



Python 程式設計





課程內容

- 輸出、輸入
- 資料型態&型別轉換
- 算術運算 / round / abs
- String format / f-string
- 字串操作
- 流程控制-判斷
- 流程控制-迴圈
- List
- List Slice / String Slice
- Dict
- Function

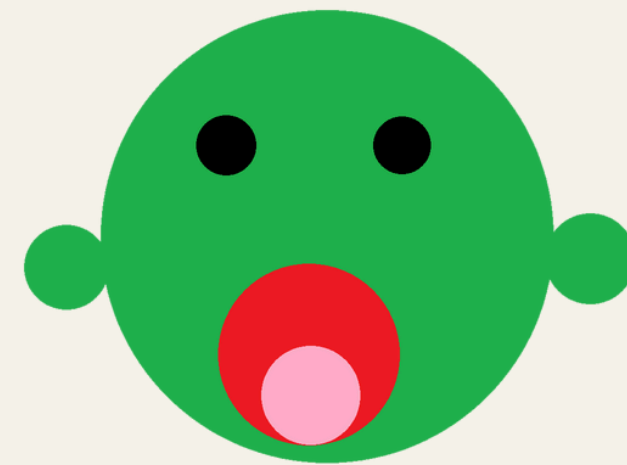
Python 是什麼？

Python 是一種程式語言，就像我們跟人溝通要說中文、英文，日文...
Python 就是其中一種跟電腦溝通的語言，我們可以透過程式讓電腦執行我們指定好的動作。



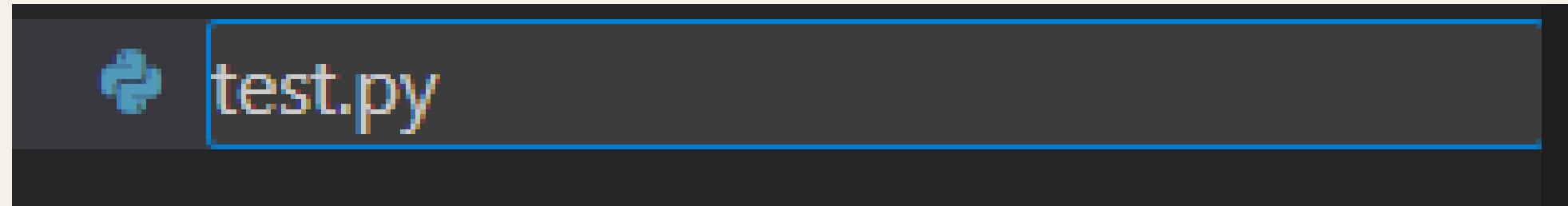
請你輸出 "Hello world"

"Hello world"



