Julia 超新手教學 II

by 杜岳華

Outline

- Collections
- String and Operators
- Functions
- Types

Collections

同類型的變數不只有一個怎麼辦？

Arrays

在程式語言當中最基本的集合或是資料結構

Create an array

In [1]:
```
x = []
```

Out[1]:
```
0-element Array{Any,1}
```

Homogeneous: 同質性，Array中只能放入屬於同一型別的物件

In [2]:
```
Any[]
```

Out[2]:
```
0-element Array{Any,1}
```

In [3]:
```
Int64[]
```

Out[3]:
```
0-element Array{Int64,1}
```

Type inference on array

In [4]:
```
x = [1, 2, 3]
```

Out[4]:
```
3-element Array{Int64,1}:
 1
 2
 3
```

In [5]:
```
x = [1, 1.2]
```

Out[5]:
```
2-element Array{Float64,1}:
 1.0
 1.2
```

Specified array type

In [6]:
```
Int8[1, 2, 3, 4]
```

Out[6]:
```
4-element Array{Int8,1}:
 1
 2
 3
 4
```

In [7]:
```
Array{Int8, 1}(5)  # 尚未初始化
```

```
MethodError: no method matching Array{Int8,1}(::Int64)
Closest candidates are:
  Array{Int8,1}() where T at boot.jl:413
  Array{Int8,1}(!Matched::UndefInitializer, !Matched::Int64) where T at boot.jl:394
  Array{Int8,1}(!Matched::UndefInitializer, !Matched::Int64...) where {T, N} at boot.jl:400
  ...

Stacktrace:
 [1] top-level scope at In[7]:1
```

Indexing

Index starts from 1.

□ □ □

1 2 3

In [8]:
```
x
```

Out[8]:
```
2-element Array{Float64,1}:
 1.0
 1.2
```

In [9]:
```
x[1]
```

Out[9]:
```
1.0
```

In [10]:
```
x[2]
```

Out[10]:
```
1.2
```

In [11]:
```
length(x)
```

Out[11]:
```
2
```

In [12]:
```
x = [6.0, 3.2, 7.6, 0.9, 2.3]
```

Out[12]:
```
5-element Array{Float64,1}:
 6.0
 3.2
 7.6
 0.9
 2.3
```

In [13]:
```
x[1:2]
```

Out[13]:
```
2-element Array{Float64,1}:
 6.0
 3.2
```

In [14]:
```
x[3:end]
```

Out[14]:
```
3-element Array{Float64,1}:
 7.6
 0.9
 2.3
```

In [15]:
```
x[1:2:end]
```

Out[15]:
```
3-element Array{Float64,1}:
 6.0
 7.6
 2.3
```

Assign value

In [16]:
```
x[2] = 7.5
```

Out[16]:
```
7.5
```

In [17]:
```
x
```

Out[17]:
```
5-element Array{Float64,1}:
 6.0
 7.5
 7.6
 0.9
 2.3
```

Useful operations

In [18]:
```
push!(x, 9.0)
```

Out[18]:
```
6-element Array{Float64,1}:
 6.0
 7.5
 7.6
 0.9
 2.3
 9.0
```

In [19]:
```
y = [10.0, 3.4]
append!(x, y)
```

Out[19]:
```
8-element Array{Float64,1}:
  6.0
  7.5
  7.6
  0.9
  2.3
  9.0
 10.0
  3.4
```

In [20]:
```
x
```

Out[20]:
```
8-element Array{Float64,1}:
  6.0
  7.5
  7.6
  0.9
  2.3
  9.0
 10.0
  3.4
```

In [21]:
```
pop!(x)
```

Out[21]:
```
3.4
```

In [22]:
```
x
```

Out[22]:
```
7-element Array{Float64,1}:
  6.0
  7.5
  7.6
  0.9
  2.3
  9.0
 10.0
```

In [23]:
```
popfirst!(x)
```

Out[23]:
```
6.0
```

In [24]:
```
x
```

Out[24]:
```
6-element Array{Float64,1}:
  7.5
  7.6
  0.9
  2.3
  9.0
 10.0
```

In [25]:
```
pushfirst!(x, 6.0)
```

Out[25]:
```
7-element Array{Float64,1}:
  6.0
  7.5
  7.6
  0.9
  2.3
  9.0
 10.0
```

