

# Chapter 4

---

## 數字與字串

# 數字

---

數值在電腦中可使用二進位，八進位，及十六進位來儲存與表示

function	meaning
bin()	字串形式回傳二進位表示式
oct()	字串形式回傳八進位表示式
hex()	字串形式回傳十六進位表示式

# 數字

---

```
icy = 12345  
  
a = bin(icy)  
print(a, type(a))  
  
b = oct(icy)  
print(b, type(b))  
  
c = hex(icy)  
print(c, type(c))
```

PAUSE

```
0b11000000111001 <class 'str'>  
0o30071 <class 'str'>  
0x3039 <class 'str'>
```

請按任意鍵繼續 . . .

# 數字

---

數字型態中，浮點數又可以分成 float, complex 二種資料型態

float 便是指最常見的浮點數，用來保存倍精度浮點數類型的數值資料

complex 則是用來儲存複數類型的數值資料

數字分成實部及虛部，以兩個 float 來表示實部與虛部 ex.  $\langle 1.2 + 3j \rangle$  或  $\langle 3000 - 5j \rangle$

# 字串

---

字串是任何一個程式語言中都相當重要的一個變數  
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  
只要不以真／假、數值表示的資料都可以用字串來表示

Python 有內建字串的函數庫，多種函數可用來處理字串

# 字串

x="Hello Python" ，共有 12 個字元。

如果想要取出 x 裡面第 3 個字元，就得下 x[2] 這個指令。

要取出第 3 個字元，[ ] 裡面是 2

程式的世界索引大多從 0 開始算，所以 0 算到 2 剛好是 3

而 index 值也可以是負數

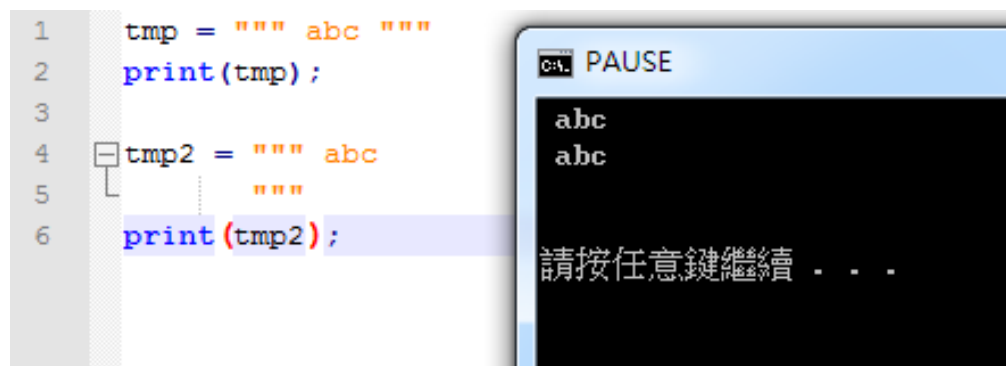
s	H	e	l	l	o		P	y	t	h	o	n
index	0	1	2	3	4	5	6	7	8	9	10	11
index	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

# 字串

---

宣告字串的方式可用一組單引號或雙引號括起來 ‘abc’, “abc”

使用三個單 ( 雙 ) 引號前後括住字串後可以任意換行



```
1 tmp = """ abc """
2 print(tmp);
3
4 tmp2 = """ abc
5         """
6 print(tmp2);
```

PAUSE

abc  
abc

請按任意鍵繼續 . . .

# 字串

---

字串本身具有順序性，也可從中間取出資料或插入資料

使用方式為 **變數名字** [ **start** : **end** : step ]

