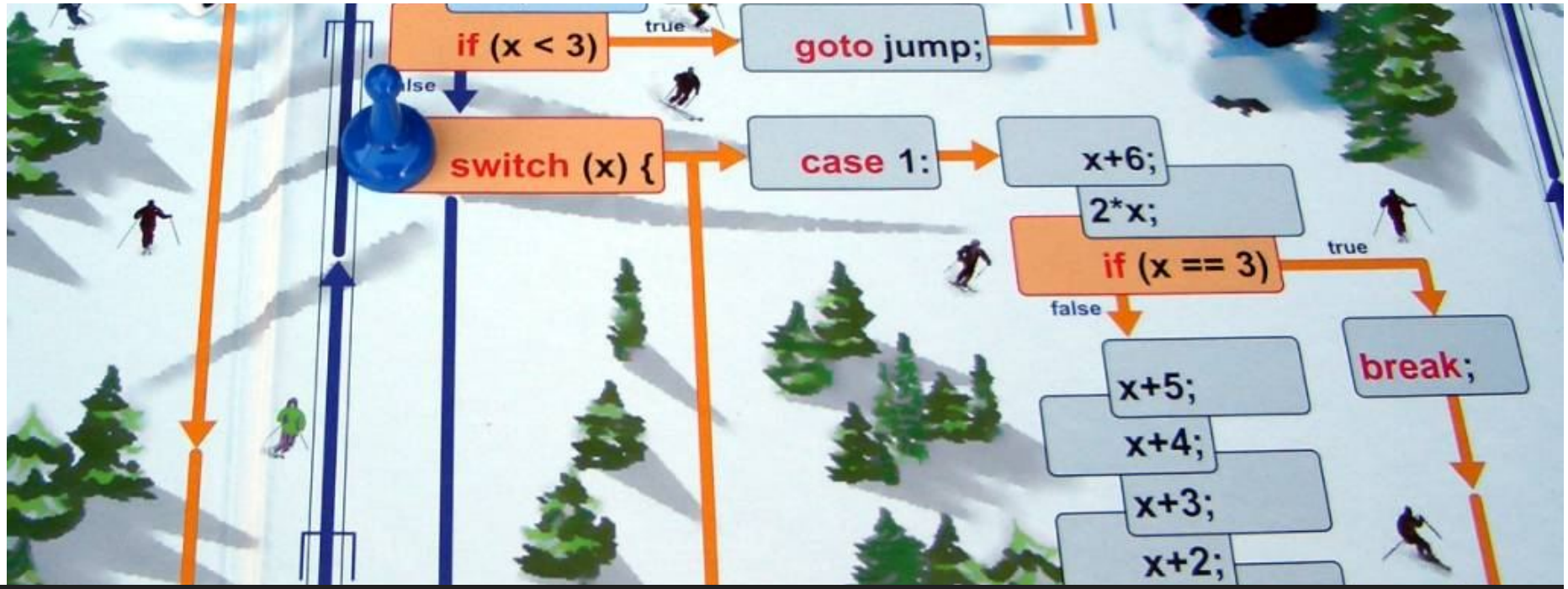


Creative Coding 2023

Instructor: Neng-Hao (Jones) Yu

Course website: https://openprocessing.org/class/83620

Functions

Definition: a function is a block of code that performs a specific task and can be called or executed multiple times from different parts of a program.

Function = Reusable code module

- Build-in functions
  - point(), line(), rect(), ellipse()…
  - background(), stroke(), fill()…
  - abs(), dist(), sq()…

Why use functions

Modularity (模組化)

- Break down code into smaller parts
- More manageable and readable
- Easy to debug

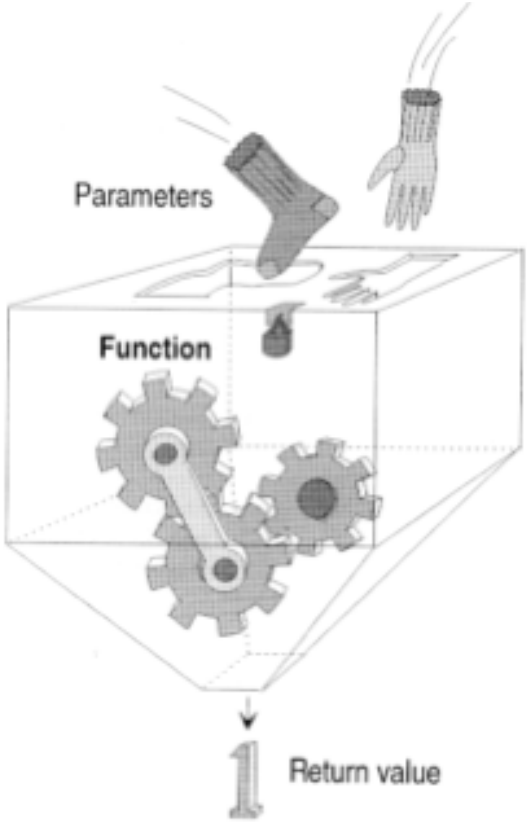
Why use functions

Reusability(可重複使用)