建立檔案

首先我們需要建立一份指令檔案，副檔名設定為.py，例如：



接下來就是快樂 py 派對時間了



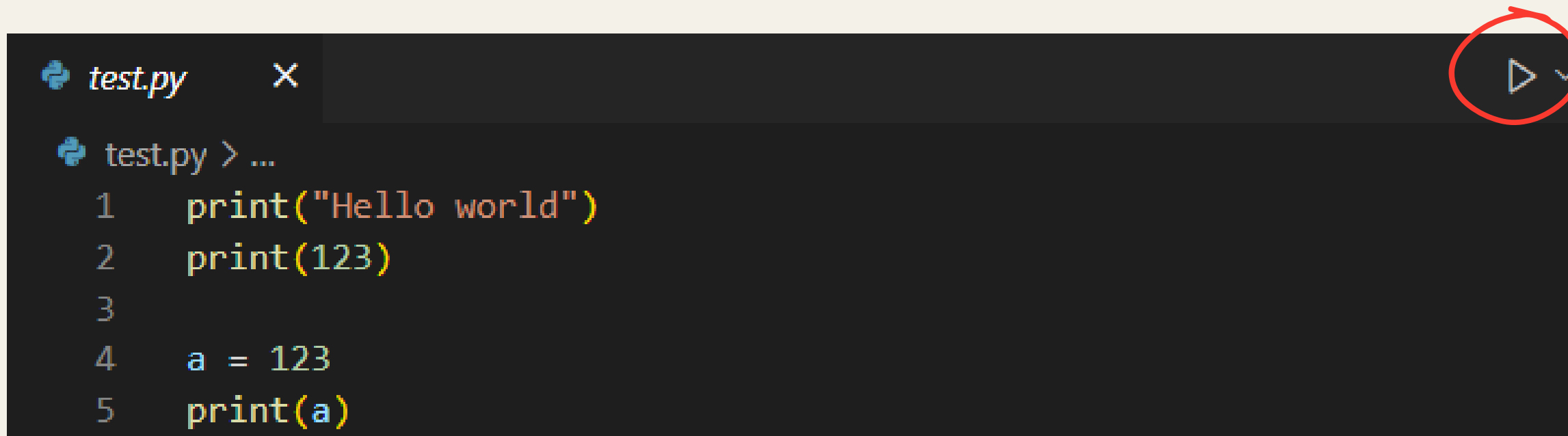
輸出

- 輸出的方式是利用 `print()` 這個函式做輸出
- () 裡面放要輸出的東西，可以是字串、變數、數字等...

```
test.py > ...  
1  print("Hello world")  
2  print(123)  
3  
4  a = 123  
5  print(a)
```

執行程式

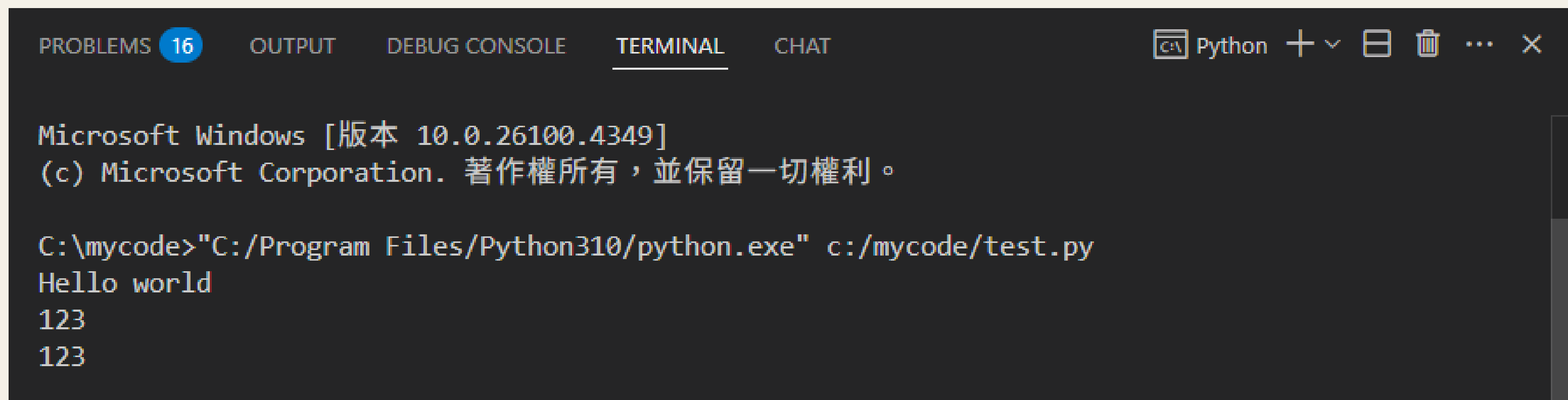
寫好語言後，我們來執行看看程式
程式碼區塊的右上角有個小小的執行按鈕，給他點下去

A screenshot of a code editor interface. At the top, there's a tab labeled 'test.py' with a close button 'X'. Below the tab, the code editor shows a Python script with five lines: '1 print("Hello world")', '2 print(123)', '3', '4 a = 123', and '5 print(a)'. In the top right corner of the editor, there is a small play button icon (a right-pointing triangle) which is circled in red. Next to it is a small checkmark icon.

```
test.py X
test.py > ...
1  print("Hello world")
2  print(123)
3
4  a = 123
5  print(a)
```

執行程式

這樣你就完成了第一支程式



```
PROBLEMS 16 OUTPUT DEBUG CONSOLE TERMINAL CHAT Python + ▾ ☐ ☒ ... ✕

Microsoft Windows [版本 10.0.26100.4349]
(c) Microsoft Corporation. 著作權所有，並保留一切權利。

C:\mycode>"C:/Program Files/Python310/python.exe" c:/mycode/test.py
Hello world
123
123
```

輸出 (ex1)

() 裡面也可以放多個參數並用 "," 隔開

```
2  print("A", "BB", "CCC")  
3  print(1, "+", "1", "= 2")
```


輸出

我們還可以在後面設定結尾字元與換行間隔字串

`end="換行字串"`

`sep="間隔字串"`

這兩個參數預設值是 `end="\n", sep=" "`

```
7   print("Hi", end=".")
8   print()
9   print(1, "1", sep="+")
10  print(1, "2", sep="+", end="=3")
```

