Python
字串操作

臺北科技大學資訊工程系

字串格式化

- '字串'.format(參數)
  - 字串內以{}代換 format的參數-- 字串資料
  - 代換以 0, 1, 2..位置識別，或者以符號識別

```
def format00():
    str='{0} is {1}!!'.format('Justin', 'caterpillar')            #第0個對應後面第0個參數
    print(str)                                                     #'Justin is caterpillar!!'
    str='{real} is {nick}!!'.format(real = 'Justin', nick = 'caterpillar')   #以符號對應
    print(str)                                                     #'Justin is caterpillar!!'
    str='{real} is {nick}!!'.format(nick = 'caterpillar', real = 'Justin')   #符號順序無關
    print(str)                                                     #'Justin is caterpillar!!'
    str='{0} is {nick}!!'.format('Justin', nick = 'caterpillar')
    print(str)                                                     #'Justin is caterpillar!!'
    str='{name} is {age} years old!'.format(name = 'Justin', age = 35)
    print(str)                                                     #'Justin is 35 years old!'
```

字串格式化

- format() ，使用{} 放參數

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
print(myorder.format(quantity, itemno, price))
myorder = "I want to pay {2} dollars for {0} pieces of item {1}."
print(myorder.format(quantity, itemno, price))
```

```
I want 3 pieces of item 567 for 49.95 dollars.
I want to pay 49.95 dollars for 3 pieces of item 567.
```

字串格式化

```
import math
import sys
def format01():
    str=math.pi
    print(str)                                                    #3.14159265359'
    str=format(math.pi, '.18f')                                   #PI 取小數點18位
    print(str)                                                    #3.141592653589793116
    str='PI = {0.pi}'.format(math)                                #第0個對應後面第0個參數
    print(str)                                                    #'PI = 3.14159265359'
    str='My platform is {pc.platform}'.format(pc = sys)           #以符號 pc 對應 sys
    print(str)                                                    #'My platform is win32'
    str='My platform is {0.platform}. PI = {1.pi}.'.format(sys, math)
    print(str)
    #'My platform is win32. PI = 3.14159265359.'
    str='element of index 1 is {0[1]}'.format([20, 10, 5])        #第0個對應後面第0個參數
    print(str)                                                    #'element of index 1 is 10'
    str='My name is {person[name]}'.format(person = {'name' : 'Justin', 'age' : 35})
    print(str)                                                    #'My name is Justin'
```

字串格式化

- str.ljust(length, [char])
  - 將字串向靠左對齊
  - []可選的參數
  - char是預設空格，補齊符號
- str.rjust(length, [char])
  - 將字串向靠右對齊
- str.center(length, [char])
  - 將字串向中間對齊

```python
s="banana"
s1=s.rjust(20)
s2=s.ljust(20)
s3=s.center(20)
print(s1, "is my favorite fruit.")
print(s2, "is my favorite fruit.")
print(s3, "is my favorite fruit.")
print(s.rjust(20, "O"))
print(s.ljust(20, "O"))
print(s.center(20, "O"))
```

```
              banana is my favorite fruit.
banana               is my favorite fruit.
       banana        is my favorite fruit.
OOOOOOOOOOOOOObanana
bananaOOOOOOOOOOOOOO
OOOOOOObananaOOOOOOO
```

字串格式化

- 三種格式化 (%-formatting, str.format, f-string)
  - '%s'，後面接 tuple資料型別，轉成字串資料型別

```
a, b, d = 2, 3, 'ans'
s = d + ':'+ str(a) + '/' + str(b) + '=' + str(round(a/b, 3))
print(s)
x = '%s: %d/%d =%.3f' % (d, a, b, a/b)
print(x)
y = '{d1}: {a1}/{b1} = {c1:.3f}'.format(a1=a, b1=b, c1=a/b, d1=d)
print(y)
z = f'{d}: {a}/{b} = {a/b:.3f}'
print(z)
```

```
ans: 2/3 =0.667
ans: 2/3 =0.667
ans: 2/3 = 0.667
ans: 2/3 = 0.667
```

```
a = list('Apple')
print(a)
content='%s'*len(a) %tuple(a)
# len(a) =5
# '%s'*5   '%s %s %s %s %s'
#  '%s %s %s %s %s' %('A', 'p','p', 'l', 'e')
print(content)
```

