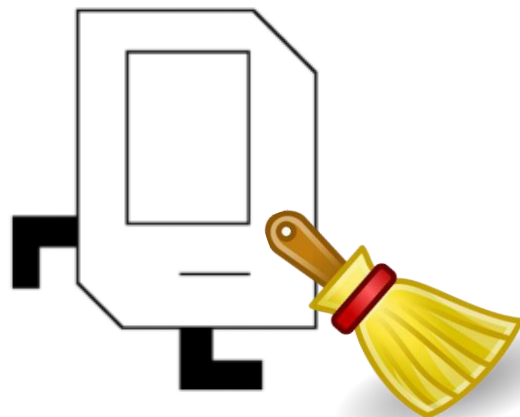




# Control Flow

CS106A, Stanford University

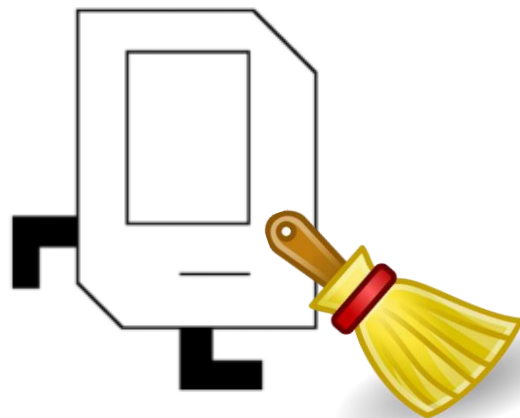
# Housekeeping I



- Class website: <http://cs106a.stanford.edu>
- Section sign-ups (sections start next week)
  - Sign-up at: <http://cs198.stanford.edu> (will be on CS106A page)
  - Sign-ups start Thurs., March 31 at 5pm; end Sun., April at 5pm
  - Not first-come, first-served, but make sure to sign-up
- Assignment #0 still open (over 350 responses so far)
  - ~70% have done 0-10 hours of programming
  - You are an amazing group of people!



# Housekeeping II



- Please send OAE letters to Juliette and me
- Application open for CS100A (link on CS106A website)
  - 1-unit supplementary section for stronger foundation
- We are using “ed” discussion forum
  - Link on top right corner of CS106A class web page
- LaIR Helper Hours start this Sunday (April 3)
  - Located in Durand Building, Room 353



# Black LaIR

## Come to Black LaIR for Assignment Help!

Black LaIR is open to **everyone** for **conceptual** and **debugging help** on **CS106A Assignments**. We hope to see you there!

### What:

Virtual, one-on-one **conceptual** and **debugging help** sessions for CS106A and CS106B held through **QueueStatus** and **Zoom**.

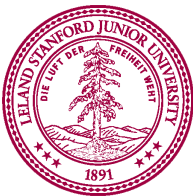
### When:

- **Tuesdays:** 5 - 8 PM PST
- **Thursdays:** 5 - 8 PM PST
- **Saturdays:** 12 - 3 PM PST


CS106A QueueStatus: <https://queuestatus.com/queues/753>

Visit <https://blackincs.stanford.edu/black-lair> to learn more, and reach out to [ajarno@stanford.edu](mailto:ajarno@stanford.edu) if you have any questions or concerns!

Stanford University



# Install PyCharm

 CS106A   HANDOUTS   LECTURES   ASSIGNMENTS   SECTIONS



## CS106A Programming Methodology

Spring Quarter 2022

Lecture MWF 12:15-1:15pm in Hewlett 200

### TEACHING TEAM

Mehran Sahami



Instructor

### ANNOUNCEMENTS






#### ACE Application

Last updated yesterday by Juliette

#### CS106A ACE Section

If you are taking CS106A and feel you would benefit from extra practice in addition to your default CS106A section, consider applying for CS100A!

### QUICK LINKS

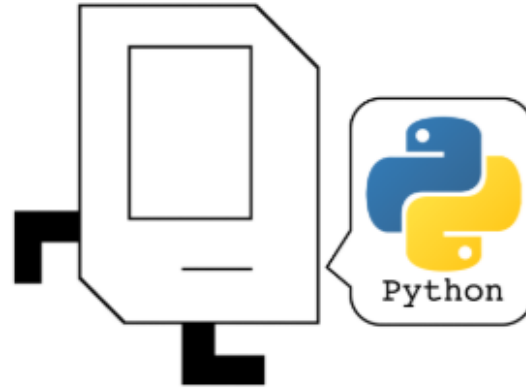
-  Discussion Forum
-  Canvas Page (lecture recordings)
-  Karel Reader
-  **PyCharm** PyCharm
-  Python Guide

Please follow instructions *closely*.  
Post on Ed if you have problems.

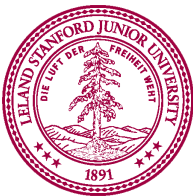
Sahami, CS106A, Stanford University



# Using Karel and Assignment 1

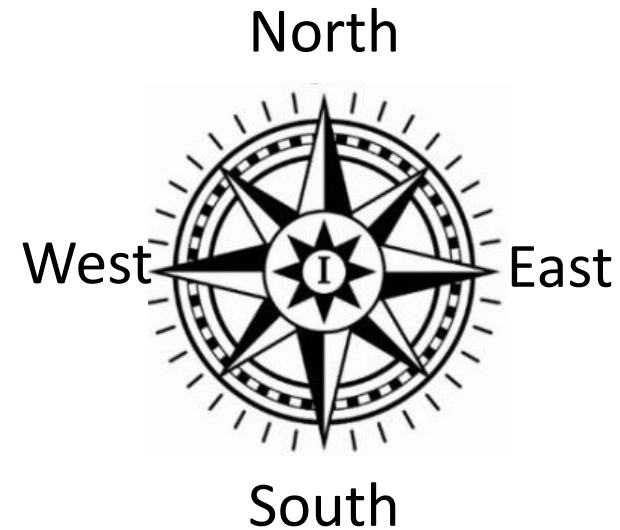
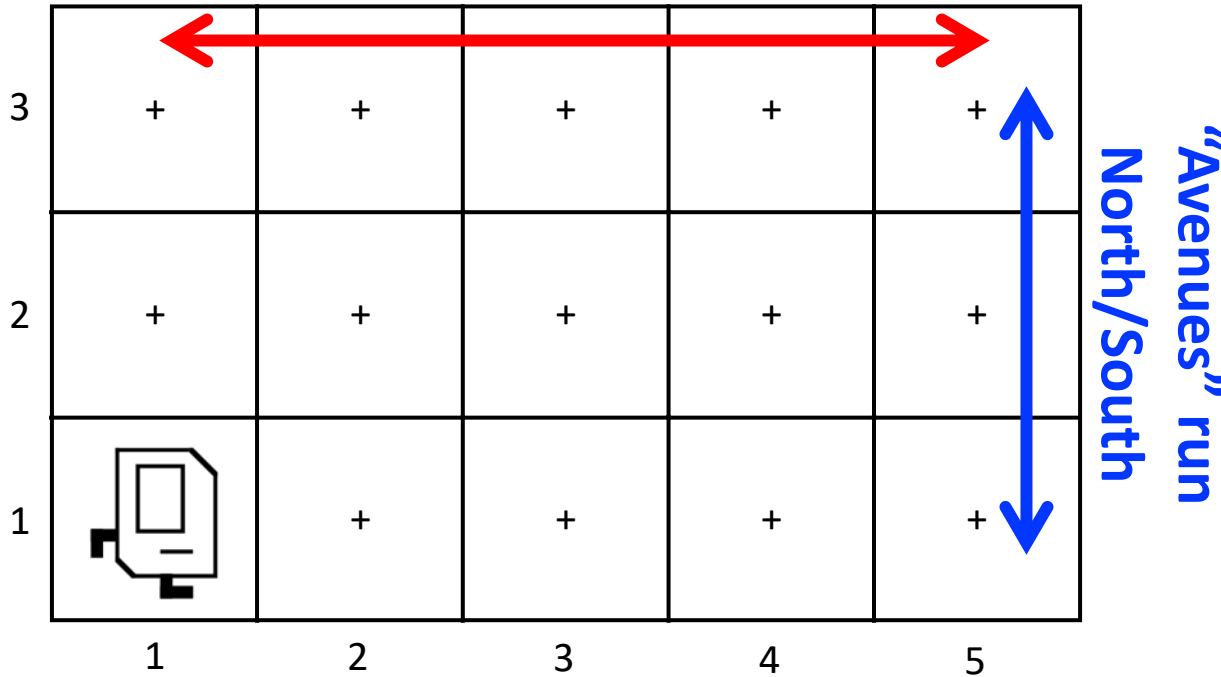


- Reading: Should read the “Karel Reader” on class website
- Handout #3: “Honor Code”
- Handout #4: “Using Karel with PyCharm”
  - Tells you how to get started with writing Karel programs
- Handout #5: “Assignment 1”
  - Set of Karel programs for you to write
  - Due 12:15pm on Friday, April 8th
- Only use features of Karel in the course reader
  - No other features of Python may be used in Karel programs!

