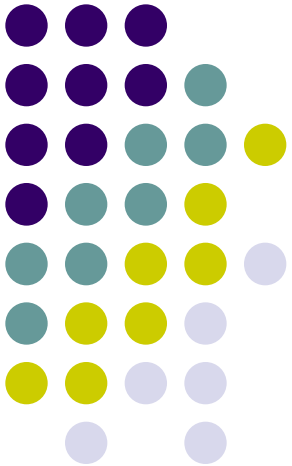
第三章
變數與資料型態



認識變數與常數

認識Java的基本資料型態

格式化列印資料

學習如何由鍵盤輸入資料

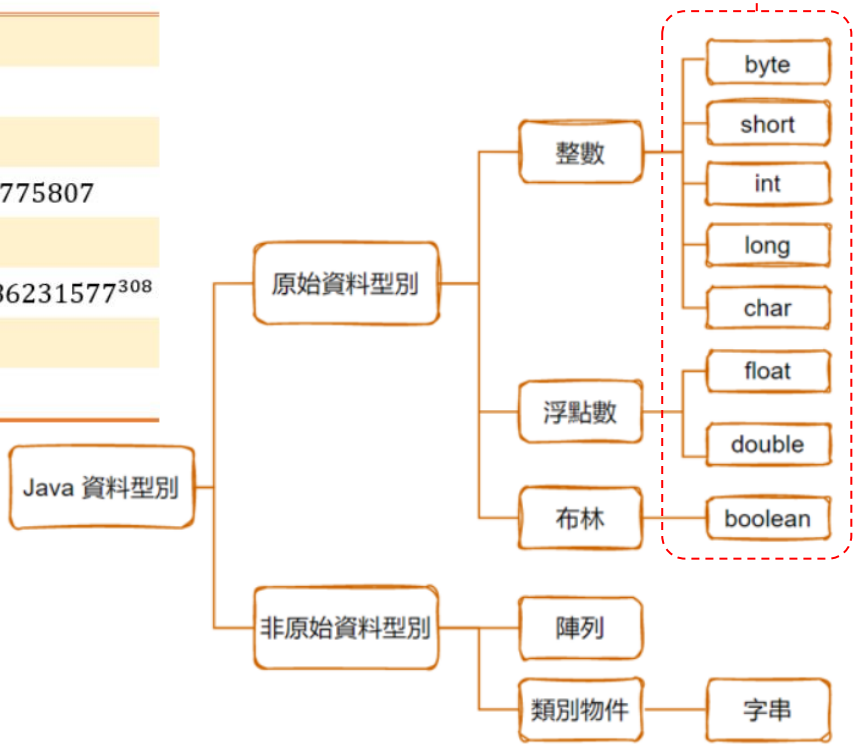
資料型別

- Java的資料型別：

• Java 的原始資料型別

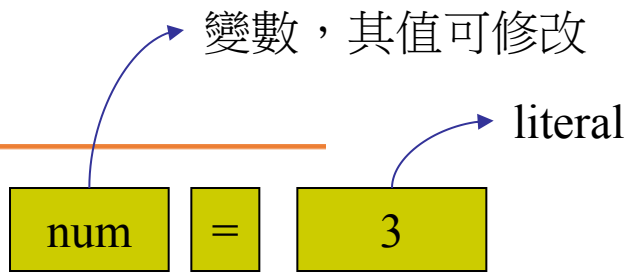
資料型別	說明	位元組	表達範圍
byte	位元組	1	−128~127
short	短整數	2	−32768~32767
int	整數	4	−2147483648~2147483647
long	長整數	8	−9223372036854775808~9223372036854775807
float	浮點數	4	-3.4029234^{38}~-3.4029234^{38}
double	倍精度	8	-1.79769313486231^{308}~$1.1.79769313486231577^{308}$
char	字元	2	0～65535('\u0000'~'\uFFFF')
boolean	布林	1	只能使用 true 或 false