這三個參數都是由使用者自行選擇性輸入的

當沒有輸入參數時，**預設值** **start = 0**, **end** 為字串長度, **step** 為 1



# 字串

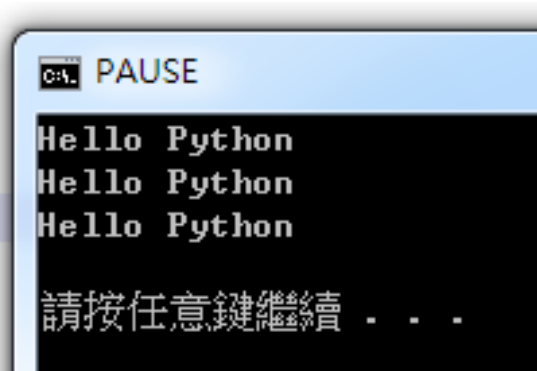
---

當沒有輸入參數時，預設值 `start = 0`，`end` 為 `len(string)`，`step` 為 1。

Note: `print(s)`, `print(s[:])` 與 `print(s[0:len(s):1])` 輸出的結果都是一樣的

```
test = "Hello Python"

print(test)
print(test[:])
print(test[0:len(test):1])
```



```
PAUSE
Hello Python
Hello Python
Hello Python
請按任意鍵繼續 . . .
```

# 字串

---

`string[ start : end : step ]`, 來點變化吧 !!!

```
print(test[4::]) #從index 4 開始輸出字串 test
print(test[::-2]) #每一次疊代 跳2個index
print(test[-1::-1]) # 反過來輸出
print(test[0:6:] + test2) # Hello Allen Iverson
```

PAUSE

```
o Python
HloPto
nohtyP olleH
Hello Allen Iverson
```

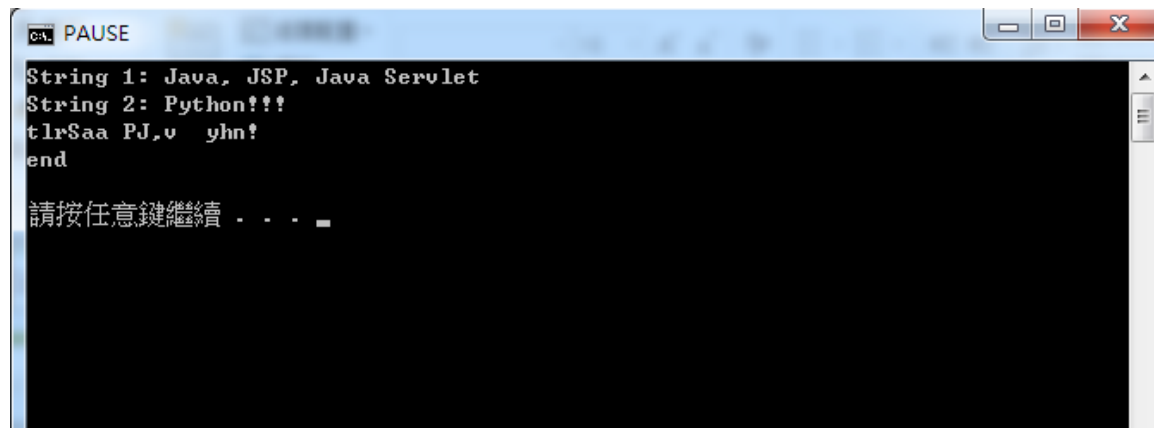
請按任意鍵繼續 . . .

# 隨堂練習

---

請寫出 Python 程式

1. “String 1, String 2” 使用 input()
2. 輸出為 tlrSaa PJ,v yhn!



The screenshot shows a Python interpreter window with a black background and white text. The text displayed is as follows:

```
String 1: Java, JSP, Java Servlet
String 2: Python!!!
tlrSaa PJ,v yhn!
end
請按任意鍵繼續 . . .
```

# 字串

## 幾個常用的字串函數

<code>split([sep=None, maxsplit=-1])</code>	字串以 <code>sep</code> 分割 , <code>maxsplit</code> 為子字串最大數量
<code>count(sub[, start[,end]])</code>	計算 <code>sub</code> 子字串出現的次數  <code>start</code> 為起始索引值 , <code>end</code> 為結束索引值
<code>find(sub[, start[,end]])</code>	回傳 <code>sub</code> 子字串第一次出現的索引值 , 若沒有則回傳 -1  <code>start</code> 與 <code>end</code> 定義範圍
<code>index(sub[, start[,end]])</code>	回傳 <code>sub</code> 子字串第一次出現的索引值  若沒有則回傳 <code>ValueError</code> , <code>start</code> 與 <code>end</code> 定義範圍
<code>replace(old, new[,count])</code>	字串中 <code>old</code> 子字串以 <code>new</code> 子字串替換 。 <code>count</code> 值  代表的 <code>old</code> 子字串會被替換掉幾次。