- Duplicated code (copy/paste) is not good
  - Need to maintain it in multiple places
- Better to put duplicate code in a new function and ‘call’ it from multiple places

How to use Functions

- Functions can take in **input** values called arguments or **parameters**, perform some operations on those inputs, and **return** an **output** value.
  - `int a = abs(-15);`
  - `float r = radians(135);`
  - `float n = dist(x1, y1, x2, y2)`



Function declaration & Function call

return type

function name

```
void sayHi() {
    println("Hi");
}
```

Function declaration

```
sayHi();     // Hi
sayHi();     // Hi
sayHi();     // Hi
```

Function calls

Parameter passing

return type
void: no return value

function name

parameter / argument

```
void sayHi(String name) {

    println("Hi " + name);

}

sayHi("Jones");    // Hi Jones

sayHi("Alice");    // Hi Alice

sayHi("Joe");      // Hi Joe
```

Parameter passing

return type

function name

multiple parameters / arguments

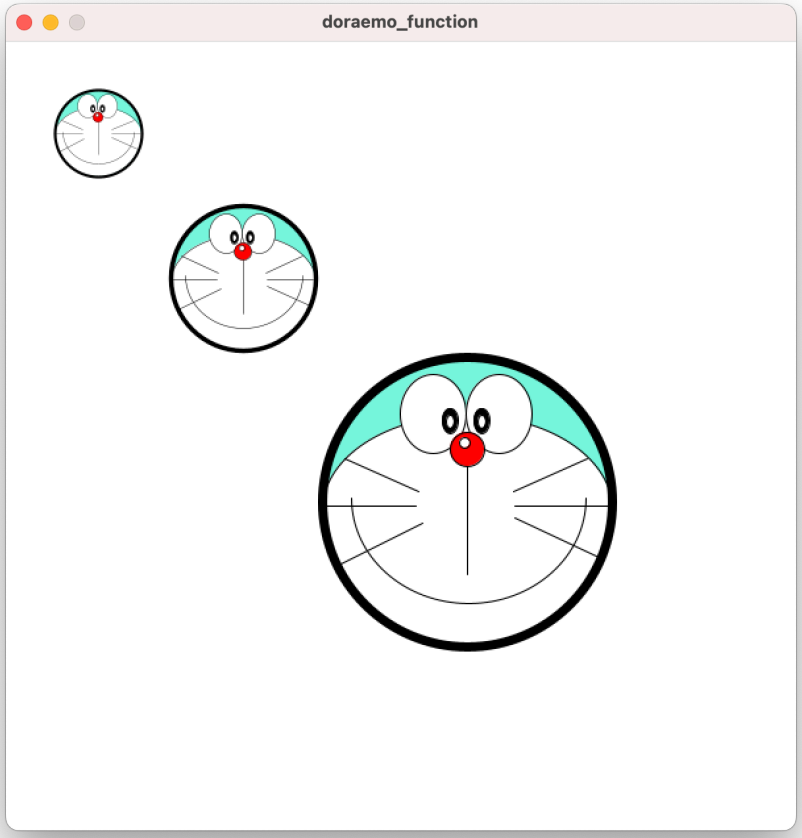
```
void circle(int x, int y, int diameter) {

  ellipse(x, y , diameter, diameter);

}


circle(150, 150, 50);
```

Exercise: draw Doraemo at giving location & size

Starter code: https://openprocessing.org/sketch/974749

1. doraemo(x,y);
2. doraemo(x,y,size);

Hint: pushMatrix(), popMatrix(), translate(), scale()

