蟒蛇
Python 3.9
String

基礎程式設計

Fundamental Programming

-- Python 3.9 String

Yung-Chen Chou Ph.D.
iSchool, Feng Chia University
Aug. 20, 2021

字串 (String)

- 使用單引號「'」與雙引號「"」所包夾的文字，在Python 會被視為字串
  - 字串內文字可以儲存Unicode 編碼的文字，支援中文
- 單引號內使用雙引號，可以正確顯示雙引號，雙引號內也可以使用單引號，也可以正確顯示單引
- 使用 3 個單引號或 3 個雙引號 可用來顯示多行文字，且每行前面的空白也會正常顯示，換行字元也會被保留

```
>>> str1 = '人工智慧學位學程'
>>> str2 = " ' 創能學院' in 'FCU' "
>>> str3 = '''  逢甲大學
...   創能學院
...   人工智慧學位學程  '''
>>> print( str1 )
人工智慧學位學程
>>> print( str2 )
 ' 創能學院' in 'FCU'
>>> print( str3 )
  逢甲大學
  創能學院
  人工智慧學位學程
```

字串運算子 (String operator)

- 串接字串:
  - 使用「 + 」將兩個字串併接起來
- 複製字串:
  - 使用「 * 」對字串進行複製
- 存取字串:
  - 使用「 [ ] 」取出字串元素, 例 str1[0] 可取出 str1 字串中的第 1 個元素
- 切割字串:
  - 使用「 [開始 : 結束 : 間隔] 」進行字串切割, 從「開始」字元到「結束」字元的前一個(即不包含結束的字元), 每隔「間隔」個字元取一個字元出來

```
>>> str1 = "Feng Chia "
>>> str2 = "University "
>>> str3 = str1 + str2
>>> print( str3 )
Feng Chia University
```

```
>>> str4 = str1 * 3
>>> print( str4 )
Feng Chia Feng Chia Feng Chia
```

```
>>> str5 = "abcdefghijklmnopqrstuvwxyz"
>>> print( str5[0] )
a
>>> print( str5[25] )
z
>>> print( str5[-1] )
z
```

```
>>> print( str5[:] )
abcdefghijklmnopqrstuvwxyz
>>> print( str5[12:] )
mnopqrstuvwxyz
>>> print( str5[:12] )
abcdefghijkl
>>> print( str5[8:12] )
ijkl
>>> print( str5[8:12:2] )
ik
>>> print( str5[8:-3:3] )
iloru
>>> print( str5[::-1] )
zyxwvutsrqponmlkjihgfedcba
>>> print( str5[-1::-1] )
zyxwvutsrqponmlkjihgfedcba
```

字串運算子 (String operator)

- 串接多行
  - 使用「\」串接多行字串
  - Python 程式中同一行的程式碼過長，在該行最後加上 \ 字元，就可以繼續於下行續寫程式，如些來就會被 Python 視為是同一行程式

```
>>> str6 = '逢甲大學
  File "<stdin>", line 1
    str6 = '逢甲大學
                 ^
SyntaxError: EOL while scanning string literal
>>> str6 = '逢甲大學 \
... 創能學院 \
... 人工智慧學位學程'
>>> print( str6 )
逢甲大學 創能學院 人工智慧學位學程
```

未加 \ 字元，所以出現錯誤訊息

```
>>> print( str1, str2, \
... str3, str5)
Feng Chia  University  Feng Chia University  abcdefghijklmnopqrstuvwxyz
```

字串內建函式 (Build-in function)

- 字串切割: 字串.split(切割字元)

```
split(self, /, sep=None, maxsplit=-1)
    Return a list of the words in the string, using sep as the delimiter string.

    sep
      The delimiter according which to split the string.
      None (the default value) means split according to any whitespace,
      and discard empty strings from the result.
    maxsplit
      Maximum number of splits to do.
      -1 (the default value) means no limit.
```

```
>>> str7 = 'Taiwan,Asia,America,Japan'
>>> list7 = str7.split(',')
>>> print( list7 )
['Taiwan', 'Asia', 'America', 'Japan']
>>> print( list7[2] )
America
```

字串內建函式 (Build-in function)

