2.8. String

Fig. 2.8.1 Photo by Kelly Sikkema on Unsplash

Note

Outline

1. Introduction
2. Examples
    a. Ex1: String Cut
    b. Ex2: String Cut More
    c. Ex3: What a Beautiful and Peaceful World
    d. Ex4: Reverse String
    e. Ex5: ord() vs. chr()
    f. Ex6: String vs. Bytes
3. Build-in Functions
    a. bytes.split([sep=None, maxsplit=-1])
    b. bytes.count(sub[, start[, end]])
    c. bytes.find(sub[, start[, end]])
    d. bytes.index(sub[, start[, end]])
    e. bytes.replace(old, new[, count])
    f. bytes.join(iterable)
    g. bytes.strip([chars]), bytes.lstrip([chars]), bytes.rstrip([chars])
4. Ex7: Summary
5. Exercise

Note

Roadmap

1. This topic: String

myMaze

Basic Syntax																
Encoding	Data Type							Output					Input		Remark	Identation
	Number				String	Container*	Type Casting	print()	Format			Escape	Assignment	input()		
	int	float	bool	complex*					%	format()	F-String					

2. Course: Python 1
3. Subject: Programming
4. Field
    a. Software Engineering (SE)
    b. Computer Science and Information Engineering (CSIE)
    c. Electrical/Electronics Engineering (EE)

1. 字串是程式語言中相當重要的一部分， 在傳達資訊給使用者及獲取使用者的輸入資料時經常被使用到。
2. 只要不以真 / 假、數值表示的資料都可以用字串來表示。
3. 在使用字串時， Python 本身有內建一種叫函數庫的東西， 裡面已經幫普羅設定好多種函數可以直接使用。
4. 不只是字串有函數庫， 其他資料型別也有類似的函數庫。
5. 字串有索引值（index）的概念， 也就是當給定一個index值， Python便會回傳在字串中該 index 值所代表的字元。
6. 如下表所示， 值得一提的是， index 值也可以是負數。

s[0]	s[1]	s[2]	s[3]	s[4]	s[5]	s[6]	s[7]	s[8]	s[9]	s[10]	s[11]
H	e	l	l	o	,		W	o	r	l	d
s[-12]	s[-11]	s[-10]	s[-9]	s[-8]	s[-7]	s[-6]	s[-5]	s[-4]	s[-3]	s[-2]	s[-1]

7. 在前面章節有介紹過， 定義字串的方式可以使用一組單引號或雙引號括起來， 又或者使用連續三個單(雙)引號在前後包住。
8. 此外， 使用三個單(雙)引號前後括住的字串還可以任意換行， 例如：

```
string ='''使用三個單( 雙) 引號
前後括住的字串可以任意換行'''
```

9. 字串本身具有順序性， 並且可以從中間取出或插入部分資料。

10. 這裡會使用到前面章節所提到的[ ] 運算子， 形式是str[start: end: step]。
11. 這三個屬性都是選擇性的選項屬性， 可以自由選擇是否使用。
12. 當沒有輸入時， 預設值start為0， end為len(str)， 也就是str的最後一個index值再加上1， step為1。
13. 換言之， print(s)、print(s[::]) 與print(s[0:len(s):1]) 等三者所得到的結果都會是一樣的， 以下舉例說明之：

2.8.1. Examples

2.8.1.1. Ex1: String Cut

1. Code+Output

Code Output

```
1  s = '0123456'
2  print(s)
3  print(s[0: 0])
4  print(s[0: 1])
5  print(s[0: 2])
6  print(s[1: 0])
7  print(s[1: 1])
8  print(s[1: 2])
9  print(s[: 0])
10 print(s[: 1])
11 print(s[: 2])
```

2.8.1.2. Ex2: String Cut More

1. Code+Output

p0401String.py Answer

Listing 2.8.1.2.1 /src/String/p0401String.py

```
1  #coding=utf-8
2  '''
3  Created on 2015年8月27日
4
5  @author: cph
6  '''
7  s = "Hello, World"
8  print(s)            # 直接印出s
9
```