Random array

In [26]:
```
x = rand(5)
```

Out[26]:
```
5-element Array{Float64,1}:
 0.21453141264164777
 0.33911519878720675
 0.1736605111037608
 0.924091604483507
 0.9313225278498267
```

In [27]:
```
sort(x)
```

Out[27]:
```
5-element Array{Float64,1}:
 0.1736605111037608
 0.21453141264164777
 0.33911519878720675
 0.924091604483507
 0.9313225278498267
```

In [28]:
```
x
```

Out[28]:
```
5-element Array{Float64,1}:
 0.21453141264164777
 0.33911519878720675
 0.1736605111037608
 0.924091604483507
 0.9313225278498267
```

In [29]:
```
sort!(x)
```

Out[29]:
```
5-element Array{Float64,1}:
 0.1736605111037608
 0.21453141264164777
 0.33911519878720675
 0.924091604483507
 0.9313225278498267
```

In [30]:
```
x
```

Out[30]:
```
5-element Array{Float64,1}:
 0.1736605111037608
 0.21453141264164777
 0.33911519878720675
 0.924091604483507
 0.9313225278498267
```

由大到小

In [31]:
```
sort(x, rev=true)
```

Out[31]:
```
5-element Array{Float64,1}:
 0.9313225278498267
 0.924091604483507
 0.33911519878720675
 0.21453141264164777
 0.1736605111037608
```

依絕對值大小排序

In [32]:
```
x = randn(10)
```

Out[32]:
```
10-element Array{Float64,1}:
 -1.1965110518069177
 -2.5673236261556553
 -0.8304415553505384
 -0.41761711155050135
 -0.5774989482136595
 -0.6295961416534364
  0.5457415460575774
  0.6613047934817022
  0.8509853754213738
 -0.0008691151310652138
```

In [33]:
```
sort(x, by=abs)
```

Out[33]:
```
10-element Array{Float64,1}:
 -0.0008691151310652138
 -0.41761711155050135
  0.5457415460575774
 -0.5774989482136595
 -0.6295961416534364
  0.6613047934817022
 -0.8304415553505384
  0.8509853754213738
 -1.1965110518069177
 -2.5673236261556553
```

Iteration

In [34]:
```
for i in x
    println(i)
end
```

```
-1.1965110518069177
-2.5673236261556553
-0.8304415553505384
-0.41761711155050135
-0.5774989482136595
-0.6295961416534364
0.5457415460575774
0.6613047934817022
0.8509853754213738
-0.0008691151310652138
```

Quiz 1

請造出一個陣列，當中的數值是均勻分佈，從-345到957.6

提示：$y = \frac{x - min(x)}{max(x) - min(x)}$

其中一個答案

In [35]:
```
(957.6 - (-345)) * rand(10) .+ (-345)
```

Out[35]:
```
10-element Array{Float64,1}:
 -334.48734547212007
  107.67339218509932
 -301.6266125118421
  579.6664242648574
   -3.3131282827195605
 -218.11748943365666
 -197.9126563168624
  795.3918456081692
  760.8636633468809
  739.1140171035474
```

Quiz 2

請造出一個陣列，當中的數值是服從常態分佈

其中一個答案

In [36]:
```
randn(10)
```

Out[36]:
```
10-element Array{Float64,1}:
 -0.8872584043918716
 -1.5664265699276245
  0.10207006688085296
 -0.15338627790089324
 -1.1024843383816594
 -0.3169308401620576
  0.46143380744414936
  0.3572804043192466
  1.9469758269259256
 -0.9657416007563445
```

Quiz 3

請造出一個陣列，當中的數值是服從常態分佈，μ=3.5，σ=2.5

提示：$y = \frac{x-\mu}{\sigma}$

其中一個答案

In [37]:
```
2.5 * randn(10) .+ 3.5
```

Out[37]:
```
10-element Array{Float64,1}:
 5.024562407066564
 2.937093585978778
 3.844077484294863
 4.55676041417134
 8.69544296211776
 2.648919459352561
 3.4218683932152327
 1.0016142723499661
 3.6143944979691147
 1.5689530575021093
```

