

# Chapter 4

## 基本輸入與輸出

## 4.1 Python的輔助說明help()

- help( )函數可以列出某一個Python的指令或函數的使用說明。

```
>>> help(print)
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file: a file-like object (stream); defaults to the current sys.stdout.
    sep: string inserted between values, default a space.
    end: string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

## 4.2 格式化輸出資料使用print()

1. 使用`%`：適用Python 2.x ~ 3.x。 (非常類似 C, C++ 的用法)
2. 使用`{}`和`format()`：適用Python 2.6 ~ 3.x。 (非常類似 C#的用法)
3. 使用`f-strings`：適用Python 3.6(含)以上。

## 4.2 格式化輸出資料使用print()

- 基本語法

`print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)`

- `value`: 想輸出的資料，可一次輸出多筆資料，各資料以逗號隔開。
- `sep`: 當初出多筆資料的時候，各筆資料中間的分隔符號，預設是空格。
- `end`: 資料輸出完後所插入的字元，預設是換行字元。
- `file`: 資料輸出位置，預設是螢幕(`sys.stdout`)。也可設定輸出至檔案或設備。
- `flush`: 是否即時將緩衝區的資料輸出。

```
f = open("123.txt", "w")
print("123456789", file = f)
```

執行後開啟123.txt檔案，發現“123456789”未被寫入，檔案內容為空。只有`f.close()`後才將內容寫進檔案中。如果加入`flush = True`，即上面程式碼改為：

```
f = open("123.txt", "w")
print("123456789", file = f, flush = True)
```

不用`f.close()`即可將內容寫進檔案中

`flush`引數主要是重新整理，預設`flush = False`，不重新整理，如上面例子，`print`到`f`中的內容先存到記憶體中，當檔案物件關閉時才把內容輸出到 123.txt 中；而當`flush = True`時它會立即把內容重新整理存到 123.txt 中。

## 4.2 格式化輸出資料使用print()

- 基本語法

```
1 num1 = 123
2 num2 = 456
3 num3 = num1 + num2
4 print("數值相加為:", num3)
5 str1 = str(num1) + str(num2)
6 print("字串相加為:", str1, sep=" ~~~^~~~ ")
```

數值相加為: 579  
字串相加為: ~~~^~~~ 123456

```
1 num1 = 123
2 num2 = 456
3 num3 = num1 + num2
4 print("數值相加為:", num3, end="\t")
5 str1 = str(num1) + str(num2)
6 print("字串相加為:", str1, sep=" ~~~^~~~ ")
```

數值相加為: 579      字串相加為: ~~~^~~~ 123456

## 4.2 格式化輸出資料使用print()

- 使用 % 格式化輸出：

`print("...輸出格式區..." % (變數系列區, ...))`

- 輸出格式區中可放置變數系列區相對應的格式化字元：
  - %d：格式化整數輸出。
  - %f：格式化浮點數輸出。
  - %x：格式化16進位整數輸出。
  - %X：格式化大寫16進位整數輸出。
  - %o：格式化8進位整數輸出。
  - %s：格式化字串輸出。
  - %e：格式化科學記號e的輸出。
  - %E：格式化科學記號大寫E的輸出。

```
1 name = "Curtis"
2 score = 90
3 count = 1
4 print("%s 的第 %d 次考試成績為 %d 分." % (name, count, score))
```

Curtis 的第 1 次考試成績為 90 分.

## 4.2 格式化輸出資料使用print()

- 使用 % 格式化輸出：

```
1 name = "Curtis"  Curtis 的第 1 次考試成績為 90 分.
2 score = 90
3 count = 1
4 outString = "%s 的第 %d 次考試成績為 %d 分."
5 print(outString % (name, count, score))
```

```
1 name = "Curtis"  %s 的第 %d 次考試分數是 %d 分
2 score = 90      Curtis 的第 1 次考試分數是 90 分
3 count = 1
4 outString = "%s 的第 %d 次考試分數是 %d 分"
5 print(outString)
6 out = outString % (name, count, score)
7 print(out)      為字串填值的方式
```

```
1 x = 100
2 print("x:\n整數 %d\n浮點數 %f\n字串 %s\n" % (x, x, x))
3 y = 90.9
4 print("y:\n整數 %d\n浮點數 %f\n字串 %s\n" % (y, y, y))
```

