商管程式設計 109-1
TA Lab 7

2020/11/02-11/06

影片助教 彭晨

Agenda

- 字串資料型態
- 常用字串函數

字串資料型態

字串資料型態

1. 字(符)串即為一串字符。
2. 字串兩側需要用雙或單引號包住。
3. 型態為 **<class 'str'>**
4. **input()** 函數傳入的預設值型態為字串。
5. 使用 **len()** 函數可以取得字串之長度。
   (一字串長度即為在此字串中的字元數目)

```
string1 = "New York"
print(type(string1))

string2 = input()
print(type(string2))

print(len(string1))
print(len(string2))
```

```
<class 'str'>
London
<class 'str'>
8
6
```

取出字串中的字元

- 字串中的每個字元可以用其對應位置 (index) 取出。
- 字串的位置 (index) 由第一個字元開始依序為 0, 1, ...。
- 字串的位置亦可由最後一個字元開始，用負數標記。

字串	T	O	K	Y	O	!
正數 index	0	1	2	3	4	5
負號 index	-6	-5	-4	-3	-2	-1

取出字串中的子字串--Slicing

使用**string[start_index : end_index]**
取出位置從 **start_index** 到 **end_index-1** 的子字串。
若不指定 **start_index** 則電腦預設為 **start_index** 為 0。

字串	S	a	n		F	r	a	n	c	l	s	c	o
正數 index	0	1	2	3	4	5	6	7	8	9	10	11	12
負數 index	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
string3 = "San Francisco"
print(string3[2:4])
print(string3[2:])
print(string3[:4])
print(string3[2:-1])
```

```
n
n Francisco
San
n Francisc
```

取出字串中的子字串--Slicing

使用**string[start_index : end_index : stepSize]**

則會取出位置從 **start_index** 到位置 **end_index** 以前(不包含)的子字串中，每 **stepSize** 取出字元輸出成最後的字串。

```
string4 = "To Boston,Tokyo,Seattle,Los Angeles Toronto"
print( string4[2:10]   )
print( string4[2:10:]  )
print( string4[2:10:2] )
print( string4[2:10:3] )
```

```
Boston,
Boston,
otn
sn
```

字串的運算

連接兩字串可以使用 + 來連接。

重複一字串可以使用 * 來重複。

```
string1 = "NTU"
string2 = "PDOGS"
print(string1 + string2)
print(string1 * 3)
```

```
NTUPDOGS

PDOGSPDOGSPDOGS
```

尋訪一字串中的字元

```
string = "New York"
for character in string:
    print(character, end=" ")
print()
for index, character in enumerate(string):
    print(index, character, end=" ")
print()
for index in range(len(string)):
    print(string[index], end=" ")
```

```
N e w   Y o r k

0 N 1 e 2 w 3   4 Y 5 o 6 r 7 k

N e w   Y o r k
```

常用字串函數

常用字串函數

方法	內容
`str.capitalize()`	大寫字串第一個字元。
`str.upper()`	大寫字串所有字元。
`str.lower()`	小寫字串所有字元。
`str.title()`	大寫每個單詞的第一個字元。
`str.count("x")`	計算字串 `"x"` 在另外一個字串中出現的次數。
`str.find("x")`	檢查字串 `"x"` 是否包含於另外一個字串中。若存在則回傳其第一次出現的起始位置，若不存在則回傳 -1。
`str.replace("x", "y")`	取代字串中的特定子字串。將 `"x"` 以 `"y"` 取代。

常用字串函數

```
string = "new york boston london TOKYO"
print(string.capitalize())
print(string.upper())
print(string.lower())
print(string.title())
print(string.count("on"))
print(string.find("london"))
print(string.find("seattle"))
print("melbourne" in string)
print("tokyo" in string)
print(string.replace("boston", "Kyoto"))
```

```
New york boston london tokyo
NEW YORK BOSTON LONDON TOKYO
new york boston london tokyo
New York Boston London Tokyo
3
16
-1
False
False
new york Kyoto london TOKYO
```

常用字串函數

方法	內容
str.split()	根據特定字串去將另一字串分割，將分割的結果存成 list。
"y".join(x)	以 "y" 串接 list x 中的所有字串。

```
string = " new york.boston.london,tokyo "
print(string.split('.'))
print(string.split(','))
print(string.split())
print(string.split('wow'))
string_list = ['New
York','London','Boston']
print(' '.join(string_list))
print('!!!'.join(string_list))
```

```
[' new york', 'boston', 'london,tokyo ']
[' new york.boston.london', 'tokyo ']
['new', 'york.boston.london,tokyo']
[' new york.boston.london,tokyo ']
New York London Boston
New York!!!London!!!Boston
```

指定字串輸出格式 .format()

- .format() 在字串指定位子插入要印出的變數

```
# Format a string with positional arguments


print('{0} {1} cost ${2}'.format(6, 'bananas', 1.74))

# 6 bananas cost $1.74
```

```
# Format a string with keyword arguments

print('{quantity} {item} cost ${price}'.format(\
quantity=6, item='bananas', price=1.74))
# 6 bananas cost $1.74
```

指定字串輸出格式 .format()

- .format() 指定小數列印格式

```
print("{:.2f}".format(3.1415926))  # 保留小數點後兩位
# 3.14

print("{:+.2f}".format(3.14159))  # 帶正負符號保留小數點後兩位
# +3.14

print("{:+.4f}".format(-3.14159)) # 帶正負符號保留小數點後四位
# -3.1416

print("{:.0f}".format(-3.14159))  # 不帶小數
# -3
```

指定字串輸出格式 .format()

- .format() 對齊