Sets

數學上的集合

In [38]:
```
x = Set([1, 2, 3, 4])
```

Out[38]:
```
Set([4, 2, 3, 1])
```

In [39]:
```
push!(x, 5)
```

Out[39]:
```
Set([4, 2, 3, 5, 1])
```

In [40]:
```
pop!(x)
```

Out[40]:
```
4
```

In [41]:
```
x
```

Out[41]:
```
Set([2, 3, 5, 1])
```

Exists

In [42]:
```
3 in x
```

Out[42]: true

In [43]:
```
4 in x
```

Out[43]: false

Equivalent

In [44]:
```
x == Set([3, 2, 1, 5])
```

Out[44]:
```
true
```

Iteration

In [45]:
```
for i in x
    println(i)
end
```

```
2
3
5
1
```

Quiz 4

請告訴我以下資料有幾種數值

[8, 4, 1, 2, 9, 4, 5, 4, 5, ...]

In [46]:
```
x = rand([1, 2, 4, 5, 8, 9], 50);
```

In [47]:
```
Set(x)
```

Out[47]:
```
Set([2, 9, 8, 4, 5, 1])
```

Dictionaries

key-value 的資料結構

In [48]:
```
x = Dict("1" => 1, "2" => 2, "3" => 3)
```

Out[48]:
```
Dict{String,Int64} with 3 entries:
  "1" => 1
  "2" => 2
  "3" => 3
```

In [49]:
```
x["1"]
```

Out[49]:
```
1
```

In [50]:
```
x["A"]
```

```
KeyError: key "A" not found

Stacktrace:
 [1] getindex(::Dict{String,Int64}, ::String) at ./dict.jl:478
 [2] top-level scope at In[50]:1
```

Add new pair

In [51]:
```
x["4"] = 4
```

Out[51]:
```
4
```

In [52]:
```
x
```

Out[52]:
```
Dict{String,Int64} with 4 entries:
  "4" => 4
  "1" => 1
  "2" => 2
  "3" => 3
```

Overwrite

In [53]:
```
x["1"] = 5
```

Out[53]:
```
5
```

In [54]:
```
x
```

Out[54]:
```
Dict{String,Int64} with 4 entries:
  "4" => 4
  "1" => 5
  "2" => 2
  "3" => 3
```

keys and values

In [55]:
```
keys(x)
```

Out[55]:
```
Base.KeySet for a Dict{String,Int64} with 4 entries. Keys:
  "4"
  "1"
  "2"
  "3"
```

In [56]:
```
values(x)
```

Out[56]:
```
Base.ValueIterator for a Dict{String,Int64} with 4 entries. Values:
  4
  5
  2
  3
```

Iteration

In [57]:
```
for (k, v) in x
    println(k, "->", v)
end
```

```
4->4
1->5
2->2
3->3
```