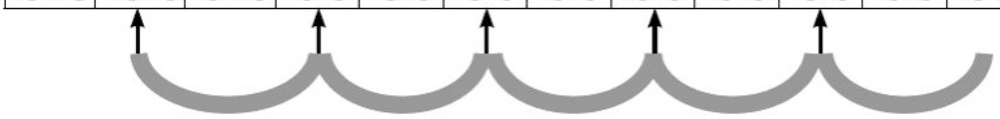
```
print(s[2:])        # 設定為[2:]，代表從index = 2 的字元到最後一個字元
print(s[0:7] + "The Beautiful" + s[6:12])   # 利用+ 運算子將多個字串連接起來
print(s[::1])       # 設定為[::1]，表示從第0 個字元到最後一個字元，以1 為間隔取值
print(s[::2])       # 設定為[::2]，表示從第0 個字元到最後一個字元，以2 為間隔取值
print(s[::-1])      # 設定為[::-1]，會將整個字串顛倒輸出，請參考後續的範例
print(s[-1:0:-2])   # 設定為[-1:0:-2]，從-1 開始往回到0，以-2 間隔取值
```

2. 從上面的程式碼可以看出， 在第三次輸出時， 菲絲恩將s字串的兩個切段中間加入新的字串， 並且將三個子字串用+運算子作連接， 這是其中一種將子字串插入已經存在字串的方式。

3. 而最後一次輸出則是s[-1: 0: -2]， 這可能有些難理解，以下將以圖形說明之：

表 4-2 str 的 index 比對表

H	e	l	l	o	,		W	o	r	l	d
s[-12]	0s[-11]	s[-10]	s[-9]	s[-8]	s[-7]	s[-6]	s[-5]	s[-4]	s[-3]	s[-2]	s[-1]



s[-1:0:-2] = s[::-2] == "drW,le"

2.8.1.3. Ex3: What a Beautiful and Peaceful World

1. Code+Output

p0401String-20210709.py | Output

Listing 2.8.1.3.1 /src/String/p0401String-20210709.py

```
'''
Created on 2021年7月9日

@author: cph
'''
sPos = "012345678901234567890123456789012345678901234567"
sStr = "What a beautiful and peaceful world~~~"
sPosN= "876543210987654321098765432109876543210987654321"
print('Original String:', sStr)
print()
print('[1] Get "beautiful" word...')
print('Answer: [', sStr[7: 16: 1], ']')
print('Answer: [', sStr[7: 16: ], ']')
```

```
print('Answer: [', sStr[7: 16], ']')
print('Answer: [' + sStr[7: 16: 1] + ']')
print('Answer: [' + sStr[7: 16: ] + ']')
print('Answer: [' + sStr[7: 16] + ']')
print()
print('[2] Get "lufituaeb" word...')
print('Answer: [' + sStr[15: 6: -1] + ']')
print('Answer: [' + sStr[-23: -32: -1] + ']')
print('Answer: [' + sStr[15: -32: -1] + ']')    # Seldom used
print('Answer: [' + sStr[-23: 6: -1] + ']')     # Seldom used
print()
print('[3] Get "peaceful" word...')
print('Answer: [' + sStr[21: 29] + ']')
print()
print('[4] Get "lufecaep" word...')
print('Answer: [' + sStr[28: 20: -1] + ']')
print('Answer: [' + sStr[-10: -18: -1] + ']')
print('Answer: [' + sStr[28: -18: -1] + ']')    # Seldom used
print('Answer: [' + sStr[-10: 20: -1] + ']')    # Seldom used
print()
print('[5] Switch "beautiful" and "peaceful" words...')
print('Answer: [What a peaceful and beautiful world~~~]')
# Haha, not good
print('Answer: [' + sStr[: 7] + sStr[21: 30] + sStr[17: 21] +
sStr[7: 17] + sStr[30: ] + ']')
print('Answer: [' +
      sStr[: 7] +
      sStr[21: 30] +
      sStr[17: 21] +
      sStr[7: 17] +
      sStr[30: ] +
      ']')

sPart1 = sStr[: 7]
sPart2 = sStr[7: 17]
sPart3 = sStr[17: 21]
sPart4 = sStr[21: 30]
sPart5 = sStr[30: ]
print('Answer: [' + sPart1 + sPart4 + sPart3 + sPart2 +
sPart5 + ']')
```

