# Karel's Built in Commands 卡羅爾的內建指令

```
move();
移動();

turnLeft();
左轉();

putBall();
放球();

takeBall();
拿球();
```

# Functions 函式

Writing a function is like teaching karel a new word.寫一個函式就像教卡羅爾一個新的字。

Naming Functions: You can name your functions whatever you want, but you can't have spaces in the function name. 命名函式：你可以依自己喜好命名函式，但是函式名稱中不可有空格。

Remember that each open bracket **{** must match with a close bracket **}** 記得每一個左括號 **{** 必須有一個對應的右括號 **}** 。

```
function turnRight(){
      turnLeft();
      turnLeft();
      turnLeft();
}

函式 右轉(){
   左轉();
   左轉();
   左轉();
}

function turnAround(){
      turnLeft();
      turnLeft();
}

函式 轉身(){
   左轉();
   左轉();
 }

function yourFunctionName() {
```

```
  // Code that will run when you make a call to
  // this function.
}

函式 你的函式名稱() {
  // 將執行的程式碼
  // 每當你叫出此函式
}
```

# Conditional Statements 條件性陳述

Remember that each open bracket **{** must match with a close bracket **}**

**記住每一個左括號{** 必須有個對應的右括號 **}**

```
if (condition) {
  //code that will run if the condition is true
}
如果 (條件) {
  //在此條件為正確的情況下，將會執行的程式碼
}


if (condition) {
  //code that will run if the condition is true
} else {
  //code that will run if condition is not true
}
如果 (條件) {
  //在此條件為正確的情況下，將會執行的程式碼
} 否則 {
  //在此條件為不正確的情況下，將會執行的程式碼
}
```

### Example of if statements “如果”陳述之例子

```
if(frontIsClear()){
  move();
}
```

```
如果(前方無阻礙物()){
  移動();
}
```

```
if(ballsPresent()){
  takeBall();
}else{
  move();
}
```

```
如果(有球()){
  拿球();
}否則{
  移動();
}
```

## Karel Conditions 卡羅爾相關的條件

Don't forget the `()` at the end 記住要以`()` 結尾

| | |
|---|---|
| `frontIsClear()` | `frontIsBlocked()` |
| `leftIsClear()` | `leftIsBlocked()` |
| `rightIsClear()` | `rightIsBlocked()` |
| `前方無障礙物()` | `前方有障礙物()` |
| `左方無障礙物()` | `左方有障礙物()` |
| `右方無障礙物()` | `右方有障礙物()` |
| | `notFacingNorth()` |
| `facingNorth()` | `notFacingSouth()` |
| `facingSouth()` | `notFacingEast()` |
| `facingEast()` | `notFacingWest()` |
| `facingWest()` | `不面向北方()` |
| `面向北方()` | `不面向南方()` |
| `面向南方()` | `不面向東方()` |
| `面向東方()` | `不面向西方()` |
| `面向西方()` | |
| `ballsPresent()` | `noBallsPresent()` |
| `有球()` | `沒有球()` |

# Loops 迴圈

Remember that each open bracket **{** must match with a close bracket **}**

**記住每一個左括號{** 必須有個對應的右括號 **}**

## While Loops “當…”的迴圈

```
while (CONDITION) {
  // Code that will run while the CONDITION is true.
  // Once the CONDITION is no longer true,
  // it will stop.
}
```

```
當 (條件) {
  // 當該條件為正確時，將會執行的程式碼
  // 一旦該條件不再正確時
  // 此程式碼將會停止
}
```

## Example of while loops “當…”迴圈之例子

```
/* This moves Karel to a wall */
while(frontIsClear()){
      move();
}
```

```
/* 這使卡羅爾移向一面牆*/
當(前方無障礙物()){
      移動();
}
```

## For Loops “重複” 的迴圈

```
for (var i=0; i < COUNT; i++) {
  // Code that will run 'COUNT' times
}
```

```
重複 (變數 i=0; i < 總數; i++) {
  // 將會執行程式碼“總數”次
}
```

## Example of for loops “重複”迴圈的例子

```
/* This puts down 10 balls */
for(var i = 0; i < 10; i++){
      putBall();
}
```

```
/* 放下十顆球 */
重複(變數 i = 0; i < 10; i++){
      放球();
}
```

You can have multiple statements or function calls in a for loop. 你可以在“重複”迴圈裡使用多重的陳述或函數。

```
/* This puts down five balls and
   moves after each one */
for(var i = 0; i < 5; i++){
      putBall();
      move();
}
```

```
/* 放下五顆球
   每放下一顆球後移動 */
重複(變數 i = 0; i < 5; i++){
      放球();
      移動();
}
```

Use for-loops when you want to repeat something a **fixed** number of times.當你想讓某個事件重複特定的次數，使用“重複”迴圈。

Use while-loops when you want to repeat something as long as a condition is true. 當你想在該條件為正確的情況下重複某個事件，使用“當…”迴圈。

# Comments 註釋

```
/* A multi-line comment describes your code
 * to someone who is reading it. */
```

```
// Use single line comments to clarify code.


/* 以多行註釋描述你的程式碼
 * 給你的讀者 */

// 使用單行註釋釐清程式碼
```