Strings

字串是很常用到的物件

但是字串並不是最基本的元素

Characters

字元是組成字串的基本單元

In [58]:
```
'A'
```

Out[58]:
```
'A': ASCII/Unicode U+0041 (category Lu: Letter, uppercase)
```

In [59]:
```
'a'
```

Out[59]:
```
'a': ASCII/Unicode U+0061 (category Ll: Letter, lowercase)
```

字元用單引號，字串用雙引號

In [60]:
```
typeof('A')
```

Out[60]:
```
Char
```

In [61]:
```
typeof("A")
```

Out[61]:
```
String
```

字元其實是用相對應的整數表示的

In [62]:
```
Int('A')
```

Out[62]:
```
65
```

In [63]:
```
Char(65)
```

Out[63]:
```
'A': ASCII/Unicode U+0041 (category Lu: Letter, uppercase)
```

In [64]:
```
Int('B')
```

Out[64]:
```
66
```

字元能適用加法嗎？

In [65]:
```
'A' + 1
```

Out[65]:
```
'B': ASCII/Unicode U+0042 (category Lu: Letter, uppercase)
```

In [66]:
```
'C' - 2
```

Out[66]:
```
'A': ASCII/Unicode U+0041 (category Lu: Letter, uppercase)
```

字元可以比較大小嗎？

In [67]:
```
'C' > 'A'
```

Out[67]:
```
true
```

In [68]:
```
'a' > 'A'
```

Out[68]:
```
true
```

In [69]:
```
Int('a')
```

Out[69]:
```
97
```

In [70]:
```
'a' - 'A'
```

Out[70]:
```
32
```

Strings

In [71]:
```
x = "Hello World!"
```

Out[71]:
```
"Hello World!"
```

In [72]:
```
"""Hello World!"""
```

Out[72]:
```
"Hello World!"
```

In [73]:
```
"""Hello
World
!
"""
```

Out[73]:
```
"Hello\nWorld\n!\n"
```

Indexing

In [74]:
```
x[1]
```

Out[74]:
```
'H': ASCII/Unicode U+0048 (category Lu: Letter, uppercase)
```

In [75]:
```
x[end-1]
```

Out[75]:
```
'd': ASCII/Unicode U+0064 (category Ll: Letter, lowercase)
```

In [76]:
```
x[3:5]
```

Out[76]:
```
"llo"
```

Unicode and UTF-8

In [77]:
```
s = "\u2200 x \U2203 y"
```

Out[77]:
```
"∀ x ∃ y"
```

In [78]:
```
s[1]
```

Out[78]:
```
'∀': Unicode U+2200 (category Sm: Symbol, math)
```

In [79]:
```
s[2]
```

```
StringIndexError("∀ x ∃ y", 2)

Stacktrace:
 [1] top-level scope at In[79]:1
```

用來告訴你下一個index

In [80]:
```
nextind(s, 1)
```

Out[80]:
```
4
```

In [81]:
```
s[4]
```

Out[81]:
```
' ': ASCII/Unicode U+0020 (category Zs: Separator, space)
```

Operators

In [82]:
```
length("123456")
```

Out[82]: 6

Interpolation

In [83]:
```
x = "Today"
y = "Sunday"
string(x, " is ", y)
```

Out[83]: "Today is Sunday"

In [84]:
```
"$x is $y"
```

Out[84]: "Today is Sunday"

In [85]:
```
"1 + 2 = $(1 + 2)"
```

Out[85]: "1 + 2 = 3"

Equivalent

In [86]:
```
"1 + 2 = 3" == "1 + 2 = $(1 + 2)"
```

Out[86]:
```
true
```

Contains substring

In [87]:
```
occursin("na", "banana")
```

Out[87]:
```
true
```

Repeat

In [88]:
```
repeat(x, 10)
```

Out[88]:
```
"TodayTodayTodayTodayTodayTodayTodayTodayTodayToday"
```

Join strings

In [89]:
```
join(["apples", "bananas", "pineapples"], ", ", " and ")
```

Out[89]:
```
"apples, bananas and pineapples"
```

Split strings

In [90]:
```
split("1,2,3,4,5,6", ",")
```

Out[90]:
```
6-element Array{SubString{String},1}:
 "1"
 "2"
 "3"
 "4"
 "5"
 "6"
```

Replace

In [91]:
```
replace("Hello, world!", "world" => "Julia")
```

Out[91]:
```
"Hello, Julia!"
```

Quiz 5

如果我們要把以下的文字解析成電腦可以運算的數字，要怎麼做呢？

In [92]:
```
matrix = """1, 2, 3, 4
5, 6, 7, 8
9, 10, 11, 12"""
```

Out[92]:
```
"1, 2, 3, 4\n5, 6, 7, 8\n9, 10, 11, 12"
```

其中一個答案：

我們要對文字做處理，可以先針對不同行先切分，所以分隔符是 "\n"，這是代表 換行 的符號，他也是一種跳脫字元，在 Julia 中，跳脫字元會以 \ 做起始，他可以用來表示那些不可列印的字元。

In [93]:
```
rows = split(matrix, "\n")
```

Out[93]:
```
3-element Array{SubString{String},1}:
 "1, 2, 3, 4"
 "5, 6, 7, 8"
 "9, 10, 11, 12"
```

接著，可以用兩層的 for 迴圈分別去處理列以及每一個元素，要把每一列也依據分隔符切開，切開後的元素需要經由 parse 函式來轉成整數，然後把整數存進陣列中。

In [94]:

```
A = Int64[]
for row in rows
    elements = split(row, ", ")
    for e in elements
        append!(A, Meta.parse(e))
    end
end
```

In [95]:
```
A
```

Out[95]:
```
12-element Array{Int64,1}:
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
 11
 12
```

Functions

What is function?

