

Python程式設計

林奇賦 daky1983@gmail.com

Outline

- 字串處理

字串處理

- 脫逸字元
 - \' 跳脫單引號
 - \" 跳脫雙引號
 - \\ 跳脫反斜線
 - \n 換行
 - \t ASCII水平Tab字元
 - \b ASCII退格字元

字串處理

- 字串型態的切片 (Slice)

- Ex:

```
>>> toast="PYTHONSLICE"  
>>> toast[-5]          ... -3 -2 -1  
'S'  
>>> toast[-5:]  
'SLICE'  
>>> toast[:6]  
'PYTHON'  
>>> toast[:5]+toast[5:]  
'PYTHONSLICE'
```

- Ex:

```
>>> 'PYTHONSLICE'[:6]==toast[:6]  
True
```

字串處理

- 元組型態的切片

- Ex:

```
>>> toast="PYTHONSLICE"  
>>> tuples=toast[0:3],toast[3:6],toast[6:9],toast[9:11]  
>>> tuples  
(('PYT', 'HON', 'SLI', 'CE'))  
>>> tuples[0]  
(('PYT', 'HON', 'SLI', 'CE'))  
>>> tuples[2:4]  
(('SLI', 'CE'))  
>>> tuples[1][0]  
(('HON', 'SLI', 'CE'))
```

- Ex:

```
>>>  
("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")[3]  
(('Wednesday', 'Thursday', 'Friday', 'Saturday'))
```

字串處理

- 序列型態的切片

- Ex:

```
>>>
["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"][random.randint(0,6)]
'Saturday'
```

- Ex:

```
>>>
days=["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"]
>>> days[2]= '星期二'
>>> print(days[2])
星期二
```

字串處理

- 字典型態的切片

- Ex:

```
>>>
```

```
days={1:"Sunday",2:"Monday",3:"Tuesday",4:"Wednesday",5:"Thursday",6:"Friday",7:"Saturday"}
```

```
>>> days[2],days[3],days[4]
```

```
('Monday', 'Tuesday', 'Wednesday')
```

- 數字型態的切片

- 必須先用`str()`函數轉換成字串

字串處理

- 字串函數處理與格式化
 - 旗標(Flag)指定格式化的字串變數
 - Ex:

```
>>> print("會員編號1:%d, 會員編號2:%d" % (10, 20))
```

```
會員編號1:10, 會員編號2:20
```

```
>>> print("會員編號2:%(#2)d, 會員編號1:%(#1)d"%{"#1":10,  
"#2":20})
```

```
會員編號2:20, 會員編號1:10
```

```
>>> print("會員編號1:%(num1)d, 會員編號  
2:%(num2)d"%{"num1":10, "num2":20})
```

```
會員編號1:10, 會員編號2:20
```


字串處理

- 字串函數處理與格式化

- 補0

- Ex:

```
>>> print("會員編號:%(#)08d" % {"#" : 123456})  
會員編號:00123456
```

- Ex:

```
>>> print("%8.2f" % (123.456))  
123.46
```

- Ex:

```
>>> money=987.98  
>>> print("$%*.2f" % (7, money))  
$ 987.98
```

字串處理

- 字串函數處理與格式化

- 輸出格式不只接受單純的數字和字串型態的變數，也可帶入整個字典型態變數

- Ex:

```
>>> name={"game":"xbox", "apple":"iphone", "camera":"nikon"}  
>>> print("%(apple)s, %(camera)s, %(game)s" %name)  
iphone, nikon, xbox
```