```
print("{:>10d}".format(20))  # 向右對齊，寬度為10
#          20
# 列印出的 20 前面會有 8 個空格

print("{:<10d}".format(20)) # 向左對齊，寬度為10
#20
# 列印出的 20 後面會有 8 個空格

print("{:^10d}".format(20)) # 置中對齊，寬度為10
#     20
# 列印出的 20 前後各有 4 個空格
```

指定字串輸出格式 .format()

- .format() 補字元

```
print("{:0>2d}".format(8))  # 以 0 填充左邊，寬度為 2
# 08

print("{:X<10d}".format(20))  # 以 X 填充右邊，寬度為10
# 20XXXXXXXX
```

重點整理

字串資料型態

- 字串即為一串字符。
- 字串兩側需要用雙或單引號包住。
- 型態為 `<class 'str'>`
- `input()` 函數傳入的預設值型態為字串。
- 使用 `len()` 函數可以取得字串之長度（字元數目）。
- 可以用 `+` 來連接、用 `*` 來重複。

取用字串的方法

- 用對應位置 (index) 取出字串中的字元。
- 使用 `string[start_index : end_index : stepSize]` 取出位置從 `start_index` 到 `end_index-1` 的子字串，`stepSize` 用來指定步長。
- 用 `for` 迴圈取出。

```
string1 = "New York"
print(string1[0])
print(string1[0:5])
print(string1[0:5:2])
for character in string1:
    print(character, end=" ")
```

```
N
New Y
NwY
N e w  Y o r k
```

常用字串函數

方法	內容
`str.capitalize()`	大寫字串第一個字元。
`str.upper()`	大寫字串所有字元。
`str.lower()`	小寫字串所有字元。
`str.title()`	大寫每個單詞的第一個字元。
`str.count("x")`	計算字串 `"x"` 在另外一個字串中出現的次數。
`str.find("x")`	檢查字串 `"x"` 是否包含於另外一個字串中。若存在則回傳其第一次出現的起始位置，若不存在則回傳 -1。
`str.replace("x", "y")`	取代字串中的特定子字串。將 `"x"` 以 `"y"` 取代。
`str.split()`	根據特定字串去將另一字串分割，將分割的結果存成 list。
`"y".join(x)`	以 `"y"` 串接 list `x` 中的所有字串。

格式化輸出 str.format()

- .format() 在字串指定位置插入要印出的變數

```
print('{} {} cost ${}'.format(6, 'bananas', 1.74))
print('{1} {0} cost ${2}'.format('bananas', 6, 1.74))
# 6 bananas cost $1.74
```

- 數字格式化

```
print('{:.2f}'.format(3.1415926))  # 3.14
```

數字	格式	輸出	內容
3.1415926	{:.2f}	3.14	保留小數點後兩位
3.1415926	{:+.2f}	+3.14	帶符號保留小數點後兩位
-1	{:+.2f}	-1.00	帶符號保留小數點後兩位
2.71828	{:.0f}	3	取整數

Practice 1

Practice 1

- 請定義一個函數 `countCharacter(str_)`，接收一個字串參數。
- 找出字串中重複出現最多次的字元，並印出其出現次數與該字元。
- 重複出現最多次的字元只會有一個。
- 若沒有重複出現的字元，則印出 N。使用 `countCharacter(input())`來測試。
- Sample 1: 輸入 `庭院深深深幾許` 輸出 `3,深`
- Sample 2: 輸入 `How about you?` 輸出 `3,o`
- Sample 3: 輸入 `Thank you` 輸出 `N`

```python
def countCharacter(str_):
    character_list = []
    for character in str_: # 將字元加入一個list
        if character_list.count(character) == 0: # 沒出現過才加
            character_list.append(character)
    # 記錄最多次
    max_count = 0
    max_character = ""
    for character in character_list: # 找出現最多次的字元
        if str_.count(character) > max_count:
            max_count = str_.count(character)
            max_character = character
    if max_count == 1: # 最多次數是1即是沒有重複
        print("N")
    else:
        print(max_count, max_character, sep=",")

countCharacter(input())
```

Practice 2

Practice 2

- 所謂凱薩密碼，就是將原本的英文字母對應到偏移過後所對應到的英文字母，舉例而言，若偏移量為 3，則 A 對應到 D，B 對應到 E，依此類推，而 X 對應到回 A。
- 請定義一個函數 `caesar(str_, n)`，輸入為一字串以及偏移量，輸出為加密過後的字串。
- Sample 1: 輸入 `caesar('apple', 3)` 輸出 `dssoh`
- Sample 2: 輸入 `caesar('Apple', 3)` 輸出 `Dssoh`
- Hint: `ord()` 和 `chr()`

Solution 2

```
def caesar(str_, n):
    result = ""
    for i in range(len(str_)):
        char = str_[i]
        if char.isupper():
            shift = (ord(char) + n - ord('A')) % 26 + ord('A')
            result += chr(shift)
        else:
            shift = (ord(char) + n - ord('a')) % 26 + ord('a')
            result += chr(shift)
    return result
```

Practice 3

Practice 3

- 定義一個函數 `myPalindrome`，接收一個字串參數，
- 用遞迴的方式檢查此字串是否為對稱。
- 效果等於 **`s == s[::-1]`**

- Sample 1: 輸入 `123456` 輸出 `False`
- Sample 2: 輸入 `anna` 輸出 `True`
- Sample 3: 輸入 `step on no pets` 輸出 `True`

Solution 3

```
def myPalindrome(string):
    if len(string) <= 1:
        return True
    else:
        if string[0] == string[-1]:
            return myPalindrome(string[1:-1])
        else:
            return False

print(myPalindrome(input()))
```