```
int x = 10; # 常數定義法，這是註解

flo y = 10.1 + int2flo(x) / 3.0;  # 浮點數

str name = "John"; # 字串，預設用utf-8

bool c = True; # 布林值，或是 False

'''
這是多行註解

函數上面的多行註解會轉成 docString，先用 markdown 做子語言吧；

底下爲列表
'''
List(int) a_list = [1, 2, 3, 4, 5];

Array(int) a_array = array![1, 2, 3, 4, 5];

# 以下是 doc string 的範例：

'''
Descr : find the sqrt sum of x and y
Arguments :
 - x : lfs value
 - y : rhs value
Example:
    sqrtsum(3, 4) == 25; # True
'''
fn sqrtSum = int x, int y -> int :
             int z = x ** 2 + y ** 2;
             return z;

fn isJohn = str s -> bool :
             return case {
               #print! 是巨集。!結尾是巨集名稱。 ++ 是字串相加
               s == john -> {print!("Hi, " ++ s ++ "!\n");
                             True;}
               else -> False;
               };

#不返回值(void)的時候也要標註 return;
fn printCat = void -> void :
                       print!("cat!");
                       return ;

# 多型：
# @{} vars of Type with constraints
fn map = @(A, B  in {x | contains(x.attrs, "Any")}) # or @(A, B in Any)
              (List A) origList; ( A -> B ) aFunction -> (List B) :
          return match origList{
```

```
            [] -> origList;
            [x:xs] -> cons(aFunction(origList),
                           map(origList[1:], aFunction));
        };

# 定義自定型別：
type Person = Person(str name, int age);
type TrafficLight = Red | Yellow | Green;
type List = @{A in Any} Nil | Cons a (List A);

Person peter = Person{"Peter", 17};

debug!(peter);
```