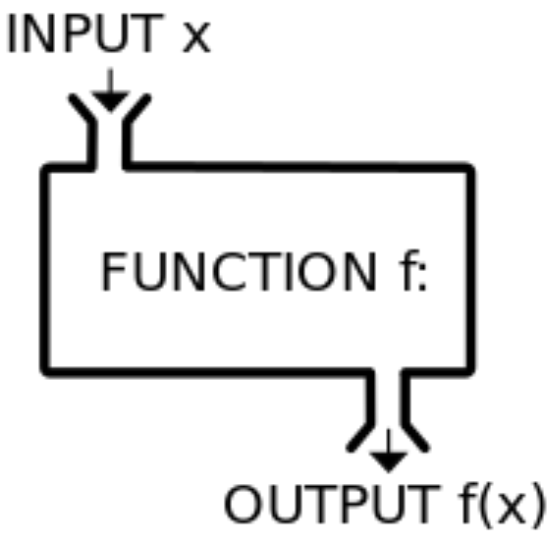


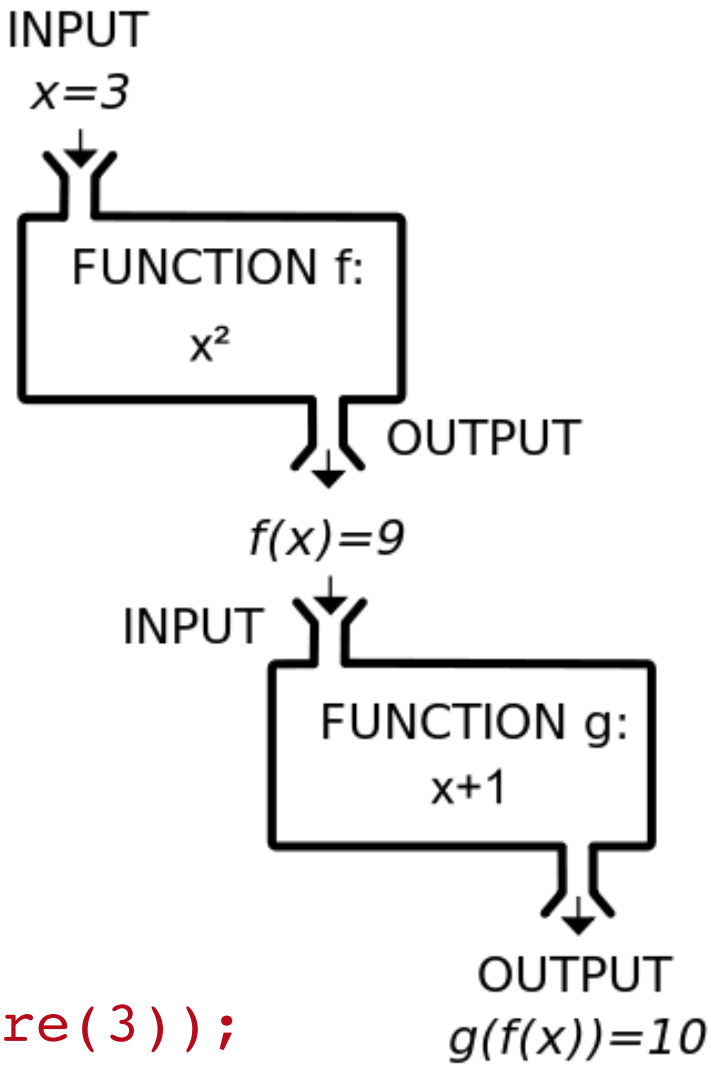
Return value

return type function name parameter / argument

```
int square(int num) {
    return(num * num );
}
int area = square(5); // call function
//returns the int value 25
```

Chained function calls






```
int y = add1(square(3));
```

Exercise

1. **Calculating the area of a triangle**
   - `input: base and height`
   - `output: triangle's area`

2. **Calculating the sum of all values in an array**
   - `input: integer array`
   - `output: sum`

Variable Scope

```
// global variables
int score = 10;
int level = 5;

void funcA() {
  // local variable
  int i = 100;
}

void funcB(){
  // local variable
  int n = 200;
}
```

Local scope

Local scope

Global scope

Function scope

Local scope

```
float area(float hh, float ww) {

    return(hh * ww );

}

float rect = area(6,5);

//returns the float value 30



println (hh); // error: undefined variables
```

The return statement terminates the function

```
float area(float hh, float ww) {
    return(hh * ww );
    println("after return");
 // Error: unreachable code.
}
```

Be aware of incomplete return

```
Boolean isOdd(int num) {
    if (num % 2 == 0){
      return false;
    }
}
println(isOdd(2));
// error: Function does not return a value.
```

Fix the incomplete return

```
Boolean isOdd(int num) {
    if (num % 2 == 0){
      return false;
    } else{
      return true;
    }
}
```

Exercise: Lottery game

You must choose **6 numbers** from **01 to 49** for your bet.

1. Design '**isExist**' function:
   if the input number already exist in the array, return true;

2. Design '**winningNumbers**' function:
   Randomly select **six winning numbers without duplicates** for the Lotto draw.

3. Design '**isWin**' function:
   If three or more (including three numbers) of your six selected numbers match the six numbers drawn for that Lotto draw, return true.

Function overloading

- one function can perform different tasks.
  - same function name
  - different types of input/output
  - different numbers of inputs

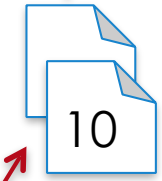
```
int addXY(int x, int y){
    return x+y;
}

println( addXY(3, 2) );
// 5
```

```
float addXY(float x, float y){
    return x+y;
}

println( addXY(5.5, 2) );
// 7.5
```

Pass by value vs pass by reference

- Primitive data types are passed by value
  - String, int, float, Boolean
- Complex data types are passed by reference
- Incidentally, these rules apply to variable assignments, too.

```
float cArea(float r) {
  return (PI * r * r);
}
float radius = 10;
println( cArea(raduis) );
```




```
void shuffle(int[]arr){
… }
int[] myArray= {0,1,2};

shuffle(myArray);
```

Define functions in practice

- a block of codes
  - no input parameters
  - no output value
- a dynamic module
  - with input parameters
  - output the computed value

Toward Reusability

- A good function does only one task

Exercise: Redesign the chickenRun game with functions

Starter code: https://openprocessing.org/sketch/1899403

Test code:

```
case GAME_RUN:
    background(10,110,16);
    // start and end area
    showSafeArea();
    // show life
    showLife();
    // show chicken
    image(imgChicken, x, y);
    // check destination
    checkDest();
    showCars();
    break;
```