# 字串

---

`String.split([sep=None, maxsplit = -1])`

將 **sep** 的值為分割的依據，如果 **sep** 沒有設值則會把空白字元當作分割依據

**maxsplit** 是最多分割幾次，預設為 -1

如果有給定非負整數的值 (ex:2) ，則會分割成最多 2+1 個子字串

# 字串

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

```
String = "An apple a day, keeps a doctor away"

print(String.split(sep=" "))
print()

print(String.split())
print()

print(String.split(maxsplit = 1))
print()

print(String.split(sep=" ", maxsplit = 5))
print()
```

PAUSE

```
['An', 'apple', 'a', 'day,', 'keeps', 'a', 'doctor', 'away']
['An', 'apple', 'a', 'day,', 'keeps', 'a', 'doctor', 'away']
['An', 'apple a day, keeps a doctor away']
['An', 'apple', 'a', 'day,', 'keeps', 'a doctor away']
```

請按任意鍵繼續 . . .

# 字串

---

`String.count(sub[, start[, end]])`

計算字串中有關 sub 子字串出現的次數，而 sub 一定要給值

start 從給定的索引值開始計數

end 代表計數的動作到給定的索引值前即停止，**並不包含**給定的索引位置

# 字串


String.count(sub[, start[, end]])

```
String = "An apple a day, keeps a doctor away"

print(String.count("a") )

print(String.count("a", 0, 20) )

print(String.count("e", 8) )
```



PAUSE

6  
3  
2

請按任意鍵繼續 . . .



# 字串

---

`String.find(sub[, start[, end]])`

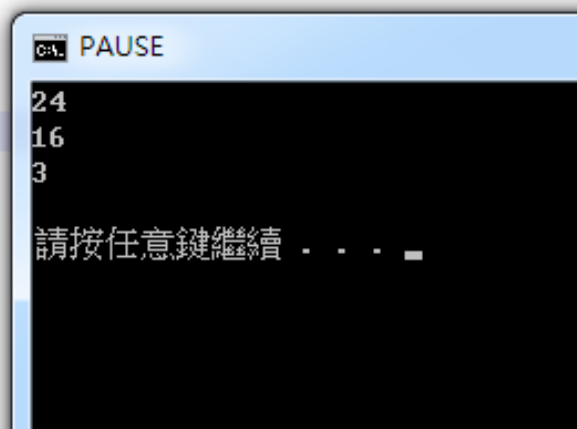
在字串裡面找尋第一次出現 sub 子字串的索引值

```
String = "An apple a day, keeps a doctor away"

print(String.find("doctor"))

print(String.find("keeps", 5) )

print(String.find("apple", 0 , 10))
```



# 字串

---

`String.index(sub[, start[, end]])`

和 `String.find()` 差異在若沒有找到子字串的時候不會回傳 -1, 而是回傳「`ValueError`」

```
String = "An apple a day, keeps a doctor away"
```

```
#print(String.find("doctor"))
```

```
#print(String.find("keeps", 5) )
```

```
#print(String.find("apple", 0 , 10))
```

```
print(String.index("apple"))
```

```
print(String.index("curry"))
```

PAUSE

```
3
Traceback (most recent call last):
  File "E:\course_code\python_code\ch-3\find.py", line 12, in <module>
    print(String.index("curry"))
ValueError: substring not found
```

請按任意鍵繼續 . . .

# 字串

`String.replace(old, new[, count])`

將舊的子字串換成新的子字串

count 若有給值 ( **ex count = 2** ), String 前兩次出現的 old 子字串將會被 new 子字串取代

第三次以後出現的 old 子字串將不會被取代

```
String = "An apple a day, keeps a doctor away"

print(String.replace("apple", "澎個" ) )

print(String.replace("doctor", "小護士" ) )

print(String.replace("a", "X", 3) )
```

PAUSE

```
An 澎個 a day, keeps a doctor away
An apple a day, keeps a 小護士 away
An Xpple X dXy, keeps a doctor away
```

請按任意鍵繼續 . . .