方法	描述
<code>str.capitalize()</code>	回傳將 str 改成首字母大寫，其餘字母小寫的字串
<code>str.center(width[, fillchar])</code>	回傳一個將 str 設置字串中央，長度 width 的新字串， fillchar 為填充字元，預設為空格
<code>str.count(sub[, start[, end]])</code>	計算 sub 出現的次數， start 為起始計算索引值， end 為結束索引值
<code>str.encode(encoding="utf-8", errors="strict")</code>	回傳 encoding 版本的 bytes 物件
<code>str.endswith(suffix[, start[, end]])</code>	判斷 str 是否以 suffix 結尾
<code>str.expandtabs([tabsize])</code>	將 tab 符號以 tabsize 的空格數替換
<code>str.find(sub[, start[, end]])</code>	回傳 sub 第一次出現的索引值
<code>str.format(*args, **kwargs)</code>	進行格式化字串運算
<code>str.index(sub[, start[, end]])</code>	回傳 sub 第一次出現的索引值
<code>str.isalnum()</code>	判斷字串中的字元是否至少一個是字母或數字
<code>str.isalpha()</code>	判斷字串中的字元是否至少一個是字母
<code>str.isdecimal()</code>	判斷字串中所有字元是否是十進位數字
<code>str.isdigit()</code>	判斷字串中所有字元是否是數字
<code>str.isidentifier()</code>	判斷字串是否可作為合法的識別字
<code>str.islower()</code>	判斷字串中所有字母字元是否都是小寫字母
<code>str.isnumeric()</code>	判斷字串中所有字元是否是數字
<code>str.isprintable()</code>	判斷字串中所有字元是否都屬於可見字元

方法	描述
<code>str.isspace()</code>	判斷字串是否為空格字元
<code>str.istitle()</code>	判斷字串是否適合當作標題
<code>str.isupper()</code>	判斷字串中所有字母字元是否都是大寫字母
<code>str.join(iterable)</code>	回傳將 <code>str</code> 連結 <code>iterable</code> 各元素的字串
<code>str.ljust(width[, fillchar])</code>	回傳將 <code>str</code> 在寬度 <code>width</code> 向左對齊的字串， <code>fillchar</code> 為填充字元，預設為空格
<code>str.lower()</code>	將 <code>str</code> 的英文字母都改成小寫
<code>str.lstrip([chars])</code>	回傳將 <code>str</code> 左邊具有 <code>chars</code> 字元去除的拷貝版本， <code>chars</code> 預設為空格符號
<code>static str.maketrans(x[, y[, z]])</code>	回傳 <code>x</code> 與 <code>y</code> 配對的 Unicode 編碼字典，若有提供 <code>z</code> ， <code>z</code> 中的字元會跟 <code>None</code> 配對
<code>str.partition(sep)</code>	以 <code>sep</code> 分割 <code>str</code> 為三個部份，結果回傳具有三個子字串的序對
<code>str.replace(old, new[, count])</code>	將 <code>str</code> 中的 <code>old</code> 子字串以 <code>new</code> 代換
<code>str.rfind(sub[, start[, end]])</code>	尋找最右邊的 <code>sub</code> ，也就是索引值最大的 <code>sub</code>
<code>str.rindex(sub[, start[, end]])</code>	尋找最右邊的 <code>sub</code> ，也就是索引值最大的 <code>sub</code>
<code>str.rjust(width[, fillchar])</code>	回傳將 <code>str</code> 在寬度 <code>width</code> 向右對齊的字串， <code>fillchar</code> 為填充字元，預設為空格

方法	描述
<code>str.rpartition(sep)</code>	以 <code>sep</code> 從最右端分割 <code>str</code> 為三個部份，結果回傳具有三個子字串的序對
<code>str.rsplit([sep[, maxsplit]])</code>	將 <code>str</code> 從最右端以 <code>sep</code> 分割成子字串，回傳儲存子字串的串列， <code>maxsplit</code> 為子字串最多的數量
<code>str.rstrip([chars])</code>	從 <code>str</code> 的最右端中移除 <code>chars</code> 字元，預設為空白字元
<code>str.split([sep[, maxsplit]])</code>	將 <code>str</code> 以 <code>sep</code> 分割成子字串，回傳儲存子字串的串列， <code>maxsplit</code> 為子字串最多的數量
<code>str.splitlines([keepends])</code>	將 <code>str</code> 以新行符號分割成子字串，回傳儲存子字串的串列
<code>str.startswith(prefix[, start[, end]])</code>	判斷 <code>str</code> 是否以 <code>prefix</code> 開頭
<code>str.strip([chars])</code>	從 <code>str</code> 中移除 <code>chars</code> 字元，預設為空白字元
<code>str.swapcase()</code>	將 <code>str</code> 中的英文字母進行大小寫轉換
<code>str.title()</code>	將 <code>str</code> 轉換成作為標題的字串
<code>str.translate(map)</code>	將 <code>str</code> 中的字元以 <code>map</code> 中配對的字元轉換
<code>str.upper()</code>	將 <code>str</code> 的英文字母都改成大寫
<code>str.zfill(width)</code>	回傳以 <code>0</code> 塞滿 <code>width</code> 的新字串
<code>str.rpartition(sep)</code>	以 <code>sep</code> 從最右端分割 <code>str</code> 為三個部份，結果回傳具有三個子字串的序對

字串函數使用

- 字串函數使用

- `string.capitalize()` 函數

- 變數第一個字轉變為大寫

- Ex:

```
>>> print("how are you?".capitalize())
```

```
How are you?
```

字串函數使用

- 字串函數使用
 - `string.center(width)`函數
 - `width`引數決定對齊的總長度
 - Ex:

```
>>> text1="first line....."
```

```
>>> text1.center(50)
```

```
'          first line.....          '
```

字串函數使用

- 字串函數使用

- `string.count(sub[, start[, end]])`

- 回傳此字串裡有多少個sub引數字元

- Ex:

```
>>> text='abbggccdeefgggijklgglmo'
```

```
>>> text.count('g')
```

```
7
```

```
>>> text.count('g',4,-4)
```

```
5
```


字串函數使用

- 字串函數使用

- `str.endswith(suffix[, start[, end]])`

- 判斷字串內是否有符合`suffix`引數的值

- Ex:

```
>>> images="xbox.gif, iphone.jpg"
```

```
>>> images.endswith(".jpg")
```

```
True
```

```
>>> images.endswith(".gif",0, 8)
```

```
True
```

```
>>> images.endswith(".gif")
```

```
False
```

字串函數使用

- 字串函數使用

- `string.find(s, sub[, start[, end]])`

- 搜尋字串變數裡符合sub引數的字元位置

- Ex:

```
>>> text='abcdefgabcdefg'
```

```
>>> text.find('a')
```

```
0
```

```
>>> text.find('a',1)
```

```
7
```

字串函數使用

- 字串函數使用

- `str.format(format_string, *args, **kwargs)`

- 將輸入的`format_string`引數變數進行格式化
 - 不支援旗標的格式
 - Ex:

- ```
>>> "{o} makes a full man, and {1} an exact man.".format("Reading", "writing")
```

- ```
'Reading makes a full man, and writing an exact man.'
```

- 沒有強調限制 "{ }" 符號內的名稱一定要數字
 - Ex:

- ```
{a}...{b}....format(a="Reading", b="writing")
```

- 允許對參照關係符號定義寬度，長度不夠會自動填滿

# 字串處理

- 字串函數使用

- `string.index(s, sub[, start[, end]])`

- 與`string.find()`類似，差異在當s字串變數內搜尋不到sub字串會回傳`ValueError`錯誤訊息

- Ex:

```
>>> text="abcdeabcde"
```

```
>>> text.index('d', 4)
```

```
8
```

# 字串函數使用

- 字串函數使用
  - `str.isalnum()`
    - 判斷該變數裡的內容是否為[a-z]、[A-Z]與[0-9]的字元
    - 不可以判別多行宣告
  - `str.isalpha()`
    - 與`str.isalnum()`的差異在於這個函數只接受字串內有英文字母

# 字串函數使用

- 字串函數使用
  - `str.isdigit()`
    - 判斷字串內的數字
  - `str.islower()`
    - 判斷字串變數內的字元是否全部都是小寫
  - `str.isspace()`
    - 判斷字串變數是否為空白字元

# 課堂練習

- 使用者可以輸入任意數字 $n$
- 當輸入的 $n$ 不為數字，提示使用者輸入型態錯誤，並且重新讓使用者繼續輸入
- 若輸入的值為數字，將其`print`至螢幕上
- `ex.`
  - `n=100`

# 字串函數使用

- 字串函數使用
  - `str.istitle()`
    - 判斷字串變數裡的第一個字是否為大寫
    - 如果宣告一句英文句子，句子裡的每一個單字都會判斷
  - `str.isupper()`
    - 判斷字串變數內的所有字母都必須要大寫
    - 不會理會特殊字元



# 字串函數使用

- 字串函數使用

- `string.ljust(s, width)`

- 將傳入的s字串進行向左對齊，width引數是指定對齊的總寬度

- Ex:

```
>>> text="abcdefghijkl"
```

```
>>> text.ljust(20)
```

```
'abcdefghijkl '
```

