

# CSE 1061 **Introduction to Computing**

## Lecture 3

Week 2

Fall 2015



Department of Computing  
The School of EE & Computing  
Adama Science & Technology University

## Characteristics of Python

### Instruction set

Arithmetic and logical operations

+, -, \*, /, and \*\*

and, or, not

Assignment

Conditionals

Iterations

Input/output

} for defining  
expressions

**No pointers**

**No declarations**

- A small grid-like 2D world

### Basic actions

move (): moving one grid forward

turn\_left (): turning left by 90°

pick\_beeper(): pick ing up beepers

drop\_beeper(): putting down beepers

## **Our own instructions: functions**

### Comments

# OUTLINE

---

**Conditionals**

**Iterations**

**for\_loops**

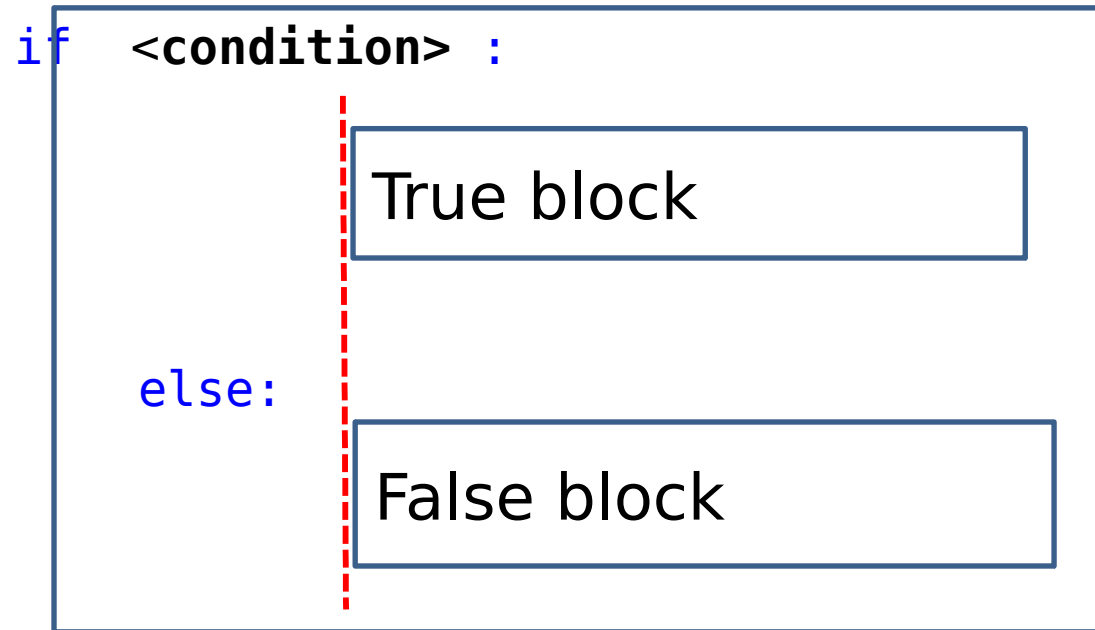
**while\_loops**

Reading assignment of the text book :

**Chapter 4 Conditionals and recursion**

**Chapter 6 Iteration**

# CONDITIONALS



<**condition**> has a “True” or “False” value, representing true or false, respectively .

**If** it is true, the **True block** is executed; **otherwise**, the False block is executed.



What will be printed ?

```
if True:
    print "CSE1061 is my favorite course"
else:
    print "Every pre student will receive an A+"
```

Now, do you understand?

```
if 5 > 3:
    print " CSE1061 is my favorite course"
else:
    print "Every pre student will receive an A+"
```

Now, what will happen?

```
if False:
    print "Every pre student will receive an A+"
```

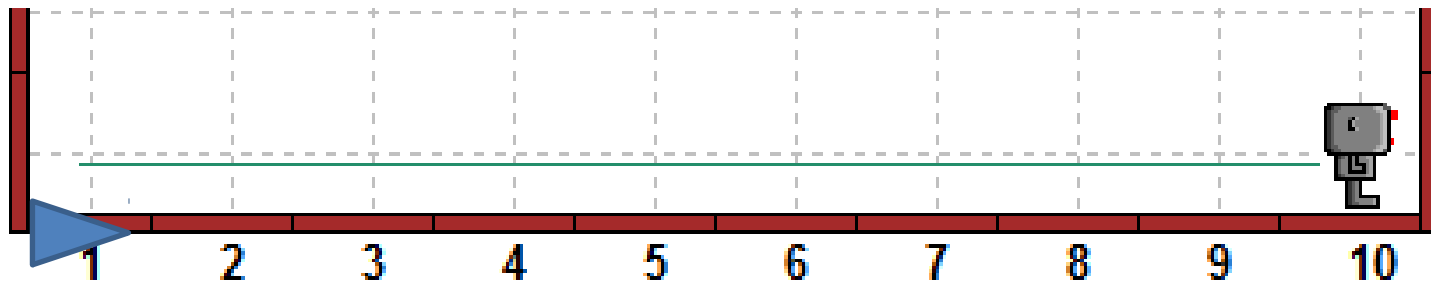
The keyword `not` inverts the sense of the condition: `not True` is False, and `not False` is True.

What is the output?

```
>>print not 3 < 5
```

# SENSING A BEEPER

We want Hubo to make 9 steps and pick all **beepers** on the way. However, we do not know where beepers are. If there is **no beeper** at a grid point, then **hubo.pick\_beeper()** causes an error.



How to sense a beeper?

Use **hubo.on\_beeper()**





Move and pick a beeper if any.

Take a step forward.

Check if there is a beeper

or  
Move forward 9 steps.  
At each step, move  
if yes, pick it up. and pick up a  
beeper  
if any.

```
def move_and_pick():  
    hubo.move()  
    if hubo.on_beeper():  
        hubo.pick_beeper()
```

```
for i in range(9):  
    move_and_pick()
```

No False block!!

Let's do the opposite: we want to drop a beeper, but only if there is no beeper at the current location.

if **not** `hubo.on_beeper()`:

`hubo.drop_beeper()`

**Ex) Program: "0 move\_and\_pick.py"**



# FOLLOWING THE BOUNDARY

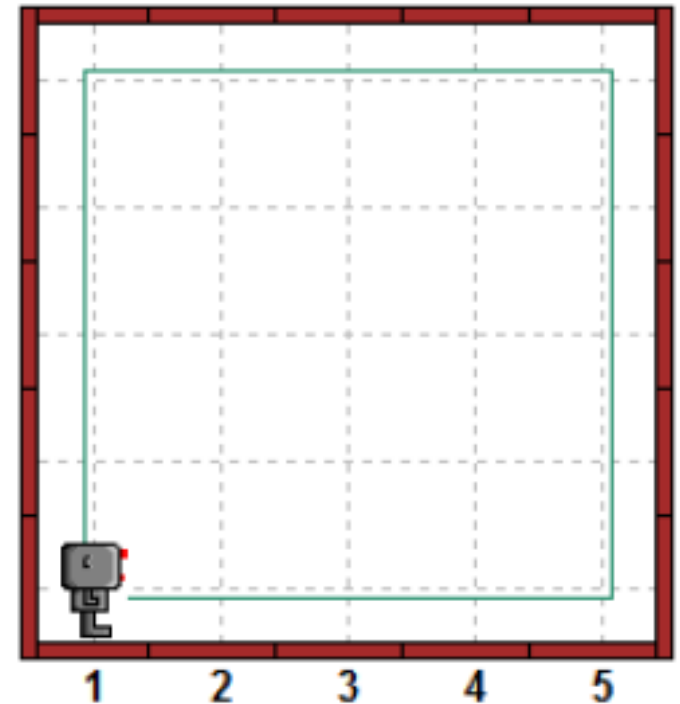
Hubo tries to follow the boundary of the world: He moves forward if there is no wall; otherwise, turn to the left.

```
from cs1robots import *  
create_world(avenues =5, streets = 5)  
hubo = Robot()  
hubo.set_trace("blue")
```

```
def move_or_turn():  
    if hubo.front_is_clear():  
        hubo.move()  
    else:  
        hubo.turn_left()
```

```
for i in range(20):  
    move_or_turn()
```

**Why 20 ?**



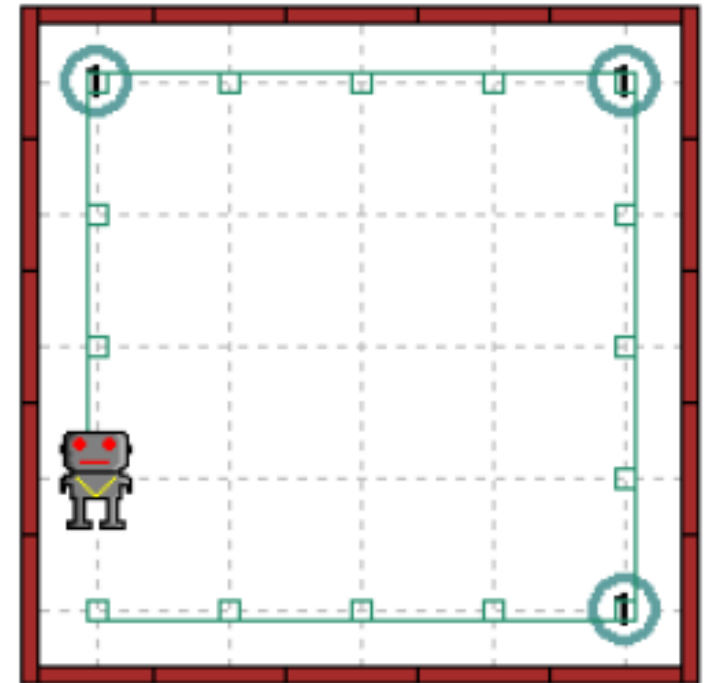
# MOVING AND DANCING

```
hubo = Robot(beeper=5)
```

```
def dance():  
    for i in range(4):  
        hubo.turn_left()  
  
def move_or_turn():  
    if hubo.front_is_clear():  
        dance()  
        hubo.move()  
    else:  
        hubo.turn_left()  
        hubo.drop_beeper()
```

```
For i in range(18):  
    move_or_turn()  
dancing.py
```

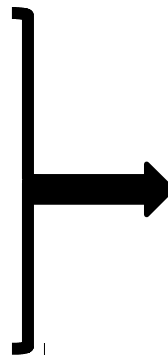
## Ex: 1



# MULTIPLE CHOICES

**elif** combines **else** and **if** to express many alternatives without complicated indentation.

```
if hubo.on_beeper():
    hubo.pick_beeper()
elif hubo.front_is_clear():
    hubo.move()
elif hubo.left_is_clear():
    hubo.turn_left()
elif hubo.right_is_clear():
    turn_right()
else:
    turn_around()
```

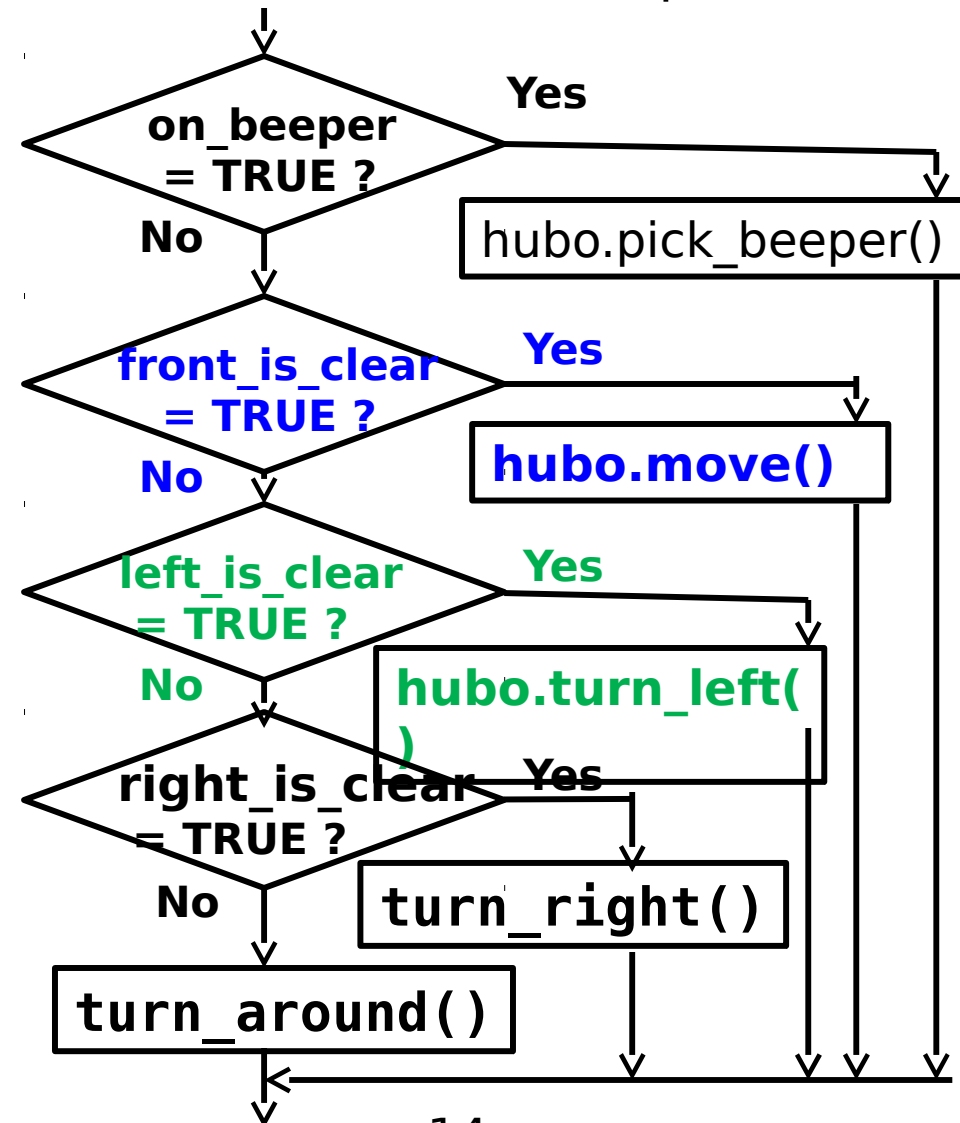


```
else:
    if hubo.front_is_clear():
        hubo.move()
    else:
        if hubo.left_is_clear():
            hubo.move()
```

# MULTIPLE CHOICES

**elif** combines **else** and **if** to express many alternatives without complicated indentation.

```
if hubo.on_beeper():  
    hubo.pick_beeper()  
elif hubo.front_is_clear():  
    hubo.move()  
  
elif hubo.left_is_clear():  
    hubo.turn_left()  
  
elif hubo.right_is_clear():  
    turn_right()  
  
else:  
    turn_around()
```



# WHILE-LOOPS

---

**while** <condition>:

block

The block is executed as long as <**condition**> is True;  
**otherwise**, it is skipped.

A **while-loop repeats** instructions as long as a **condition is true**.

```
while not hubo.on_beeper():  
    hubo.move()
```

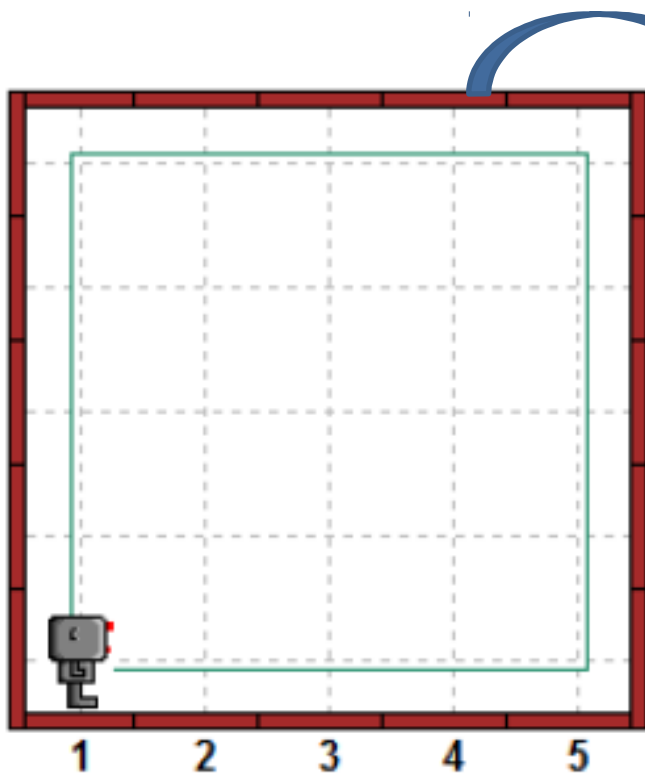
**Move forward as  
long as there is no  
beeper**

A **for-loop repeats** some instructions a **fixed number of times**.

```
for i in range(9):  
    hubo.move()
```



Let's write a program to let the robot walk around the boundary of the world until he comes back to the starting point



Think your solution for this problem □  
your method □  
your **Algorithm** for solving a problem !!

1. Put down a beeper to mark the starting point
2. **Repeat steps 3 and 4** while no beeper found
3. If not facing a wall, move forward
4. Otherwise, turn left
5. **Finish** when we found the beeper

```
hubo.drop_beeper()
```

```
hubo.move() Why this?
```

```
while not hubo.on_beeper():
```

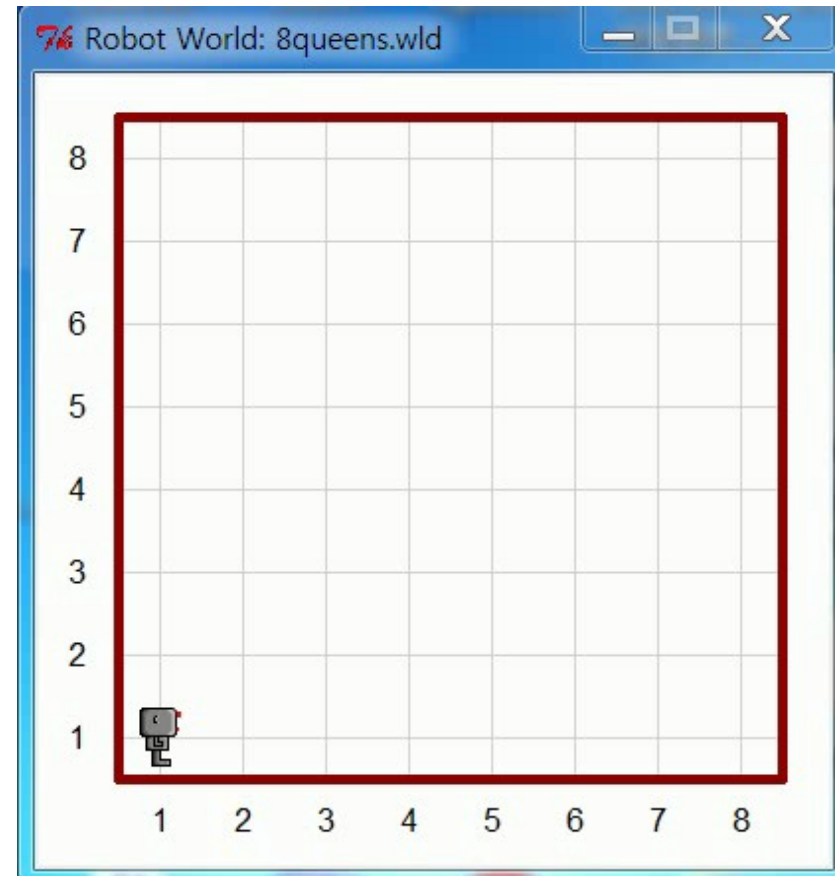
```
    if hubo.front_is_clear():
```

```
        hubo.move()
```

```
    else:
```

```
        hubo.turn_left()
```

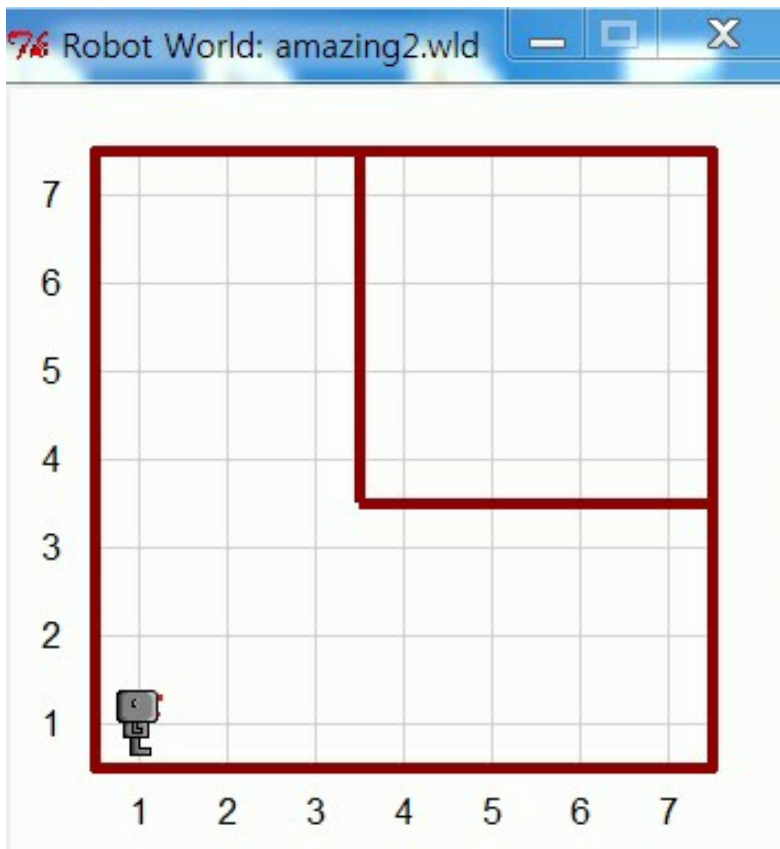
```
hubo.turn_left()
```



Does this program always work?

Well, .....

## How about this case?



```
hubo.drop_beeper()  
hubo.move()  
while not  
hubo.on_beeper():  
    if  
hubo.front_is_clear():  
        hubo.move()  
    else:  
        hubo.turn_left()  
hubo.turn_left()
```

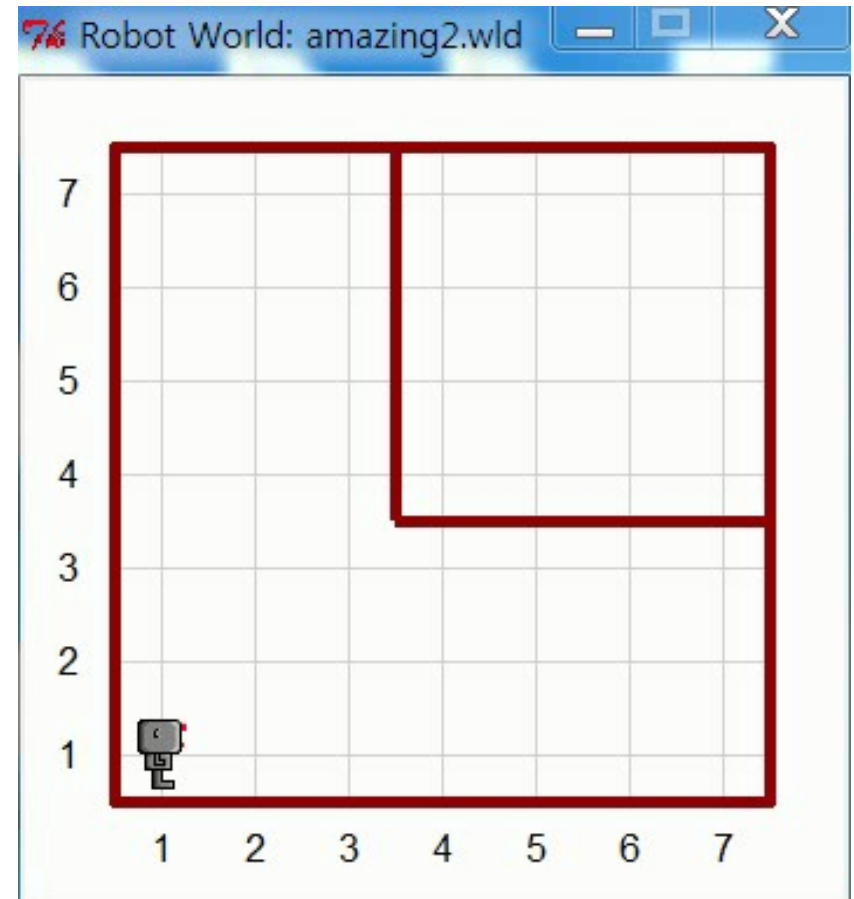
Try the code in the previous page with "amazing2.wld" and see if the previous code works.

Sometimes we need a right  
turn!

## Does this work?

Well, .....

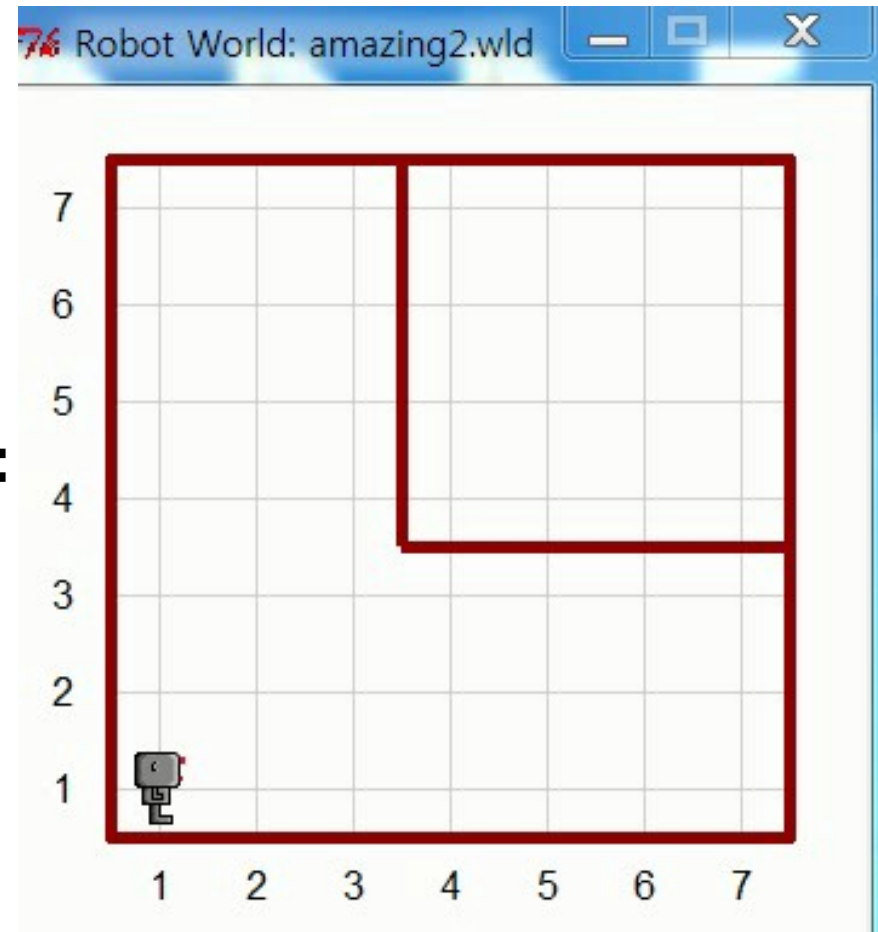
```
hubo.drop_beeper()
hubo.move()
while not hubo.on_beeper():
    if
hubo.right_is_clear():
    turn_right()
    elif
hubo.front_is_clear():
    hubo.move()
    else:
        hubo.turn_left()
hubo.turn_left()
```



Infinite loop!

## How about this?

```
hubo.drop_beeper()  
hubo.move()  
while not hubo.on_beeper():  
    if hubo.right_is_clear():  
        turn_right()  
        hubo.move()  
    elif  
hubo.front_is_clear():  
        hubo.move()  
    else:  
        hubo.turn_left()
```

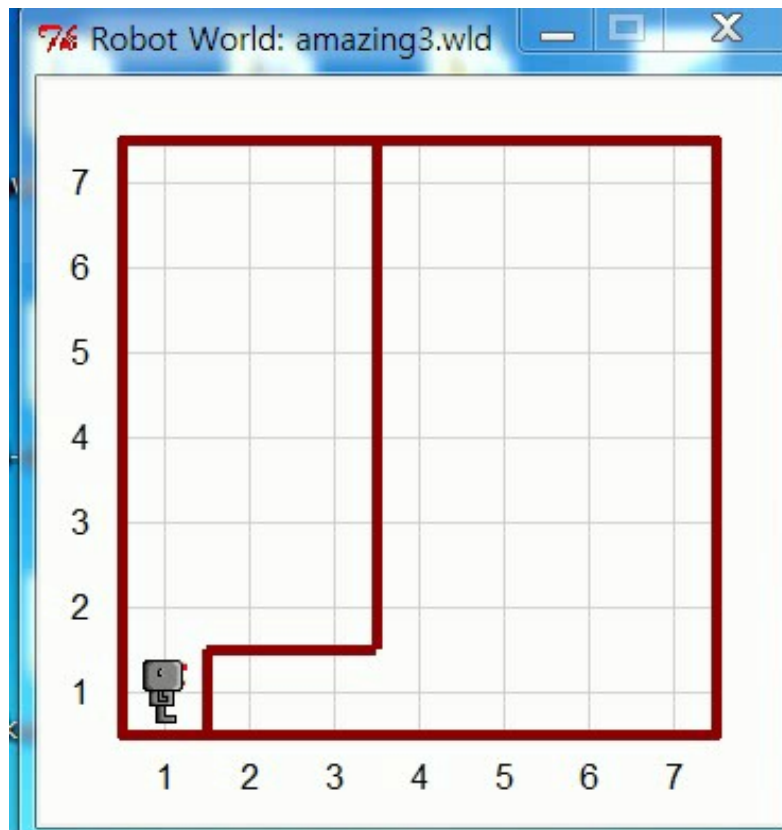


Does this always work?

## How about this case?

```
hubo.drop_beeper()  
hubo.move()  
while not hubo.on_beeper():  
    if hubo.right_is_clear():  
        turn_right()  
        hubo.move()  
    elif  
hubo.front_is_clear():  
    hubo.move()  
else:  
    hubo.turn_left()
```

## Does this work?

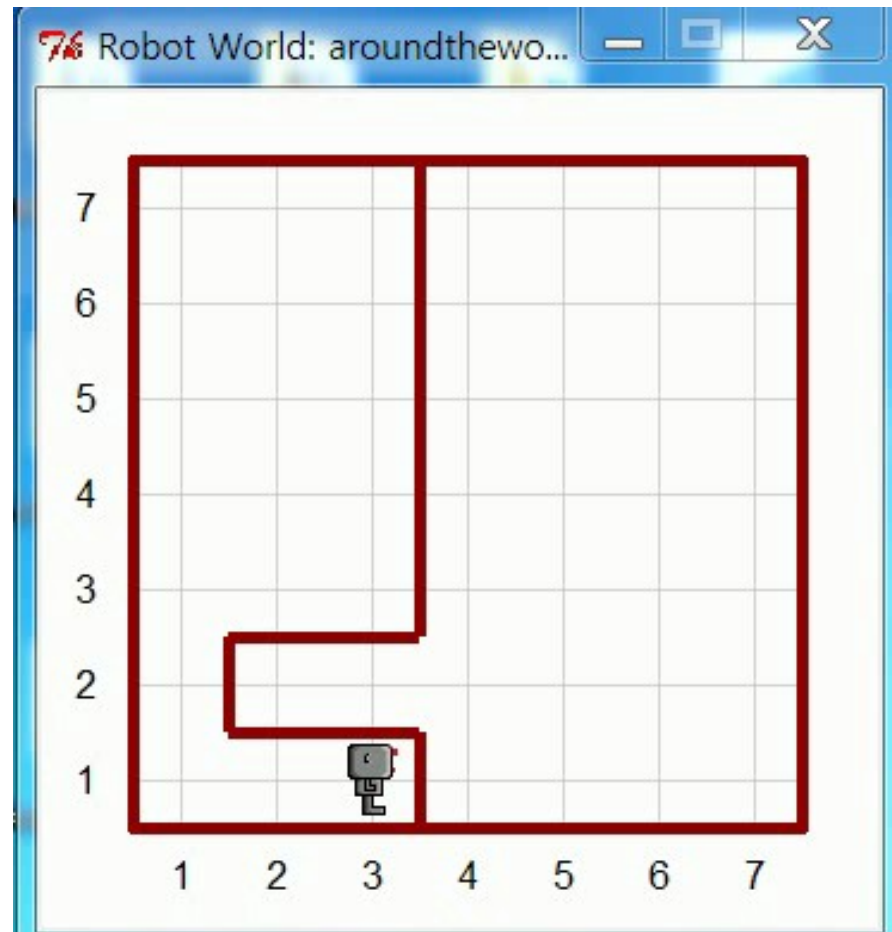
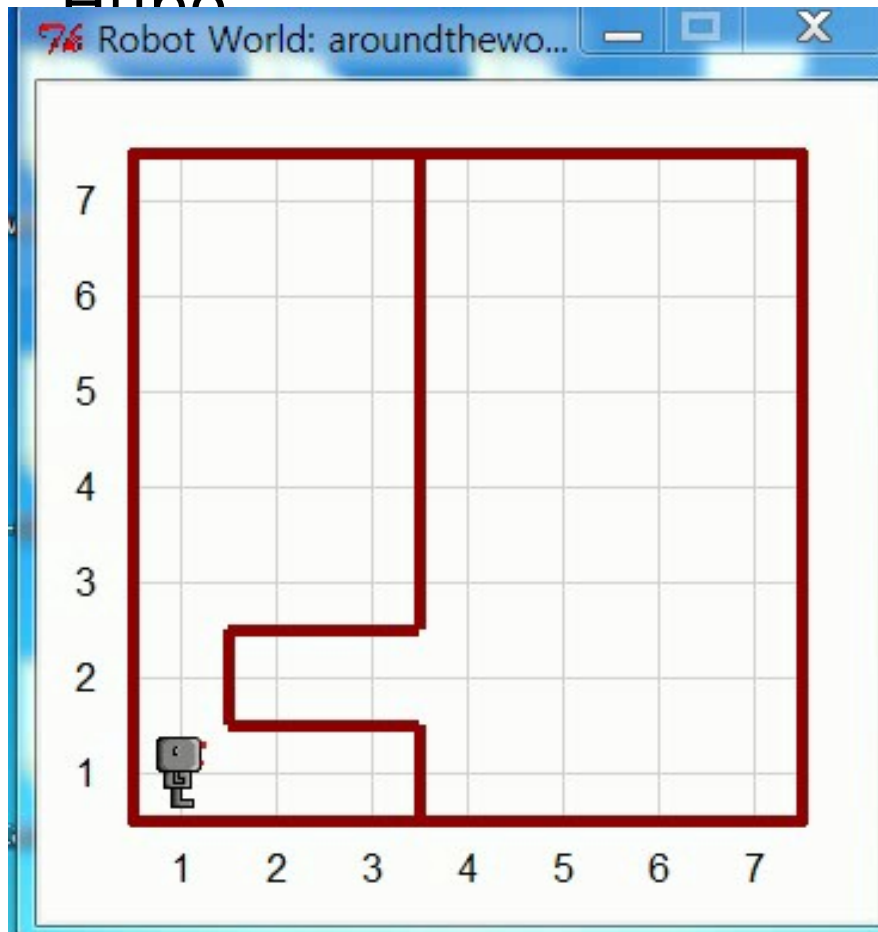


```
hubo.drop_beeper()
while not
hubo.front_is_clear():
    hubo.turn_left()
hubo.move()
while not hubo.on_beeper():
    if hubo.right_is_clear():
        turn_right()
        hubo.move()
    elif
hubo.front_is_clear():
        hubo.move()
    else:
        hubo.turn_left()
Hubo.turn_left()
```

Does this always work?

**Still not perfect !**

Very **sensitive** to the **initial position** of  
Hubo







```
hubo.drop_beeper()
while not
hubo.front_is_clear():
    hubo.turn_left()
hubo.move()
```

← `def`  
`mark_and_move():`

```
while not
hubo.on_beeper():
    if hubo.right_is_clear():
        turn_right()
        hubo.move()
    elif
hubo.front_is_clear():
        hubo.move()
    else:
```

← `def follow_wall():`

```
mark_and_move()
while not
hubo.on_beeper():
    follow_wall()
hubo.turn_left()
```

```
        hubo.turn_left()
Hubo.turn_left()
```

# COMMENTS FOR HUMANS

---

One of the **secrets** of writing **good, correct, elegant programs** is to write them as if you wrote them for a **human reader**, not a computer. Let's clean up our program

.

How ? By adding comments !



```
"""  
This program lets the robot go around his world counter-  
clockwise, stopping when he comes back to his starting point.  
"""
```

```
#Turn right.
```

```
def turn_right():  
    for i in range(3):  
        hubo.turn_left()
```

```
#Mark the starting point and move
```

```
def mark_and_move():  
    hubo.drop_beeper()  
    while not hubo.front_is_clear():  
        hubo.turn_left()  
    hubo.move()
```

```
(continued)
```



---

#Follow the wall at each iteration.

```
def follow_wall():
    if hubo.right_is_clear():
        # turn right to follow the wall
        right_turn()
        hubo.move()
    elif hubo.front_is_clear():
        # move forward while following the wall
        hubo.move()
    else:
        # turn left to follow the wall
        hubo.turn_left()
```

(continued)



---

#Begin actual move.

mark\_and\_move()

#Follow the entire wall.

while not hubo.on\_beeper():

    follow\_wall()

Hubo.turn\_left()

# STEPWISE REFINEMENT

---

1. Start with a **primitive program** that solves a simple problem.
2. Make **small changes**, one at a time to generalize the program.
3. Make sure that each change **does not invalidate** what you have done before.
4. Add appropriate **comments** (not just repeating what the instruction does).
5. Choose **descriptive names**.