```
['A', 'p', 'p', 'l', 'e']
Apple
```

字串格式化

- ^ (居中)、< (左對齊)、> (向右對齊)、{:,}(分隔數字)調整輸出

```
d = 'ans'
e = 'US$'
a = 980000
x = '{d1:^10}={a1:<10,d}-{e1:>10}'.format(a1=a, d1=d, e1=e)
print(x)
```

```
ans    =980,000   -       US$
```

字串重整

- string.strip([chars])
  - 將string字串變數裡的左右兩邊空白刪除掉
  - chars參數可決定要刪除的符號

```
t1 = "      aaaaa      bbbbbb    aaa          ccccc "
t2 = "aaaaabbbbbb  aaa          ccccc aaaaa"
print(t1.strip())
print(t2.strip('a'))
```

```
aaaaa   bbbbbb  aaa     ccccc
bbbbbb  aaa     ccccc
```

- string.lstrip([chars]): 左邊空白刪除
- string.rstrip([chars]): 右邊空白刪除
  - string.rstrip('c'): 右邊'c'子字串去掉，直到空白

```
s1 = "        aaaaa        bbbbbb      aaa ccccc cc"
s2 = "        aaaaa        bbbbbb      aaa ccccc "
print(s1.lstrip())
print(s2.rstrip())
print(s1.rstrip('c'))
```

```
aaaaa   bbbbbb  aaa ccccc cc
        aaaaa   bbbbbb  aaa ccccc
        aaaaa   bbbbbb  aaa ccccc
```

字串重整

- string.zfill(width)
  - 將string變數內字串前補0，直到string變數的長度等於width參數設定的長度

```
s = "50"
print(s.zfill(3))
print(s.zfill(10))
print(s.zfill(0))
```

```
050
0000000050
50
```

字串搜尋

- string.count(sub[, start[, end]])
  - 回傳此字串裡有多少個sub子字串

```
text='abbggccdeefgggijklgglmo'
print(text.count('g'))
print(text.count('g',4,-4))
```

```
7
5
```

```
images="xbox.gif, iphone.jpg"
print(images.startswith("xbox"))
print(images.startswith(".gif"))
print(images.startswith("iphone",10, 20))
```

```
True
False
True
```

- str.startswith(prefix[, start[, end]])
  - 判斷傳入的prefix字串是否為開始字串
- str.endswith(suffix[, start[, end]])
  - 判斷字串內是否有符合suffix引數的值

```
images="xbox.gif, iphone.jpg"
print(images.endswith(".jpg"))
print(images.endswith(".gif"))
print(images.endswith(".gif",0, 8))
```

```
True
False
True
```

Exercise

- 計算字串有多少單一元素和空白

```
s='Given a string and count how many characters and spaces in the
string'
print(len(s))
words=s.[redacted](' ')
print(words)
```

Exercise

- 計算字串的元素個數
  - Sample String : 'google.com'
  - Expected Result : {'g': 2, 'o': 3, 'l': 1, 'e': 1, '.': 1, 'c': 1, 'm': 1}
  - Sample string: 'thequickbrownfoxjumpsoverthelazydog'
  - Expected output :
    - o 4
    - e 3
    - u 2
    - h 2
    - r 2
    - t 2

```
inputStr = "google.com"
countDict = {}
for char in inputStr:
  countDict[char]=inputStr.______(char)
print(countDict)
```

字串搜尋

- sring.rfind(sub[,start[, end]])
  - 從右到左尋找，sub預計要搜尋的字串
- string.find(sub[, start[, end]])
  - 從左到右尋找
  - 搜尋字串變數裡符合sub的字串位置
  - 找不到回傳-1.
- string.rindex(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

```
s = "Mi casa, su casa."
print(s.find("casa"))
print(s.find("haha"))
print(s.rfind("casa"))
print(s.find("casa",5))
```

```
3
```

```
text='abcdefgabcdefg'
print(text.find('a'))
print(text.find('a',1))
print(text.rfind('e'))
```

```
0
7
11
```

```
s = "Mi casa, su casa."
print(s.rindex("casa"))
print(s.rindex("casa",2,10))
#print(s.rindex("haha"))
```