本章要介紹的資料型別



變數的使用

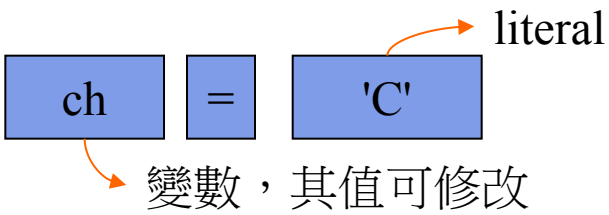
- 下面是變數使用的範例：




```
01  // Ch3_1, 變數的設值
02  public class Ch3_1{
03     public static void main(String[] args){
04        int num=3;      // 宣告 num 為 int 型別的變數，並設值為 3
05        char ch='C';    // 宣告 ch 為 char 型別的變數，並設值為'C'
06        System.out.println(num+" is an integer");
07        System.out.println(ch+" is a character");
08     }
09  }
```

執行結果：

```
3 is an integer
C is a character
```



整數

- 不帶有小數的數字即為整數，可分為byte、short、int及long四種
- 宣告整數變數的注意事項：
  - 整數的值必須在該型別可表達的範圍內
  - 整數後面加上一個L，代表長整數（如 -12345678900L 或 66000000000L）
  - long常數（如 3600L）不能設定給int、short或byte型別的變數存放

```
byte value=20;              // 宣告 byte 型別的變數 value，並設值為 20
short id=-3200;             // 宣告 short 型別的變數 id，並設值為-3200
int num=0;                  // 宣告 int 型別的變數 num，並設值為 0
int len=16_384;             // 宣告 int 型別的變數 len，並設值為 16384
long largeNum=0L;           // 宣告 long 型別的變數 largeNum，並設值為 0L
long huge=6_432_201_505L;   // 宣告 long 型別的變數 huge，並設值為 6432201505
```

有錯的範例

- 初學者常犯錯誤：

```
01  //Ch3_2,有錯誤的範例
02  public class Ch3_2{
03      public static void main(String[] args){
04        byte num=-360;          // 錯誤，byte 容許的最小值為-128
05        short area=40000;       // 錯誤，short 容許的最大值為 32767
06        int value=600L;         // 錯誤，600L 是長整數
07        long width=12345678900; // 錯誤，12345678900 後面要加 L
08
09        System.out.println(num+" is of type byte");
10        System.out.println(area+" is of type short");
11        System.out.println(value+" is of type int");
12        System.out.println(width+" is of type long");
13    }
14  }
```

```
/* output----------------------------------
-36 is of type byte
4000 is of type short
600 is of type int
12345678900 is of type long

----------------------------------------*/
```

問題 4　輸出　偵錯主控台　終端機

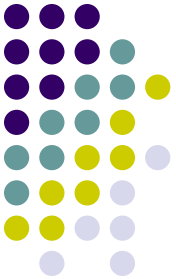
Ch3_2.java 4

- Type mismatch: cannot convert from int to byte Java(16777233) [第 4 行，第 18 欄]
- Type mismatch: cannot convert from int to short Java(16777233) [第 5 行，第 20 欄]
- Syntax error on token "600L", delete this token Java(1610612968) [第 6 行，第 19 欄]
- The literal 12345678900 of type int is out of range Java(536871066) [第 7 行，第 20 欄]

將 4~7 行修改如下

```
04      byte num=-36;
05      short area=4000;
06      int value=600;
07      long width=12345678900L;
```

以二進位、八進位和十六進位表示整數



```
01  //Ch3_3, 二進位、八進位和十六進位的使用
02  public class Ch3_3{
03      public static void main(String[] args){
04        byte bin=0b1111;          // 以二進位表示的整數
05        short oct=02001;          // 以八進位表示的整數
06        int hex1=0xff12;          // 以十六進位表示的整數
07        long hex2=0x12345ffeL;    // 以十六進位表示的長整數
08
09        System.out.println("bin= " + bin);
10        System.out.println("oct= " + oct);
11        System.out.println("hex1= " + hex1);
12        System.out.println("hex2= " + hex2);
13      }
14  }
```