當有些程式行為需要不斷被重複使用，只需要更改行為的一部份即可

這些行為就可以被**抽出來（abstract）**，成為 function

讓這部份程式可以有更**廣泛的（generic）**用處，而不是**狹隘而特定的（specific）**

In [96]:
```
function f(x, y)
    return x + y
end
```

Out[96]:
```
f (generic function with 1 method)
```

In [97]:
```
f(1, 2)
```

Out[97]:
```
3
```

當你呼叫函式 f(1, 2) 的時候， x=1 與 y=2 會被傳送給 f 。

函式就會進行後續的運算，並把運算結果透過 return 進行回傳。

當函數被呼叫，記憶體會空出一塊空間給函式，是函式的運算空間。

In [98]:
```
f(f(1, 2), 3)
```

Out[98]: 6

當以上函式被呼叫，最內部的函式 f(1, 2) 會先被運算，等運算結果回傳之後，才運算外層的函式 f(3, 3) 。

短小輕巧的函式在Julia很常見

In [99]:
```
h(x, y) = x + y
```

Out[99]:
```
h (generic function with 1 method)
```

In [100]:
```
h(1, 2)
```

Out[100]:
```
3
```

Specify input and output datatype

In [101]:
```
function g(x::Int64, y::Int64)::Int64
    return x + y
end
```

Out[101]:
```
g (generic function with 1 method)
```

In [102]:
```
g(1, 2)
```

Out[102]:
```
3
```

In [103]:
```
g(1.2, 2.3)
```

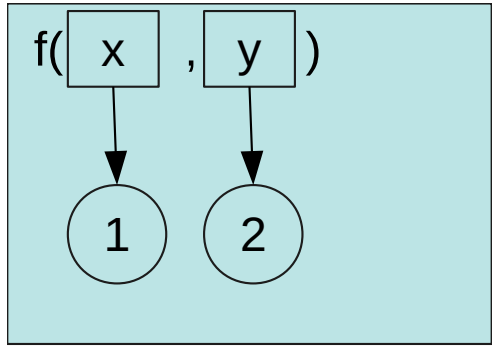
```
MethodError: no method matching g(::Float64, ::Float64)

Stacktrace:
 [1] top-level scope at In[103]:1
```