```
12
3
```

Exercise

- 找出字串中 “USA” 個數，大小寫視為相同
  - input_str = "Welcome to USA. usa awesome, isn't it?“
  - Expected outcome: The USA count is: 2

```
inputString = "Welcome to USA. usa awesome, isn't it?"
substring = "USA"
tempString = inputString.lower()
count = tempString.count(         lower())
print("The USA count is:", count)
```

- 找到 "Hello, World"字串中第一個和最後一個'o' 和 ','位置

```
s = "Hello, World"
print(s.f   ("o"))
print(s.r   l("o"))
print(s.f   (","))
print(s.r   l(","))
```

```
4
8
5
5
```

字串轉換

- string.translate(map)
  - 將 string 中的字串以 map 中配對的字串轉換，
    - 搭配map=str.maketrans(from, to)

```
fromStr = "aeiou"
toStr = "12345"
str = "this is string example....wow!!!"
map=str.maketrans(fromStr, toStr)
#map=str.maketrans(fromStr, toStr,'w')
print(str.translate(map))
```

```
th3s 3s str3ng 2x1mpl2....4!!!
```

```
th3s 3s str3ng 2x1mpl2....w4w!!!
```

```
dict= {"a": "123", "b": "456", "c": "789"}
string = "abcdef"
map=string.maketrans(dict)
print(string.translate(map))
```

```
123456789def
```

Exercise

- 找字串中最大和最小長度的word
- 反轉字串中的 words
- 輸入逗號隔開的一串 word，輸出排序好不重複的 word
  - Sample Words : red, white, black, red, green, black
  - Expected Result : black, green, red, white

```
s = input('enter a string: ')
print(s▢-1])
words=s.▢()
for word in words▢-1]:
  print(word, end=' ')
```

Exercise

- 找基因序列 (從頭找，每次都找最短的基因)
  - 若DNA序列由A, C, G, T四個字母組成的字串(string)，
  - 基因(gene)是隱藏於DNA序列中的部分片段(子字串)。
  - 給定一DNA序列(長度小於50)，找出在裡面的基因；規則：
    - 1.前面是ATG，後面接TAG、TAA、或TGA；
      - 例如ATGTTTTAA。
    - 2.長度為3的倍數，其中未含有ATG、TAG、TAA或TGA。
  - 例如，給定DNA序列，其中包含兩個基因，CCATGTTTTAACCATGCCTAAATGGGGCGTTAGTT：
    - 基因TTT前面為ATG，後面接著TAA，長度3，其中未含ATG、TAG、TAA或TGA。
    - 基因GGGCGT前面為ATG，後面接著TAG，長度6，其中未含ATG、TAG、TAA或TGA。
  - 撰寫程式輸入DNA序列，找出該序列中所有基因。

Exercise

```
def check(gen):
    if len(gen)==0:                          #空的基因
        return False
    for tag in ['ATG', 'TAG','TAA','TGA']: #不能含有這些tag
        if gen.find(tag)>-1:
            return False
    if len(gen)%3==0:                        #長度3倍數
        return True
    return False
```

Exercise

```
def findGen(dna):
    startTag, endTags = 'ATG', ['TAG','TAA','TGA']
    length = len(dna)
    startIndex=dna.find(startTag)+3          #基因前面是 startTag 'ATG'
    endIndex=length+1                        #初始化結束點
    for tag in endTags:                      #從三個 end tag 找出最小符合基因字串
        endTemp=dna.find(tag, startIndex)
        if endTemp!=-1 and endTemp<endIndex:  #找最小符合的 end tag 的點
           endIndex = endTemp
    if endIndex==length+1 or startIndex<3:  #找不到前後tag
        return 0, 0, 'None'
    gen = dna[startIndex:endIndex]            #找到基因
    if check(gen):                            #驗證基因
        return 1, endIndex+3, gen
    return 2, startIndex+3, 'None'
```

Exercise

```
def finAllGen(dna):
    i, count = 0, 0
    print('==>')
    while (True):
        dna=dna[i:]              #往後找基因
        b, i, gen = findGen(dna) #b=1找到, i往後找索引
        if (b==1):
            print(gen)
            count=count+1        #找到幾個基因
        elif (b==0):             #找到最後沒找到
            break
    if count==0:                 #完全沒找到
        print('沒有基因')

finAllGen('CCATGTTTTAACCATGCCTAAATGGGGCGTTAGTT')
finAllGen('TAAGATGAATGA')
finAllGen('ATGAAATGA')
finAllGen('ATGTGAATGAAATGA')
finAllGen('TTATGTTAAAAGGATGTTAATGTAAGGGCGTTAGTT')
finAllGen('AATAGATGTTTAAGTGATATGGGGATGTCATAGATGCCCTTCACCTAA')
```

```
==>
TTT
GGGCGT
==>
沒有基因
==>
AAA
==>
AAA
==>
沒有基因
==>
TCA
CCCTTCACC
```

字串函數 1/3

Method	Description
capitalize()	Converts the first character to upper case
casefold()	Converts string into lower case
center()	Returns a centered string
count()	Returns the number of times a specified value occurs in a string
encode()	Returns an encoded version of the string
endswith()	Returns true if the string ends with the specified value
expandtabs()	Sets the tab size of the string
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
format_map()	Formats specified values in a string
index()	Searches the string for a specified value and returns the position of where it was found
isalnum()	Returns True if all characters in the string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabet
isdecimal()	Returns True if all characters in the string are decimals

字串函數 2/3

isdigit()	Returns True if all characters in the string are digits
isidentifier()	Returns True if the string is an identifier
islower()	Returns True if all characters in the string are lower case
isnumeric()	Returns True if all characters in the string are numeric
isprintable()	Returns True if all characters in the string are printable
isspace()	Returns True if all characters in the string are whitespaces
istitle()	Returns True if the string follows the rules of a title (i.e., Check if each word start with an upper case letter)
isupper()	Returns True if all characters in the string are upper case
join()	Joins the elements of an iterable to the end of the string
ljust()	Returns a left justified version of the string
lower()	Converts a string into lower case
lstrip()	Returns a **left trim** version of the string
maketrans()	Returns a translation table to be used in translations
partition()	Returns a tuple where the string is parted into **three parts**
replace()	Returns a string where a specified value is replaced with a specified value

字串函數 3/3

rfind()	Searches the string for a specified value and returns the last position of where it was found
rindex()	Searches the string for a specified value and returns the last position of where it was found
rjust()	Returns a right justified version of the string
rpartition(sep)	以 sep 從最右端分割 str 為三個部份，結果回傳具有三個子字串的序對
rsplit([sep[,maxsplit]])	將 str 從最右端以 sep 分割成子字串，回傳儲存子字串的 串列， maxsplit 為子字串最多的數量
rstrip([chars])	從 str 的最右端中移除 chars 字串，預設為空白
split([sep[, maxsplit]])	將 str 以 sep 分割成子字串，回傳儲存子字串的串列，maxsplit 為子字串最多的數量
splitlines([keepends])	將 str 以新行符號分割成子字串，回傳儲存子字串的串列
startswith(prefix[, start[, end]])	判斷 str 是否以 prefix 開頭
strip([chars])	從 str 中移除 chars 字串，預設為空白
swapcase()	將 str 中的英文字母進行大小寫轉換
title()	Converts the first character of each word to upper case
translate()	Returns a translated string
upper()	將 str 的英文字母都改成大寫
zfill(width)	回傳以 0 塞滿 width 的新字串

跳脫符號

Code	Result	Example	Result

\'	Single Quote	txt = 'It\'s alright.'
		print(txt)
			It's alright.

\\	Backslash	txt = "This will insert one \\ (backslash)." print(txt)	This will insert one \ (backslash).

\n	New Line	txt = "Hello\nWorld!"
		print(txt)
			Hello
			World!

\r	Carriage Return	txt = "Hello\rWorld!"
		print(txt)
			Hello
			World!

\t	Tab	txt = "Hello\tWorld!"
		print(txt)
			Hello World!

\b	Backspace	txt = "Hello \bWorld!"
		print(txt)
			HelloWorld!

\f	Form Feed		forces the printer to eject the current page and to continue printing at the top of another

\ooo	Octal value	txt = "\110\145\154\154\157"
		print(txt)
			Hello

\xhh	Hex value	txt = "\x48\x65\x6c\x6c\x6f"
		print(txt)
			Hello

END