- 連結字串: 連結字串.join(要串接的串列)

```
join(self, iterable, /)
    Concatenate any number of strings.

    The string whose method is called is inserted in between each given string.
    The result is returned as a new string.

    Example: '.'.join(['ab', 'pq', 'rs']) -> 'ab.pq.rs'
```

```
>>> list7
['Taiwan', 'Asia', 'America', 'Japan']
>>> str8 = '-'.join( list7 )
>>> print( str8 )
Taiwan-Asia-America-Japan
```

字串內建函式 (Build-in function)

- 字串取代: 字串.replace(原始字串, 取代字串)

```
replace(self, old, new, count=-1, /)
    Return a copy with all occurrences of substring old replaced by new.

      count
        Maximum number of occurrences to replace.
        -1 (the default value) means replace all occurrences.

    If the optional argument count is given, only the first count occurrences are
    replaced.
```

```
>>> str8
'Taiwan-Asia-America-Japan'
>>> str9 = str8.replace( '-' , ' # ' )
>>> print( str9 )
Taiwan # Asia # America # Japan
```

字串內建函式 (Build-in function)

- 字串尋找: 字串.find(要找的字串)

```
find(...)
    S.find(sub[, start[, end]]) -> int

    Return the lowest index in S where substring sub is found,
    such that sub is contained within S[start:end].  Optional
    arguments start and end are interpreted as in slice notation.

    Return -1 on failure.
```

```
>>> str7
'Taiwan,Asia,America,Japan'
>>> str7.find('Ame')
12
>>> str7.find('i')
2
>>> str7.find('x')
-1
```

字串內建函式 (Build-in function)

- 從右側開始尋找第一個符合的字串位置
  - 字串.rfind(要找的字串)

```
rfind(...)
    S.rfind(sub[, start[, end]]) -> int

    Return the highest index in S where substring sub is found,
    such that sub is contained within S[start:end].  Optional
    arguments start and end are interpreted as in slice notation.

    Return -1 on failure.
```

```
>>> str7
'Taiwan,Asia,America,Japan'
>>> str7.rfind('Ame')
12
>>> str7.rfind('i')
16
>>> str7.rfind('x')
-1
```

字串內建函式 (Build-in function)

- 判斷字串首是否符合特定字串
  - 字串.startswith(要找的字串)

```
startswith(...)
    S.startswith(prefix[, start[, end]]) -> bool

    Return True if S starts with the specified prefix, False otherwise.
    With optional start, test S beginning at that position.
    With optional end, stop comparing S at that position.
    prefix can also be a tuple of strings to try.
```

```
>>> str7
'Taiwan,Asia,America,Japan'
>>> str7.startswith( 'Tai' )
True
>>> str7.startswith( 'Jap' )
False
>>> str7.startswith( 'T' )
True
>>> str7.startswith( 'J' )
False
```

字串內建函式 (Build-in function)

- 判斷字串尾是否符合特定字串:
  - 字串.endswith(要找的字串)

```
endswith(...)
    S.endswith(suffix[, start[, end]]) -> bool

    Return True if S ends with the specified suffix, False otherwise.
    With optional start, test S beginning at that position.
    With optional end, stop comparing S at that position.
    suffix can also be a tuple of strings to try.
```

```
>>> str7
'Taiwan,Asia,America,Japan'
>>> str7.endswith( 'pan' )
True
>>> str7.endswith( 'wan' )
False
>>> str7.endswith( 'n' )
True
```

字串內建函式 (Build-in function)

- 字串統計:
  - 字串.count(要找的字串)

```
count(...)
    S.count(sub[, start[, end]]) -> int

    Return the number of non-overlapping occurrences of substring sub in
    string S[start:end].  Optional arguments start and end are
    interpreted as in slice notation.
```

```
>>> str7
'Taiwan,Asia,America,Japan'
>>> str7.count( 'an' )
2
>>> str7.count( 'i' )
3
>>> str7.count( 'A' )
2
>>> str7.count( 'a' )
6
```

字串內建函式 (Build-in function)

- 字串置中對齊:
  - 字串.center(顯示字串寬度數值)

```
center(self, width, fillchar=' ', /)
    Return a centered string of length width.

    Padding is done using the specified fill character (default is a space).
```

```
>>> str8 = '0123456789'
>>> str8.center( 15 )
'   0123456789  '
>>> str8.center( 11 )
' 0123456789'
>>> str8.center( 12 )
' 0123456789 '
```

字串內建函式 (Build-in function)

