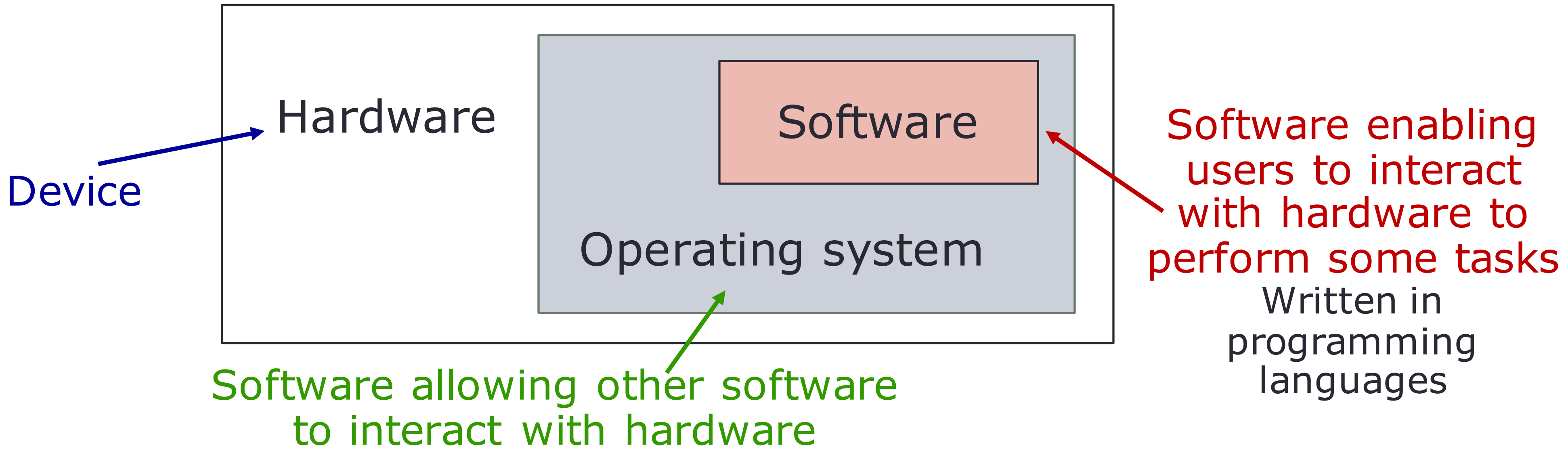
Algorithm and Ambiguity

CS 1111
Introduction to Programming

Spring 2019

Computing is Everywhere

Computing



Art of computer science (problem solving)