2.8.1.4. Ex4: Reverse String

1. Code+Output

p0401String-20211026.py | Output

Listing 2.8.1.4.1 /src/String/p0401String-20211026.py

```
#coding=utf-8
'''
Created on 2021年10月26日
```

```
4
5   @author: cph
6   '''
7   print('0123456789012345')
8   s = "Hello, World"
9   print(s)
10  print(s[: : -1])
11  print(s[len(s): : -1])
12  print(s[len(s): -(len(s) + 1): -1])
13  print(s[len(s): -13: -1])
```

2.8.1.5. Ex5: ord() vs. chr()

1. Code+Output

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

p0408OrdChr.py | Output

Listing 2.8.1.5.1 /src/String/p0408OrdChr.py

```
1   '''
2   author: cph
3   since: 2023020
4   '''
5   cChr = 'A'
6   print(f'cChr = {cChr}')
7   print(f'ord(cChr) = {ord(cChr)}')
8   print(f'chr(ord(cChr)) = {chr(ord(cChr))}')
9   print('---------------------------')
10  cChr = 'Z'
```

```
print(f'cChr = {cChr}')
print(f'ord(cChr) = {ord(cChr)}')
print(f'chr(ord(cChr)) = {chr(ord(cChr))}')
print('---------------------------')
cChr = 'a'
print(f'cChr = {cChr}')
print(f'ord(cChr) = {ord(cChr)}')
print(f'chr(ord(cChr)) = {chr(ord(cChr))}')
print('---------------------------')
cChr = 'z'
print(f'cChr = {cChr}')
print(f'ord(cChr) = {ord(cChr)}')
print(f'chr(ord(cChr)) = {chr(ord(cChr))}')
print('---------------------------')
cChr = '\n'
print(f'cChr = {cChr}')
print(f'ord(cChr) = {ord(cChr)}')
print(f'chr(ord(cChr)) = {chr(ord(cChr))}')
print('---------------------------')
cChr = '/'
print(f'cChr = {cChr}')
print(f'ord(cChr) = {ord(cChr)}')
print(f'chr(ord(cChr)) = {chr(ord(cChr))}')
print('---------------------------')
```

2.8.1.6. Ex6: String vs. Bytes

1. Python v3.x use UTF-8 as default coding.
2. There are two popular string types.

    a. string: Unicode inside the memory.

    b. bytes: binary data.

3. If we want to transmit data or save data to disk, we have to transform data from string to bytes.

```
str.encode([encoding='utf8'][, error='strict'])
```

where [error] has the following options, such as

> a. strict: as default
>
> b. ignore
>
> c. replace
>
> d. xmlcharrefrelpace

4. Code+Output

p0408StringEncode.py | Output

Listing 2.8.1.6.1 /src/String/p0408StringEncode.py

```
'''
author: cph
since: 20230718
'''
sStr = "Hello, 我愛上Python了..."
print(f'sStr 1: {sStr}')
print(f'sStr 1 length: {len(sStr)}')
sStr = sStr.encode('utf8')
print(f'sStr 2: {sStr}')
print(f'sStr 2 length: {len(sStr)}')
sStr = sStr.decode('utf8')
print(f'sStr 3: {sStr}')
print(f'sStr 3 length: {len(sStr)}')
```

2.8.2. Build-in Functions

1. str 本身有幾種常用的函數，在這裡列表說明：

Function	Description
bytes.split([sep=None,maxsplit=-1])	將字串以sep分割成子字串， 回傳儲存子字串的list， maxsplit為子字串最多的數量。
bytes.count(sub[, start[,end]])	計算sub出現的次數並以int型態回傳， start為起始計算索引值， end為結束索引值
bytes.find(sub[, start[,end]])	回傳sub第一次出現的索引值， 若sub不在其中則回傳-1。 start值與end值定義尋找的範圍。
bytes.index(sub[, start[,end]])	回傳sub第一次出現的索引值， 若sub不在其中則回傳ValueError。 start值與end值定義尋找的範圍。
bytes.replace(old, new[,count])	將bytes中的old子字串以new子字串代換。

Function	Description
	如果有給定count值， 則只有被給定的count值所代表的old子字串會被替換掉。
bytes.join(iterable)	將bytes中的old子字串以new子字串代換。 如果有給定count值， 則只有被給定的count值所代表的old子字串會被替換掉。
bytes.strip([chars])	將bytes中左邊與右邊的多餘餘空白全部移除。chars預設為空白， chars可以為想要刪除的字元組合。
bytes.lstrip([chars])	將bytes中左邊的多餘空白全部移除。 chars預設為空白， chars可以為想要刪除的字元組合。
bytes.rstrip([chars])	將bytes中右邊的多餘空白全部移除。 chars預設為空白， chars可以為想要刪除的字元組合。

2. 在上表所看見的[ ]中所包含之文字是可以自由選擇要不要打的， 如果需要裡面的功能， 就把需要的[ ]內容加上去， 反之則可以省略。

2.8.2.1. bytes.split([sep=None, maxsplit=-1])

1. bytes.split()的作用在於將字串作分割， 裡面有兩個選項屬性可以使用。
2. sep就是分割器， sep的值就是想拿來作分割器的值。 如果沒有給定sep值或是將sep值給予None值， 則會把空白字元當作分割器， 並且在回傳時自動把所有空白字元都刪除掉。
3. maxsplit是最多分割幾次， 預設為-1。 如果有給定非負整數的值， 例如給定2， 則會分割成最多2+1個子字串， 其範例如下所示：
4. Code+Output 1

p0402StringSplit.py | Output

Listing 2.8.2.1.1 /src/String/p0402StringSplit.py

```
1 #coding=utf-8
2 '''
3 Created on 2015年8月27日
4
```

```
@author: cph
'''
# 菲絲恩故意在各單字之間放置兩個半型空白
string = "Apple  is  a  kind  of  fruit."
print(string.split(sep =" "))       # 使用半型空白當作分割器
print(string.split())               # 不設定分割器，會自動以空白字元當作分割器
                                    # 並在回傳時捨棄子字串中的空白字元
print(string.split(sep = None))     # 跟不設定分割器的效果相同
print(string.split(maxsplit = 0))   # 設定最多只分割0 次，
                                    # 因此不會執行分割的動作
print(string.split(maxsplit = 2))   # 設定最多只分割2 次，
                                    # 因此會回傳3 個子字串
print(string.split(sep = None, maxsplit = 4))   # 設定最多只分割4 次，
                                    # 因此會回傳5 個子字串
```

5. Code+Output 2

p0402StringSplit-20220228.py | Output

Listing 2.8.2.1.2 /src/String/p0402StringSplit-20220228.py

```
#coding=utf-8
'''
Created on 2015年10月16日

@author: cph
@note: 在正常字串中，透過空白切割字串，且同時要切割出空白
'''
sA, sB = input("請輸入兩個數字【例如，3 5】: ").split()
print(sA, sB)
```

See also

1. Official Python Document: split
2. Official Python Document: lsplit
3. Official Python Document: rsplit

2.8.2.2. bytes.count(sub[, start[, end]])

1. bytes.count()的作用乃在於計算字串中有關sub子字串出現的次數。 有兩個選擇性選項屬性：start與end。

2. sub是一定要給值的， 否則Python不知道要在字串中尋找什麼來計數。
3. start負責接收int值， 代表從index給定的int 值開始計數。
4. end則負責接收int值， 代表計數的動作到index給定的int值之前即停止， 所以並不包含給定的int值的位置。
5. 這裡要特別注意的是， 當使用bytes.count()時， 是沒辦法只給定end值而不給定start值。以下提供示範程式碼作為說明：
6. Code+Output

p0403StringCount.py | **Output**

Listing 2.8.2.2.1 /src/String/p0403StringCount.py

```
'''
Created on 2015年8月27日

@author: cph
'''
# 只要問心無愧，無端的指責可以一笑置之。
i = "012345678901234567890123456789012345678901234567890123456789"
s = "A clear conscience laughs at false accusation."
print(s.count("A"))          # 在s 字串中統計大寫字母A出現的次數
print(s.count("a"))          # 在s 字串中統計小寫字母a 出現的次數
print(s.count("a", 6))       # 從index = 6之後開始統計
print(s.count("a", 0, 36))   # 從index = 0之後開始統計到index =
36
                             # 不包含36喔~~~
```

See also

1. Official Python Document: count

2.8.2.3. bytes.find(sub[, start[, end]])

1. bytes.find()的作用是在字串裡面找尋第一次出現sub子字串的index值， 一樣也有兩個選擇性選項屬性， 跟bytes.count()一樣是start跟end， 也都是接收int值。 以下提供示範程式碼來加以說明：
2. Code+Output

p0404StringFind.py | **Output**

Listing 2.8.2.3.1 /src/String/p0404StringFind.py

```
'''
Created on 2015年8月27日

@author: cph
'''
# 患難見真情
i = "01234567890123456789012345678901234567890123456789"
s = "A friend in need is a friend indeed."
print(s.find("friend"))          # 找尋s中friend子字串出現的index
print(s.find("friend", 3))       # 給定start值為3
print(s.find("friend", 3, 22))   # 給定start值為3且end值為22

print(s.find("in"))          # 找尋s中in子字串出現的index
print(s.find("in", 13))      # 給定start值為13
print(s.find("in", 13, 22))  # 給定start值為13且end值為22
```

3. 從上面的示範程式碼可以知道， find()會找尋母字串中， 子字串第一次完整出現的index值，所回傳的值是子字串第一個字母的index值。
4. 如果在搜尋範圍中沒有找到子字串，會回傳-1。

See also

a. Official Python Document: find

2.8.2.4. bytes.index(sub[, start[, end]])

1. bytes.index() 的功能跟str.find 幾乎一模一樣， 所擁有的選擇性選項屬性也相同； 唯一的差異點就是str.index在搜尋範圍中沒有找到子字串的時候不會回傳-1， 而是會回報一個名為【ValueError】的錯誤，例如：
2. Code+Output

p0405StringIndex.py | Output

Listing 2.8.2.4.1 /src/String/p0405StringIndex.py

```
'''
Created on 2015年8月27日

@author: cph
'''
# 玉不琢，不成器
i = "01234567890123456789012345678901234567890123456789"
s = "An uncut gem goes not sparkle."
print(s.index("o"))          # 使用index() 在s 裡面尋找‘o’，
                             # 並回傳'o' 第一次出現的index 值
```

```
print(s.index("t", 15))      # 給定start 值來尋找't'
print(s.index("a", 5, 15))  # 給定start 值及end 值，在其中尋找'a'
```

3. 如同上面所提到，index()跟find()的功能差別在於找不到子字串時回傳的資料不相同。

    a. find()會回傳-1，因此使用find()來找尋子字串不需要處理Error訊息。

    b. 相對的，使用index()就要準備Error 訊息的處理方式。

    c. 有關Error的錯誤處理，在後面的例外處理章節中會有詳細說明。

See also

1. Official Python Document: index

2.8.2.5. bytes.replace(old, new[, count])

1. bytes.replace()的功能是將舊的子字串old替換成新的子字串new， 有一個選擇性選項屬性：count。
2. count屬性如果有被給定int值(例如在此先假設int值為2)， str裡面前兩次出現的old子字串將會被new子字串取代， 第三次(如果有的話)以後出現的old子字串將不會被取代， 如下範例所示：
3. Code+Output 1

p0406StringReplace.py | Output

Listing 2.8.2.5.1 /src/String/p0406StringReplace.py

```
'''
Created on 2015年8月27日

@author: cph
'''
# 儲蓄方為賺錢之道
s = "A penny saved is a penny earned."
print(s.replace("n", "l"))      # 使用replace()將n用l取代
print(s)                        # replace()並不會真的更改s所參照物件的值
print(s.replace("penny", "dollar"))     # replace()將penny用dollar取代
print(s.replace("penny", "dollar", 1))  # 給定count值為1，
                                        # 故第二次出現的penny並不會被取代
```

> **Note**
>
> a. String s is still the same after replace() operation.
>
> b. That is, the replaced string is return by a new string.

4. Code+Output 2

p0402StringSplit2.py | Output

Listing 2.8.2.5.2 /src/String/p0402StringSplit2.py

```
#coding=utf-8
'''
Created on 2015年10月16日

@author: cph
@note: 在正常字串中，透過空白切割字串，且同時要切割出空白
'''
string = "Apple is a kind of fruit."
string = string.replace(" ", "* *")
print(string)
print(string.split(sep ="*"))       # 使用半型空白當作分割器
```

> **See also**
>
> 1. Official Python Document: replace

2.8.2.6. bytes.join(iterable)

1. iterable是疊代，例如序對、串列、集合、字典等等。
2. Code+Output

Code | Output

```
sStr  = "     I am a     Python man.       "
print("s = [" + sStr + "]")
lStr = sStr.split()                     # 切割成串列
print(lStr)
print("s = [" + " ".join(lStr) + "]")   # 使用空白來串接
print("s = [" + "*".join(lStr) + "]")   # 使用星號來串接
print("s = [" + "-".join(lStr) + "]")   # 使用減號來串接
print("s = [" + "__".join(lStr) + "]")  # 使用下標來串接
```

See also

1. Official Python Document: join

2.8.2.7. bytes.strip([chars]), bytes.lstrip([chars]), bytes.rstrip([chars])

1. Remote the pre-/post-spaces from a string, but NOT care about the duplicate spaces inside the string.
2. It do NOT replace the input source string.
3. Code+Output

Code Output

```
s = "     I am a     Python man.       "
print("s = [" + s + "]")
print("s = [" + s.rstrip() + "]")
print("s = [" + s.lstrip() + "]")
print("s = [" + s.strip() + "]")
print("s = [" + s + "]")
```

Note

1. A nice way to remove the duplicate spaces or special characters inside a string.
2. That is, it uses split() and join() of string.
3. Hence, we can split a string to a string list, then join all elements from the string list.
4. Source

```
s = "     I am a     Python man.       "
print("s = [" + s + "]")
print("s = [" + " ".join(s.split()) + "]")
```

5. Answer

```
s = [     I am a     Python man.       ]
s = [I am a Python man.]
```

See also

1. Official Python Document: strip
2. Official Python Document: lstrip
3. Official Python Document: rstrip

2.8.3. Ex7: Summary

1. 到了語言鑑定的考場， 菲絲恩安撫緊張的普羅說：「沒有什麼難的，把我昨天教你的拿出來用就對了！」

語言鑑定題目：

1. 令 line1 = “Welcome to Python World Game.” 請使用str.split以空格做分割。
2. 令 line2 = “Can you can a can as a canner can can a can?” 請使用str.count 算出有幾個’a’。
3. 令 line3 = “Few free fruit flies fly from flames.” 請使用str.find 或str.index 找出”fly” 第一次出現的位子。
4. 令 line4 = “Czngrxtulxtizns, yzu hxve pxssed the czmpetitizn.” 請使用str.replace 把’z’ 替換成’o’，以及把’x’替換成’a’。

1. Code+Output

Code Output

```
line1 = "Welcome to Python World Game."
print(line1.split())        # 以空格做分隔

line2 = "Can you can a can as a canner can can a can?"
print(line2.count("a"))     # 計算有幾個"a"

line3 = "Few free fruit flies fly from flames."
print(line3.find("fly"))    # 找出"fly"第一次出現的位置

line4 = "Czngrxtulxtizns, yzu hxve pxssed the czmpetitizn."
line5 = line4.replace("z", "o")   # 把"z"替換成"o"
print(line5.replace("x", "a"))    # 把"x"替換成"a"

print(line4.replace("z", "o").replace("x", "a"))
print(line4.replace("x", "a").replace("z", "o"))
```

Note

Usage: () vs. [] vs. {}

Symbol	Usage	Examples
()	Operator Tuple Function calls	if ((a = 1) and (b =2)): t = (1, 2, 3) r = f(a, b, c)
[]	String Index String Operation List Document	s = str[4] s = str[0: len(str): 2] l = [1, 2, 3] f(a=0 [,b=”[, c=None]])
{}	Set Dictionary	s = {1, 2, 3} d = {1: ‘a’, 2: ‘b’}

2.8.4. Exercise

2.8.4.1. Ex. 1

Question Code

```
請查詢附錄X的表格，將以下句子全部輸出成小寫，
"PLEASE CONVERT THIS SENTENCE TO LOWER CASE."
```

2.8.4.2. Ex. 2

Question Code

```
請修正下列程式碼，使其能正確執行：

bornYear = input("請輸入你的出生年份：")
nowYear = input("請輸入今年的年份：")
age = nowYear - bornYear
print("你今年%d歲" % age)
```

2.8.4.3. Ex. 3

Question Code

```
請修正下列程式碼，使其能正確執行：

gift = input("請輸入禮物份數：")
children = input("請輸入小朋友個數：")
oneGet = gift // children
print(gift + "個禮物分給" + children +"個小朋友，
每個小朋友可以分到" + oneGet + "個禮物")
```

2.8.4.4. Ex. 4

Question Code

```
金融卡完整卡號是16碼，
但不是每間銀行的帳號長度都會達16碼，
當未達16碼時，前面須補上0。
試著寫一段程式碼，
當輸入卡號「314159265359」時，
會輸出16位的整數。
```

2.8.4.5. Ex. 5

Question **Code**

```
請讓使用者個別輸入當天的年份、月份及日期，
再以格式化方式輸出。
假設今天是2010年1月1日，
使用者分別輸入「2010」、「1」、「1」，
然後程式再輸出2010.01.01。
```

Fig. 2.8.4.5.1 Photo by Randolfo Santos on Google Map, March 2022.

Note

1. Start: 20120311
2. System Environment:

Listing 2.8.4.5.1 requirements.txt

```
sphinx==7.1.2                           # Sphinx
graphviz>=0.20.1                        # Graphviz
sphinxbootstrap4theme>=0.6.0            # Theme: Bootstrap
sphinx-material>=0.0.35                 # Theme: Material
sphinxcontrib-plantuml>=0.25            # PlantUML
sphinxcontrib.bibtex>=2.5.0             # Bibliography
sphinx-autorun>=1.1.1                   # ExecCode: pycon
sphinx-execute-code-python3>=0.3        # ExecCode
btd.sphinx.inheritance-diagram>=2.3.1   # Diagram
sphinx-copybutton>=0.5.1                # Copy button
sphinx_code_tabs>=0.5.3                 # Tabs
sphinx-immaterial>=0.11.3               # Tabs

#-----------------------------------------------------
#-- Library Upgrade Error by Library Itself
#   >> It needs to fix by library owner
#   >> After fixed, we need to try it later
#-----------------------------------------------------
pydantic==1.10.10                       # 2.0: sphinx compiler
error, 20230701

#-----------------------------------------------------
#-- Minor Extension
#-----------------------------------------------------
sphinxcontrib.httpdomain>=1.8.1         # HTTP API

#sphinxcontrib-blockdiag>=3.0.0         # Diagram: block
#sphinxcontrib-actdiag>=3.0.0           # Diagram: activity
#sphinxcontrib-nwdiag>=2.0.0            # Diagram: network
#sphinxcontrib-seqdiag>=3.0.0           # Diagram: sequence

#-----------------------------------------------------
#-- Still Wait For Upgrading Version
#-----------------------------------------------------

#-----------------------------------------------------
#-- Still Under Testing
#-----------------------------------------------------
#numpy>=1.24.2                          # Figure: numpy

#-----------------------------------------------------
#-- NOT Workable
#-----------------------------------------------------
#sphinxcontrib.jsdemo==0.1.4    # ExecCode: Need replace
add_js_file()
#jupyter-sphinx==0.4.0          # ExecCode: Need gcc compiler
```

```
#sphinxcontrib.slide==1.0.0     # Slide: Slideshare
#hieroglyph==2.1.0              # Slide: make slides
#matplotlib>=3.7.1              # Plot: Need Python >= v3.8
#manim==0.17.2                  # Diagram: scipy, numpy need
gcc
#sphinx_diagrams==0.4.0         # Diagram: Need GKE access
#sphinx-tabs>=3.4.1                     # Tabs: Conflict w/
sphinx-material
```