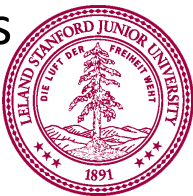


# Recall, Karel's World

“Streets” run East/West



- Grid, where “corner” is intersection of each street/avenue
- Karel is currently on corner (1, 1)
- If Karel moved forward, Karel would be on corner (2, 1)
- Karel’s beeper bag can have 0, 1, or more (up to infinite) beepers



# First Lesson in Programming Style

```
from karel.stanfordkarel import *
```

```
"""
```

```
File: StepUpKarel.py
```

```
-----
```

```
Karel program, where Karel picks up a beeper,  
jumps up on a step and drops the beeper off.
```

```
"""
```

Multi-line  
*comment*

```
def main():  
    move()  
    pick_beeper()  
    move()  
    turn_left()  
    move()  
    turn_right()  
    move()  
    put_beeper()  
    move()
```

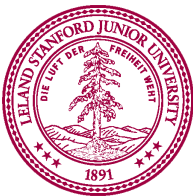
SOFTWARE ENGINEERING PRINCIPLE:  
Aim to make programs readable by *humans*

One line  
*comment*

```
# Karel turns to the right
```

```
def turn_right():  
    turn_left()  
    turn_left()  
    turn_left()
```

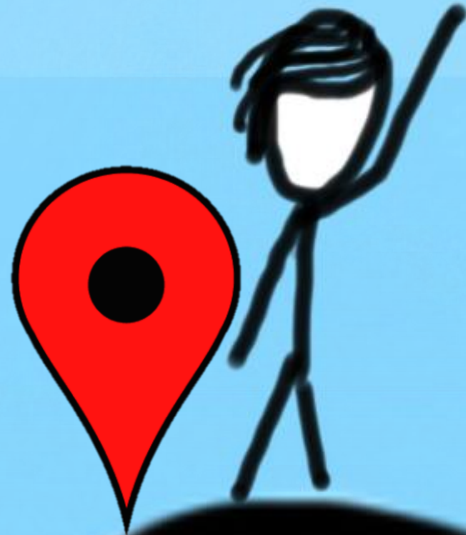
Descriptive  
*names*  
(snake\_case)



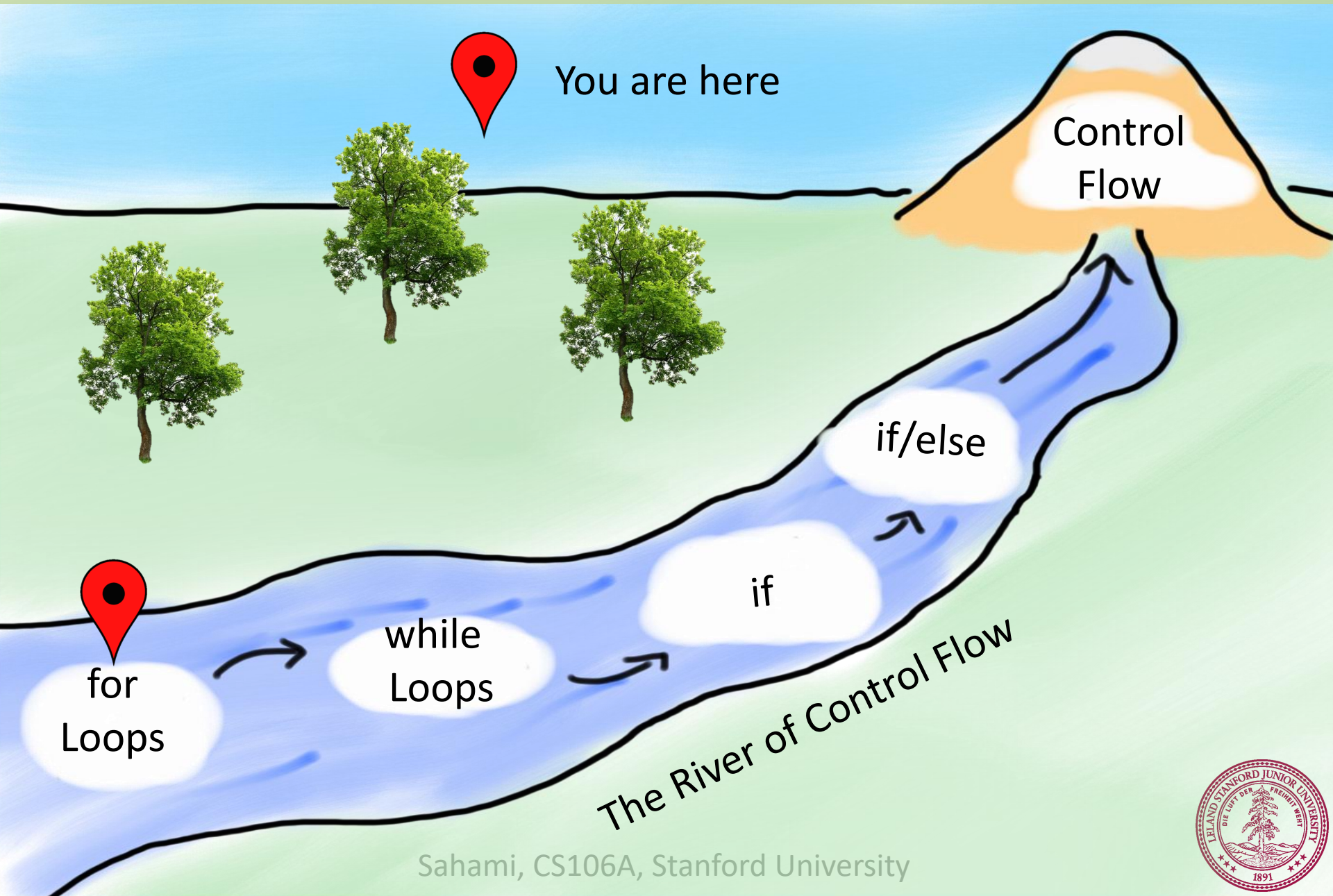


# Today's Goal

1. Code using loops and conditions
2. Trace programs that use loops and conditions




# Today's Route



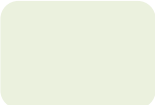

# for loop

```
for i in range(count):  
    statements           # note indenting
```



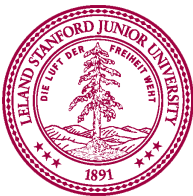
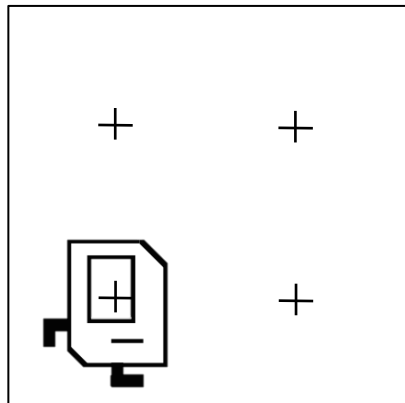
---

```
def turn_right():  
    for i in range(3):  
        turn_left()     # note indenting
```



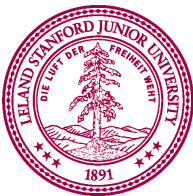
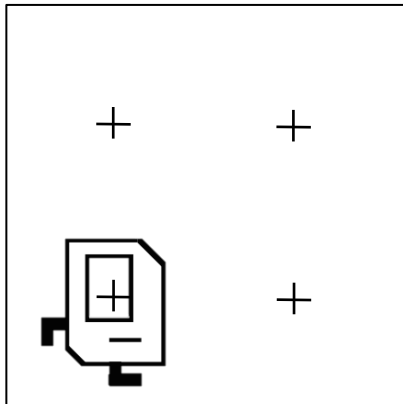
# Place Beeper Square

```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```



# Place Beeper Square

```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```



# Place Beeper Square

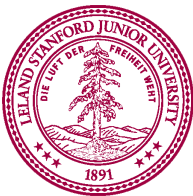
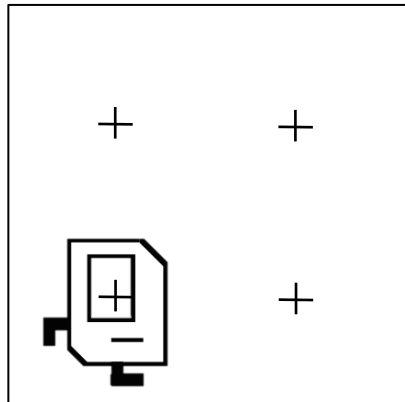
```
def main():
```

```
    for i in range(4):
```

```
        put_beeper()
```