輸出 (跳脫字元)

跳脫字元：反斜線 (\)

用來顯示一些特殊的字符，如以下常用字符

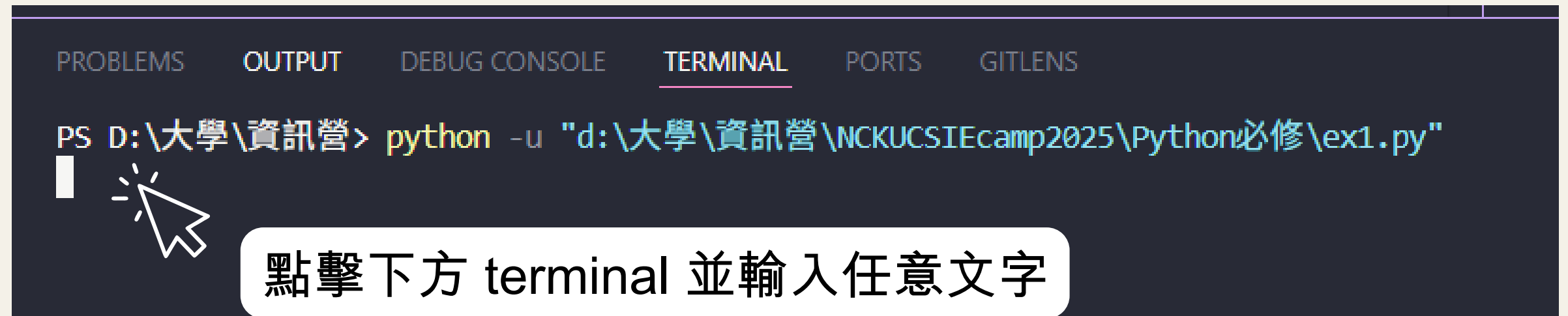
\n => 換行	\t => tab
\' => '	\" => "
\\ => \	

```
6  print("line1\nline2")
7  print("\tI'm tab")
8  print("\' is single quote")
9  print("\" is double quote")
10 print("/\._.'\\")
```

輸入 (ex2)

能透過使用 input() 取得使用者的輸入內容
並根據內容做額外操作

```
abc = input()  
print("變數abc的值:", abc)
```



練習時間 (p1)

資料基本型態 (ex3)

每一份資料都有它的型態，不同型態能進行的操作方式也不同

以下是最基本的資料型態：

1. 數值型態 (能進行數值運算) : int, float, bool
2. 字串型態 (單純文字處理) : char, string
3. 容器型態 (後續提及) : list, dict

我們可以簡單將資料傳入 `type(...)` 這個函式即可知道它的型態

```
1 print("123", "is", type("123"))
2 print(123, "is", type(123))
3 print(123.0, "is", type(123.0))
4 print(True, type(True))
```

資料型態轉換

在轉換型態之前，我們要先了解基礎型態的意義

1. 數值型態：

- a. int 表示整數數值
- b. float 表示浮點數(小數)
- c. bool 表示布林值 (0 or 1 / False or True)

2. 字串型態：

- a. char 表示字元，簡單講就是一個字
- b. string 表示字串，相當於由複數個字元所組成

資料型態轉換

型態轉換，俗稱Casting，也就是變更資料原始的型態
但轉換時要特別注意，由於轉換後的型態與原始資料不同
因此，可能導致資料內容變動

```
9   # 資料型態轉換
10  print(type(float(123)), end=" : ")
11  print(float(123)) # 123.0
12
13  print(type(int(123.5)), end=" : ")
14  print(int(123.5)) # 123
15
16  print(type(float(True)), end=" : ")
17  print(float(True)) # 1.0
18
19  print(type(bool(0.5)), end=" : ")
20  print(bool(0.5)) # True
```

數值資料間可以任意轉換，但內容也會轉換成該型態的意義

※ bool：只要數字 > 0 即為 True，反之為 False

資料型態轉換

但當你想要將進行 "字串型態" 轉換成 "數值型態" 時
需根據數值意義範圍 (float > int) 進行轉換
字串中數值的型態，只能轉成範圍相等或較大者，否則會報錯
並且要確保字串中只有單純數值，無其他文字或空字串

```
26 print(type(int("123")), end=" : ")
27 print(int("123")) # 123
28
29 print(type(float(123.45)), end=" : ")
30 print(float("123.45")) # 123.45
31
32 print(type(float("123")), end=" : ")
33 print(float("123")) # 123.0
34
35 print(type(bool("123.45")), end=" : ")
36 print(bool("123.45")) # True
37
38 print(type(bool("")), end=" : ")
39 print(bool("")) # False
```

```
41 print(int("123.45")) # error
42 print(int("A0")) # error
```