執行結果：

```
bin= 15
oct= 1025
hex1= 65298
hex2= 305422334
```

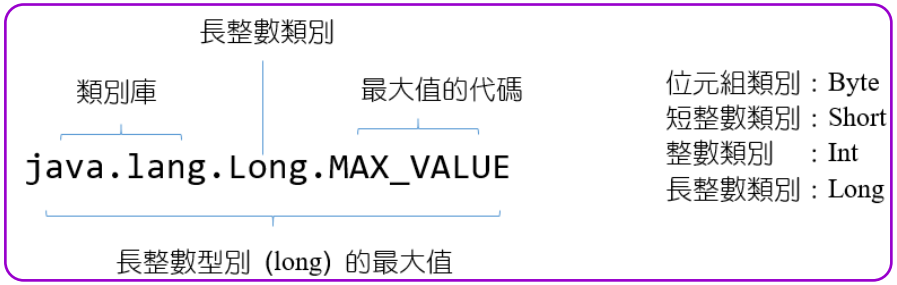
簡單易記的代碼

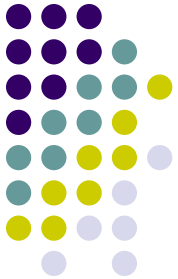
- Java把整數數值的型態視為int，超過範圍時會發生錯誤

```
01  // Ch3_4, 印出 Java 定義的整數常數之最大值
02  public class Ch3_4{
03     public static void main(String[] args){
04        long lmax=java.lang.Long.MAX_VALUE;      // long 型別的最大值
05        int imax=java.lang.Integer.MAX_VALUE;    // int 型別的最大值
06        short smax=Short.MAX_VALUE;   // 省略類別庫 java.lang
07        byte bmax=Byte.MAX_VALUE;     // 省略類別庫 java.lang
08
09        System.out.println("Max value of long  : "+lmax);
10        System.out.println("Max value of int   : "+imax);
11        System.out.println("Max value of short : "+smax);
12        System.out.println("Max value of byte  : "+bmax);
13     }
14  }
```

執行結果：

```
Max value of long  : 9223372036854775807
Max value of int   : 2147483647
Max value of short : 32767
Max value of byte  : 127
```





溢位 (overflow) 的發生 (1/3)

- 溢位：當儲存的數值超出容許範圍時

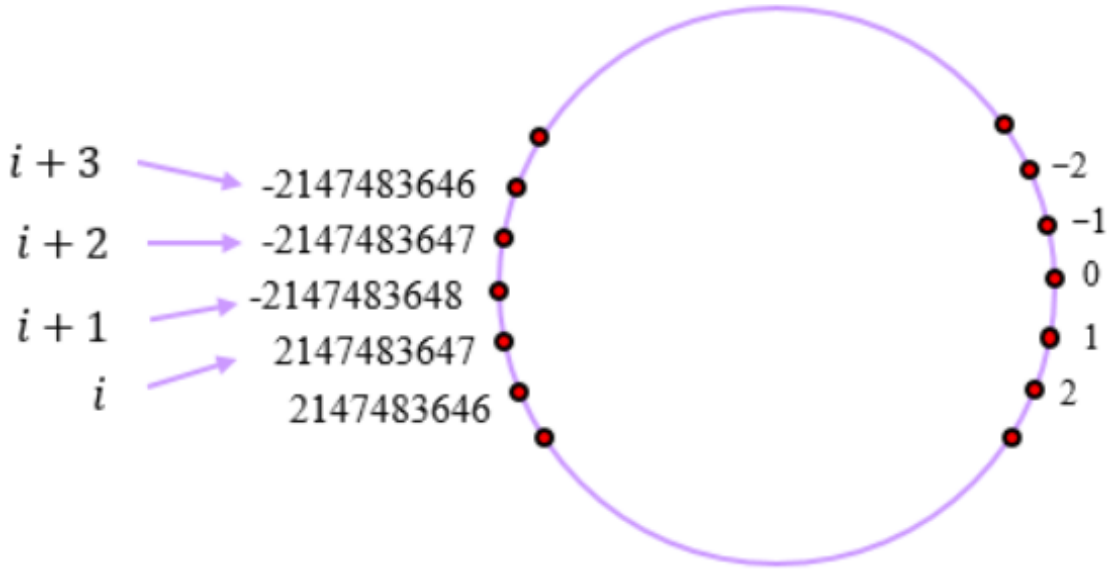
```
01  // Ch3_5, 溢位的發生
02  public class Ch3_5{
03     public static void main(String[] args){
04        int i=java.lang.Integer.MAX_VALUE;       // 將 i 設為 int 型別的最大值
05
06        System.out.println("i="+i);
07        System.out.println("i+1="+(i+1));        // 會發生溢位
08        System.out.println("i+2="+(i+2));        // 會發生溢位
09     }
10  }
```