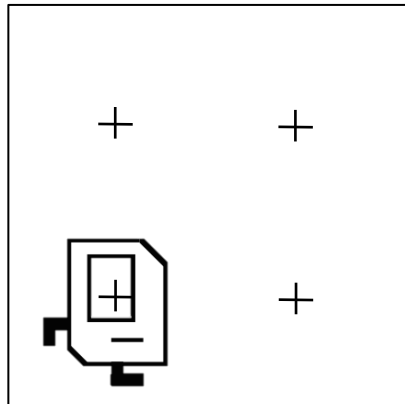
```
        move()
```

```
        turn_left()
```

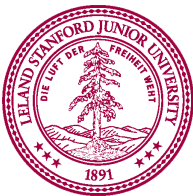


# Place Beeper Square

```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```

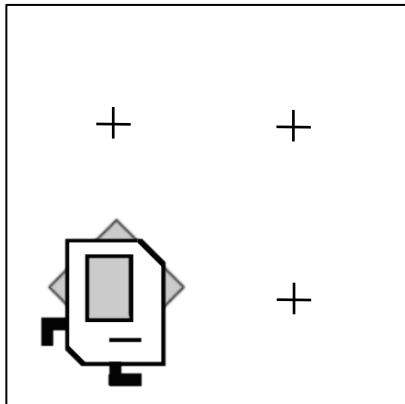


First time  
through the  
loop

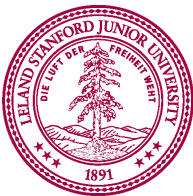


# Place Beeper Square

```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```



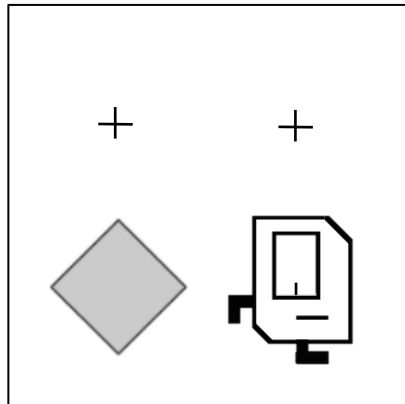
First time  
through the  
loop



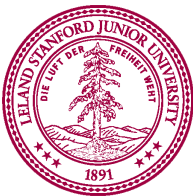


# Place Beeper Square

```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```

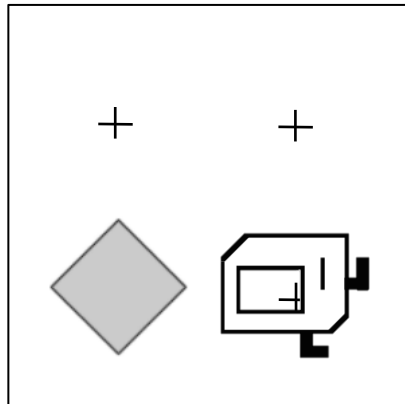


First time  
through the  
loop

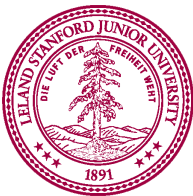


# Place Beeper Square

```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```

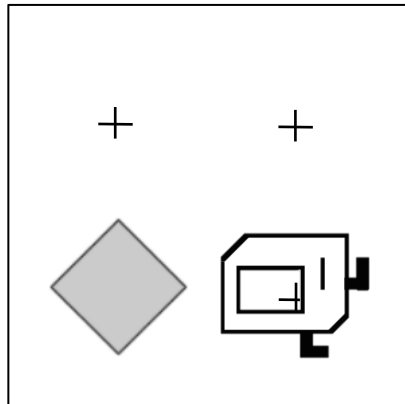


First time  
through the  
loop

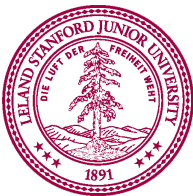


# Place Beeper Square

```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```

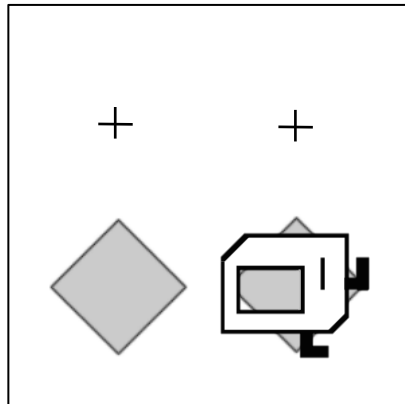


Second time  
through the  
loop



# Place Beeper Square

```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```

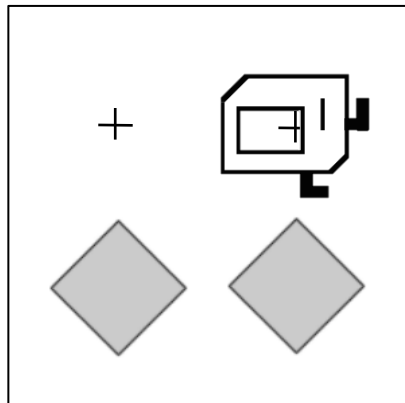


Second time  
through the  
loop



# Place Beeper Square

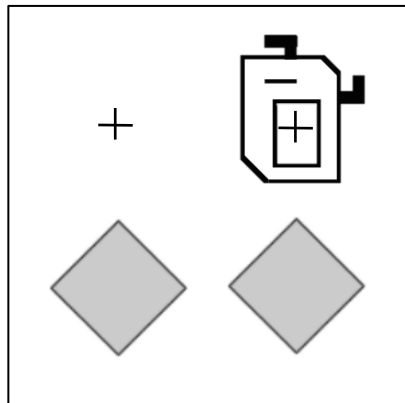
```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```



Second time  
through the  
loop

# Place Beeper Square

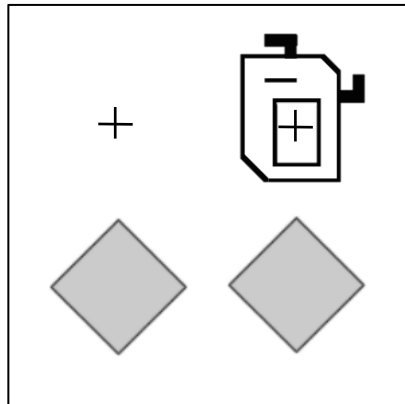
```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```



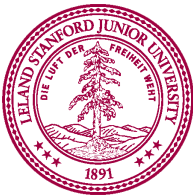
Second time  
through the  
loop

# Place Beeper Square

```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```

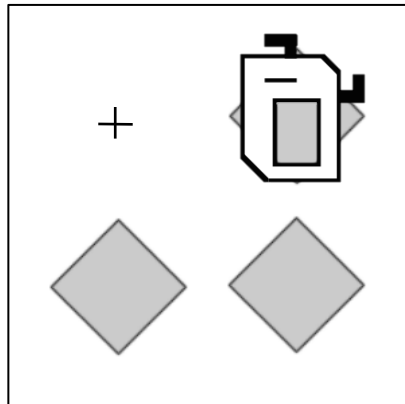


Third time  
through the  
loop

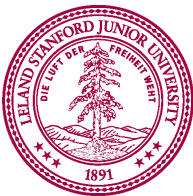


# Place Beeper Square

```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```



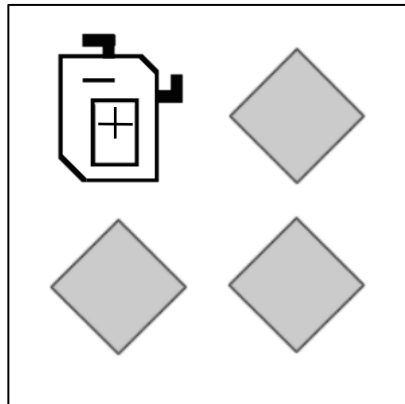
Third time  
through the  
loop



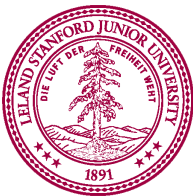


# Place Beeper Square

```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```

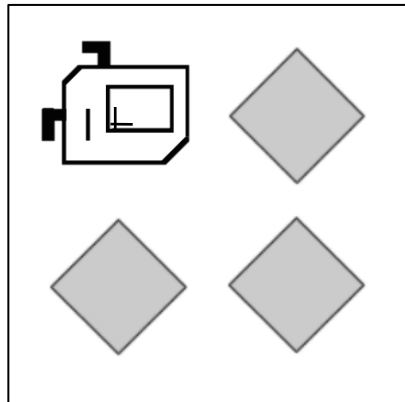


Third time  
through the  
loop

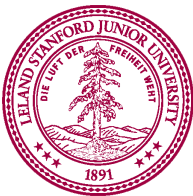


# Place Beeper Square

```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```

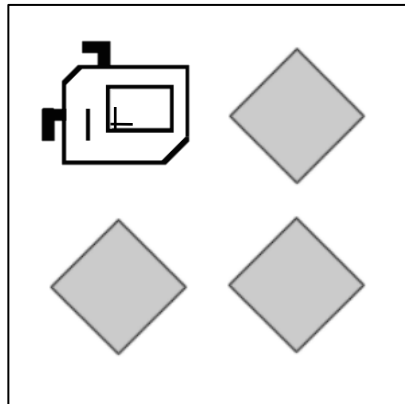


Third time  
through the  
loop



# Place Beeper Square

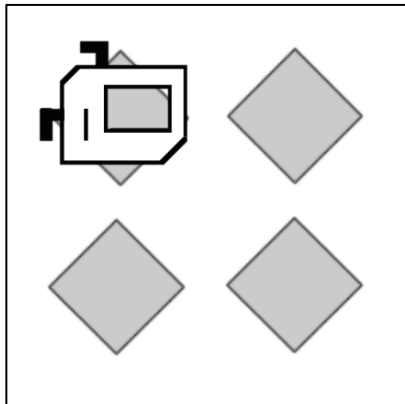
```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```



**Fourth** time  
through the  
loop

# Place Beeper Square

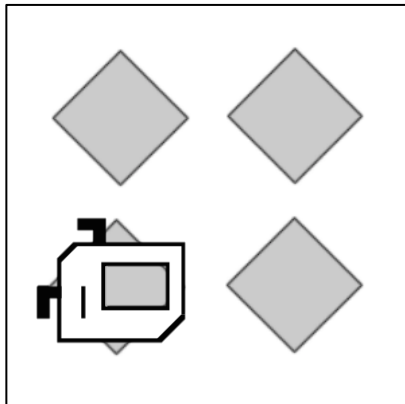
```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```



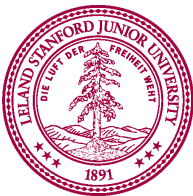
Fourth time  
through the  
loop

# Place Beeper Square

```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```

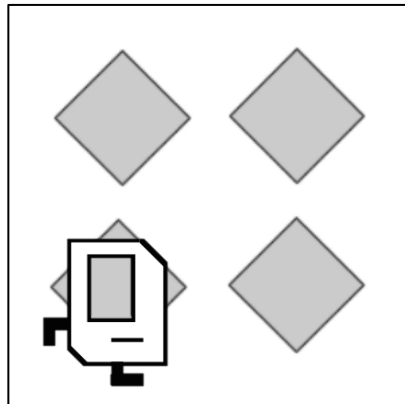


Fourth time through the loop



# Place Beeper Square

```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```



Fourth time  
through the  
loop

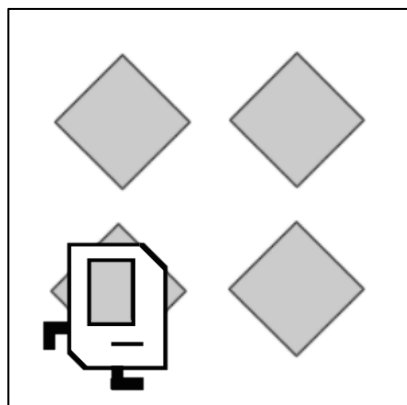


# Place Beeper Square

```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```



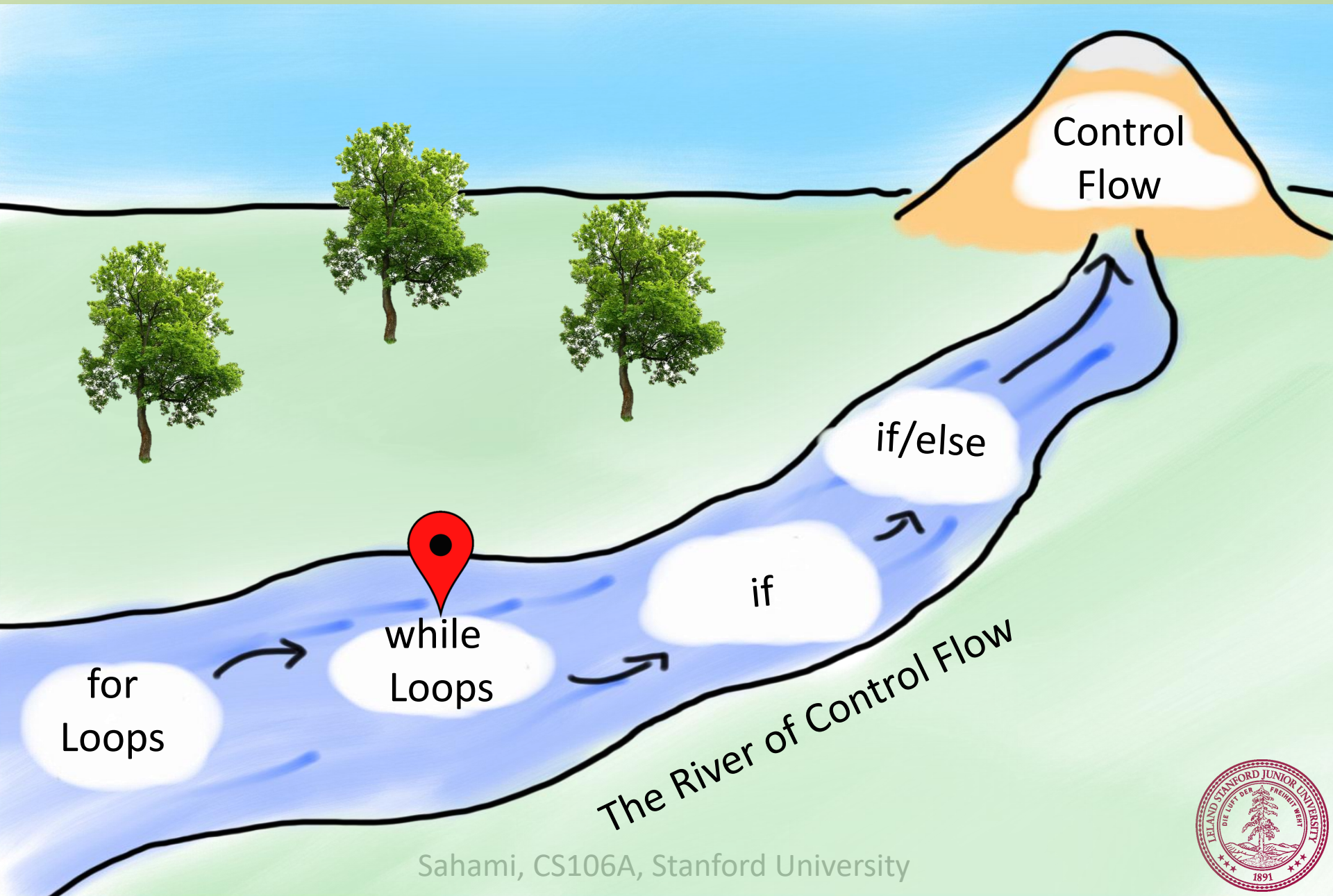
You often want the **postcondition** of a loop to match the **precondition**



Done!



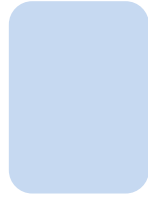
# Today's Route





# while loop

```
while condition:
```



```
    statements
```

```
    # note indenting
```

---

```
def move_to_wall():
```



```
    while front_is_clear():
```



```
        move()
```

```
        # note indenting
```

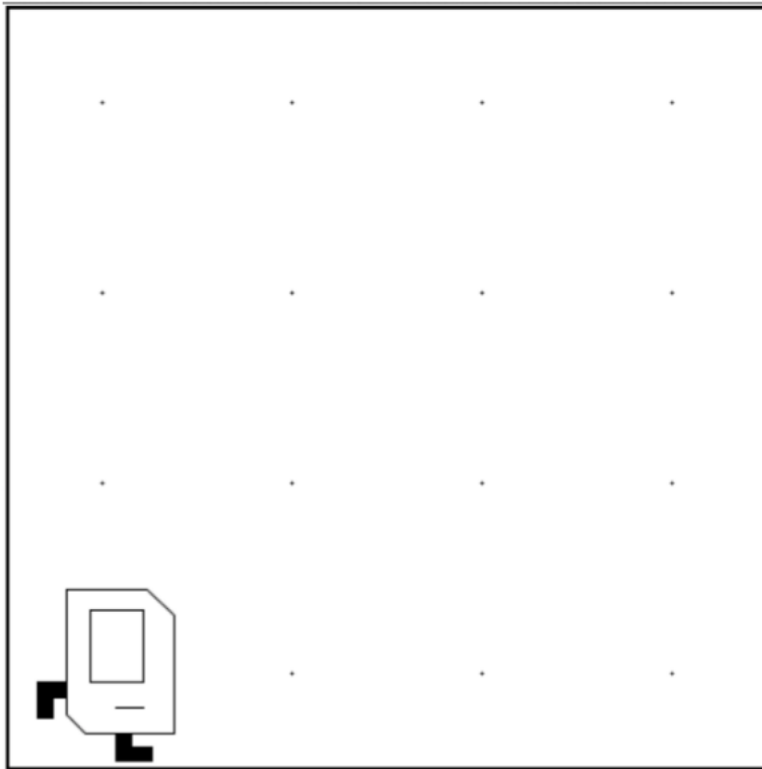
# Conditions Karel Can Check For

<i>Test</i>	<i>Opposite</i>	<i>What it checks</i>
<code>front_is_clear()</code>	<code>front_is_blocked()</code>	Is there a wall in front of Karel?
<code>left_is_clear()</code>	<code>left_is_blocked()</code>	Is there a wall to Karel's left?
<code>right_is_clear()</code>	<code>right_is_blocked()</code>	Is there a wall to Karel's right?
<code>beepers_present()</code>	<code>no_beeper_present()</code>	Are there beepers on this corner?
<code>beepers_in_bag()</code>	<code>no_beeper_in_bag()</code>	Any there beepers in Karel's bag?
<code>facing_north()</code>	<code>not_facing_north()</code>	Is Karel facing north?
<code>facing_east()</code>	<code>not_facing_east()</code>	Is Karel facing east?
<code>facing_south()</code>	<code>not_facing_south()</code>	Is Karel facing south?
<code>facing_west()</code>	<code>not_facing_west()</code>	Is Karel facing west?

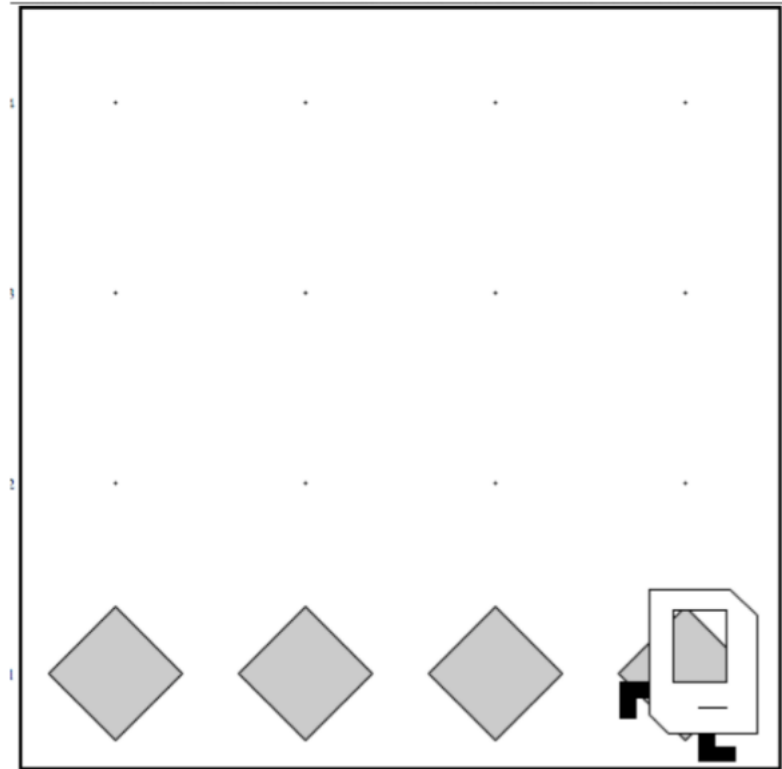
This is in Chapter 10 of the online Karel course reader

# Task: Place Beeper Line

Before



After



# Place Beeper Line

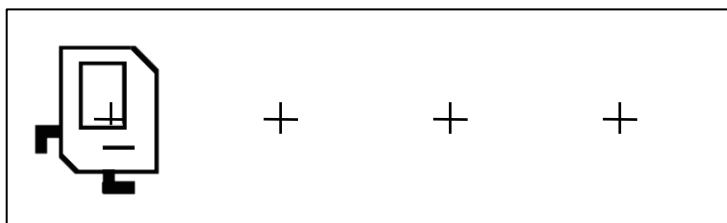
```
def main():  
    while front_is_clear():  
        put_beeper()  
        move()
```



# Place Beeper Line

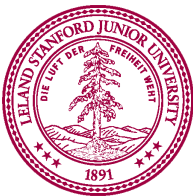
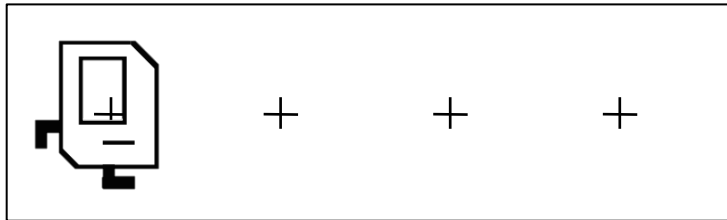
```
def main():
```

```
    while front_is_clear():  
        put_beeper()  
        move()
```



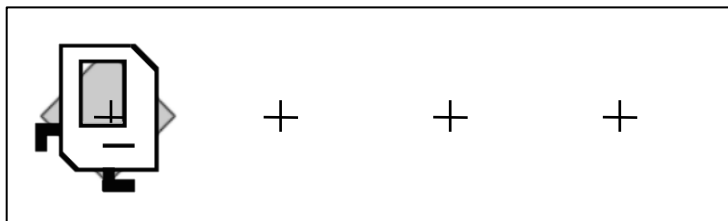
# Place Beeper Line

```
def main():  
    while front_is_clear():  
        put_beeper()  
        move()
```



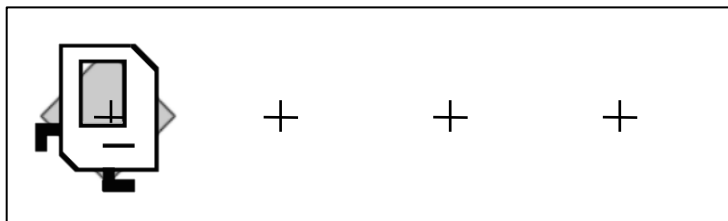
# Place Beeper Line

```
def main():  
    while front_is_clear():  
        put_beeper()  
        move()
```



# Place Beeper Line

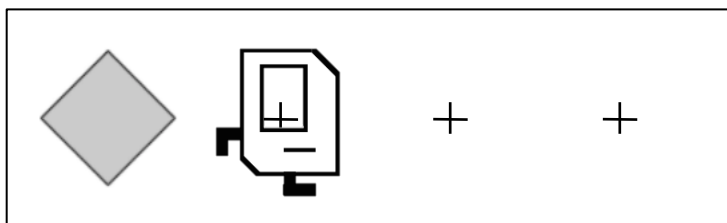
```
def main():  
    while front_is_clear():  
        put_beeper()  
        move()
```





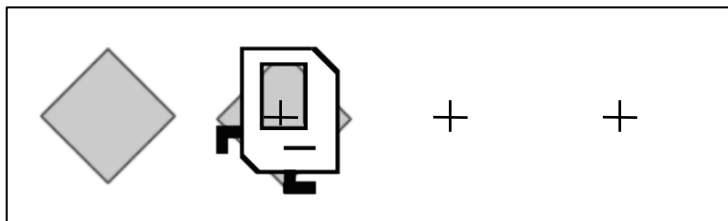
# Place Beeper Line

```
def main():  
    while front_is_clear():  
        put_beeper()  
        move()
```



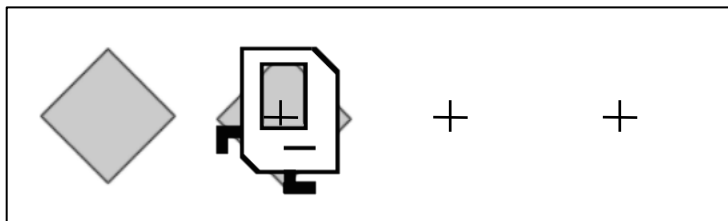
# Place Beeper Line

```
def main():  
    while front_is_clear():  
        put_beeper()  
        move()
```



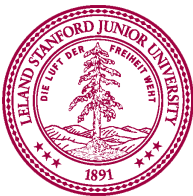
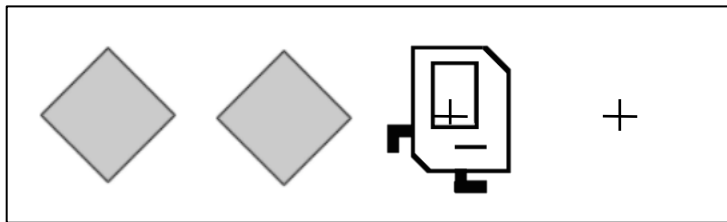
# Place Beeper Line

```
def main():  
    while front_is_clear():  
        put_beeper()  
        move()
```



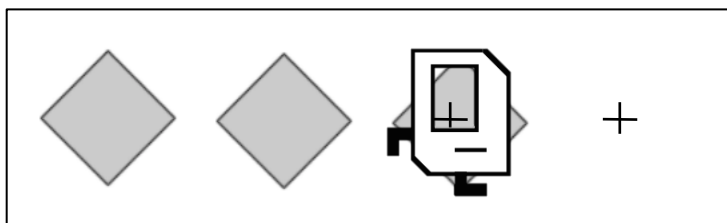
# Place Beeper Line