※ bool(str) 只是單純判斷是否為空字串

算術運算 (ex4)

接下來我們要介紹針對數值資料的運算處理

常用運算：

1. + 加
2. - 減
3. * 乘
4. / 除，回傳浮點數(float)
5. // 整除，回傳整數(int)
6. % (Mod)取餘數
7. ** 指數

```
1  a = int(input("輸入a:"))
2  b = int(input("輸入b:"))
3
4  print("a + b =", a+b)
5  print("a - b =", a-b)
6  print("a * b =", a*b)
7  print("a / b =", a/b)
8  print("a // b =", a//b)
9  print("a % b =", a%b)
10 print("a ** b =", a**b)
```

算術運算

其餘我們最常用的操作，莫過於 "四捨五入" 和 "絕對值"

四捨五入：round (num, int digit) / round (num) 若不給digit預設為0

絕對值：abs(num)

```
14 num = -0.123456
15 print(f"round({num}):", round(num))
16 print(f"round({num}, 2):", round(num, 2))
17 print(f"round({num}, 5):", round(num, 5))
18 print()
19 print("abs(5):", abs(5))
20 print("abs(-5):", abs(-5))
21 print("abs(num):", abs(num))
```

※ 由於電腦是以二進制做round，因此當遇到5時，有可能不會進位

練習時間 (p2)

字串格式化 format (ex5)

講完數值資料的操作，輪到字串資料的處理拉～

首先先來認識一下字串格式化

要了解為什麼我們會想要將字串格式化，我們先看下面兩張圖中 print 那行，你覺得哪個比較清楚？

```
name = "Alice"  
age = 20  
print("Name:", name, "Age:", age)
```

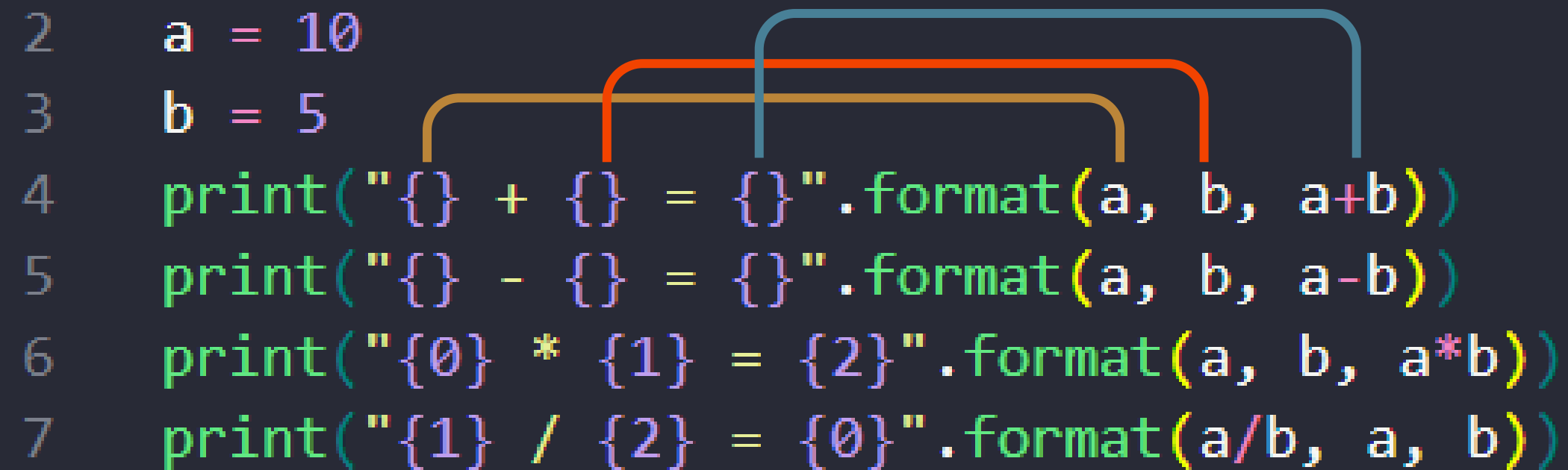
```
print("Name: {}, Age: {}".format("Alice", 20))
```

字串格式化 format

可能你會認為那兩張圖好像看起來都差不多XD
但格式化的核心功能是「能將變數的值快速寫成string」

透過在字串中加入{}，使得.format(...) 中的變數，能顯示到對應的 {} 中
同時若不填變數，也可直接填入運算過程，直接顯示運算後的值

{ }中亦可填入索引值(index)，對應右側



```
2 a = 10
3 b = 5
4 print("{} + {} = {}".format(a, b, a+b))
5 print("{} - {} = {}".format(a, b, a-b))
6 print("{0} * {1} = {2}".format(a, b, a*b))
7 print("{1} / {2} = {0}".format(a/b, a, b))
```

The code snippet demonstrates various string formatting techniques. Lines 2 and 3 define variables `a` and `b`. Lines 4 through 7 show different ways to use the `.format()` method. Colored lines (blue, orange, and yellow) connect the curly braces in the format strings to the corresponding arguments or expressions provided in the `.format()` call. For example, in line 4, the first brace connects to `a`, the second to `b`, and the third to `a+b`. In line 6, the braces are indexed: `{0}` connects to `a`, `{1}` to `b`, and `{2}` to `a*b`. In line 7, the braces are indexed: `{1}` connects to `a`, `{2}` to `b`, and `{0}` to `a/b`.

字串格式化 f-string


f-string 與format的差異只在於，f-string能直接在{}中填入變數或運算
使得字串變得更加清晰易讀

```
12     a = 10
13     b = 5
14     print(f"{a} + {b} = {a+b}")
15     print(f"{a} - {b} = {a-b}")
16     print(f"{a} * {b} = {a*b}")
17     print(f"{a} / {b} = {a/b}")
18
```

字串操作 (ex6)

python針對字串有提供相當多的操作功能
如以下：

- `len(your_str)` 取得字串長度
- `split()` 依照指定字元切割字串
- `replace()` 取代指定字串



```
1 a_str = "abc, 123, hi"
2 print(len(a_str))
3 print(f"astr.split(','): {a_str.split(',')}")
4 print(f"astr.split(', '): {a_str.split(', ')}")
5 print(f"astr.replace(', ', '-'): {a_str.replace(', ', '-')}")
```

字串操作

python針對字串有提供相當多的操作功能
如以下：

- strip() 移除字串兩邊指定字元
- lstrip() 移除字串左邊指定字元
- rstrip() 移除字串右邊指定字元
- count() 計算指定字元有幾個




```
9   b_str = "|||hello|||"
10  print(f"bstr.strip('|'): {b_str.strip('|')}")
11  print(f"bstr.lstrip('|'): {b_str.lstrip('|')}")
12  print(f"bstr.rstrip('|'): {b_str.rstrip('|')}")
13  print(f"bstr.count('|'): {b_str.count('|')}")
```


字串操作

python針對字串有提供相當多的操作功能
如以下：

- title() 將開頭英文字母轉成大寫，其餘小寫
- upper() 將英文字母轉成大寫
- lower() 將英文字母轉成小寫



```
17  c_str = "abCdEFg"
18  print(f"cstr.title(): {c_str.title()}")
19  print(f"cstr.upper(): {c_str.upper()}")
20  print(f"cstr.lower(): {c_str.lower()}")
```

字串操作

python針對字串有提供相當多的操作功能
如以下：

- index() 回傳指定字元位置，找不到會報錯
- find() 回傳指定字元位置，找不到會回傳-1

```
24 d_str = "ABCabc"
25 print(f"d_str.index('c'): {d_str.index('c')}")
26 print(f"d_str.find('C'): {d_str.find('C')}")
27 print(f"d_str.find('e'): {d_str.find('e')}")
28 print(f"d_str.index('e'): {d_str.index('e')}") # error
```

流程控制 - 判斷 (ex7)

流程控制是程式當中最重要的一環，為了讓電腦能做到像現實世界的判斷

舉例來說：

如果作業DL還有一個禮拜，你會去打LOL

但如果作業DL還有24小時，先別慌

除非作業DL剩1小時，找人借抄

語法：

if 條件式(bool):

 如果成立則執行 A

elif 條件式(bool):

 否則如果成立則執行 B

else:

 否則執行 C



流程控制 - 判斷 (邏輯運算)


如何決定條件式關乎到你的流程走向
邏輯運算的結果只會有 True 或 False

主要有以下幾種比較運算子：

- > 大於
- < 小於
- >= 大於等於
- <= 小於等於
- != 不等於
- == 等於

以及以下的邏輯運算子：

- and 相等
- or 或
- not 否(反轉)



```
2  a = 7
3  b = 5
4
5  print(a > b)
6  print(a < b)
7  print(a == b)
8  print(a != b)
9  print(not(a > b))
10
11 print("="*10)
12
13 print(True and True)
14 print(False and False)
15 print(True and False)
16
17 print(True or True)
18 print(False or False)
19 print(True or False)
```

流程控制 - 判斷

要特別注意，Python他是以縮排的空格數決定層級
相同縮排者，在相同層級

```
24     deadline = 1
25
26  ✓ if deadline >= 24*7 :
27      |     print("Play LOL")
28  ✓ elif (deadline < 24*7 and deadline > 24) :
29      |     print("Chill ...")
30  ✓ else :
31      |     print("copy paste")
32
```

練習時間 (p3)

練習時間 (p4)

流程控制 - for迴圈(ex8)

為什麼我們想用程式幫我們完成某些任務？

正因為有些運算是相似且重複的動作，我們希望程式幫我們快速運算出來，因此 " 迴圈 " 就能解決這件事

語法：

```
for 迭代變數 in 範圍：  
    code...
```

常見我們會將 " 範圍 " 透過range()

流程控制 - for迴圈

常見情況我們會將 " 範圍 " 透過 range(stop) 來做

```
2  print("range(5):")
3  for i in range(5):
4      print(i)
```

而range()除了上述形式外，還有range(start, stop), range(start, stop, step)

```
6  print("\nrage(1,5):")
7  ✓ for i in range(1,5):
8      print(i)
```

```
10 print("\nrage(0,5,2):")
11 ✓ for i in range(0,5,2):
12     print(i)
13
14 print("\nrage(4,-1,-1):")
15 ✓ for i in range(4,-1,-1):
16     print(i)
```

練習時間 (p5)

練習時間 (p6)

額外練習：p7~p12

流程控制 - while迴圈

當然迴圈的方式肯定不止一種
大家有注意到 for 迴圈有什麼缺點嗎？

沒錯，他必須明確知道要執行多少次，但如果我只知道結束條件呢？
這時你會需要用到 while 迴圈

語法：

while 條件式:
 code...

當條件式為True時，就會一直執行，直到code裡面有相關操作導致條件式為False，才會結束迴圈

流程控制 - while迴圈

我們也可以透過 while 迴圈做到像剛剛 for 迴圈的事情

```
21  i = 0
22  while i < 5:
23      print(i)
24      i += 1
```

當然我們也可以讓條件恆為True，透過 break 來達到結束迴圈的行為

```
26  i = 0
27  while True:
28      print(i)
29      i += 1
30      if i >= 5:
31          break
```

流程控制 - 迴圈控制

在迴圈中，可能在某些條件下，我們不想讓部分後續操作執行
想要直接跳出迴圈，或跳到下一次迴圈

此時，我們可以使用 `break` 和 `continue`，達到控制迴圈

```
38  for i in range(1, 5):
39      if i == 2:
40          continue
41      if i == 4 :
42          break
43      print(i)
```

```
47      i = 0
48  ✓ while i < 5:
49      i += 1
50  ✓      if i == 2:
51          continue
52  ✓      if i == 4 :
53          break
54      print(i)
```

資料容器 List

我們現在知道 "資料" 可以儲存在 Variable 中，但是如果今天我有 1000 筆資料，我要開 1000 個變數來儲存嗎？

這顯然沒有什麼效率，所以我們需要一個容器儲存，並利用 index 來存取這些資料

資料容器 List (*ex9*)

可以像這樣先把資料放進去，並用 index 操作

```
1  alist = [1, 2, 3, 4, 5]
2
3  print(alist)
4  print(alist[0])
5  alist[1] = 3
6  print(alist)
```

資料容器 List (*ex9*)

也可以利用 `list.append(data)` 來把資料加入該 list

```
1  alist = []
2
3  for i in range(1, 10):
4      alist.append(i * 2)
5
6  print(alist)
```

List 操作 (ex10)

List 和字串一樣提供許多操作函數

如以下：

- `len(your_list)` 取得 List 長度
- `append(data)` 新增 data 到 List 尾部
- `insert(index, data)` 插入資料至 index 的位置

```
1  alist = [1,5,4,8,9,8,10]
2
3  print("len(alist):", len(alist))
4
5  alist.append(0) #將 0 新增到alist尾部
6  print("append(0):", alist)
7
8  alist.insert(1, "hi") #在1號位置插入 "hi"
9  print('insert(1, "hi")', alist)
```

List 操作 (ex10)

List 和字串一樣提供許多操作函數

如以下：

- pop() 取出尾部資料 (回傳尾部資料後，刪除該資料)
- pop(index) (回傳 index位置資料後，刪除該資料)

```
11  print("pop()回傳:", alist.pop()) #取出尾部資料
12  print("pop():", alist)
13
14  print("pop(1)回傳:", alist.pop(1)) #取出1號位置資料
15  print("pop(1):", alist)
```

List 操作 (ex10)

List 和字串一樣提供許多操作函數

如以下：

- `max(your_list)` 回傳 List 中最大的值
- `min(your_list)` 回傳 List 中最小的值
- `count(data)` 回傳該資料在 List 中有幾個
- `remove(data)` 刪除該資料 (只會刪除一個，最左方的該資料)

```
17  print("max(alist):", max(alist))
18  print("min(alist):", min(alist))
19
20  print("count(8)回傳:", alist.count(8)) #回傳資料8在alist中有幾個
21
22  alist.remove(8) #刪除資料8
23  print("remove(8):", alist)
```

List 操作 (ex10)

List 和字串一樣提供許多操作函數

如以下：

- `index(data)` 回傳該資料的位置，找不到會報錯
- `reverse()` 反轉 List
- `sort()` 升冪排序資料
- `sort(reverse=True)` 降冪排序資料

```
25  print("index(4)回傳:", alist.index(4)) #回傳資料4的位置，找不到時報錯
26
27  alist.reverse() #也可以 alist = alist[::-1]
28  print("reverse():", alist)
29
30  alist.sort() #排序(預設升冪)
31  print("sort():", alist)
32
33  alist.sort(reverse=True) #排序(降冪)
34  print("sort(reverse=True):", alist)
```

二維 List (ex11)

List 是可以放在 List 內的，可以想像成一維資料是一條線，二維資料就是一個面，有x y座標，舉個例子：

```
1  alist = []  
2  for i in range(3):  
3      |    alist.append([3, 2, 1])
```

迭代 List (ex11)

我們可以利用 for 迴圈來取 List 的資料，這是很常見的作法

```
1  alist = []
2  for i in range(1, 10):
3      alist.append(i * 2)
4
5  for i in range(len(alist)):
6      print(f"alist[{i}]: {alist[i]}")
```


迭代 List (ex11)

二維 List 的迭代方式

```
10  blist = []
11  for i in range(5):
12      temp = []
13      for j in range(5):
14          temp.append(j * i)
15      blist.append(temp)
16
17  for i in range(len(blist)):
18      for j in range(len(blist[i])):
19          print(f"{blist[i][j]:<2} ", end="")
20  print()
```

練習時間 (p13)

Slicing

你會發現字串和 list 好像喔，都可以用 [index] 來取資料。

其實 string 概念上就是字元的容器，把它串在一起就變成字串了，但是 string 不能用 [index] 來 "寫入" 資料，List 可以。

現在我們來講進階的 "取資料" 的方式

Slicing 其實就是把資料切片，切成一段一段的方式來取資料
只要以 [start:stop:step] 的方式來取資料即可。

字符串 Slicing (ex12)

```
1  s = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
2
3  print(f"s[0]: {s[0]}")
4  print(f"s[3]: {s[3]}")
5  print(f"s[-5]: {s[-5]}")
6  print(f"s[5:]: {s[5:]}")
7  print(f"s[-5:]: {s[-5:]}")
8  print(f"s[:5]: {s[:5]}")
9  print(f"s[5::2]: {s[5::2]}")
10 print(f"s[::-1]: {s[::-1]}")
11 print(f"s[-5:-11:-1]: {s[-5:-11:-1]}")
```

List Scling (*ex13*)

```
1  alist = []
2  for i in range(10):
3      alist.append(i)
4
5  print(f"alist[0]: {alist[0]}")
6  print(f"alist[3]: {alist[3]}")
7  print(f"alist[-5]: {alist[-5]}")
8  print(f"alist[5:]: {alist[5:]}")
9  print(f"alist[-5:]: {alist[-5:]}")
10 print(f"alist[:5]: {alist[:5]}")
11 print(f"alist[5::2]: {alist[5::2]}")
12 print(f"alist[::-1]: {alist[::-1]}")
13 print(f"alist[-5:-11:-1]: {alist[-5:-11:-1]}")
```