# 字串函數使用

- 字串函數使用
  - `string.lower()`
    - 將`string`內的字元從大寫字母轉換為小寫字母
  - `str.replace(old, new, count)`
    - 將字串內所有符合`old`引數以`new`引數的字元來替代，而`count`引數是指定只要代替的數目
  - `string.rfind(s, sub[,start[, end]])`
    - 從右到左尋找，`sub`引數是預計要搜尋的字元

# 字串函數使用

- 字串函數使用

- `string.lstrip(s[, chars])`

- 將s字串變數內左邊的多於空白字元去掉，`chars`引數必須傳入字串型態
    - `chars`決定`string.lstrip()`函數要去掉x字串變數內的那些字元，預設只會刪去空白字元
    - Python 2.2版是不能使用`chars`引數
    - Ex:

```
>>> text = " aaaaa bbbbbb aaa ccccc"
>>> text.lstrip(" a")
'bbbbbb aaa ccccc'
>>> text.lstrip(" ab")
'cccccc'
```

# 字串函數使用

- 字串函數使用

- `str.partition(sep)`

- 將字串做分割，但只會分割第一個符合`sep`引數的字元，形成 3-tuple
    - Python 2.5板新增的功能
    - Ex:

```
>>> "C:\\|D:\\|E:\\|G:\\".partition('|')
('C:\\', '|', 'D:\\|E:\\|G:\\')
>>> "C:\\|D:\\|E:\\|G:\\".partition('|')[0]
'C:\\'
>>> "C:\\|D:\\|E:\\|G:\\".partition('|')[1]
'|'
>>> "C:\\|D:\\|E:\\|G:\\".partition('|')[-1]
'D:\\|E:\\|G:\\'
```

# 字串函數使用

- 字串函數使用
  - `string.split( sep, maxsplit)`
    - 由左至右，將`string`字串變數內的字元以`sep`引數字元為分隔字元進行分割
    - 找不到符合`sep`的值，就會回傳整個字串

# 字串函數使用

- 字串函數使用
  - `str.splitlines(keepends)`
    - 將字串進行分割
    - 以“\n”和“\r”作為分割的區隔字元
    - 以序列型態回傳
    - `Keepends`引數預設`False`，設為`True`會連同脫逸字元一併回傳
  - `str.startswith(prefix[, start[, end]])`
    - 判斷傳入的`prefix`字串字元是否為開始字元

# 字串函數使用

- 字串函數使用
  - `string.strip([ chars ])`
    - 將string字串變數裡的左右兩邊的空白字元刪除掉
    - `chars`引數不為None時會決定`string.strip()`函數要刪除的字元
  - `string.swapcase()`
    - 將string字串裡的字母大小寫互轉

# 字串函數使用

- 字串函數使用
  - `string.rjust(width)`
    - 與`string.ljust()`有相反的意思
  - `str.rpartition(sep)`
    - 與`string.partition()`類似
  - `string.rsplit(s[, sep[, maxsplit]])`
    - 將字串變數s裡面的值進行分割，分割的參考字元是sep引數裡的字元，其結果以序列型態儲存



# 字串函數使用

- 字串函數使用
  - `string.rindex(s, sub[, start[, end]])`
    - 由右至左搜尋，`s`字串變數搜尋不到`sub`字串將會回傳 `ValueError`錯誤訊息
  - `string.rstrip(s[, chars])`
    - 將`x`字串變數內右邊的多餘空白字元去掉

# 字串函數使用

- 字串函數使用
  - `string.title()`
    - 將字串內所有為[a-z]的單字第一個字元轉換成大寫
  - `string.translate(map)`
    - 將 `string` 中的字元以 `map` 中配對的字元轉換
    - 搭配 `str.maketrans(from, to)`