```
def main():  
    while front_is_clear():  
        put_beeper()  
        move()
```



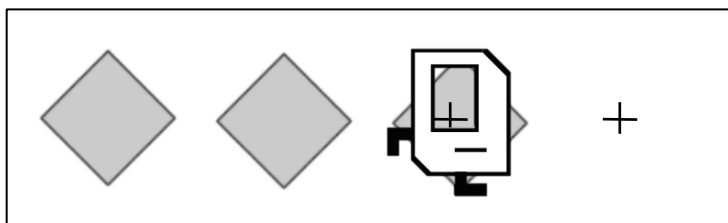
# Place Beeper Line

```
def main():  
    while front_is_clear():  
        put_beeper()  
        move()
```



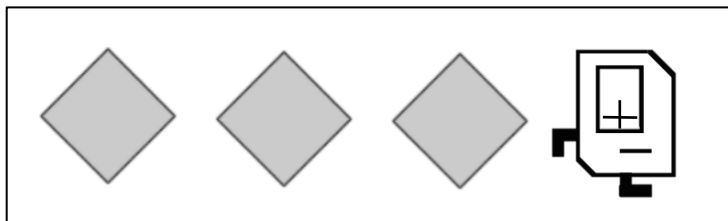
# Place Beeper Line

```
def main():  
    while front_is_clear():  
        put_beeper()  
        move()
```



# Place Beeper Line

```
def main():  
    while front_is_clear():  
        put_beeper()  
        move()
```



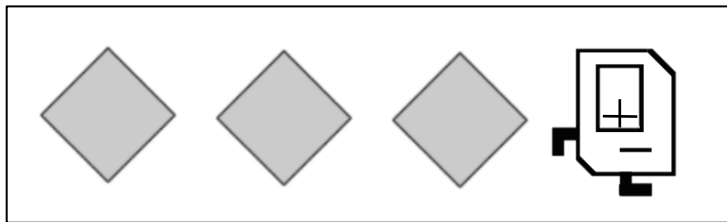
# Place Beeper Line

```
def main():  
    while front_is_clear():  
        put_beeper()  
        move()
```



BUGGY!

Done!

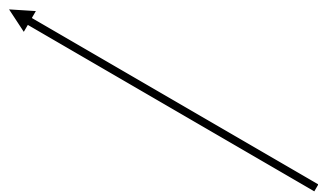




# Place Beeper Line

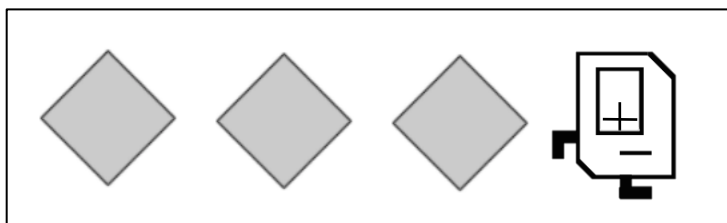


```
def main():  
    while front_is_clear():  
        put_beeper()  
        move()  
    put_beeper()           # add final put_beeper
```



Not in **while** loop

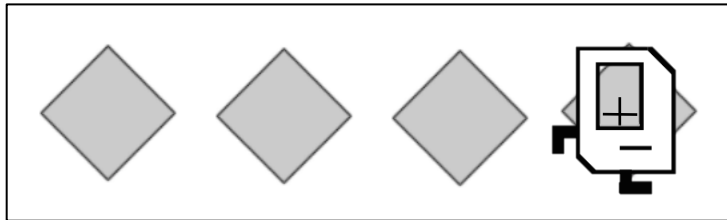
Fixed!



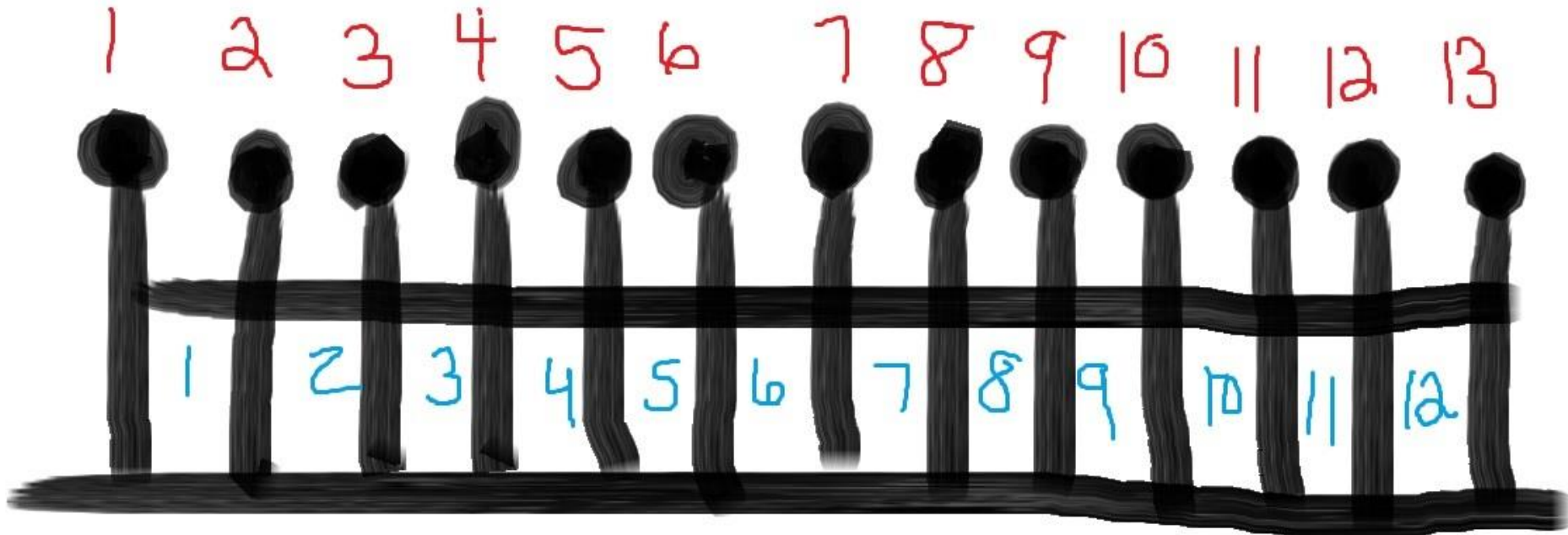
# Place Beeper Line

```
def main():  
    while front_is_clear():  
        put_beeper()  
        move()  
    put_beeper() # add final put_beeper
```

Fixed!



# Fence Post Problem

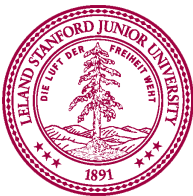


Also sometimes called an “Off By One Bug”



A program executes one line at a time.

The `while` loop checks its condition only at the start of the code block and before repeating.



# Which Loop

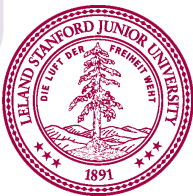
Repeat  
Process

**Know** how  
many times  
(definite loop)

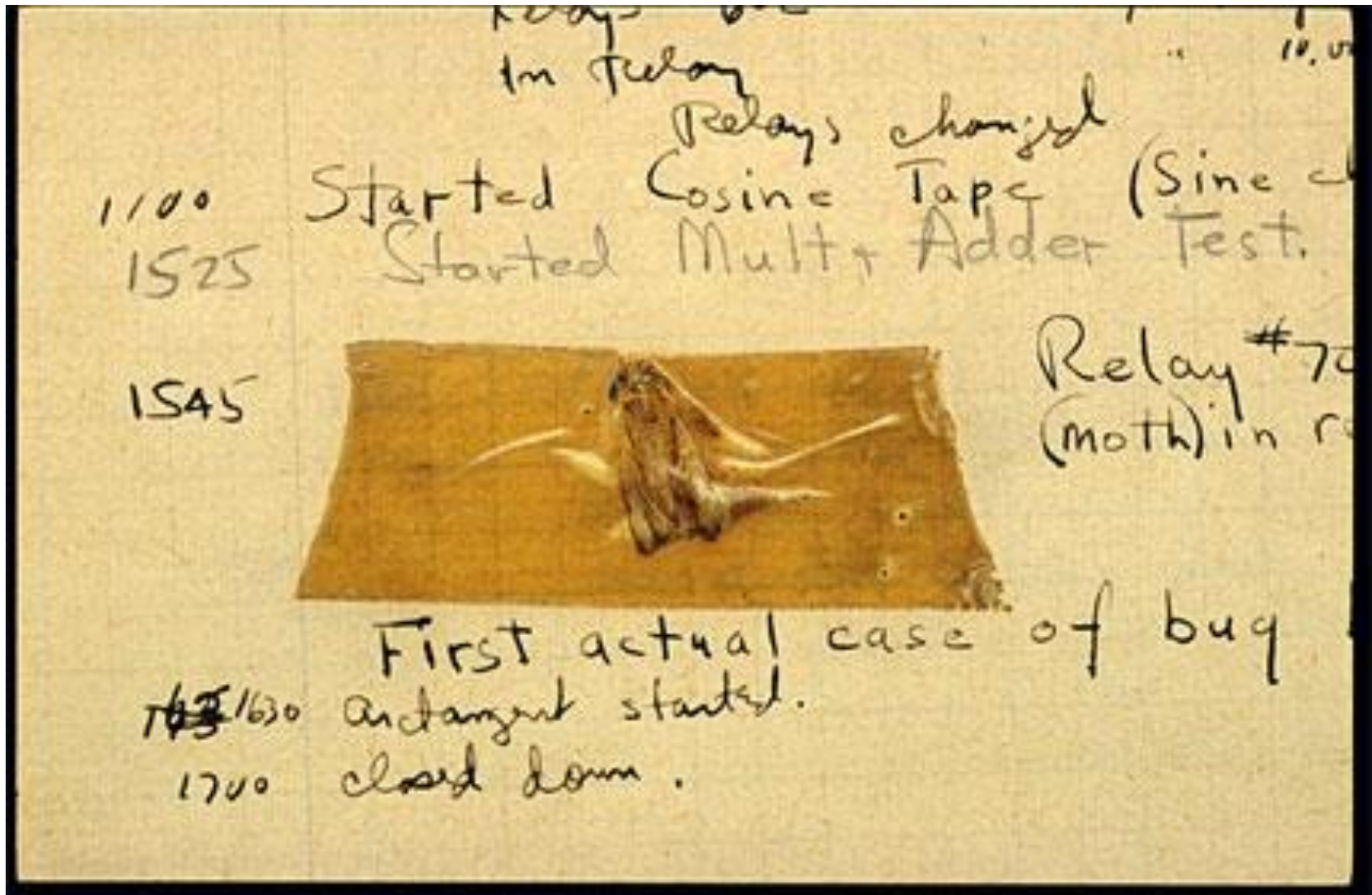
**Don't** know how  
many times  
(indefinite loop)

for Loop

while Loop



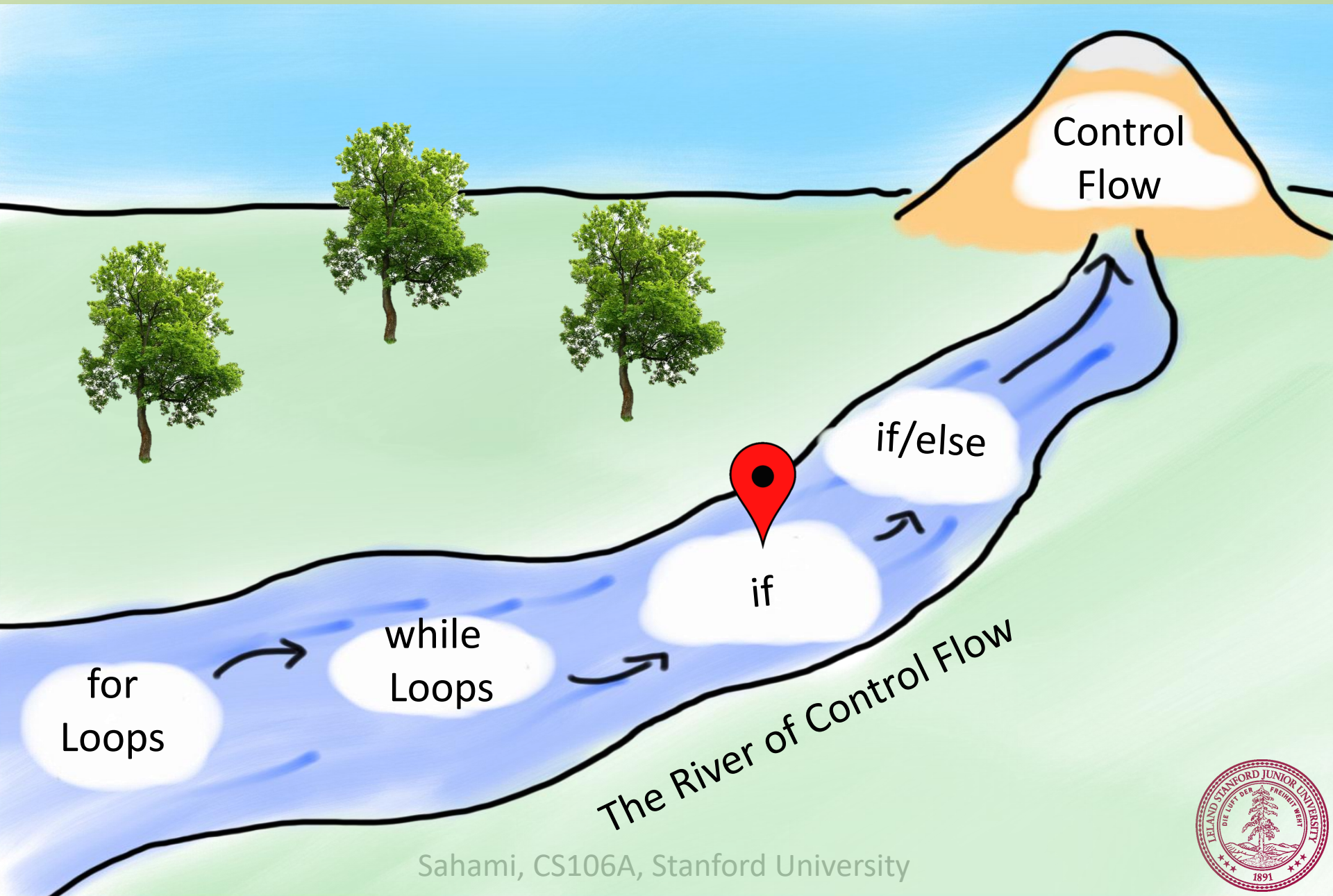
# Actual Bug from Marc II



# Grace Hopper



# Today's Route





# if statement