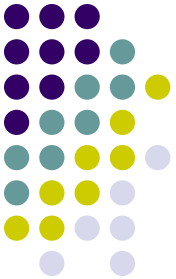
執行結果：

```
i=2147483647
i+1=-2147483648
i+2=-2147483647
```

溢位 (overflow) 的發生 (2/3)

- 下圖說明溢位的發生：





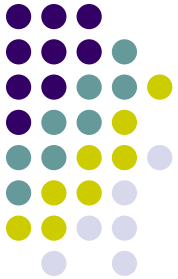
溢位 (overflow) 的發生 (3/3)

- int型態溢位處理範例

```
01  // Ch3_6, 型別的溢位處理
02  public class Ch3_6{
03     public static void main(String[] args){
04        int i=Integer.MAX_VALUE;        // 將 i 設為 int 型別的最大值
05
06        System.out.println("i="+i);
07        System.out.println("i="+(i+1));          // 會發生溢位
08        System.out.println("i+1="+(i+1L));       // 自動型別轉換
09        System.out.println("i+2="+((long)i+2)); // 強制型別轉換
10     }
11  }
```

執行結果：

```
i=2147483647
i=-2147483648
i+1=2147483648
i+2=2147483649
```



浮點數與倍精度浮點數型態 (1/3)

- 浮點數（float）長度為4個位元組

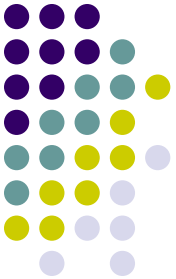
  有效範圍為 $-3.4\times10^{38}\sim3.4\times10^{38}$

- 倍精度（double）浮點數的長度為8個位元組

  有效範圍為 $-1.7\times10^{308}\sim1.7\times10^{308}$

- float與double型態的變數宣告範例如下：

```
double num=0.0;      // 宣告num為double型別的變數
float avg=2.0f;      // 宣告avg為float型別的變數
```



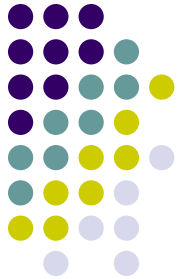
浮點數與倍精度浮點數型態 (2/3)

- float與double宣告與設值時注意事項
  - double num1=-5.6e64; // 宣告num1為double，其值為 -5.6×10^{64}
  - double num2=-6.32E16; // e 也可以用大寫的 E 來取代
  - float num3=2.478f; // 宣告 num3 為 float，並設初值為 2.478
  - float num4=2.63e64; // 錯誤，2.63×10^{64} 超過float可表示的範圍
  - float num5=6.3; // 編譯錯誤，常數6.3的型別是double
  - double num6=6.3; // 編譯成功，正確的宣告方式

```
01  // Ch3_7, 浮點數的使用
02  public class Ch3_7{
03     public static void main(String[] args){
04        float n=5.0f;    // 5.0f 為 float 型別的常數
05        double p=n*n;    // n*n 的結果會自動轉成 double，再設定給 p 存放
06        System.out.println(n+"*"+n+"="+p);   // 印出 n*n 的結果
07     }
08  }
```

執行結果：

```
5.0*5.0=25.0
```



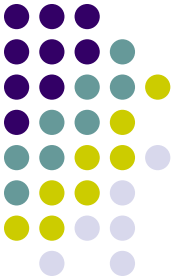
浮點數與倍精度浮點數型態 (3/3)

- 下面的範例是印出 float 與 double 的最大與最小值：

```
01  // Ch3_8, 印出浮點數的最大值和最小值
02  public class Ch3_8{
03     public static void main(String[] args){
04        System.out.println("f_max="+Float.MAX_VALUE);
05        System.out.println("f_min="+Float.MIN_VALUE);
06        System.out.println("d_max="+Double.MAX_VALUE);
07        System.out.println("d_min="+Double.MIN_VALUE);
08     }
09  }
```

執行結果：

```
f_max=3.4028235E38
f_min=1.4E-45
d_max=1.79769313486231570E308
d_min=4.9E-324
```



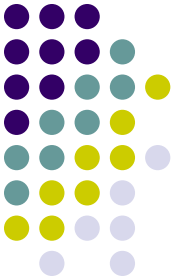
字元型態 (1/2)

- 字元型態佔 2 個位元組，用來儲存字元
- Java使用的編碼系統為Unicode(標準萬國碼)
- 宣告字元變數，並設值給它：

```
char ch='G';            // 宣告char型別的變數ch，並設值為'G'
char ch=71;             // 'G'的Unicode為71
char ch='\u0047';       // 71的16進位為47
```

- char型別與int型別之間的轉換：

```
int uc='好';            // char型別會自動轉換成int型別
int uc=(int)'好';       // 強制將char型別轉換成int
```



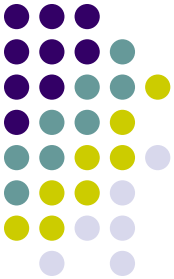
字元型態 (2/2)

- 下面的程式以不同的格式列印字元變數：

```
01  // Ch3_9, 字元型別的範例
02  public class Ch3_9{
03     public static void main(String[] args){
04        char c1='G';           // 將字面值'G'設定給 c1 存放
05        char c2=71;            // 利用 Unicode 設定 c2 為字元'G'
06        char c3='\u0047';      // 利用 16 進位的 Unicode 設定 c3 為字元'G'
07        int uni='好';          // 取得 '好' 字的 Unicode
08
09        System.out.println("c1="+c1+", c2="+c2+" ,c3="+c3);
10        System.out.println("uni="+uni);
11        System.out.println((int)'好');  // 印出字元的 Uincode
12     }
13  }
```

執行結果：

```
c1=G, c2=G ,c3=G
uni=22909
22909
```

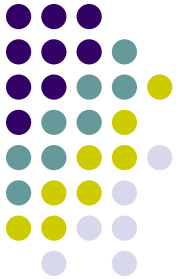


跳脫字元 (1/2)

- 反斜線「\」稱為跳脫字元
- 反斜線「\」加上控制碼，稱為跳脫序列

- 常用的跳脫序列

跳脫序列	所代表的意義	跳脫序列	所代表的意義
\f	換頁 (form feed)	\\	反斜線 (Backslash)
\b	倒退一格 (backspace)	\'	單引號 (Single quote)
\n	換行 (new line)	\"	雙引號 (Double quote)
\r	歸位 (carriage return)	\uxxxx	十六進位的 Unicode 字元
\t	跳欄 (tab)	\ddd	八進位的字元編碼，範圍在 000~377 之間（十進位為 0~255）



跳脫字元 (2/2)

- 利用跳脫序列列印字串：

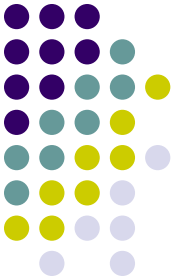
```
01  // Ch3_10, 列印跳脫序列
02  public class Ch3_10{
03     public static void main(String[] args){
04        char ch1='\042';          // 雙引號字元的八進位碼為 042
05        char ch2='\u0022';        // 雙引號字元的十六進位碼為 0022
06
07        System.out.println("\"Time is money!\"");
08        System.out.println(ch1+"Time flies."+ch1);
09        System.out.println(ch2+"Tomorrow never comes"+ch2);
10     }
11  }
```

可改成char ch1=042;

可改成char ch2=0x22;

執行結果：

```
"Time is money!"
"Time flies."
"Tomorrow never comes"
```



布林型態

- 宣告布林變數的範例：

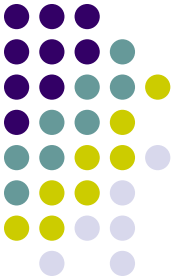
```
boolean status=true;        // 宣告布林變數status，並設值為true
```

- 在程式中印出布林值：

```
01  // Ch3_11, 印出布林值
02  public class Ch3_11{
03     public static void main(String[] args){
04        boolean status=false;      // 設定 status 布林變數的值為 false
05        System.out.println("status="+status);
06     }
07  }
```

