

Data types / Operators / Math / Mouse Input

Instructor: Neng-Hao (Jones) Yu

Time: Mon. 6:10 – 9:10pm

Place: 商院大樓 260509

Course website: <http://programming101.cs.nccu.edu.tw>

Recaps

- ▣ How to write a program?
 - ▣ Drawing APIs (Application Programming Interface)
 - ▣ `functionName(parameters)`
 - ▣ Variables
 - ▣ Naming conventions
-

Recaps

- ▣ How to write a program?
 - ▣ compose a list of instructions (statements)
- ▣ Drawing APIs (Application Programming Interface)
 - ▣ `functionName(parameters)`
 - ▣ `ellipse(x,y,width,height);`
- ▣ Variables
- ▣ Naming conventions

Variables

- ▣ Declare the variable

```
// Case sensitivity!  
int myAge;  
int birthYear;
```

- ▣ Initialize the variable

```
// assign value(right) to variable(left)  
birthYear = 1988;
```

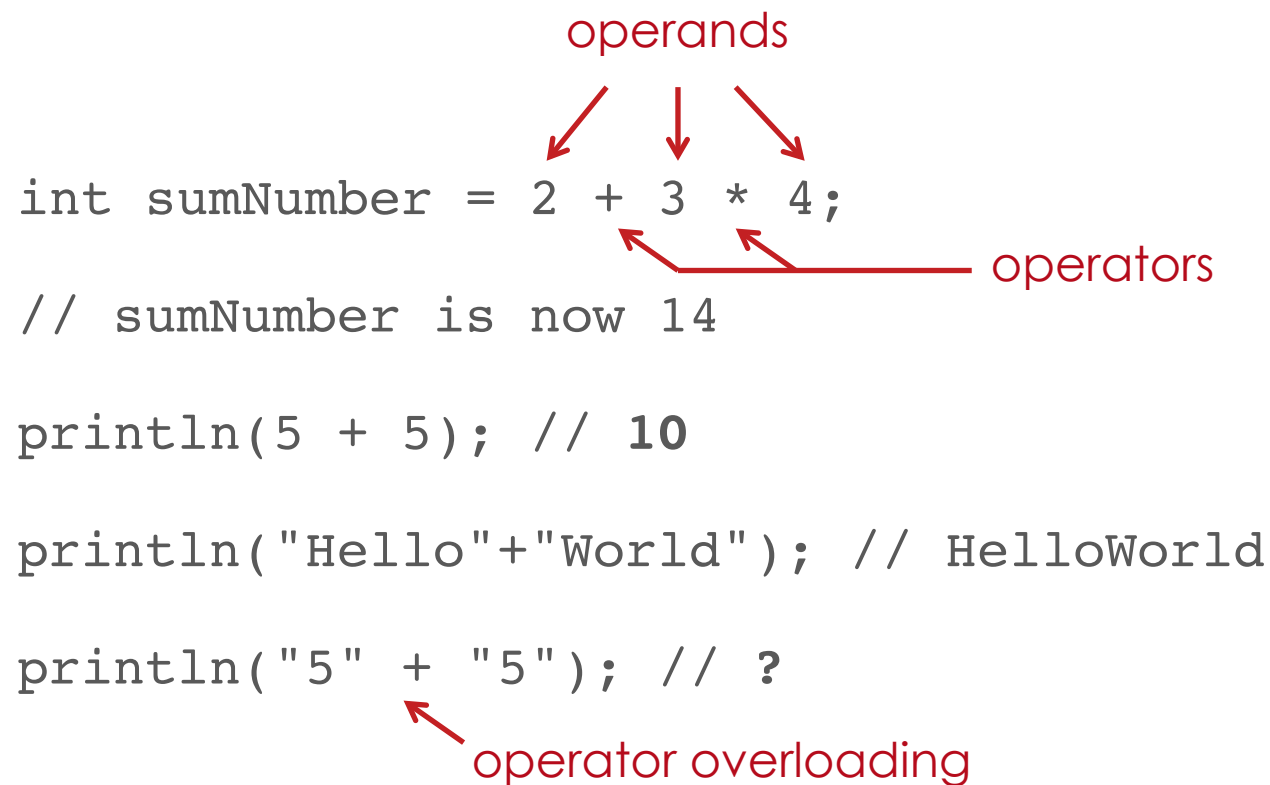
- ▣ Use the variable

```
myAge = 2014-birthYear+1;  
println(myAge);
```