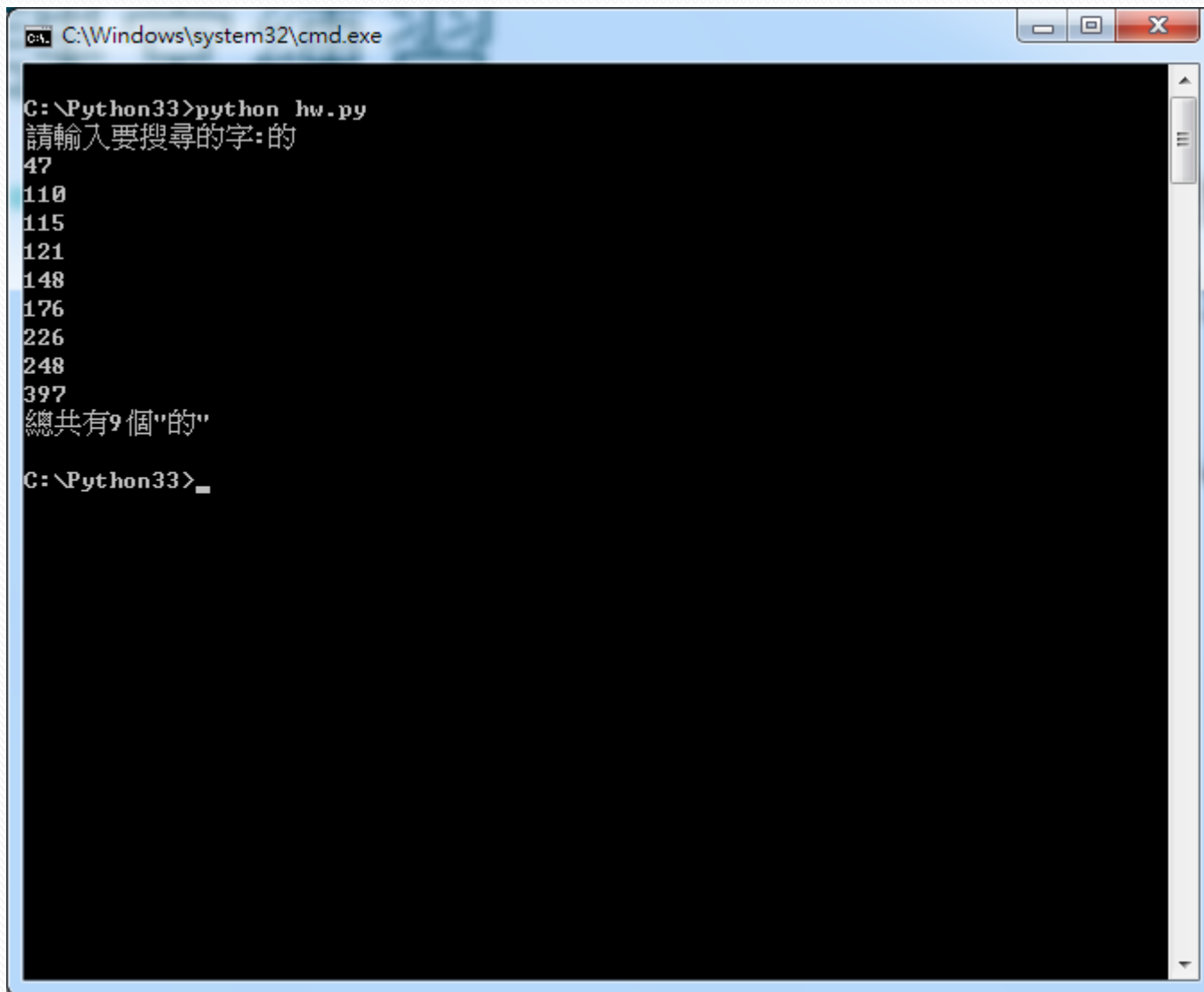
# 字串函數使用

- 字串函數使用
  - `string.upper()`
    - 將string字串變數內的字母從小寫轉換為大寫
  - `string.zfill(width)`
    - 將string變數內的字串前面補0，直到string變數的長度等於width引數設定的長度

# 課堂練習

- 題目: 在特定的文章字串中，搜尋輸入的字串
  - 條件1: 若有符合的字串，將其索引值印出 (全部印出，並非印出第一個符合的索引值)
  - 條件2: 最後印出 總共有9個"的", (若輸入的字為"的")

# 範例圖



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a blue title bar and standard Windows window controls (minimize, maximize, close). The command prompt shows the following text:

```
C:\Python33>python hw.py
請輸入要搜尋的字:的
47
110
115
121
148
176
226
248
397
總共有9個"的"
C:\Python33>_
```

The output shows a list of numbers followed by a summary statement in Chinese. The prompt "請輸入要搜尋的字:" (Please enter the word to search for:) is followed by the character "的" (de). The output lists the following numbers: 47, 110, 115, 121, 148, 176, 226, 248, and 397. The final line of output is "總共有9個"的" (There are a total of 9 occurrences of "de").