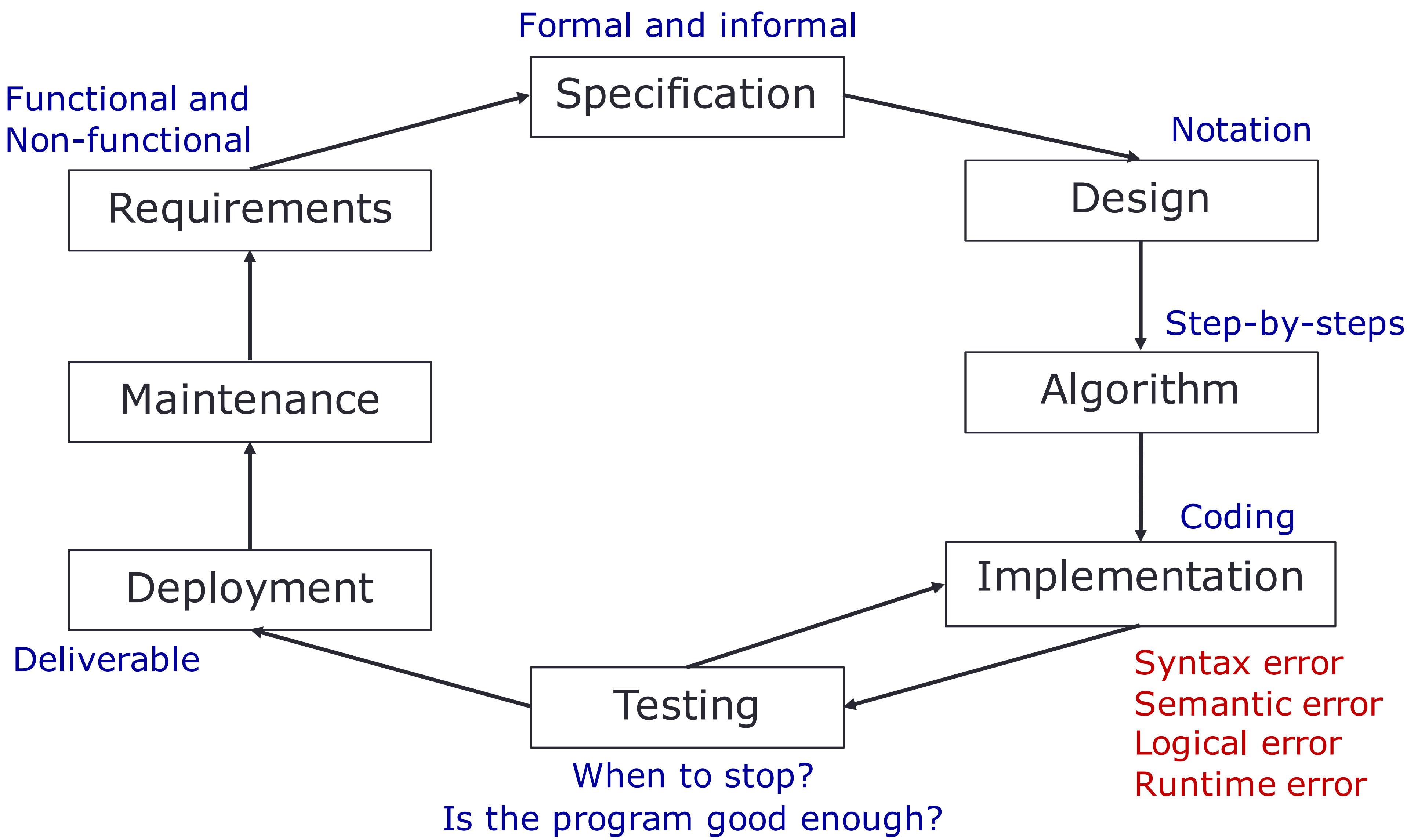
how to come up with solution

how to know if solution will work

Programming skill

how to automate solution

Software Development Life Cycle



Types of Errors

Syntax error

- Does not conform to the rules of the programming language (e.g., incorrect grammar, typo)

Semantic error

- Yields nothing meaningful (e.g., forget to divide by 100 when printing a percentage amount)

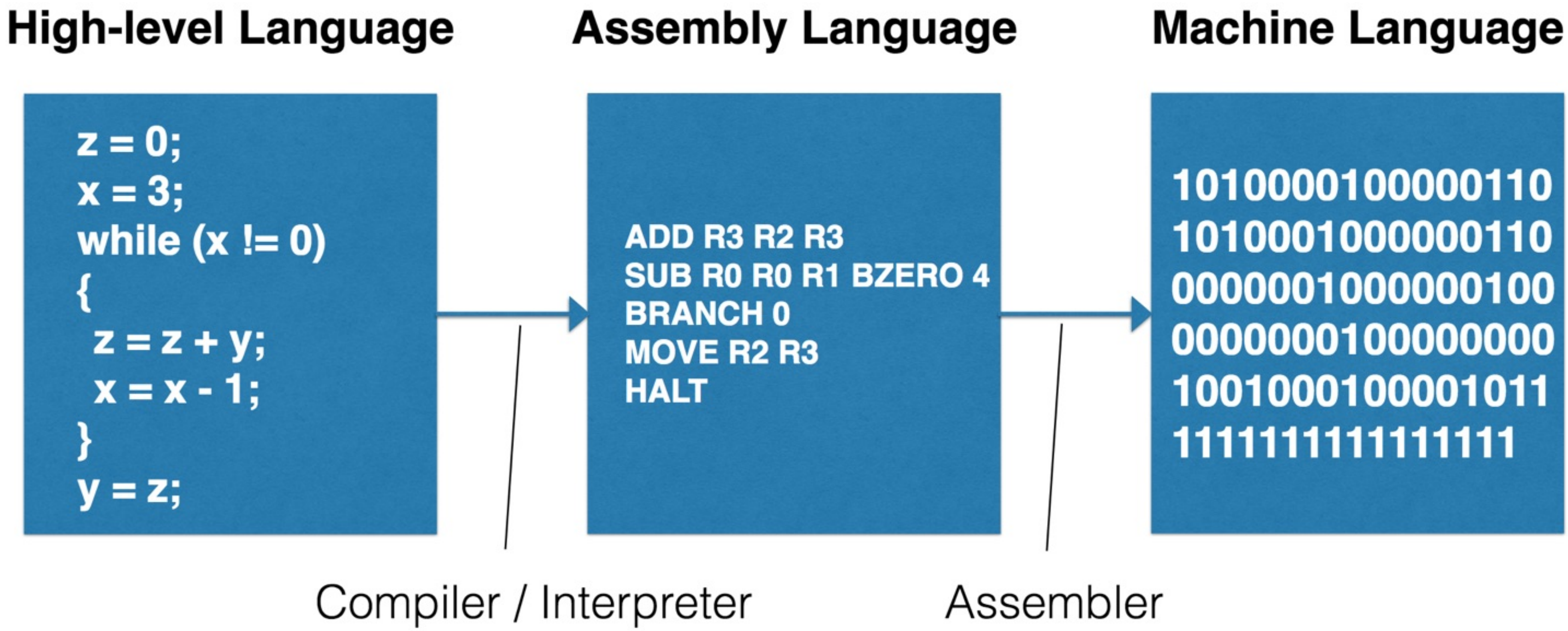
Logical error

- Causes the program to operate incorrectly, not crash
- The syntax is correct, but executes without performing the intended action, may produce incorrect output or unintended behavior

Runtime error

- Happens when running the program, generates an exception that terminates the program with an error message

Programming Languages



Algorithms

- A step by step, list of instructions that if followed exactly will solve the problem under consideration.
- Can be described in many ways. Two commonly used methods:
  - Pseudocode
  - Flowchart

Always think about a general solution, then write it in a programming language so the computer can do it.

Good Algorithms

Algorithms must be:

- Unambiguous
  - There are precise instructions for what to do at each step and where to go next.
- Executable
  - Each step can be carried out in practice.
- Terminating
  - It will eventually come to an end.

Don't think about implementation yet.
Try to focus on "how you want to solve the problem"

Pseudocode

- Pseudocode is one of the methods that can be used to represent / describe an algorithm (usually in English)
  - Informal description of an algorithm
- Not use specific programming language syntax
- Can be easily translated into a high-level programming language
- Usually include terms specifying a sequence of actions the a program will take

Control Structures

Sequence

- A series of statements that execute one after another

Condition (if)

- To decide which of the two or more different statements to execute depending on a certain condition

Repetition (loop)

- To repeat statements while certain conditions are true

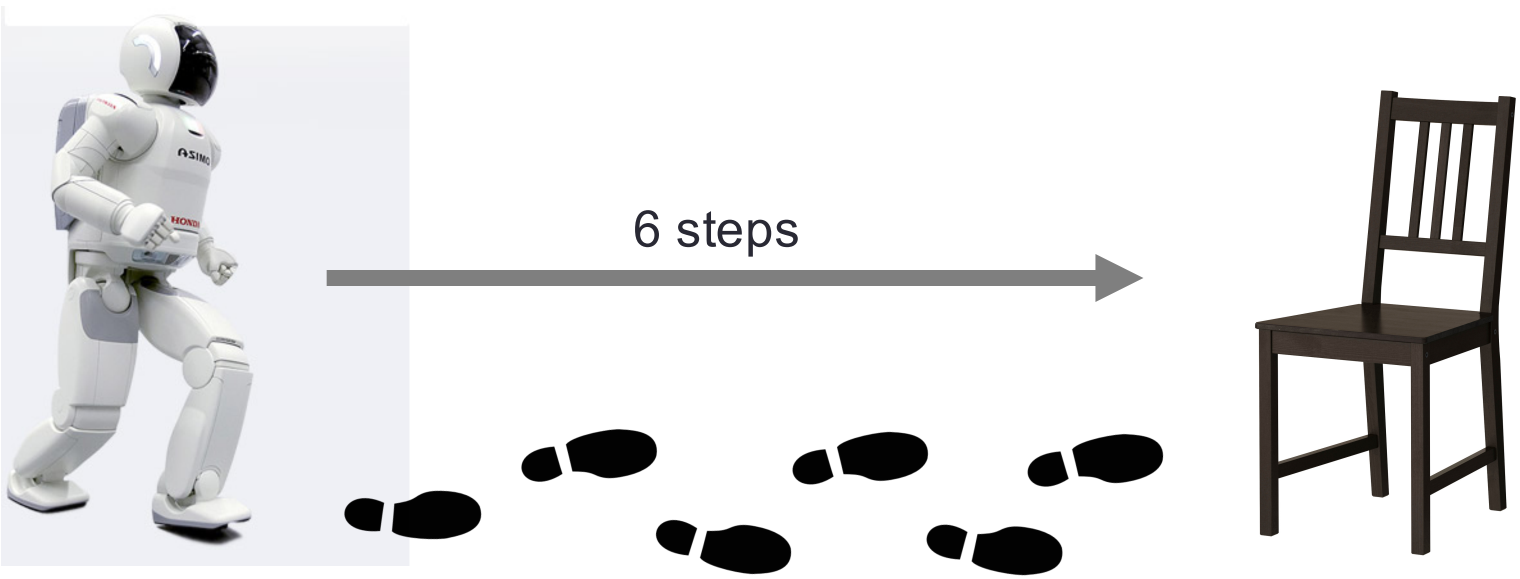
Subprogram / named action

- A small part of another program solving a certain problem
- A collection of subprograms solves the original problem

Control Structures

Sequence

- A series of statements that execute one after another



statement1
statement2
statement3
statement4
…

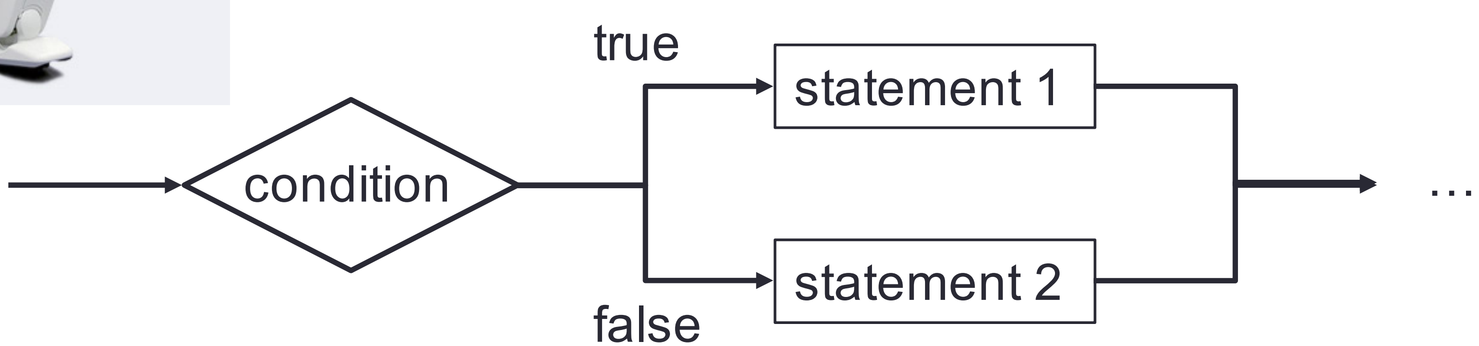
walk, walk, walk, walk, walk, walk, right-turn-180-degree, sit

Control Structures

Condition (if)

- To decide which of the two or more different statements to execute depending on a certain condition

```
If (condition):
   statement1
else:
   statement2
```



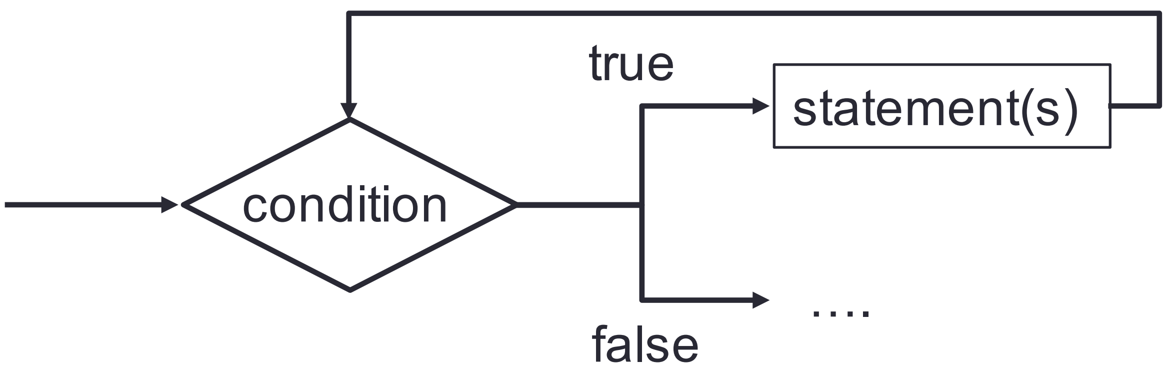
Control Structures

Repetition (loop)

- To repeat statements while certain conditions are true

? steps

- ~~Repeatedly walk 6 steps~~
- Repeatedly walk until you are in front of the chair
- Right-turn-180-degree
- Sit

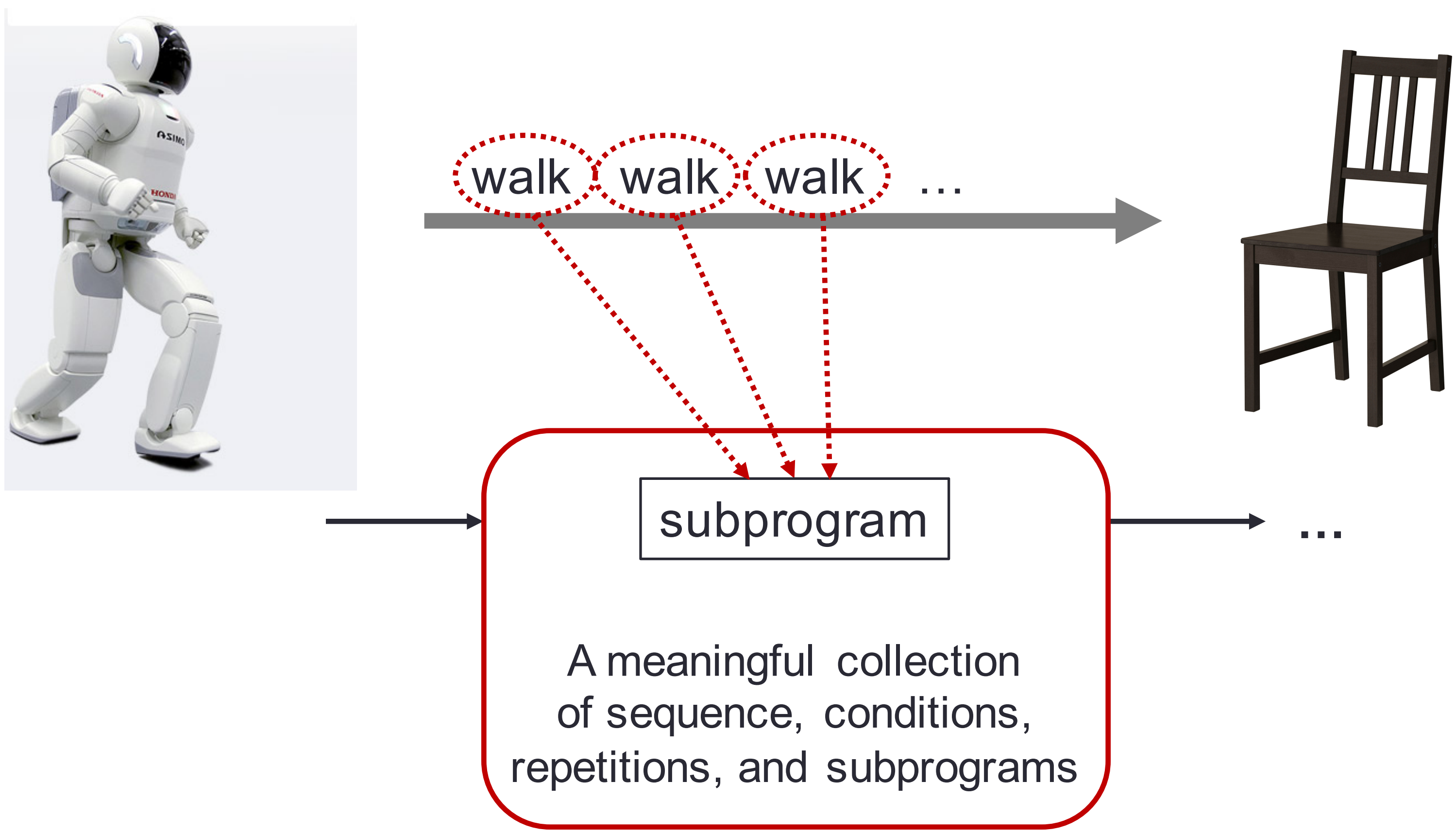



```
while (condition):
    statement1
    statement2
statement3
 …
```

