Introduction to if and if-else statements

A story

Ann: **If** you have $1000, will you please give me a half?
Bob: Of course.

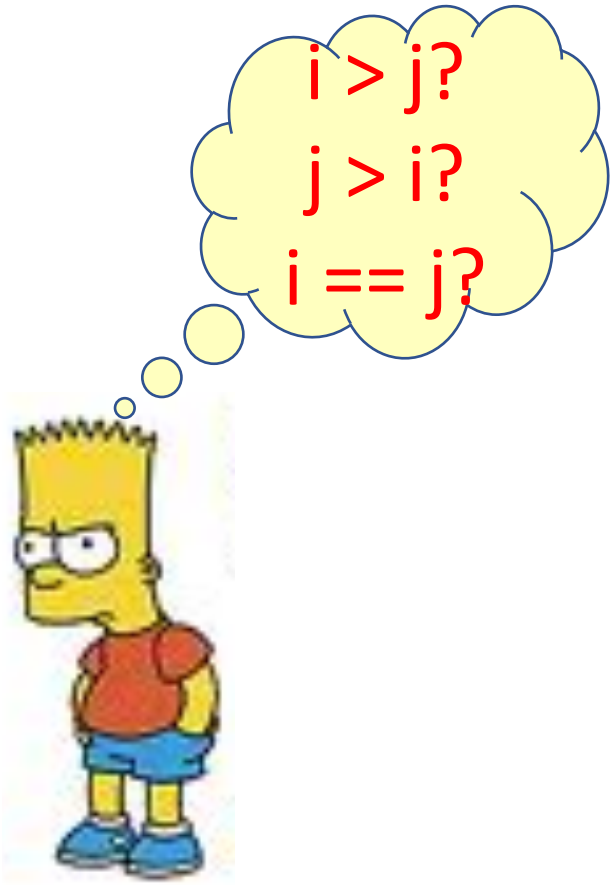
Ann: **If** you have $100, will you please give me a half?
Bob: Sure.

Ann: **If** you have $10, will you please give me a half?
Bob: No way!
Ann: Why?
Bob: I do not have $100 or more, but I do have $10.

find the max'm int

Given two integers, say i and j, respectively,

find out which one is bigger.



Who will win the competition?

Suppose we have runners: Alice and Bob. Who may win the competition?

Show Desktop.scf

If-statement

Who finishes first: Alice or Bob? The result depends on the runners' status.

(1) Possibility 1: Alice runs faster than Bob.

(2) Possibility 2: Bob runs faster than Alice.

(3) Possibility 3: There is a tie.

For results with uncertainty, we use if-statement.

if-statement

If the condition holds, then execute the statement.

If the condition does not hold, then do nothing.

```
if (condition)
  statement-run-only-if-condition-is-true;
```

Three parts of if-statement:

- Keyword if
- condition enclosed in ()
- statement-to-run-when-condition-is-true

if-statement

If the condition holds, then execute the statement.

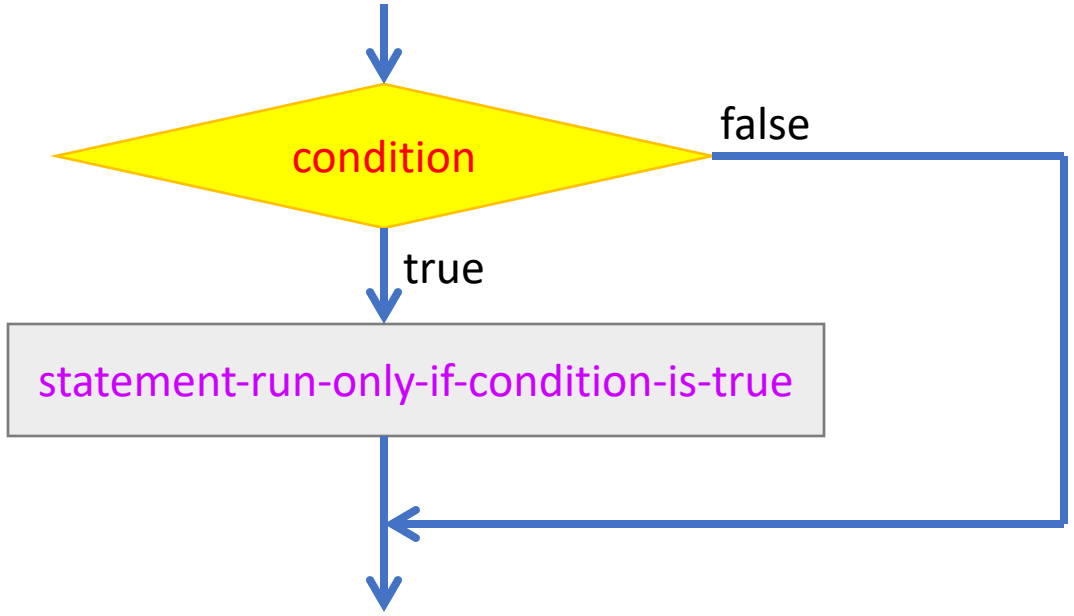
If the condition does not hold, then do nothing.

```
if (condition)
    statement-run-only-if-condition-is-true;
```

statement-to-run-when-condition-is-true may or may not be executed, depending on whether the condition is satisfied or not.

if-statement

```
if (condition)
  statement-run-only-if-condition-is-true;
```



Condition

Simplest form: compare between two items

== (equal)

!= (not equal)

>

<

>=

<=

Comparison: equality

```
if (expression1 == expression2)
    cout << "The expressions are equal.";
```

Warning:

expression1 == expression2 compares whether the two expressions have the same value, if yes, then the comparison returns true, otherwise, the comparison returns false.