```
x:
整數 100
浮點數 100.000000
字串 100

y:
整數 90
浮點數 90.900000
字串 90.9
```

浮點數以整數%d格式化後，  
小數資料將被捨去。

## 4.2 格式化輸出資料使用print()

- 使用 % 精準格式化輸出：
  - `%(+|-)nd`：格式化整數輸出。(n表示輸出長度)
  - `%(+|-)m.nf`：格式化浮點數輸出。(m表示包含小數點的輸出長度，n表示小數點保留位數。)
  - `%(+|-)nx`：格式化16進位整數輸出。
  - `%(+|-)no`：格式化8進位整數輸出。
  - `%(-)ns`：格式化字串輸出。
  - `%(-)m.ns`：m是輸出字串寬度，n是顯示字串長度，n小於字串長度時會有裁減字串的效果。
  - `%(+|-)e`：格式化科學記號e輸出。
  - `%(+|-)E`：格式化科學記號大寫E輸出。
- -號表示靠左對齊，+號表示數值大於0會加上"+"符號。



## 4.2 格式化輸出資料使用print()

- 使用 % 精準格式化輸出：

```

1  num_int = 100
2  print("num_int = |%6d|" % num_int)
3  num_float = -90.9
4  print("num_float = |%7.2f|" % num_float)
5  str = "NCKU"
6  print("str = |%6s|" % str)
7  print("str = |%6.3s|" % str)
8  print("以下是寬度設定不足的輸出範例")
9  print("num_int = |%2d|" % num_int)
10 print("num_float = |%3.2f|" % num_float)
11 print("str = |%2s|" % str)
12 print("以下是靠左對齊的輸出範例")
13 print("num_int = |%-6d|" % num_int)
14 print("num_float = |%-7.2f|" % num_float)
15 print("str = |%-6s|" % str)

```

```

num_int = |   100|
num_float = |-90.90|
str = |  NCKU|
str = |   NCK|

```

以下是寬度設定不足的輸出範例

```

num_int = |100|
num_float = |-90.90|
str = |NCKU|

```

以下是靠左對齊的輸出範例

```

num_int = |100   |
num_float = |-90.90 |
str = |NCKU   |

```

## 4.2 格式化輸出資料使用print()

- {} 和 format() 方法

print(" ...輸出格式區... ".format( 變數系列區, ... ))

- 增強版的格式化輸出功能，字串使用format的方法做格式化。
- 輸出格式區使用{}表示變數。

```
name = "Curtis"
score = 90
count = 1
print("{0} 的第 {1} 次考試成績為 {2} 分".format(name, count, score))
```

Curtis 的第 1 次考試成績為 90 分.

- {} 內可帶從0開始的編號，表示在format() 中變數的順序。

```
name = "Curtis"
score = 90
count = 1
print("{0} 的第 {1} 次考試成績為 {2} 分".format(name, count, score ))

#對，但不鼓勵使用
print("{2} 的第 {1} 次考試成績為 {0} 分".format(score, count, name))
print("{1} 的第 {2} 次考試成績為 {0} 分".format(score, name, count))
```

## 4.2 格式化輸出資料使用print()

- {} 和 format() 函數
  - 在 {} 內使用具名變數

```
1 name = "Curtis"
2 score = 90
3 count = 1
4 print("{n} 的第 {c} 次考試成績為 {s} 分".format(n=name, c=count, s=score))
5
6 #對，但不鼓勵使用
7 print("{n} 的第 {c} 次考試成績為 {s} 分".format(c=count, s=score, n=name))
```

- 也可使用精準化格式設定

```
1 name = "Curtis"
2 score = 90
3 count = 1
4 print("{:7.4s} 的第 {:2d} 次考試成績為 {:3d} 分".format(name, count, score))
```

Curt    的第   1 次考試成績為   90 分

## 4.2 格式化輸出資料使用print()

- {} 和 format() 函數

- 精準化格式設定的對齊方式：靠左(<)、靠右(>)、置中(^)。

```
1 name = "Curtis"
2 score = 90
3 count = 1
4 print("{:7.4s} 的第 {:2d} 次考試成績為 {:3d} 分".format(name, count, score))
5 print("{:>7.4s} 的第 {:<2d} 次考試成績為 {:>3d} 分".format(name, count, score))
6 print("{:^7.4s} 的第 {:^2d} 次考試成績為 {:^3d} 分".format(name, count, score))
```

```
Curt   的第  1 次考試成績為 90 分
Curt  的第 1 次考試成績為 90 分
Curt   的第 1 次考試成績為 90 分
```

% 的預設是靠右對齊

- 填充字元：放在對齊字元或指定寬度前。

```
name = "Curtis"
score = 95
count = 1
print("{:7.4s} 的第 {:2d} 次考試成績為 {:3d} 分".format(name, count, score))
print("{:_>7.4s} 的第 {:_<2d} 次考試成績為 {:_>3d} 分".format(name, count, score))
print("{:_^7.4s} 的第 {:_<2d} 次考試成績為 {:_>3d} 分".format(name, count, score))
```

```
Curt   的第  1 次考試成績為 95 分
__Curt  的第 1_ 次考試成績為 _95 分
_Curt__  的第 1_ 次考試成績為 95_ 分
```

