x` 被定義在 `example_function()` 內部，它只能在該函數內部訪問，並且在函數外部無法訪問。

函數 Function - 變數範疇與命名空間

```
def function_1():
    x = 5
    print("In function_1, x =", x)

def function_2():
    x = 6
    print("In function_2, x =", x)
```

函數 Function - 變數範疇與命名空間

```
x = 8
function_1()
function_2()

print("In main, x =", x)
```

小試身手 (4)

輸出是什麼？哪些是全域變數、哪些是區域變數？

函數 Function - 變數範疇與命名空間

解答

```
def function_1():
    x = 5 # 區域變數 (local variable)
    print("In function_1, x =", x)

def function_2():
    x = 6 # 區域變數 (local variable)
    print("In function_2, x =", x)
```

函數 Function - 變數範疇與命名空間

```
x = 8 # 全域變數 (global variable)
function_1()
function_2()

print("In main, x =", x)
```

函數 Function - 遞迴函數

遞迴函數是什麼：

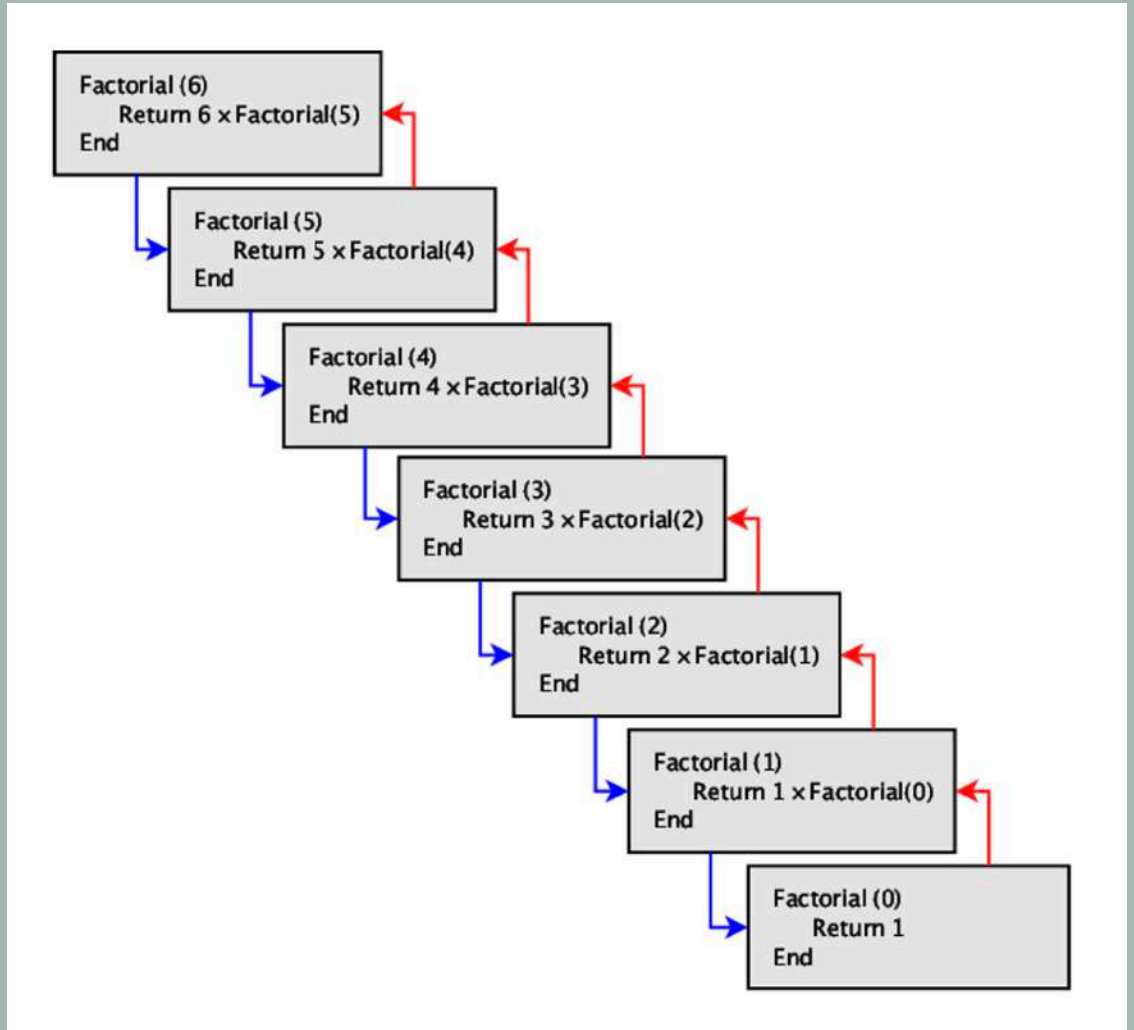
- 其定義中調用自身的函數，
- 來解決那些可以分解為相同問題的**較小版本**的問題。
- 你可能學過的遞迴函數：費氏數列、階乘運算...

函數 Function - 遞迴函數（階乘）

```
def Factorial(n):
    if n == 0:
        return 1
    else:
        return n * Factorial(n-1)
```

如果呼叫 Factorial(6)，會發生什麼事情？

函數 Function - 遞迴函數（階乘）



函數 Function - Factorial(n) 的運作

遞迴函數必須有以下兩個元素：

1. 基礎情況 (Base Case):

- 如果 $(n = 0)$，直接回傳 1。
- 這是遞迴的**終止條件**。
- 必須設定基礎情況以避免無窮遞迴。

函數 Function - Factorial(n) 的運作

2. 一般情況 (Recursive Case):

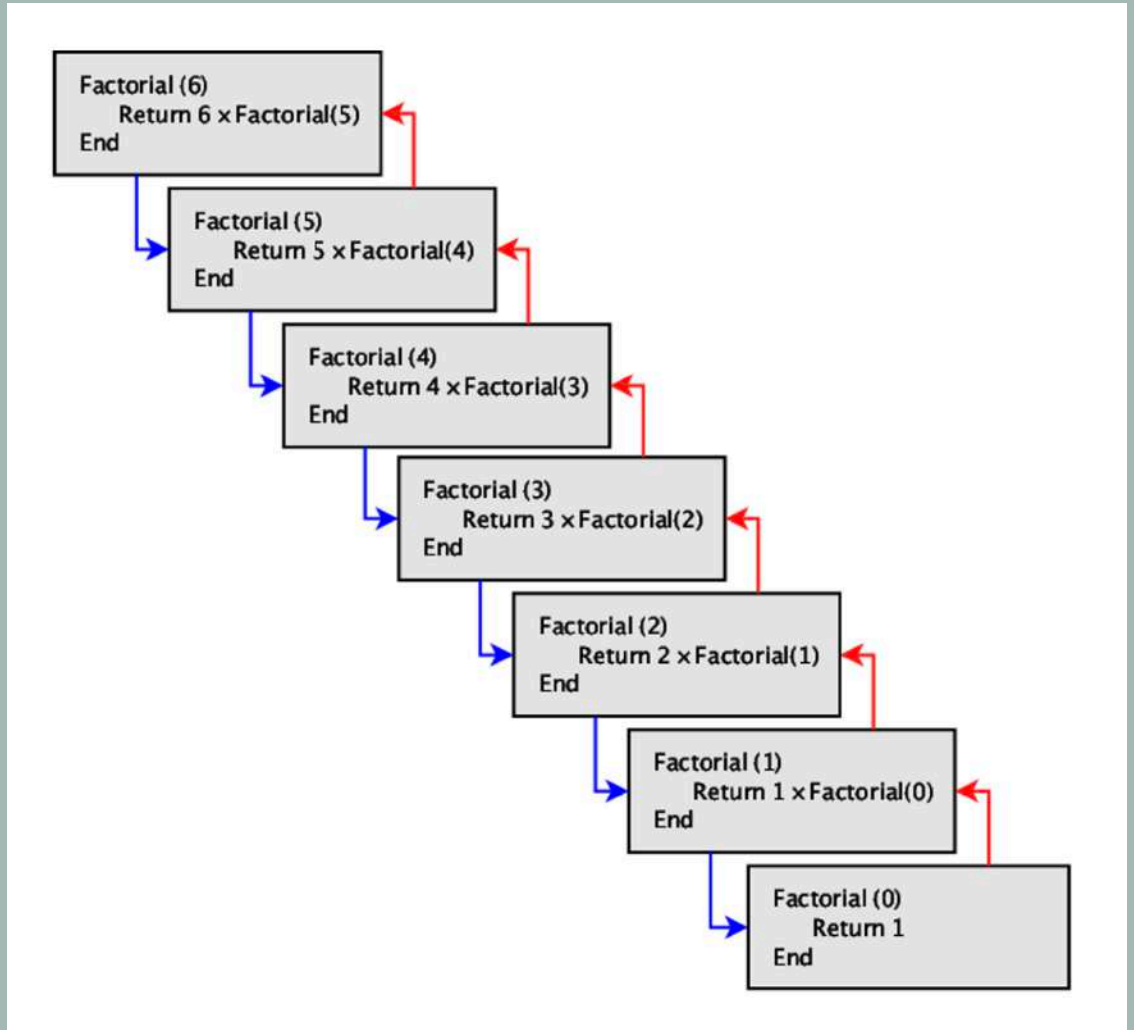
- 如果 $(n \neq 0)$，函數會呼叫自身：`n * Factorial(n - 1)`。
- 每次呼叫時，n 減少 1，直到 $(n = 0)$ 結束。

函數 Function - Factorial(n) 的運作

```
def factorial(n):
    if n == 0: # 檢查是否為 base case
        return 1
    else: # 如果未達到，則進行遞迴。
        return n * factorial(n-1)
```

逐步回傳結果，完成計算。

函數 Function - 遞迴函數（階乘）



函數 Function - 遞迴函數

小試身手 (5)

定義函數 $F(x)$

$$F(x) = \begin{cases} 1 & \text{若 } x = 1 \\ F\left(\frac{x}{2}\right) & \text{若 } x \text{ 為偶數} \\ F(x-1) + F(x+1) & \text{其餘情況} \end{cases}$$

請在講義 Jupyter Notebook 中寫出此函數，並算出 $F(14)$ 和 $F(15)$。

物件 Object

物件 Object - 基本概念

將物件類比為一個「盒子」，內部包含：

1. 屬性 (Attributes)：描述物件的資料（如名稱、顏色）。
2. 方法 (Methods)：操作物件的行為（如咆哮、吃東西）。

物件 Object - 基本概念

類別 (Class)

用來定義物件的範圍，規範所有屬性與方法的共同特徵。

例如，定義「貓咪」類別，包含：

- 屬性：名稱、毛色。
- 方法：咆哮、睡覺。

物件 Object - 基本概念

實例 (Instance)

具體化的物件，例如：

- 貓咪實例：大橘、小白。
- 它們的名稱不同，但都屬於「貓咪」類別，共享基本屬性和方法。

物件 Object - 類別定義

```python
# 使用 class 定義屬於自己的物件
class Cat:
    # 初始化方法 (__init__)：當創建物件時，會自動執行，給屬性賦初值
    def __init__(self, name, color, weight):
        self.name = name
        self.color = color
        self.weight = weight
    # 方法 (Methods)：表現物件行為，例如貓的喵喵叫、進食等動作。包含三個方法：meow, eat, and sleep。
    def meow(self):
        print(f"{self.name} says Meow!")
    def eat(self, food):
        print(f"{self.name} is eating {food}.")
    def sleep(self, hours):
        print(f"{self.name} sleeps for {hours} hours.")
```

物件 Object - 範例

```
cat1 = Cat("大橘", "橘色", 5.5)
cat2 = Cat("小白", "白色", 4.2)
cat3 = Cat("黑咪", "黑色", 6.0)

cat1.meow() # 大橘 says Meow!
cat2.eat("魚") # 小白 is eating 魚.
cat3.sleep(10) # 黑咪 sleeps for 10 hours.
```

模組與套件

模組與套件 - 動機

為什麼使用模組與套件?

1. 提升開發效率

- 重複使用他人編寫的高品質程式碼，避免重複造輪子。

2. 程式結構化

- 模組:方便將相關函數與類別集中管理。
- 套件:建立分層架構，便於組織與維護。

模組與套件 - 使用方式

匯入整個模組或套件

```
import 模組套件或名稱
```

匯入模組或套件，並給別名

```
import 模組或套件名稱 as 別名
```

匯入模組或套件中特定的函數或類別

```
from 模組或套件名稱 import 子模組, 套件或函數名稱
```

模組與套件 - 以 Numpy 為例

Why Numpy?

- Numpy 已經幫我們寫好了許多複雜的數學運算，讓我們可以不用自己寫!
- 避免重複實作複雜的數學。

我們可以同時達到程式結構化，NumPy 將矩陣運算和科學計算相關功能集中管理，簡化專案結構。

模組與套件 - 以 Numpy 為例

模組與套件 - 以 Numpy 為例

```
# 利用簡短的別名 np 增加程式開發效率
import numpy as np
```

在 Numpy 中建立陣列： `arr = np.array([1, 2, 3, 4])`

矩陣及向量運算： `result = np.dot([1, 2], [3, 4])`

數學運算： `sin_values = np.sin(arr)`

模組與套件 - 以 Numpy 為例

使用不同的套件時，Google, ChatGPT 都是你的好朋友：）

小試身手 (6)

Numpy 練習 (請看 Jupyter Notebook)