Control Structures

Subprogram / named action

- A small part of another program solving a certain problem
- A collection of subprograms solves the original problem



Activity: "If You're Happy"

Write a pseudocode to tell a robot-1111 computer to perform the "If You're Happy" song (sing, clap, stomp, shout, …)

You may assume the robot-1111 computer knows what to do when it is instructed to "sing," "clap," "stomp," "shout", …

You may review the video before writing the pseudocode if you'd prefer

https://www.youtube.com/watch?v=Im5i7EqZE1A

Now … you try …

Let’s Try – “If You’re happy”

How many times? Or until when?

Repeat

Sing “If you’re happy and you know it, clap your hands”

Repeat

How many times? Or until when?

Clap

Sing “If you’re happy and you know it, then your face will surely show it”

Sing “If you’re happy and you know it, clap your hands”

Repeat

How many times? Or until when?

Clap

[https://www.youtube.com/watch?v=Im5i7EqZE1A]

Make It Unambiguous

Sequence

Repeat 2 times

Sing "If you're happy and you know it, clap your hands"

Repeat 2 times

Clap

Sing "If you're happy and you know it, then your face will surely show it"

Sing "If you're happy and you know it, clap your hands"

Repeat 2 times

Clap

Repetition

Subprogram

[https://www.youtube.com/watch?v=Im5i7EqZE1A]

Activity: 3X + 1

Let’s pretend, you are an “awesome-robot” and follow the instructions below:

```
Let X be your age in years
Repeat as long as X is not 1:
    If X is even:
        Divide X by 2
    Otherwise:
        Multiple X by 3 and add 1
Clap as many times as you repeated
```

Now … awesome-robot … do this …

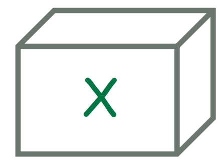
Let’s Consider

variable

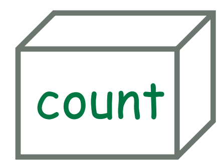
Named value that can vary over the course of doing something

```
Let X be your age in years
Repeat as long as X is not 1:
    If X is even:
        Divide X by 2
    Otherwise:
        Multiple X by 3 and add 1
Clap as many times as you repeated
```

How to tell?



Keep track of the value of X



Keep track of the number of times repeated

Make It Unambiguous

Let X be your age in years

Let count start as 0

Repeat as long as X is not 1:

    Add 1 to count

    If X is even:

        Divide X by 2,
        and remember the value as X

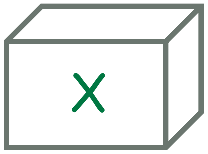
    Otherwise:

        Multiple X by 3 and add 1,
        and remember the value as X

Clap count times

assign

assign



X = 20
X = 10
X = 5
X = 16
X = 8
X = 4
X = 2
X = 1

count = 0
count = 1
count = 2
count = 3
count = 4
count = 5
count = 6
count = 7