資料容器 Dict (ex14)

Dict 是 Dictionary 的意思，這種容器結構不同於 List 是使用由 0 開始排序的 index 存取資料，而是 key value pair 的方式來存取資料，例如：

```
14  adict = {
15      "a":1,
16      "c":[1,2,3],
17      "d":{
18          "a":3,
19          "b":4
20      },
21      123:"num"
22  }
23
24  print(adict)
25  print(adict['a'])
26  print(adict['d']['b'])
```

資料容器 Dict (ex14)

當 key 存在時可以用 dict[key] = value 新增資料，若 key 存在則會覆蓋掉該 key 的 value

```
30  bdict = {}
31
32  bdict['a'] = "apple"
33  bdict['b'] = "banana"
34  bdict['c'] = "coconut"
35  print(bdict)
36
37  bdict['a'] = 'avocado'
38  print(bdict)
```

Dict 操作 (ex15)

Dict 的操作函數：

- `len(your_dict)` 回傳字典長度
- `items()` 回傳字典所有資料，結構為 `[(key,value),(key,value),(key,value)]`
- `keys()` 回傳字典所有的 Key
- `values()` 回傳字典所有的 Value

```
22  adict = {
23      "A": "Hi",
24      "B": "Hello",
25      "C": "World",
26  }
27
28  print(len(adict))
29  print(adict.items())
30  print(adict.keys())
31  print(adict.values())
```


Dict 操作 (ex15)

Dict 的操作函數：

- get(key) 回傳該 Key 對應的 Value，若無此 Key 則回傳 None

```
43  print("adict['A']:", adict['A']) #依 key:"A"，取value
44  print("adict.get('A'):", adict.get('A')) #依 key:"A"，取value
45
46  # print("adict['P']:", adict['P']) #找不到 key，報錯 KeyError
47  print("adict.get('P'):", adict.get('P')) #找不到 1ek，回傳 None
```

Dict 操作 (ex15)

Dict 的操作函數：

- `pop(key)` 回傳該 Key 對應的 Value，且會把字典內的這組資料刪除
- `pop(key, default)` 跟上面一樣，只是找不到 Key 會回傳 default

```
51  #可利用.pop(key)將資料從字典刪除
52  print("adict.pop('A'):", adict.pop('A')) #取出 key:A 的值
53  print(adict)
54
55  # print("adict.pop('P'):", adict.pop('P')) #找不到 key，報錯 KeyError
56  print("adict.pop('P'):", adict.pop('P', None)) #找不到 key，回傳 None
57  print(adict)
```

Dict 操作 (ex15)

Dict 的操作函數：

- clear() 清空字典

```
61      adict.clear()  
62      print(adict)
```

練習時間 (p14)

Function

跟數學的 函式 / 方程式 一樣，例如直線的方程式為 $y = ax+b$ ，把 x 帶入就能得到 y ，在程式中就是用 `f(arg1, arg2 arg3...)` 這樣的方式來帶入參數，等函式執行完後就能拿到回傳結果。

當然函式也可以不要有參數或回傳值，可以單純把函式當作副程式來執行，可以把複雜的邏輯拆分成抽象化的文字，讓人可以更簡單理解程式的運作流程。

你會注意到前面用的 `print`、`input` 這些都是函式，是 python 內建的 `function`。

Function (ex16)

```
1  def f():  
2      |   print("function 'f'")  
3  
4  f()
```

Function (ex17)

```
1  def f(a, b):  
2      print(f"a: {a}")  
3      print(f"b: {b}")  
4      print(f"a + b = {a+b}")  
5  
6  f(3, 8)
```

Function (ex18)

我們也可以給傳入參數設定預設值，呼叫函式的時候可以選擇是否傳入該參數，若不傳入則使用 default

```
8     def f(a, n=2):  
9         |     return a**n  
10  
11     print(f(2))  
12     print(f(2, 3))  
13     print(f(n=3, a=2))
```


Function (ex19)

同樣都是 x, y 差在哪?
alist, adict 呢?

```
1  x = 10
2  y = 100
3  alist = [1, 2, 3]
4  adict = {'a': "apple", 'b': "banana"}
5
6  def f(x = 5):
7      y = 5
8      print(f"func x: {x}, y: {y}")
9      print(alist)
10     print(adict)
11     alist[0] = 5
12     adict['b'] = "berry"
13
14  f(20)
15  print(f"global x: {x}, y: {y}")
16  print(alist)
17  print(adict)
```

練習時間 (p15)



感謝聆聽