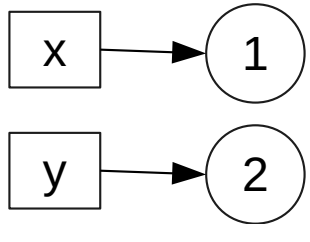
Argument passing

call-by-value

複製一份變數的值到函式中

e.g. C, primitive values in Java

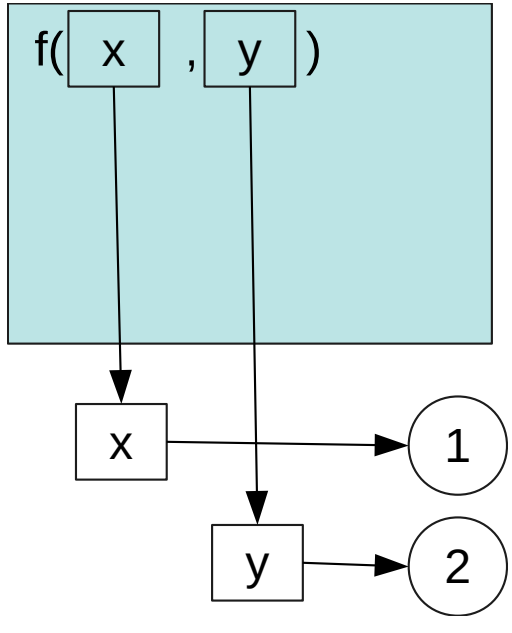




call-by-reference

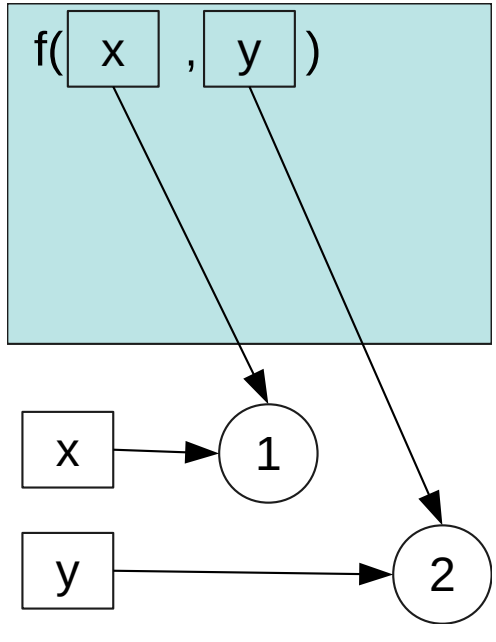
在函式中製造一個參考（reference），參考指向變數

e.g. Python, object in Java



pass-by-sharing

傳參數時，並不會複製一份給函式，但是參數本身會作為一個新的變數**綁定（bind）**到原本值的位址



如何驗證以上的行為？

In [104]:
```
println(objectid(1))
```

```
11967854120867199718
```

In [105]:
```
x = 1
println(objectid(x))
```

```
11967854120867199718
```

In [106]:
```
function sharing(x)
    println(objectid(x))
    x = 2
    println(objectid(x))
end
```

Out[106]:
```
sharing (generic function with 1 method)
```

In [107]:
```
sharing(x)
```

```
11967854120867199718
5352850025288631388
```

In [108]:
```
x
```

Out[108]:
```
1
```

Operators are functions

In [109]:
```
1 + 2 + 3 + 4 + 5 + 6
```

Out[109]: 21

In [110]:
```
+(1, 2, 3, 4, 5, 6)
```

Out[110]: 21

Anonymous functions

In [111]:
```
a = () -> println("Calling function a.")
```

Out[111]:
```
#3 (generic function with 1 method)
```

In [112]:
```
a()
```

```
Calling function a.
```

In [113]:
```
b = x -> println(x)
```

Out[113]:
```
#5 (generic function with 1 method)
```

In [114]:
```
b(5)
```

```
5
```

In [115]:
```
c = (x, y) -> x + y
```

Out[115]:
```
#7 (generic function with 1 method)
```

In [116]:
```
c(2, 3)
```

Out[116]:
```
5
```

Tuples

In [117]:
```
x = (1, 2, 3)
```

Out[117]: (1, 2, 3)

In [118]:
```
x[1]
```

Out[118]: 1

In [119]:
```
x[2:3]
```

Out[119]: (2, 3)

Tuple is immutable

In [120]:
```
objectid(x)
```

Out[120]:
```
0x1b15593be9794b02
```

In [121]:
```
objectid(x[2:3])
```

Out[121]:
```
0x35d8ea4221d4c2fc
```

Unpacking

In [122]:
```
a, b, c = x
```

Out[122]:
```
(1, 2, 3)
```

In [123]:
```
a
```

Out[123]:
```
1
```

In [124]:
```
b
```

Out[124]:
```
2
```

In [125]:
```
c
```

Out[125]:
```
3
```

Swap

In [126]:
```
b, a = a, b
```

Out[126]: (1, 2)

In [127]:
```
a
```

Out[127]: 2

In [128]:
```
b
```

Out[128]: 1

Tuple is the data structure that pass arguments to function

In [129]:
```
h(1, 2)
```

Out[129]: 3

`return` keyword

In [130]:
```
function sumproduct(x, y, z)
    return (x + y) * z
end
```

Out[130]:
```
sumproduct (generic function with 1 method)
```

In [131]:
```
function sumproduct(x, y, z)
    (x + y) * z
end
```

Out[131]:
```
sumproduct (generic function with 1 method)
```

Multiple return values

In [132]:
```
function shuffle_(x, y, z)
    (y, z, x)
end
```

Out[132]:
```
shuffle_ (generic function with 1 method)
```

Argument destruction

In [133]:
```
x = [1, 2, 3]
shuffle_(x...)
```

Out[133]: (2, 3, 1)