# 字串

## 其他字串函數

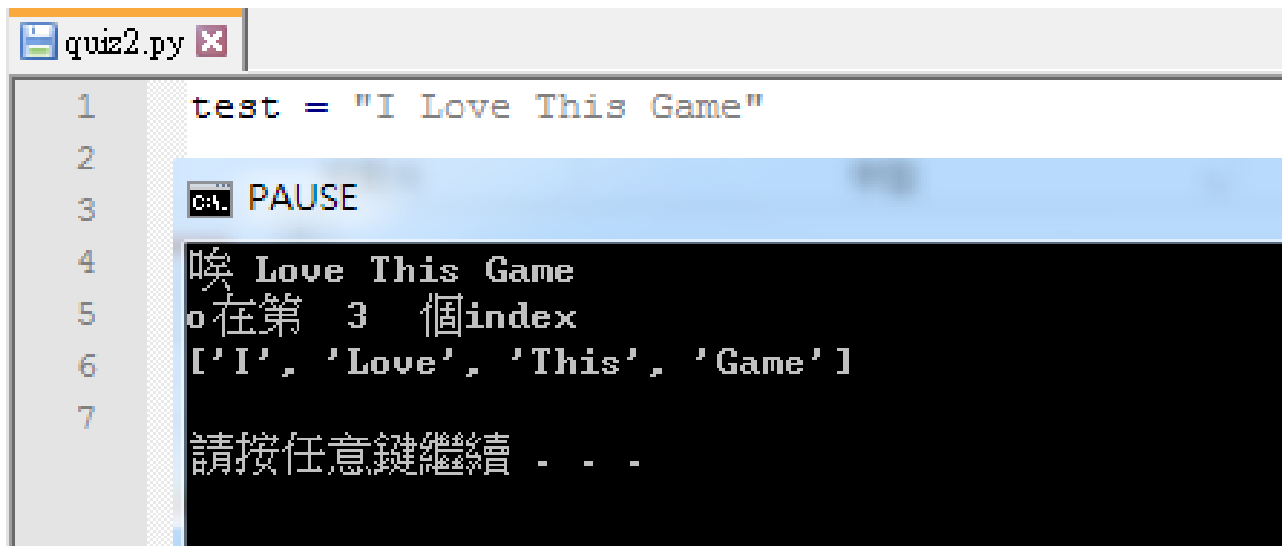
<b>isalnum()</b>	檢測字串是否僅為數字 (0-9) 或字母 (A-Z, z-a)
<b>isalpha()</b>	檢測字串是否僅為字母 (A-Z, z-a)
<b>isdigit()</b>	檢測字串是否僅為數字
<b>islower()</b>	檢測字串是否僅為小寫字母
<b>isupper()</b>	檢測字串是否僅為大寫字母
<b>swapcase()</b>	大寫字母轉小寫，小寫字母轉大寫
<b>***** len(string)</b>	<b>***** 計算字串長度 ( 這個很重要 ) *****</b>
<b>join()</b>	連接字串

# 隨堂練習

---

Test = "I Love This Game"

請寫 Python 程式輸出以下結果：



```
quiz2.py x
1 test = "I Love This Game"
2
3
4
5
6
7
```

PAUSE

唉 Love This Game  
o 在第 3 個index  
['I', 'Love', 'This', 'Game']  
請按任意鍵繼續 . . .

# 格式化

---

## 輸出格式化

以「%」與其他參數結合  
再將要輸出的資訊加以詳述

Type	meaning
%%	在字串中顯示 %
%d	10 進位整數輸出
%f	浮點數輸出
%e,%E	科學記號輸出
%o	以 8 進位整數方式輸出
%x,%X	以 16 進位整數方式輸出
%s	字串輸出
%c	單一字元輸出
%r	使用 repr() 輸出字串

# 格式化

---

輸出格式化 `print( “ 要輸出的內容 %type “ %( 變數 ) )`

```
icy = 12345
```

```
print("8進位 = %o " %(icy))
```

```
print("16進位 = %x " %(icy))
```

PAUSE

8進位 = 30071

16進位 = 3039

請按任意鍵繼續 . . .

# 格式化

浮點數的輸出也可以格式化，預設是小數點後第六位

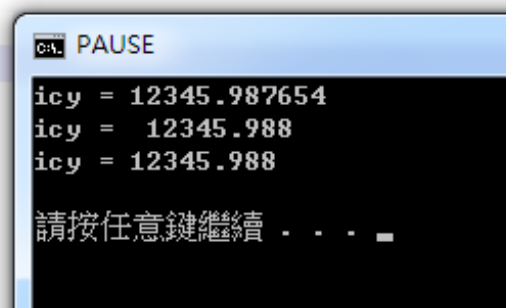
利用在「%」後面加上數字的方式，可控制輸出的長度及小數位數

```
icy = 12345.987654321

print("icy = %f" % (icy))

#.3f 指的是小數點後3位
#10 指的是輸出總長度為10，多的位數用空白取代，補在數字位置
print("icy = %10.3f" % (icy))

#-10 多的位數用空白取代，補在小數點位置
print("icy = %-10.3f" % (icy))
```



```
PAUSE
icy = 12345.987654
icy = 12345.988
icy = 12345.988

請按任意鍵繼續 . . .
```



# 格式化

字串輸出也可以格式化

在「%」後面加上數字，可以控制想輸出的長度及字元長度

```
print("%a.bs" %(Strings))
```

a: 總長度

b: 輸出幾位字元

```
Strings = "Stephen Curry 30"
```

```
print("%s" %(Strings))  
print("%.10s" %(Strings))  
print("%5.10s" %(Strings))  
print("%10.5s" %(Strings))
```

Ctrl PAUSE

```
Stephen Curry 30  
Stephen Cu  
Stephen Cu  
        Steph
```

請按任意鍵繼續 . . .

# 格式化

---

`repr()`: `%r`, 不管輸出的型態為何，都會用字串來輸出

```
Strings ="Stephen Curry 30"  
icy = 12345  
tmp = 'a'
```

```
print("%r" %(Strings) )  
print("%r" %(icy) )  
print("%r" %(tmp) )
```

PAUSE

```
'Stephen Curry 30'  
12345  
'a'
```

請按任意鍵繼續 . . .

# 格式化

## 格式化字串運算 format()

可以進行格式化輸出外，還可以針對內容進行運算

```
Strings = " {} + {} "  
  
print(Strings)  
print(Strings.format(1,2))  
print(Strings.format('a','b'))
```

```
tmp = " {Curry} "  
print(tmp)  
print(tmp.format(Curry = " Stephen Curry " ) )
```

PAUSE

```
<> + <>  
1 + 2  
a + b  
<Curry>  
Stephen Curry
```

請按任意鍵繼續 . . .

# 格式化

---

格式化字串運算 `format()`

其他參數使用方式 { 欄位 : format-spec }

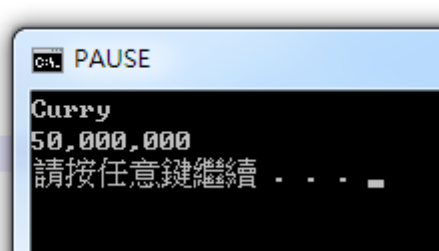
format-spec	格式化符號	meaning
align	<, >, ^	<( 靠左 ) >( 靠右 ) ^( 置中 )
sign	- +	可以使用符號 (-) (+)
width		用數值表示欄寬

# 格式化

## 格式化字串運算 `format()`

參數使用方式：欄位 · `format-spec`

```
name = "Curry"
money = 50000000
print('{0}\n{1:,d}'.format(name,money))
```



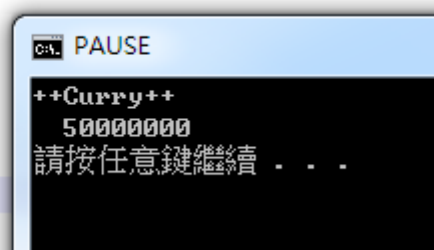
格式化符號	meaning
{0}	format 接的第一個參數
\n	換行
{1:,d}	第二個參數以及是數字的化千位數會輸出” ,”

# 格式化

## 格式化字串運算 `format()`

參數使用方式：欄位 · `format_spec`

```
name = "Curry"
money = 50000000
print('{0: +^9}\n{1: >10}'.format(name, money))
```



```
PAUSE
++Curry++
50000000
請按任意鍵繼續 . . .
```

格式化符號	meaning
{+^9}	^ 置中對齊，欄寬 9，不足 9 的補 +
{>10}	> 靠右對齊，欄寬 10，不足 10 的補空格

# 型態轉換

---

有時目前的資料型態並不能輸出想要的格式

用轉換函數將變數轉換成其他資料格式

Function	meaning
int(x)	將 x 轉換成整數
float(x)	將 x 轉換成浮點數。
str(x)	將 x 轉換為字串。
tuple(x)	將 x 轉換為 tuple。
list(x)	將 x 轉換為 list。
chr(x)	將 x 轉換為字元。

# 型態轉換

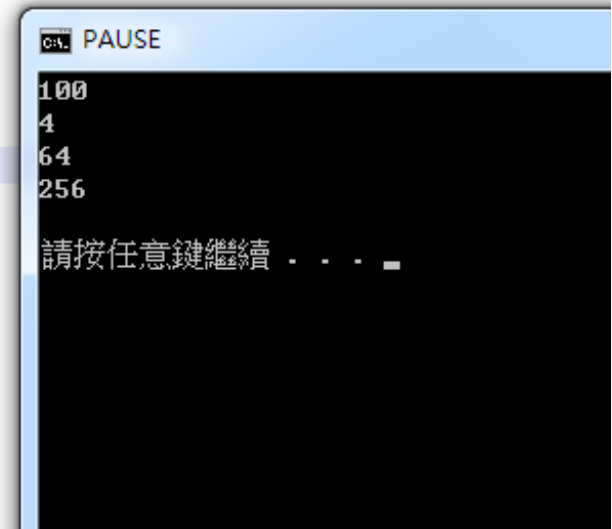
`int(test,2)`

1. 先轉成整數
2. 用 2 進位方式呈現 100
- … 以此類推

```
test = "100"

a = int(test)
b = int(test,2)
c = int(test,8)
d = int(test,16)

print(a)
print(b)
print(c)
print(d)
```



```
PAUSE
100
4
64
256
請按任意鍵繼續 . . .
```



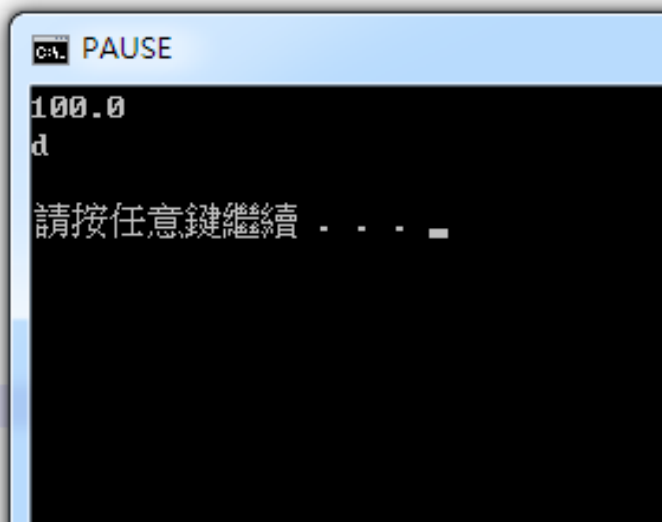
# 型態轉換

```
test = "100"  
  
a = int(test)  
b = int(test,2)  
c = int(test,8)  
d = int(test,16)
```

```
#print(a)  
#print(b)  
#print(c)  
#print(d)
```

```
e = float(a)  
f = chr(a)
```

```
print(e)  
print(f)
```



為什麼 f 輸出是 d ???