- 字串靠右對齊:
  - 字串.rjust(顯示字串寬度數值)

```
rjust(self, width, fillchar=' ', /)
    Return a right-justified string of length width.

    Padding is done using the specified fill character (default is a space).
```

```
>>> str9 = 'abcde'
>>> str9.rjust( 6 )
' abcde'
>>> str9.rjust( 7 )
'  abcde'
>>> str9.rjust( 8 )
'   abcde'
```

字串內建函式 (Build-in function)

- 字串靠左對齊:
  - 字串.ljust(顯示字串寬度數值)

```
ljust(self, width, fillchar=' ', /)
    Return a left-justified string of length width.

    Padding is done using the specified fill character (default is a space).
```

```
>>> str9 = 'abcde'
>>> str9.ljust( 6 )
'abcde '
>>> str9.ljust( 7 )
'abcde  '
>>> str9.ljust( 8 )
'abcde   '
```

字串內建函式 (Build-in function)

- 英文字串開頭第一個字大寫
  - 英文字串.capitalize()

```
capitalize(self, /)
    Return a capitalized version of the string.

    More specifically, make the first character have upper case and the rest lower
    case.
```

```
>>> str10 = 'an apple a day. ai program'
>>> str10.capitalize()
'An apple a day. ai program'
>>> str10 = 'an apple a day'
>>> str10.capitalize()
'An apple a day'
```

字串內建函式 (Build-in function)

- 英文文句中每個單字第一個字母大寫
  - 英文字串.title()

```
title(self, /)
    Return a version of the string where each word is titlecased.

    More specifically, words start with uppercased characters and all remaining
    cased characters have lower case.
```

```
>>> str11 = 'a high payload data hiding scheme by using LSB'
>>> str11.title()
'A High Payload Data Hiding Scheme By Using Lsb'
```

字串內建函式 (Build-in function)

- 英文文句中字母大小寫互換
  - 英文字串.swapcase()

```
swapcase(self, /)
    Convert uppercase characters to lowercase and lowercase characters to uppercase.
```

```
>>> str12
'A High Payload Data Hiding Scheme By Using Lsb'
>>> str12.swapcase()
'a hIGH pAYLOAD dATA hIDING sCHEME bY uSING lSB'
```

字串內建函式 (Build-in function)

- 英文文句中字母全部變大寫
  - 英文字串.upper()

```
upper(self, /)
    Return a copy of the string converted to uppercase.
```

```
>>> str12
'A High Payload Data Hiding Scheme By Using Lsb'
>>> str12.upper()
'A HIGH PAYLOAD DATA HIDING SCHEME BY USING LSB'
```

字串內建函式 (Build-in function)

- 英文文句中字母全部變小寫
  - 英文字串.lower()

```
lower(self, /)
    Return a copy of the string converted to lowercase.
```

```
>>> str12
'A High Payload Data Hiding Scheme By Using Lsb'
>>> str12.lower()
'a high payload data hiding scheme by using lsb'
```

字串內建函式 (Build-in function)

- 指定字串顯示長度，如果原本的字串長度不夠者補 0 處理
  - 字串.zfill(字串長度)

```
zfill(self, width, /)
    Pad a numeric string with zeros on the left, to fill a field of the given width.

    The string is never truncated.
```

```
>>> str1 = '123'
>>> str1.zfill( 4 )
'0123'
>>> str1.zfill( 5 )
'00123'
>>> str2 = 'abc'
>>> str2.zfill( 4 )
'0abc'
>>> str2.zfill( 5 )
'00abc'
```

字串內建函式 (Build-in function)

- 清除字串左右兩側的空白字元
  - 字串.strip( )

```
strip(self, chars=None, /)
    Return a copy of the string with leading and trailing whitespace removed.

    If chars is given and not None, remove characters in chars instead.
```

```
>>> str1 = '  123 '
>>> str1.strip()
'123'
>>> str2 = 'abc    '
>>> str2.strip()
'abc'
>>> str3 = '   123abc'
>>> str3.strip()
'123abc'
```

字串內建函式 (Build-in function)

- 清除字串左側的空白字元
  - 字串.lstrip( )

```
lstrip(self, chars=None, /)
    Return a copy of the string with leading whitespace removed.

    If chars is given and not None, remove characters in chars instead.
```

```
>>> str1 = '     Hi, Jack'
>>> str1.lstrip()
'Hi, Jack'
>>> str1.lstrip('H')
'     Hi, Jack'
>>> str1.lstrip('  H')
'i, Jack'
>>> str1.lstrip(' H')
'i, Jack'
```