執行結果：

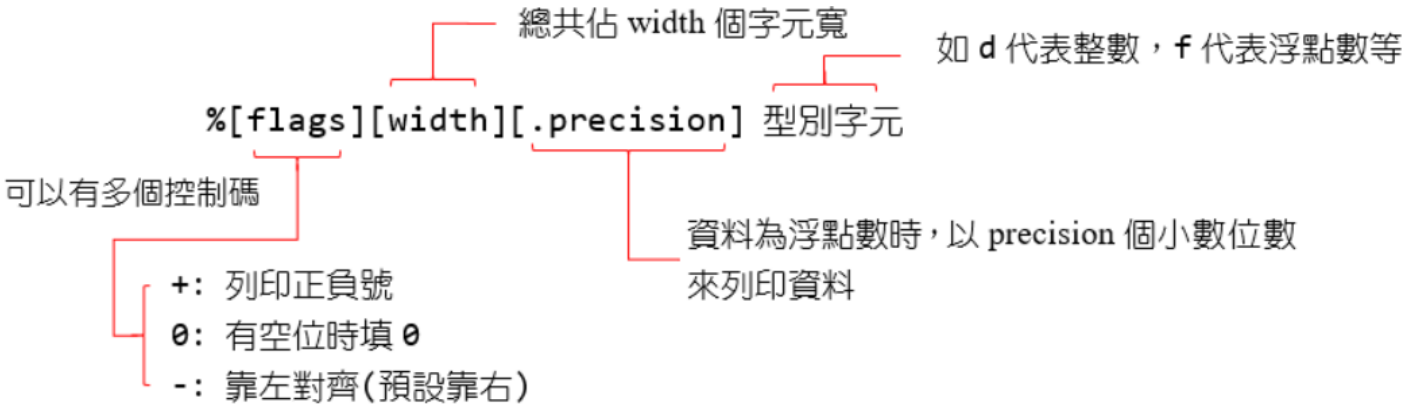
```
status=false
```

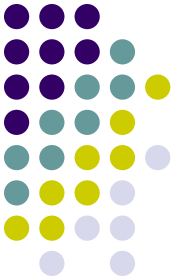


printf() 的格式碼(1/3)

printf() 的格式碼

格式碼	說明	格式碼	說明
%d	十進位整數（decimal）	%f	浮點數（float）
%o	八進位整數（octal）	%s	印出字串
%x	十六進位整數（hexadecimal），以小寫英文字顯示	%%	印出百分比符號
%X	十六進位整數（hexadecimal），以大寫英文字顯示	%c	字元（character）





printf() 的格式碼(2/3)

- 格式碼的範例：

格式碼的範例（符號 ○ 代表一個空格）

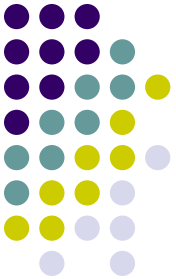
格式碼	資料	列印結果	說明
%4d	12	○○12	以 4 個字元寬靠右列印整數
%+-8d	123	+123○○○○	以 8 個字元寬、加正負號、靠左列印整數
%0+8d	123	+0000123	以 8 個字元寬、加正負號、靠右列印整數，空白處補 0
%6.2f	12.345	○12.35	以 6 個字元寬、小數點以下 2 位靠右列印浮點數
%+8.3f	12.3	○+12.300	以 8 個字元寬、小數點以下 3 位，加正負號靠右列印浮點數
%6c	'a'	○○○○○a	以 6 個字元寬靠右列印字元 'a'
%-8s	"Java"	Java○○○○	以 8 個字元寬、靠左列印字串 'Java'

列印於第二個格式碼的位置

```
int a=12;
double b=34.567;
System.out.printf("a=%3d, b=%08.2f\n", a, b);
```

列印於第一個格式碼的位置

輸出結果：a=○12, b=00034.57

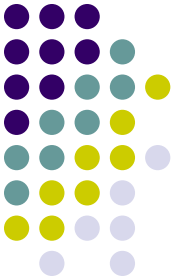


printf() 的格式碼(3/3)

```
01  // Ch3_12, 格式化列印
02  public class Ch3_12{
03     public static void main(String[] args){
04        byte bt=65;
05        float ft=3.14f;
06        double db=567.1234;
07
08        System.out.printf("bt=%c\n",bt);          // 列印字元
09        System.out.printf("bt=%+05d\n",bt);       // 列印整數
10        System.out.printf("oct=%o, hex=%x\n",bt, bt);  // 以不同進位數列印
11        System.out.printf("ft=%7.4f\n",ft);       // 列印浮點數
12        System.out.printf("db=%f\n",db);          // 列印倍精度浮點數
13     }
14  }
```

執行結果：

```
bt=A
bt=+0065
oct=101, hex=41
ft= 3.1400
db=567.123400
```



輸入資料的基本架構 (1/3)

- 載入Scanner類別，才能從鍵盤輸入資料：

```
import java.util.Scanner;   // 載入Scanner類別
```

- 輸入資料前，必須先建立Scanner類別的物件，再利用物件的函數傳回讀取的資料：

```
Scanner scn=new Scanner(System.in);   // 建立Scanner類別的物件scn
int age=scn.nextInt();                // 以int型別傳回讀取的資料
```

輸入資料的基本架構 (2/3)

- 輸入資料時Scanner類別提供的函數：

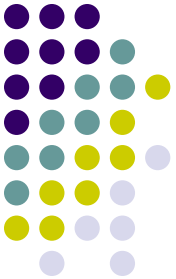
由鍵盤輸入資料時，常用的相對應型別之函數

函數	說明	函數	說明
nextByte()	以 byte 型別傳回讀取的資料	nextFloat()	以 float 型別傳回讀取的資料
nextShort()	以 short 型別傳回讀取的資料	nextDouble()	以 double 型別傳回讀取的資料
nextInt()	以 int 型別傳回讀取的資料	next()	以 String 型別傳回讀取的資料
nextLong()	以 long 型別傳回讀取的資料	nextLine()	以 String 型別傳回讀取的資料

ex: abc def
只會讀到 abc
abc def 都會讀到

- 使用Scanner類別輸入資料的範例：

```
Scanner scn=new Scanner(System.in);     // 宣告 Scanner 類別的物件
int i;                                  // 宣告整數變數
double d;                               // 宣告倍精度浮點數
i=scn.nextInt();                        // 輸入整數數值，給 i 存放
d=scn.nextDouble();                     // 輸入倍精度浮點數數值，給 d 存放
```



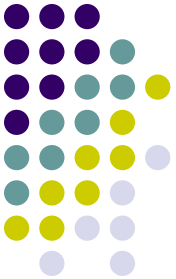
輸入資料的基本架構 (3/3)

- 由鍵盤輸入字串的範例

```
01  // Ch3_13, 由鍵盤輸入資料
02  import java.util.Scanner;          // 載入 Scanner 類別
03  public class Ch3_13{
04     public static void main(String[] args){
05        Scanner scn=new Scanner(System.in);  // 宣告 Scanner 類別的物件
06
07        System.out.print("What's your name? ");
08        String name=scn.next();      // 輸入字串
09        System.out.print("How old are you? ");
10        int age=scn.nextInt();       // 輸入整數
11        System.out.print(name+",  "+age+" years old.");
12        scn.close();                 // 將 scn 關閉
13     }
14  }
```

執行結果：

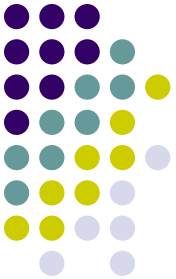
```
What's your name? Junie
How old are you? 16
Junie, 16 years old.
```



輸入數值--不合型態的輸入

- 若是需要輸入整數，卻輸入浮點數16.5，則會出現如下列的錯誤訊息：

```
Input an integer: 16.5
Exception in thread "main" java.util.InputMismatchException
        at java.base/java.util.Scanner.throwFor(Scanner.java:939)
        at java.base/java.util.Scanner.next(Scanner.java:1594)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2258)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2212)
```



以Scanner類別輸入字元

- 用next() 取得字串後，再利用charAt(0) 函數取出字串中第0個字元即可：

```
01   Scanner scn=new Scanner(System.in);
02   String str=scn.next();        // 輸入字串
03   char ch=str.charAt(0);        // 取出字串索引為 0 的字元
```


-The End-