# 型態轉換

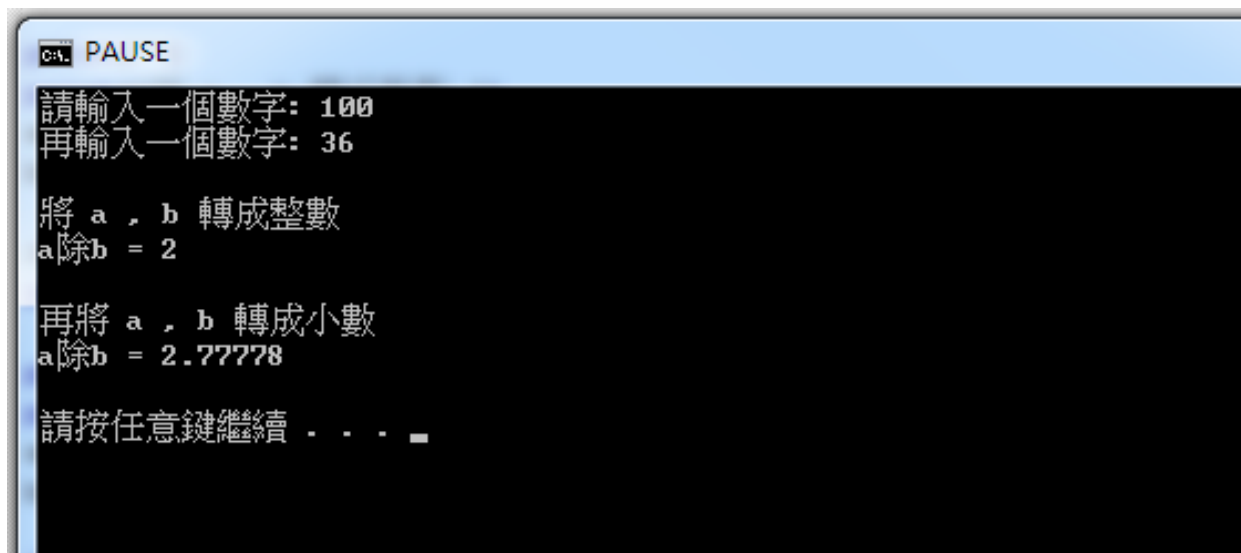
	A	B	C	D	E	F	G	H	I	J	K	L
1	10進制	16進制	字元	10進制	16進制	字元	10進制	16進制	字元	10進制	16進制	字元
2	33	21	!	57	39	9	81	51	Q	105	69	i
3	34	22	"	58	3A	:	82	52	R	106	6A	j
4	35	23	#	59	3B	;	83	53	S	107	6B	k
5	36	24	\$	60	3C	<	84	54	T	108	6C	l
6	37	25	%	61	3D	=	85	55	U	109	6D	m
7	38	26	&	62	3E	>	86	56	V	110	6E	n
8	39	27	'	63	3F	?	87	57	W	111	6F	o
9	40	28	(	64	40	@	88	58	X	112	70	p
10	41	29	)	65	41	A	89	59	Y	113	71	q
11	42	2A	*	66	42	B	90	5A	Z	114	72	r
12	43	2B	+	67	43	C	91	5B	[	115	73	s
13	44	2C	,	68	44	D	92	5C	\	116	74	t
14	45	2D	-	69	45	E	93	5D	]	117	75	u
15	46	2E	.	70	46	F	94	5E	^	118	76	v
16	47	2F	/	71	47	G	95	5F	_	119	77	w
17	48	30	0	72	48	H	96	60	`	120	78	x
18	49	31	1	73	49	I	97	61	a	121	79	y
19	50	32	2	74	4A	J	98	62	b	122	7A	z
20	51	33	3	75	4B	K	99	63	c	123	7B	{
21	52	34	4	76	4C	L	100	64	d	124	7C	
22	53	35	5	77	4D	M	101	65	e	125	7D	}
23	54	36	6	78	4E	N	102	66	f	126	7E	~
24	55	37	7	79	4F	O	103	67	g	127	7F	
25	56	38	8	80	50	P	104	68	h	128	80	

Ascii code, from [isvincent.pixnet.net](http://isvincent.pixnet.net)

# 隨堂練習

---

1. 輸入二個數字
2. 輸出以下結果：



```
CA. PAUSE
請輸入一個數字: 100
再輸入一個數字: 36

將 a , b 轉成整數
a除b = 2

再將 a , b 轉成小數
a除b = 2.77778

請按任意鍵繼續 . . .
```