等價於 shuffle_(1, 2, 3)

Vectorizing functions

適用 operators 跟 functions

In [134]:
```
x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
```

In [135]:
```
x .^ 2
```

Out[135]:
```
10-element Array{Int64,1}:
   1
   4
   9
  16
  25
  36
  49
  64
  81
 100
```

User-defined function

In [136]:
```
f(x) = 3x
```

Out[136]:
```
f (generic function with 2 methods)
```

In [137]:
```
f.(x)
```

Out[137]:
```
10-element Array{Int64,1}:
  3
  6
  9
 12
 15
 18
 21
 24
 27
 30
```

Quiz 6

撰寫簡短的程式計算 $f(x, y) = x^2 + y^2 + 5xy + 3$ 的結果，並將以下的數值帶入求值：

In [138]:
```
data = [(1, 1), (2, 3), (-78, 96), (0, 7), (6, 6)]
```

Out[138]:
```
5-element Array{Tuple{Int64,Int64},1}:
 (1, 1)
 (2, 3)
 (-78, 96)
 (0, 7)
 (6, 6)
```

其中一個答案

In [139]:
```
f(x, y) = x^2 + y^2 + 5x*y + 3
```

Out[139]:
```
f (generic function with 2 methods)
```

In [140]:
```
f.(data)
```

```
MethodError: no method matching *(::Int64, ::Tuple{Int64,Int64})
Closest candidates are:
  *(::Any, ::Any, !Matched::Any, !Matched::Any...) at operators.jl:502
  *(::T<:Union{Int128, Int16, Int32, Int64, Int8, UInt128, UInt16, UInt32, UInt64, UInt8}, !Ma
tched::T<:Union{Int128, Int16, Int32, Int64, Int8, UInt128, UInt16, UInt32, UInt64, UInt8}) wh
ere T<:Union{Int128, Int16, Int32, Int64, Int8, UInt128, UInt16, UInt32, UInt64, UInt8} at in
t.jl:54
  *(::Union{Int16, Int32, Int64, Int8}, !Matched::BigInt) at gmp.jl:463
  ...

Stacktrace:
 [1] f(::Tuple{Int64,Int64}) at ./In[136]:1
 [2] _broadcast_getindex_evalf at ./broadcast.jl:574 [inlined]
 [3] _broadcast_getindex at ./broadcast.jl:547 [inlined]
 [4] getindex at ./broadcast.jl:507 [inlined]
 [5] copy at ./broadcast.jl:782 [inlined]
 [6] materialize(::Base.Broadcast.Broadcasted{Base.Broadcast.DefaultArrayStyle{1},Nothing,type
of(f),Tuple{Array{Tuple{Int64,Int64},1}}}) at ./broadcast.jl:748
 [7] top-level scope at In[140]:1
```

In [141]:
```
f(tup::Tuple) = f(tup...)
```

Out[141]:
```
f (generic function with 3 methods)
```

In [142]:
```
f.(data)
```

Out[142]:
```
5-element Array{Int64,1}:
     10
     46
 -22137
     52
    255
```

Types

In [143]:
```
struct Point
    x::Float64
    y::Float64
end
```

In [144]:
```
p = Point(3.0, 4.0)
```

Out[144]:
```
Point(3.0, 4.0)
```

In [145]:
```
p.x
```

Out[145]:
```
3.0
```

In [146]:
```
p.y
```

Out[146]:
```
4.0
```

In [147]:
```
import Base.length
```

In [148]:
```
length(p::Point) = sqrt(p.x^2 + p.y^2)
```

Out[148]:
```
length (generic function with 85 methods)
```

In [149]:
```
length(p)
```

Out[149]:
```
5.0
```

Quiz 7

定義時間的型別，當中需要紀錄小時、分鐘跟秒。定義 `format` 函式，可以將時間物件格式化成 "HH:MM:SS" 輸出。

Q & A

In [ ]: