*Control Flow
- Assignment
- Sequencing

Control Flow - Assignment

<var> = <expression>
<var> ← <expression>

$$\underbrace{\langle var\rangle}_{LHS} := \underbrace{\langle expression\rangle}_{RHS}$$

$x = y + w;$

r-value : var name refers to its value
l-value : var name refers to its location

Value and Reference Model for variables
(see last class's notes for definition)

Java ⌐ uses value model for all its built-in types: int, float, long, double, boolean, char
└ uses reference model for all objects + strings, arrays

$\underline{int}\ x = 10; \xrightarrow[\text{model}]{\text{value}} x\ \overset{int}{\boxed{10}}$

$int[\,]\ a = \{10, 12, 14\}; \xrightarrow[\text{model}]{\text{ref}} a\ \overset{ref\ int[\,]}{\boxed{\ }} \to \overset{0}{\boxed{10}}\overset{1}{\boxed{12}}\overset{2}{\boxed{14}}\ int$

Also ~~Array~~ ArrayList<int> a = new Arraylist<int>();

~~for(int x : a)~~
~~a.add~~

This is illegal because all elements of an Arraylist have to be objects

```
for(int i = 0; i < 10; i++)
    a.add(i);
```

Correct Version
ArrayList<Integer> a = new Arraylist<Integer>();

```
for (int i=0; i < 10; i++)
    A.add ((Integer) i);
```

or
A.add(new Integer(i));

It also allows: A.add(i); // Implicit conversion

In some languages, assignment is an <u>operator</u>.

i.e. <var> = <expression>
is also an expression.

In C, C++, Java we can write

$$\underbrace{x = \underbrace{a = \underbrace{b = 4}_{4}}_{4}}_{4};$$

<u>Watch out!</u>
Use of assignment as expression combined with how a language handles booleans can lead to errors!
for example,
In C, C++
- assignment is an expression (see above)
- booleans are handled as integer values
  i.e. 0 — false
  non-zero — true

So we can write

```
if (a = b) {      //a equals b??
   |
}
```

shouldve been ==
but this is <u>not</u> a ~~syntax~~ error!

Python
- has a boolean type bool
- True/False (btw False < True!)

Also
- any string other than "" is true
- Any number other than 0 is true
- Any list, tuple, or dictionary except [],(), {} is true

AND
```
if (a < b < c):
```
$\equiv$

is equivalent to:
```
if (a < b and b < c):
```
$\equiv$

In C,

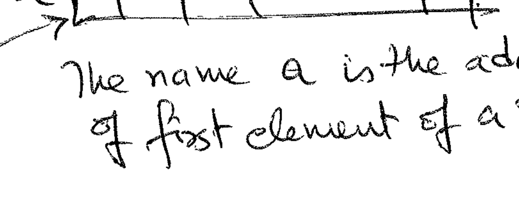
$$\text{if } (\underbrace{\underbrace{a < b}_{\text{yields } 0 \text{ or } 1} < c}_{0/1 < c})\ \{ \quad \text{No syntax error}$$

```
}
```

$\boxed{\text{BUG}}$

Combination Assignment Operators
+=, -=, *=, /=, %=   C, C++, Java, Python

Pointer Variables in C
```
int a[10];
int *p;
```
p is a pointer to an int (aka reference)



The name a is the address of first element of a[]

We can do
```
p = a;   or   p = &a[0];
```
Also, C allows: p++ and p+=3;

Multi-way Assignment

width, height = 500, 75

This is valid in Python

In general, we can do

var1, var2, ... = e1, e2, - . . . .

Also, we can do

$x, y = y, x$ #swaps x and y!

Functions ~~can~~ in Python can return multiple values (tuples)

e.g. found, index = search(L, item)

[Variable Initialization]

When a variable is defined, is it initialized?

e.g. int x;

What is the value of x?

Java
- boolean : false
- int : 0   char : \0
- float : 0.0
- String : null

C
- int : 0
- char : \0
- float : 0.0

What about Python?

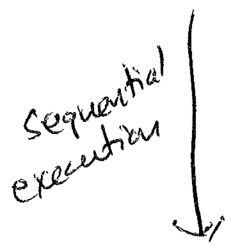
What about arrays?

int[10] a;
initialized to 0 for int
Similarly for other types

int a[10];
only static arrays are initialized.

Control Flow - Sequencing

All instructions in Imperative PLs are carried out in the sequence written. i.e.

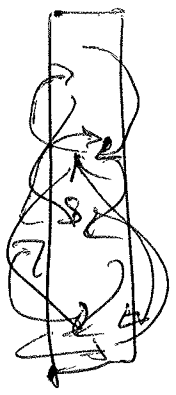


do this
then do this
and then this
etc.

Unstructured Flow : FORTRAN

```
10  IF (a .LE. b) GOTO 25
15   min = b
20   GOTO 30
25   min = a
30   ----
```

$\boxed{\text{Spaghetti code}}$



Famous Paper: GOTO Statements Considered Harmful
Introduced Structured Programming
- structured statements - if, loops
- exit from a loop - break, continue
- returning from function calls - return
- returning/exiting from a deeply nested block/fn-call
  - exceptions & exception handling

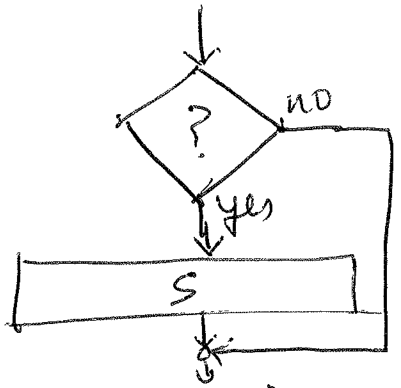
Structured Flow/Programming

Essentials
- sequencing
- selection
- iteration
+ functions
+ abstractions

Control Flow - selection

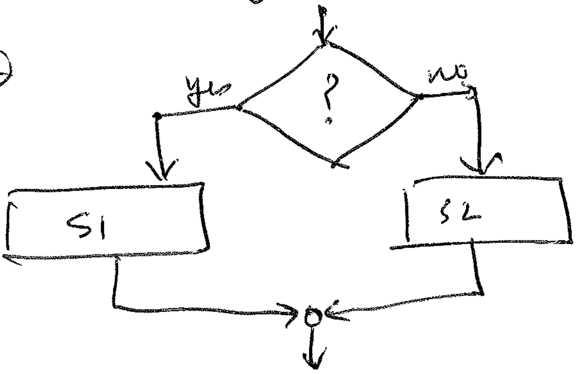
- allows a choice among a set of statements.

①



```
if <condition> then
begin
        S
end
```

②



```
if <condition> then
begin    S1
end
else begin
        S2
    end.
```

Designs

C, C++, Java

```
if ( <condition> ) {
        S1
}
else {          <- optional
        S2
}
```

Python

```
if <condition>:
      S1
else:
      S2
```

Also elif see lab#2

LISP

```
(cond (<c1> <s1>)
      (<c2> <s2>)
      :
      (<cn> <sn>))
```

Dangling-else problem if c1 if c2 S1 else S2

```
if  c1                  if  c1
    if                      if c2
        c2      OR              S1
else                        else
    S2                          S2
```

Solution

else associates with the closest unmatched if

OR use braces

Dangling-else-solutions

C

```
if (C1) {
    if (C2)
        S1;
    else
        S2;
}
```

(1)

OR

```
if (C1) {
    if (C2)
        S1;
}
else
    S2;
```

(2)

By default a dangling-else matches the closest if - i.e. (1)

Q. what about Python??