Variable naming conventions

- ▣ Variable names **cannot be a reserved word** or keyword.
 - ▣ catch, this, class, throw ... (<http://processing.org/reference/>)
- ▣ Variable names must **start with a letter**, an underscore, or a dollar sign
- ▣ Variable names **cannot use special characters** (except _ and \$ in certain situations)
- ▣ Variable names must be **unique**
- ▣ Use **descriptive variable names** so that the content of any variable is obvious
- ▣ Use **camelCase**: totalScore, mySecretNumber

Operators and operands



The diagram illustrates the components of a code snippet. Red arrows point from the word "operands" to the numbers 2, 3, and 4 in the expression `2 + 3 * 4`. Another red arrow points from the word "operators" to the `+` and `*` symbols in the same expression. A third red arrow points from the word "operator overloading" to the `+` symbol in the expression `"5" + "5"`.

```
int sumNumber = 2 + 3 * 4;  
// sumNumber is now 14  
  
println(5 + 5); // 10  
  
println("Hello"+"World"); // HelloWorld  
  
println("5" + "5"); // ?
```

operands

operators

operator overloading

Arithmetic operators

- + Addition : Adds numeric expressions
- Subtraction : Negates or subtracts numeric expressions
- * Multiplication : Multiplies two numerical expressions
- / Division : Divides expression1 by expression2
- % Modulo : Calculates the remainder of division
- ++ Increment : Adds 1 to a numeric expression
- Decrement : Subtracts 1 from a numeric expression

Modulo (%) → **NOT** Percentage

- ▣ Calculates the remainder when one number is divided by another.

- ▣ `int a = 5 % 4; // Sets 'a' to 1`

- ▣ `int row = 33 % 10; // calculate row position`

- ▣ `int col = 33 / 10; // calculate column position`

- ▣ Modulo is extremely useful for keeping numbers within a boundary such as keeping a shape on the screen.

- ▣ `int newPosition = x % width;`

Type of operators

- ▣ Interfix: $5 + 3 - 2$
- ▣ Prefix: -55
- ▣ Postfix: $\text{foo}++ \rightarrow \text{foo} = \text{foo} + 1$

- ▣ Unary $\rightarrow ++$ (one operand)
- ▣ Binary $\rightarrow + - * /$ (two operands)
- ▣ Ternary $\rightarrow ? :$ (three operands)

Prefix vs Postfix

```
int xNum = 0;  
  
println(++xNum); // 1  
  
println(xNum); // 1
```

```
int yNum = 0;  
  
println(yNum++); // 0  
  
println(yNum); // 1
```

Compound assignment operators

```
var foo=5;
```

```
foo += 5;    // foo = foo + 5;
```

```
--=
```

```
*=
```

```
/=
```

```
%=
```

Operator precedence

```
int first = (1 + 2) * 7;  
  
int second = 5 * --first % 3;  
  
println(second);  
  
println(first);
```

Name	Symbol	Examples
Parentheses	()	a * (b + c)
Postfix, Unary	++ -- !	a++ --b !c
Multiplicative	* / %	a * b
Additive	+ -	a + b
Relational	> < <= >=	if (a > b)
Equality	== !=	if (a == b)
Logical AND	&&	if (mousePressed && (a > b))
Logical OR		if (mousePressed (a > b))
Assignment	= += -= *= /= %=	a = 44

Data types

- ▣ The way values of that type can be stored.
 - ▣ `int`: integer
 - ▣ `float`: floating-point number
 - ▣ `char`: single character
 - ▣ `String`: string of words
- ▣ Advantages:
 - ▣ less bugs
 - ▣ efficient memory allocation



Data types

■ Size and value range of data types:

	Name	Description	Range of values
32bits	<i>int</i>	Integers (whole numbers)	−2,147,483,648 to 2,147,483,647
32bits	<i>float</i>	Floating-point values	−3.40282347E+38 to 3.40282347E+38
1bit	<i>boolean</i>	Logical value	true or false
16bits	<i>char</i>	Single character	A–z, 0–9, and symbols
	<i>String</i>	Sequence of characters	Any letter, word, sentence, and so on
	<i>PImage</i>	PNG, JPG, or GIF image	N/A
	<i>PFont</i>	VLW font; use the Create Font tool to make	N/A
	<i>PShape</i>	SVG file	N/A


Type conversions

- ▣ **Implicit** conversion, also called **coercion**, is sometimes performed at runtime. It happens in
 - ▣ In assignment statements
 - ▣ In expressions using certain operators, such as the addition (+) operator
- ▣ **Explicit** conversion, also called **casting**, occurs when your code instructs the compiler to treat a variable of one data type as if it belongs to a different data type.


```
int a;                b= a / 2.0;

float b;              // or

a= 55;                b= (float)a / 2;

b= a / 2;  // or

println(b);           b= float(a) / 2;

// what is the value of b?
```

<http://processing.org/examples/datatypeconversion.html>

Constants

- ▣ Must be initialized
- ▣ Cannot change the data in it
- ▣ Constant names are all caps with words separated by an underscore, e.g.

```
final float MILES_KM_CONVERSION_VALUE = 1.61;
```

```
final float PI = 3.14;
```

```
// convert mile to km
final float MILES_KM_CONVERSION_VALUE = 1.61;

float km = 30 * MILES_KM_CONVERSION_VALUE;

println(km);
```

Exercise (5mins):

Write a program to calculate the area of a circle given a radius

Build-in constants and variables

```
size (200,200);  
float x = width/2;  
float y = height/2;  
float d = width * 0.8;  
arc(x, y, d, d, 0, QUARTER_PI);  
arc(x, y, d-20, d-20, 0, HALF_PI);  
arc(x, y, d-40, d-40, 0, PI);  
arc(x, y, d-60, d-60, 0, TWO_PI);
```

Static mode vs Dynamic mode

▣ Dynamic mode:

```
int x;
```

```
void setup() {  
    size(300, 300);  
}
```

run once at the beginning

```
void draw() {  
    //background(0);  
    ellipse(mouseX, mouseY, 80, 80);  
}
```

update forever

build-in variables

Exercise (6mins)

- ▣ draw a ball
 - ▣ keep it moving from left to right
 - ▣ Bonus: moving alien
-

Math

▣ `abs()`

▣ `ceil()`

▣ `floor()`

▣ `round()`

▣ `sq()`

▣ `pow()`

▣ `min()`

▣ `max()`

▣ `norm()`

▣ `constrain()`

▣ `map()`

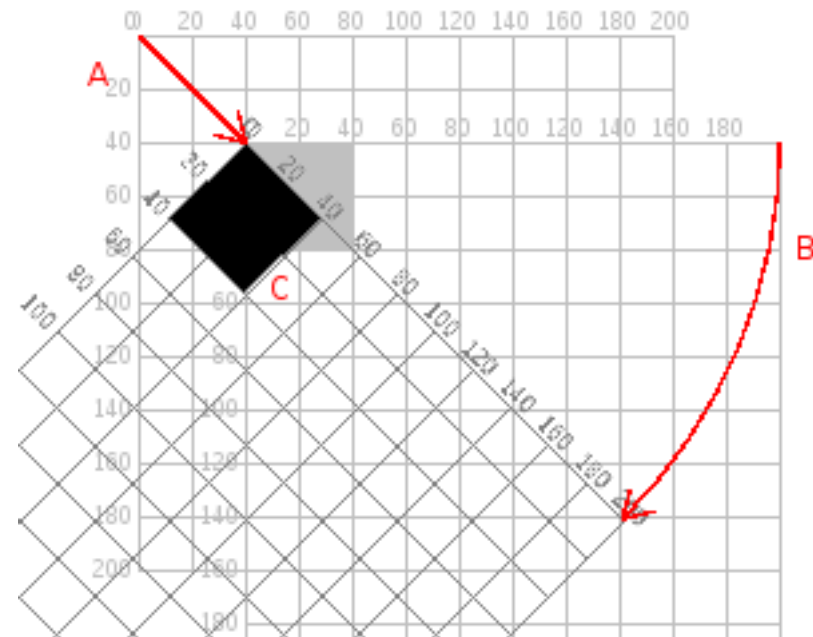
▣ `dist()`

<http://processing.org/reference/>

[https://gist.github.com/jonesfish/
7ab805f1da4f7ba1bf6a](https://gist.github.com/jonesfish/7ab805f1da4f7ba1bf6a)

Math

- ▣ `translate()`
- ▣ `rotate()`
- ▣ `scale()`
- ▣ `pushMatrix()`
- ▣ `popMatrix();`



<https://gist.github.com/jonesfish/3e93c7c11b08795654bc>

Math

- ▣ `radians()`
- ▣ `degrees()`
- ▣ `sin()`
- ▣ `atan2(y, x)`

Exercise (5mins) : draw a sin wave

<https://gist.github.com/jonesfish/1c6b2c76652e8a963f91>

Random Numbers

```
float r = random(50);  
  
// generate a random number from 0.0 to 50.0  
// starting at 0, and up to (but not including) 50.0  
  
int dice = int(random(6));  
  
// roll a dice
```

<https://gist.github.com/jonesfish/6b462ccea0f24f92cfaf1>

Assign1 – Slot Machine

- ▣ A: 出現 777的機率為 10%
- ▣ B:
 - ▣ 每次按下 "Roll" 扣50分
 - ▣ 按下 "Stop" 後以水果重複出現次數計算倍率分數（該水果數量 * 該水果的基本分數）
- ▣ C:
 - ▣ 不同水果代表不同分數
 - ▣ 開始時有500分，按下 "Roll" 啟動遊戲
 - ▣ 按下 "Stop" 後隨機指派九宮格內的水果
 - ▣ 結算分數並顯示（加總每一格水果所代表的分數）





Slot Machine

Introduction

角子機、老虎機或全名吃角子老虎機（英語：**slot machine**，另稱拉霸機或柏青嫂（日文），是一種經常在賭場見到的賭博機器，甚至有專玩角子機的娛樂場所，玩法是硬幣投入，接著會隨機出現不同圖案，如停定時如出現符合相同或特定相同圖案連線者，即依其賠率勝出。同一公司經營之角子老虎機通常會聯網，以投注額釐定大獎（**Jackpot**）金額，為增加吸引力，賭場還附有特大顯示屏顯示在當眼處，隨時更新之大獎金額以作招徠。（來源:Wikipedia）

作業下載連結：[Assign1.zip](#)
作業繳交說明：[Help-SubmitAssign.pdf](#)

點此下載作業

Requirement

- **A級條件:**
 1. 出現 777 的機率為 10%
- **B級條件:**
 1. 每次按下 "Roll it" 扣 50 分
 2. 按下 "Stop it" 後以水果重複出現次數計算倍率分數（該水果數量 * 該水果的基本分數）
- **C級條件:**
 1. 不同水果代表不同分數
 2. 開始時有 500 分，按下 "Roll it" 啟動遊戲
 3. 按下 "Stop it" 後隨機指派九宮格內的水果
 4. 結算分數並顯示（加總每一格水果所代表的分數）







請貼入遊戲play.html連結

繳交遊戲連結

使用方式

1. 將下載後的 assign1.zip解壓縮。
2. 打開 assign1.pde 檔案。
3. 找到 // ----- 的註解處，開始加入你的程式碼。
4. 註解共有三處分別在以下區段：
 - ▣ 宣告變數
 - ▣ 按下 Roll 時
 - ▣ 按下 Stop 時
5. 在本機測試正常後再繳交上傳。（繳交方式助教會再補充slide）

Slot Machine APIs

fruitId: 0	1	2	3	4	5
					
score: 60	10	20	30	40	50

```
mahcine.setSlotFruit(int slotPosition, int fruitId);  
mahcine.getSlotScore(int slotPosition);  
machine.getFruitCount(int fruitId);  
machine.probability(float p);
```

Slot Machine

Score : 0 ← `totalScore`

`slotPosition: 0`

1

2



Roll

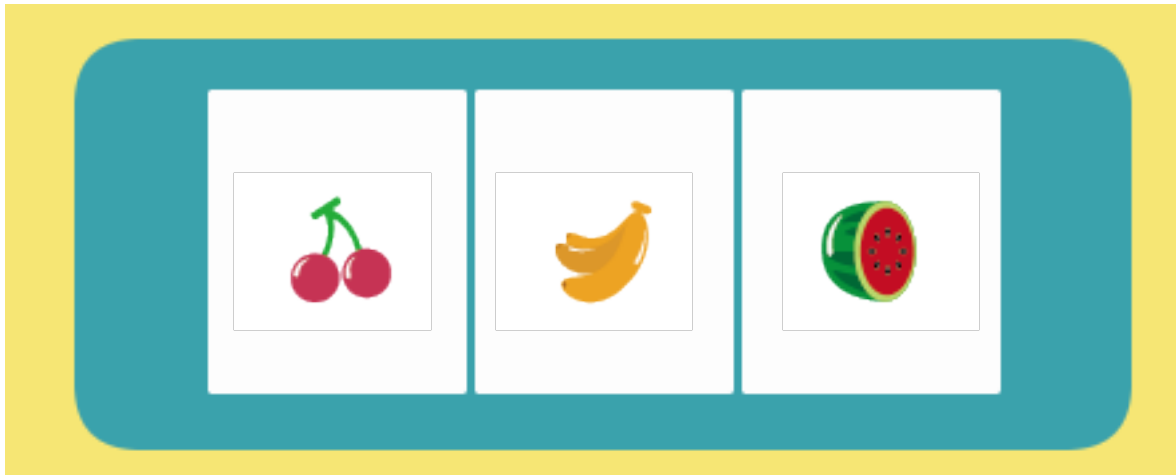
設定某一slot的水果編號

```
machine.setSlotFruit(0,2);  
machine.setSlotFruit(1,5);  
machine.setSlotFruit(2,4);
```

slotPosition: 0

1

2



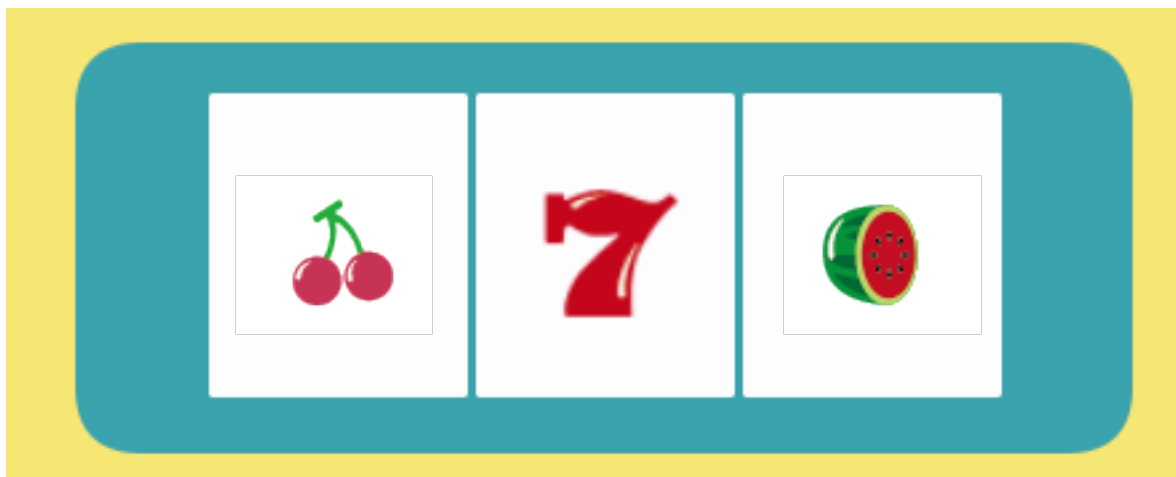
取得某一slot裡的水果代碼

```
int a = mahcine.getSlotScore(0); // a=2  
int b = mahcine.getSlotScore(1); // b=0  
int c = mahcine.getSlotScore(2); // c=4
```

slotPosition: 0

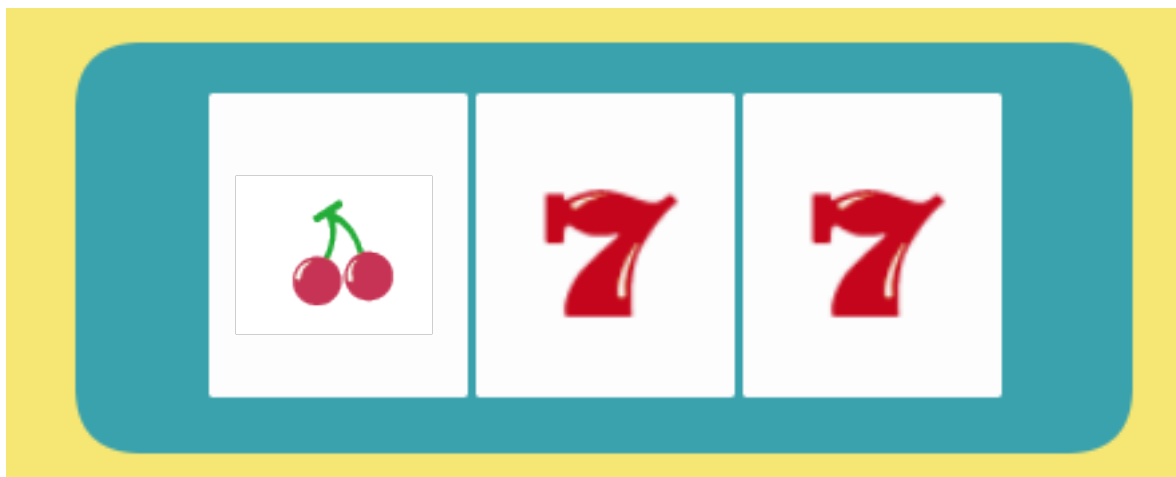
1

2



計算機器中，某一水果出現的個數

```
int m = machine.getFruitCount(0);    //lucky seven  
int n = machine.getFruitCount(2);    //cherry  
int o = machine.getFruitCount(5);    //banana  
// m = 2;  
// n = 1;  
// o = 0;
```



指定 1 出現的機率

```
int result = machine.probability(0.1);  
// result 有 0.1 (10%) 的機率會得到 1  
// result 有 0.9 (90%) 的機率會得到 0
```

```
result = machine.probability(0.5);  
// result 有 0.5 (50%) 的機率會得到 1  
// result 有 0.5 (50%) 的機率會得到 0
```