輸出Hello World

In [1]:

```
'Hello World'
```

Out[1]:

```
'Hello World'
```

輸出數值與文字

雖然可以直接輸出結果，一般會使用print()函式來印出值

print(值或運算式)

In [6]:

```
30
```

Out[6]:

```
30
```

In [13]:

```
#print(值或運算式)
print(42)
```

```
42
```

In [7]:

```
#數值相加
print(3+8)
```

```
11
```

算符與運算式

加、減、乘、除、求除法的商、求除法的餘數、次方、小括號

In [8]:

```
print(3+5)
print(3-5)
print(3*5)
print(3/5)
print(3//5)
print(3%5)
print(3**5)
print((3+5)*3+5)
```

```
8
-2
15
0.6
0
3
243
29
```

字串對字串、字串對數字

In [9]:

```
#字串加字串
print('3'+'5')
```

```
35
```

In [10]:

```
#字串乘數字
print('3'*5)
```

```
33333
```

In [11]:

```
'3'*5
```

Out[11]:

```
'33333'
```

變數

寫程式時會重複使用到某些資料，一再輸入會很麻煩。我們可以給這些資料取個名字，以後呼叫該名字時就能使用該資料了，這些名稱就稱為變數

變數值 = 值。等號 (=) 不是數學上的相等，而是將等號右邊的值指派(assign)給左邊的名稱。= 稱為指派算符 (assignment operator)

變數名稱由文字組成，可包括大小寫英文字母、數字和底線，使用時也不需要用單引號或雙引號括住它。但變數命名有些限制: 1. 不能數字開頭 2. 不能使用Python保留的關鍵字(比如 if 、for等) 3. 不要使用已經定義的函示名稱(比如print、list等)，這麼做會使該名稱變成變數，使你無法再使用原函式(直到重新啟動Python環境為止)

In [14]:

```
n = '加油'
print(n)
```

加油

更新變數的值

更新變數值的方式，使用=來賦予一個新值(因此第一次使用=時會建立該名稱的變數，在這之後使用=則會改變變數的值)

In [20]:

```
#建立變數名稱x
x = 1
print(x)

#修改x的值
x = 5
print(x)

#修改x的值(將x加1)
x= x +1
print(x)

#寫法等同於 x = x + 1
x += 1
print(x)
```

```
1
5
6
7
```

Python 的基本型別有以下幾種 : 1. int (數值) 2. float (浮點數) 3. str (字串) 4. bool (布林值)

type(變數名稱或值)

In [2]:

```
a = 50

#用print()將type()傳回的結果印出
print(type(a))

b =  3.14
print(type(b))

c = 'Hello'
print(type(c))

#變數指向不同型別資料，其型別就隨之改變
```

```
<class 'int'>
<class 'float'>
<class 'str'>
```

跨型別運算問題

In [5]:

```
'7' * 5
```

Out[5]:

```
'77777'
```

In [6]:

```
#字串跟整數相加會產生錯誤
'8' + 6

#因為字串只能跟字串相連
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-6-6fc7e0b7cd5c> in <module>
      1 #字串跟整數相加會產生錯誤
----> 2 '8' + 6
      3
      4 #因為字串只能跟字串相連

TypeError: can only concatenate str (not "int") to str
```

In [7]:

```
#數字加字串也會出錯
5 + '4'

#因為不支援讓整數加字串
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-7-969aefef54bc> in <module>
      1 #數字加字串也會出錯
----> 2 5 + '4'

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

對不同型別資料來說，算符可能會有不同意義。例如：對字串而言, +是串接 (concatenate)而不是相加

型別轉換 1. int(值)：轉換成整數 2. float (值)：轉換成浮點數 3. str(值)：轉換成字串

ps：若要將字串轉為數值，字串內容必須是合法的數值，否則轉換時將產生錯誤 。浮點數轉為整數時，會令小數點被直接捨去

In [9]:

```
#將字串'3'轉換為整數 3
print(int('2') + 8)
print('2' + str(7))
print(float(2) + 6)
print(int(3.14) + 4)
```

```
10
27
8.0
7
```

比較算符

值 1 == 值 2。此式子稱為比較運算式，會回傳布林值(bool)型別的值(True或False)

Python提供的比較算符 1. == 等於 2. != 不等於 3. > 大於 4. >= 大於等於 5. < 小於 6. <= 小於等於

In [11]:

```
print(1+1 !=3)
print(4+6 ==10)

x =3
y= 8
print(x+y >7)
print(x+y <=7)
```

```
True
True
True
False
```

if 判斷式

if 敘述語法：

```
  if 條件判斷式 :
      程式區塊
```

ps : if ...這行結尾必須加上冒號 (:) , 程式區塊也必須向右縮排----------使用縮排來區別程式區塊是Python語言特色
補充 : 在Colab中，若程式碼結尾是冒號，按下enter換行後編輯器會自動加上縮排

In [1]:

```
n = 10
print('檢查數字....')
if n >= 10:
    print('數字超過門檻')
```

```
檢查數字....
數字超過門檻
```

if ..... else.....語法：

```
   if 條件判斷式 :
       程式區塊
   else :
       程式區塊
```

In [2]:

```
n = 5
print('檢查數字...')
if n >=10:
    print('數字超過門檻')
else :
    print('數字未達門檻')
```

```
檢查數字...
數字未達門檻
```

if....elif...else.....語法：

```
   if 條件判斷式 1 :
```

```
        程式區塊
    elif 條件判斷式 2 :
        程式區塊
    elif 條件判斷式 3 :
        程式區塊

    ....
    else:
        程式區塊
```

補充 : if...elif...是由上往下判斷，因此前面不成立的條件在後面就隱含成立，不須再判斷一次了。因此在撰寫if...elif...else時，要特別注意條件判
斷式的順序內容

In [3]:

```
n = 16

if n >= 20:
    print(n,'是很大的數字')
elif n >= 15:
    print(n,'是中等的數字')
elif n >= 10:
    print(n, '是普通的數字')
else :
    print(n,'是較小的數字')
```

16 是中等的數字

and 、or、not 。可以在 if 或 elif 後面使用多個條件判斷式，並用and 以及 or 算符把它們串聯起來：

語法：

```
條件判斷式 1 and 條件判斷式 2
條件判斷式 1 or 條件判斷式 2
```

ps : 用 and 串連的條件判斷式，兩者接為True時才傳回True，否則傳回False
or 串連的條件判斷式，只要有任一為True時就會傳回True
not 算符可用來反轉條件判斷式的布林值。語法 : not 條件判斷式

In [7]:

```
x = 13
y = 27

# y<14 不成立，故整個式子得到 False
print(x >8 and y < 14)

# x>8 成立，故就算 x <14 不成立，整個式子仍為True
print(x >8 or y < 14)

#not y<14 使得 False轉True，故整個式子得到True
print(x >8 and not y < 14)
```

```
False
True
True
```

In [8]:

```
#判斷是否為閏年 (判斷規則為[四年一閏，百年不閏，四百年再閏])

year = 2020
if (year % 4 == 0 and year % 100 != 0 ) or year %400 ==0 :
    print(year,'是閏年')
else :
    print(year,'是平年')
```

```
2020 是閏年
```