Neither expression1 nor expression2 is changed.

Compare two expressions

```
if (5*5 == 6*5)
    cout << "5 * 5 == 6 * 5" << endl;
```

== (equality comparison) vs. = (assignment)

5*5 == 6*5; //invalid syntax: 5*5 == 6*5 is an expression, not an independent statement.

5*5 = 6*5; // = is assignment operator

// invalid syntax: the left-hand side of an assignment expression

//must be a variable name.

Compare two expressions

(1) (6*5 == 6*5) returns true;

(2) 6*5 = 6*5; is wrong in syntax: the left-hand side of an assignment expression must be a variable name.

(3)
```
int j = 5;
cout << (j == 5) << endl;
    //print out 1 (true).
cout << (j = 6) << endl;
    //() cannot be omitted. Print out 6.
```

Warning: if-statement ends with ;

An if-statement has three parts:

- if
- condition enclosed in parentheses ()
- statement-to-run-only-if-condition-is-true

- An if-statement ends with a semicolon ;
- semicolon ; follows statement-to-run-only-if-condition-is-true.

Warning, warning: If-statement

So …

(1) A ; follows an if-statement.

(2) **indent** statement-to-run-only-if-condition-is-true with if (condition) part.

```
if (condition)
    statement-to-run-only-if-condition-is-true;
```

Here ; end the whole if-statement.

Warning, warning: If-statement

What is the output?

```
(3')    if (the sun rises from the west) ;
            cout <<
                "You will give me one million.";
```

The above code are in fact **TWO** statements:

```
(3'')   if (the sun rises from the west)

            ;
        cout << "You will give me one million.";
```

if-else statement

If the condition holds, then execute the statement-run-only-if-condition-is-true

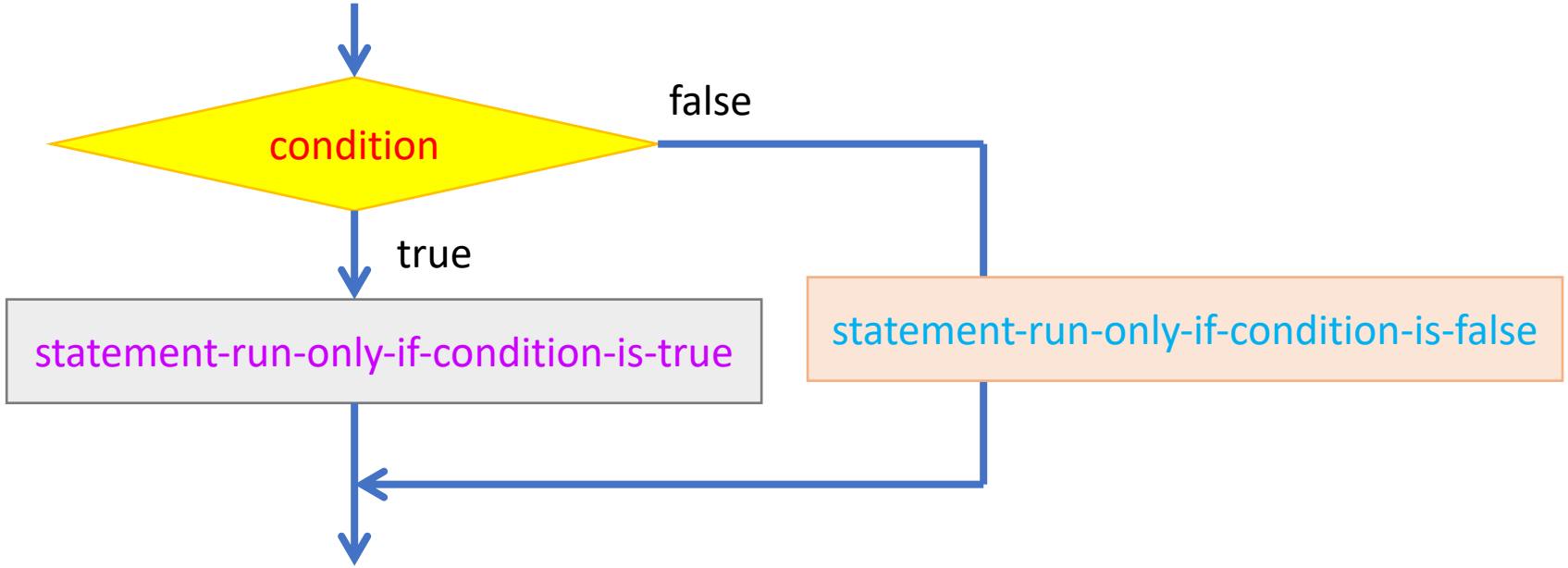
If the condition does not hold, then ~~do nothing~~.

Execute statement-run-only-if-condition-is-false

```
if (condition)
    statement-run-only-if-condition-is-true;
else statement-run-only-if-condition-is-false;
```

Flow chart of if-else statement

```
if (condition)
    statement-run-only-if-condition-is-true;
else statement-run-only-if-condition-is-false;
```



Summary

- if or if-else statement can answer questions like yes or no.
  - Is this a number a multiple of 3 or not?
- Do not put ; right after condition unless you are absolutely sure. Otherwise, an empty statement ; runs when the condition is satisfied. This means that the statement followed ; is run whether the condition holds or not.