字串內建函式 (Build-in function)

- 清除字串右側的空白字元
  - 字串.rstrip( )

```
rstrip(self, chars=None, /)
    Return a copy of the string with trailing whitespace removed.

    If chars is given and not None, remove characters in chars instead.
```

```
>>> str1 = '  Hi, Jack    '
>>> str1.rstrip()
'  Hi, Jack'
>>> str1.rstrip('k')
'  Hi, Jack    '
>>> str1.rstrip('k ')
'  Hi, Jac'
```

字串切割

(時間限制：2秒)

問題描述：

寫一個程式讓使用者輸入一段文字及一個切割用符號，程式會依指定的符號切割成數字子字串。

輸入說明：

分別輸入文字串(String)及切割符號(char)。

輸出說明：

透過切割符號切割後每一個字串一列顯示，最後必須有換行字元。

範例：

Sample Input:	Sample Output:
asia,japan,taiwan,taipei ,	asia japan taiwan taipei
Jack#David&John#Kevin #	Jack David&John Kevin

全部變小寫

(時間限制：2秒)

問題描述：

寫一個程式讓使用者輸入一段文字(String)，程式會把文字串所有的字母不管大小寫全部變成小寫字母顯示(String)。

輸入說明：

分別輸入文字串(String)。

輸出說明：

字串裡的每一個字母都變成小寫後顯示(String)，最後必須有換行字元。

範例：

Sample Input:	Sample Output:
Taiwan and U.S.A.	taiwan and u.s.a.

字串字元置換

(時間限制：2秒)

問題描述：

阿坤最討論長官老是要他改文件，因為阿坤的長官打字不太行，經常注音不選字所以錯別字特別多。請寫一個程式幫阿坤將字串中同樣的字以另外一個字替換掉。

輸入說明：

輸入一個字串(String)與 2 個字元(char)。

輸出說明：

輸出被替換後的字串(String)，最後必須有換行字元。

範例：

Sample Input:	Sample Output:
asia csie taiwan i 1	as1a cs1e ta1wan

取得字元ASCII碼

(時間限制：2秒)

問題描述：

請寫一個程式讓使用者輸入一個英文字母、標點符號或數字字符，程式回傳取得該字元的ASII值。

輸入說明：

輸入一個字元 ch (char)。

輸出說明：

輸出該字元對應的 ASCII 值 (int)，最後必須有換行字元。

範例：

Sample Input:	Sample Output:
a	97
X	88

取得中文字元萬國碼(Unicode)

(時間限制：2秒)

問題描述：

請寫一個程式讓使用者輸入一個中文字元，程式回傳取得該字元的萬國碼。

輸入說明：

輸入一個中文字元 ch (char)。

輸出說明：

輸出該字元對應的「萬國碼」(Unicode)值 (int)，最後必須有換行字元。

範例：

Sample Input:	Sample Output:
亞	4e9e
資	8cc7

取得16進制萬國碼(Unicode)對應中文字

(時間限制：2秒)

問題描述：

請寫一個程式讓使用者輸入16進制萬國碼，程式回傳該萬國碼的中文字元。

輸入說明：

輸入一個16進制整數 (int)。

輸出說明：

輸出對應的「萬國碼」(Unicode)值的字元 (char)，最後必須有換行字元。

範例：

Sample Input:	Sample Output:
8cc7	資
48e9	醜

判斷大小寫或其他字元

(時間限制：2秒)

問題描述：

輸入一個字元，判斷是大寫或小寫或是其他字元。

輸入說明：

輸入一個字元 (char)。

輸出說明：

輸出判斷結果(大寫：uppercase、小寫：lowercase、特殊字元：special character)，最後必須有換行字元。

範例：

Sample Input:	Sample Output:
A	uppercase
a	lowercase
*	special character

大小寫轉換

(時間限制：2秒)

問題描述：

輸入一個字元，轉換其大小寫輸出 (大寫 ->小寫，小寫->大寫)。

輸入說明：

輸入一個字元 (char)。

輸出說明：

將輸入轉換其大小寫輸出，最後必須有換行字元。

範例：

Sample Input:	Sample Output:
A	a
a	A