```
Traceback (most recent call last):
  File "e:\PythonTest\test.py", line 4, in <module>
    print("{:_7.4s} 的第 {:^2d} 次考試成績為 {:3d} 分".format(name, count, score))
ValueError: Invalid format specifier

1 name = "Curtis"
2 score = 90
3 count = 1
4 print("{:_7.4s} 的第 {:^2d} 次考試成績為 {:3d} 分".format(name, count, score))
5 print("{:_7.4s} 的第 {:_2d} 次考試成績為 {:3d} 分".format(name, count, score))
```

## 4.2 格式化輸出資料使用print()

- {} 和 format() 的優點
  - 實現網路爬蟲時：
    - "https://maps.apis.com/json?city=taipei&radius=1000&type=school"

```
1  str_url = "https://maps.apis.com/json?city=taipei&radius=1000&type=school"
2  print(str_url)
3  url_head = "https://maps.apis.com/json?"
4  city = "taipei"
5  radius = 1000
6  type = "school"
7  url_all = url_head + "city=" + city + "&radius=" + str(radius) + "&type=" + type
8  print(url_all)
9
10 url_all1 = url_head + "city={}&radius={}&type={}".format(city, radius, type)
11 print(url_all1)
```

```
https://maps.apis.com/json?city=taipei&radius=1000&type=school
https://maps.apis.com/json?city=taipei&radius=1000&type=school
https://maps.apis.com/json?city=taipei&radius=1000&type=school
```

## 4.2 格式化輸出資料使用print()

- f-strings 格式化字串
  - Python 3.6 以後提供。
  - 以 f 當字首，在 {} 內放變數名稱與運算式。
  - 不會有空的 {} 或是{0}, {1} 等指定變數順序的 {}。
  - 之後都會盡量用f-strings為字串格式化的方式。

```
1 city = "Taipei"      Taipei 是 Taiwan 的一個城市
2 country = "Taiwan"
3 print(f"{city} 是 {country} 的一個城市")
```

```
1 name = "Curtis"      Curt 的第 1 次考試成績為 90.50 分
2 score = 90.5
3 count = 1
4 print(f"{name:7.4}的第 {count:^5d} 次考試成績為 {score:6.2f} 分")
```

## 4.2 格式化輸出資料使用print()

- f-strings 格式化字串
  - Python 3.8 以後新增了在 {} 內增加"="符號，可以輸出變數名稱和它的值。

```
1 url_head = "https://maps.apis.com/json?"
2 city = "taipei"
3 radius = 1000
4 type = "school"
5
6 print(f"{url_head}city={city}&radius={radius}&type={type}")
7 print(f"{url_head}{city=}&{radius=}&{type=}")
8 print(f"{url_head}{city=:.6}&{radius=}&{type=:.6}")
```

```
https://maps.apis.com/json?city=taipei&radius=1000&type=school
https://maps.apis.com/json?city='taipei'&radius=1000&type='school'
https://maps.apis.com/json?city=taipei&radius=1000&type=school
```

## 4.3 輸出資料到檔案

- `open()` 函數：

`file_obj = open(file, mode = "rt")`

- `file`: 用字串列出檔案檔名，若不指定路徑則開啟在目前工作資料夾。
- `mode`: 開啟檔案的模式

第一個字母可為：

- `"r"`: 預設值，開啟檔案僅供讀取(read only)
- `"w"`: 開啟檔案供寫入，如原先檔案存在則會被複寫。
- `"a"`: 開啟檔案供寫入，如原先檔案存在則會將新資料附加在後面。
- `"x"`: 開啟一個新檔案供寫入，若原先檔案存在則會產生錯誤。

第二個字母可為：


- `"b"`: 開啟二進位檔案模式。
- `"t"`: 開啟文字檔案模式，為預設值。

- `file_obj`: 檔案物件，`print()`函數可將輸出導向此物件。不使用時，用`file_obj.close()`將檔案物件關閉。



## 4.3 輸出資料到檔案

```
1 fstream1 = open("E:/PythonTest/text.txt", mode="w")
2 print("Testing string to fstream1", file=fstream1)
3 fstream1.close()
4 fstream2 = open("E:/PythonTest/text.txt", mode="a")
5 print("Testing string to fstream2", file=fstream2)
6 fstream2.close()
```

```
7  text.txt - 記事本
8
9 檔案(F) 編輯(E) 格式(O) 檢視(V) 說明
10 Testing string to fstream1
11 Testing string to fstream2
```

## 4.4 資料輸入

- `value = input("prompt:")`
  - 與`print`相反，會從鍵盤讀取使用者輸入的資料。
  - `value` 是變數，從 `input()`讀進來的結果一律是字串資料。若要做數字運算則要轉換成整數或浮點數。

```
1 name = input("請輸入姓名：")
2 score_math = input("請輸入數學期末考成績：")
3 score_eng = input("請輸入英文期末考成績：")
4 score_ch = input("請輸入中文平均成績：")
5 score_avg = (int(score_math) + int(score_eng) + float(score_ch))/3
6 print(f"平均成績為：{score_avg}")
```

```
請輸入姓名：Curtis
請輸入數學期末考成績：90
請輸入英文期末考成績：80
請輸入中文平均成績：75.5
平均成績為：81.83333333333333
```

## 4.5 處理字串的數學運算

- `eval()` 函數：

`result = eval(expression)`

- `expression`: 公式運算字串
- `result`: 運算結果

```
1 equation = input("請輸入計算式：")
2 result = eval(equation)
3 print(f"結果為：{result}")
```

請輸入計算式：9\*3+5  
結果為：32

請輸入計算式：3+3\*\*3\*3/3  
結果為：30.0

- `eval()` 搭配 `input()` 可直接取得數值資料。
- 可用多重指定在 `eval()` 與 `input()` 上，產生一行輸入多個數值的效果。

```
1 score_math, score_eng, score_ch = eval(input("請輸入3科期末考成績："))
2 score_avg = (score_math + score_eng + score_ch)/3
3 print(f"平均成績為：{score_avg}")
```

請輸入3科期末考成績：90, 85, 83  
平均成績為：86.0

請輸入3科期末考成績：90.5, 90, 91  
平均成績為：90.5

輸入時各數字間要用","隔開

## 4.6 列出 Python 所有內建函數

- `dir()` 函數：  
`dir(__builtins__)`

```
>>> dir(__builtins__)
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'BlockingIOError', 'BrokenPipeError', 'BufferError', 'BytesWarning', 'ChildProcessError', 'ConnectionAbortedError', 'ConnectionError', 'ConnectionRefusedError', 'ConnectionResetError', 'DeprecationWarning', 'EOFError', 'Ellipsis', 'EnvironmentError', 'Exception', 'False', 'FileExistsError', 'FileNotFoundError', 'FloatingPointError', 'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError', 'ImportWarning', 'IndentationError', 'IndexError', 'InterruptedError', 'IsADirectoryError', 'KeyError', 'KeyboardInterrupt', 'LookupError', 'MemoryError', 'ModuleNotFoundError', 'NameError', 'None', 'NotADirectoryError', 'NotImplemented', 'NotImplementedError', 'OSError', 'OverflowError', 'PendingDeprecationWarning', 'PermissionError', 'ProcessLookupError', 'RecursionError', 'ReferenceError', 'ResourceWarning', 'RuntimeError', 'RuntimeWarning', 'StopAsyncIteration', 'StopIteration', 'SyntaxError', 'SyntaxWarning', 'SystemError', 'SystemExit', 'TabError', 'TimeoutError', 'True', 'TypeError', 'UnboundLocalError', 'UnicodeDecodeError', 'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError', 'UnicodeWarning', 'UserWarning', 'ValueError', 'Warning', 'WindowsError', 'ZeroDivisionError', '__build_class__', '__debug__', '__doc__', '__import__', '__loader__', '__name__', '__package__', '__spec__', 'abs', 'all', 'any', 'ascii', 'bin', 'bool', 'breakpoint', 'bytearray', 'bytes', 'callable', 'chr', 'classmethod', 'compile', 'complex', 'copyright', 'credits', 'delattr', 'dict', 'dir', 'divmod', 'enumerate', 'eval', 'exec', 'exit', 'filter', 'float', 'format', 'frozenset', 'getattr', 'globals', 'hasattr', 'hash', 'help', 'hex', 'id', 'input', 'int', 'isinstance', 'issubclass', 'iter', 'len', 'license', 'list', 'locals', 'map', 'max', 'memoryview', 'min', 'next', 'object', 'oct', 'open', 'ord', 'pow', 'print', 'property', 'quit', 'range', 'repr', 'reversed', 'round', 'set', 'setattr', 'slice', 'sorted', 'staticmethod', 'od', 'str', 'sum', 'super', 'tuple', 'type', 'vars', 'zip']
```

## 4.7 動手練習

- 輸入攝氏溫度，換算成華氏溫度後輸出。
  - 攝氏溫度 = ( 華氏溫度 - 32 ) \* 5 / 9
  - 華氏溫度 = 攝氏溫度 \* ( 9 / 5 ) + 32

- 練習 f-string 的用法：等比數列及和

- 輸入首項( $a$ )、公比( $r$ )，印出此等比數列的前四項( $n=4$ )及其和。
- 例如：

- 首項 = 2

- 公比 = 3

- 輸出畫面為：

```
首項 = 2
公比 = 3
2 + 6 + 18 + 54 = 80.0
```

等比數列求和的公式如下：
$$S_n = \frac{a(1-r^n)}{1-r}$$

- 加上輸出位數跟靠左靠右對齊