```
if condition:
```

```
    statements
```

```
    # note indenting
```

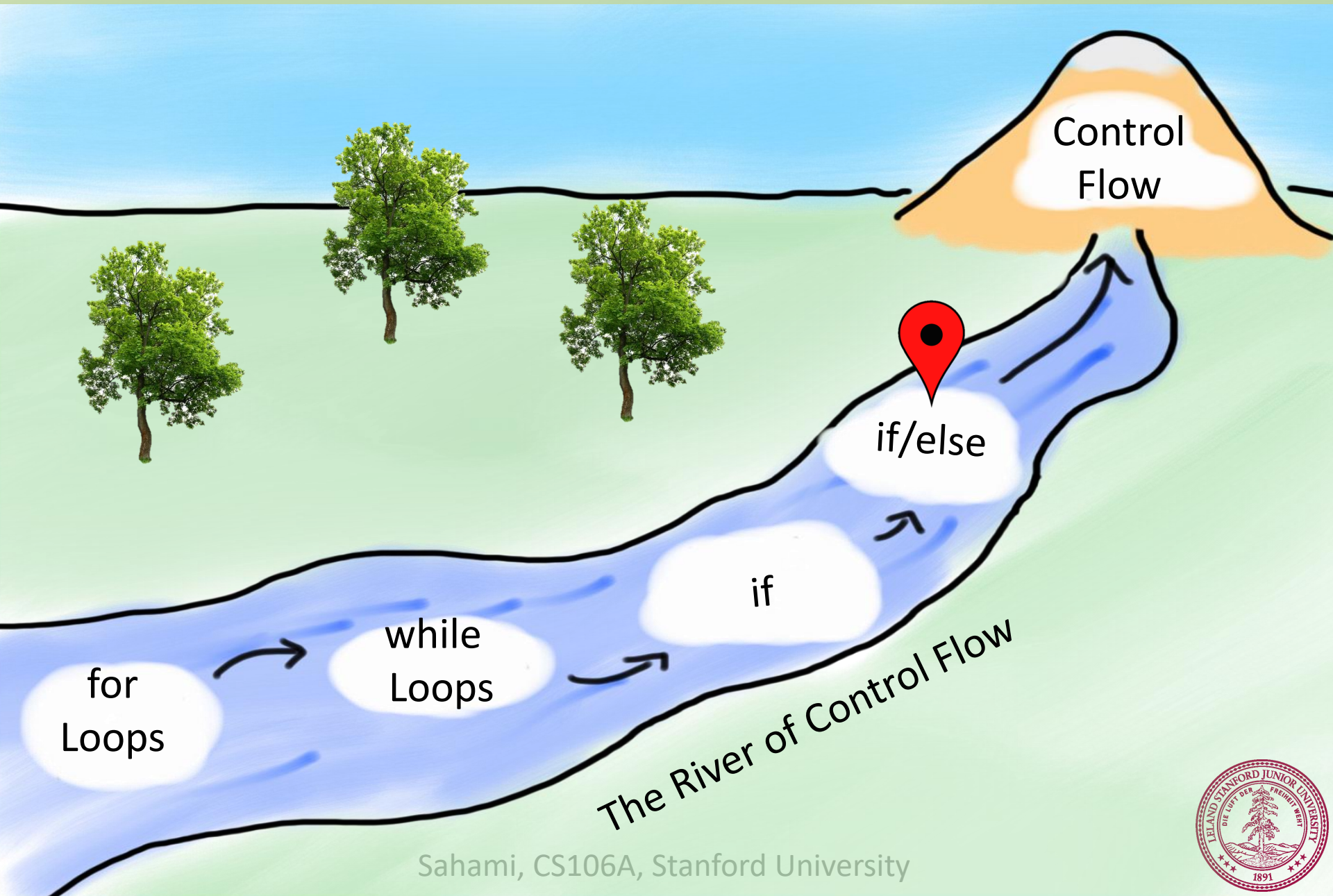
---

```
def safe_pick_up():
```

```
    if beepers_present():
```

```
        pick_beeper() # note indenting
```

# Today's Route



# if-else statement

`if condition:`

`statements                    # note indenting`

`else:`

`statements                    # note indenting`

---

```
def invert_beeper():
```

```
    if beepers_present():
```

```
        pick_beeper() # note indenting
```

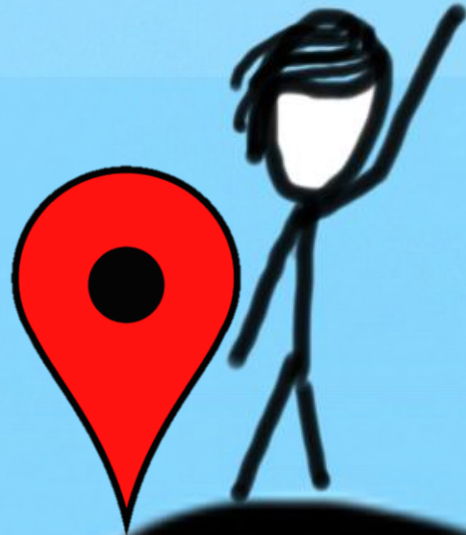
```
    else:
```

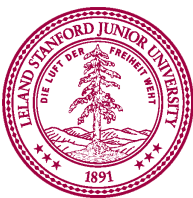
```
        put_beeper() # note indenting
```

You just learned most of  
programming "control flow"

# Today's Goal

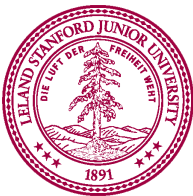
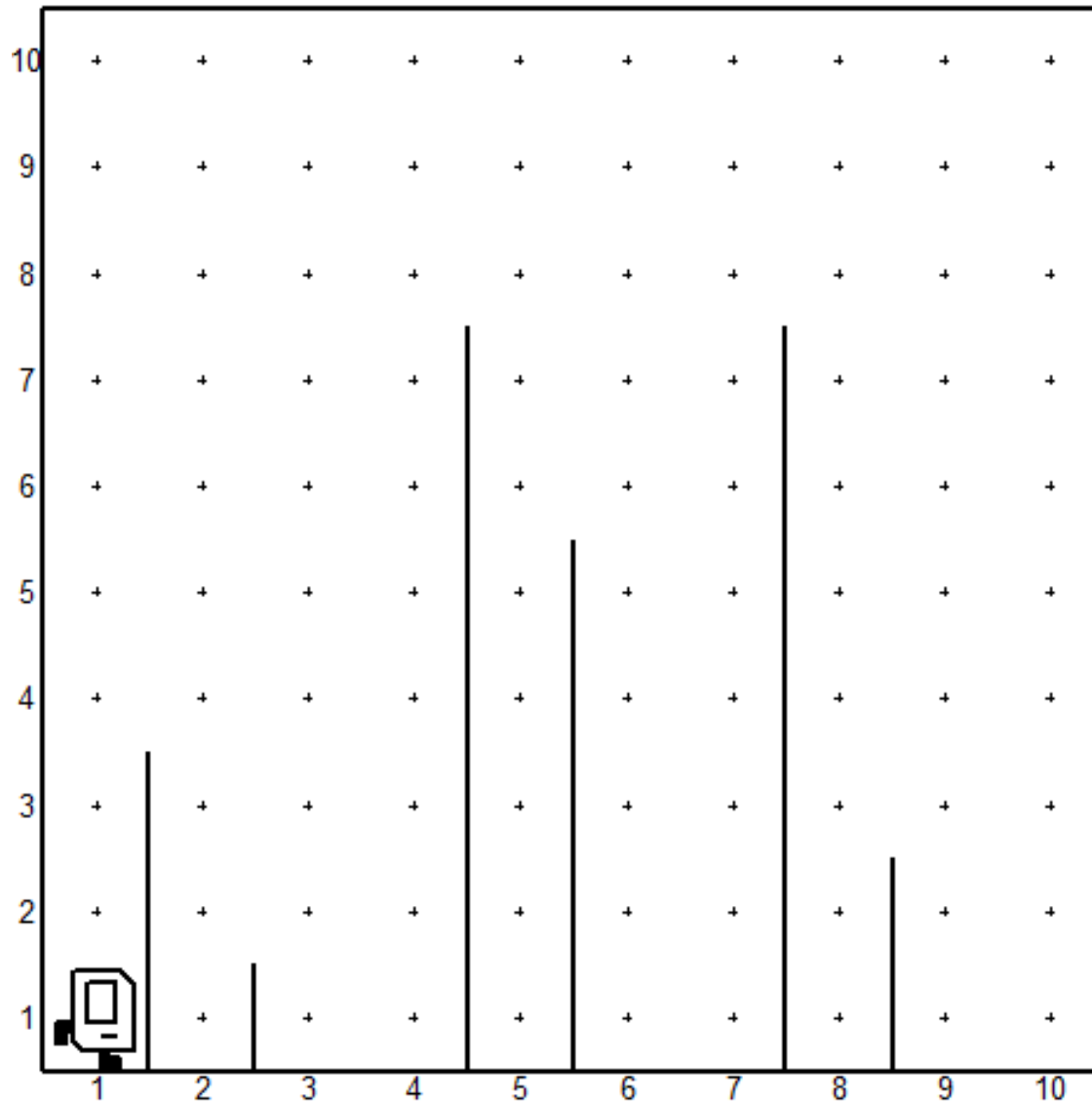
1. Code using loops and conditions
2. Trace programs that use loops and conditions





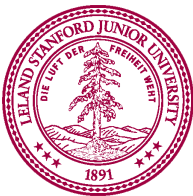
Putting it all together  
SteepChaseKarel.py

# Steeple Chase



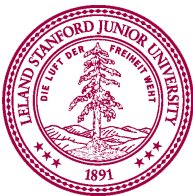
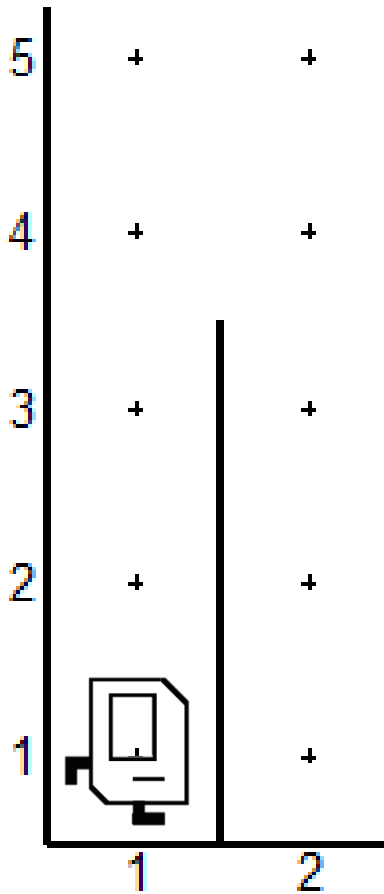


# Focus on One Steeple



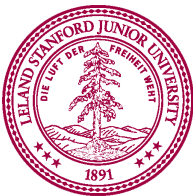
# Focus on One Steeple

`turn_left()`



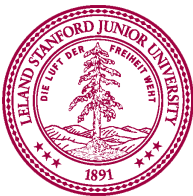
# Focus on One Steeple

`turn_left()`



# Focus on One Steeple

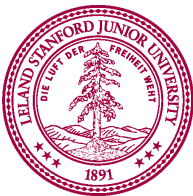
```
turn_left()  
while right_is_blocked():  
    move()
```



# Focus on One Steeple

```
turn_left()
```

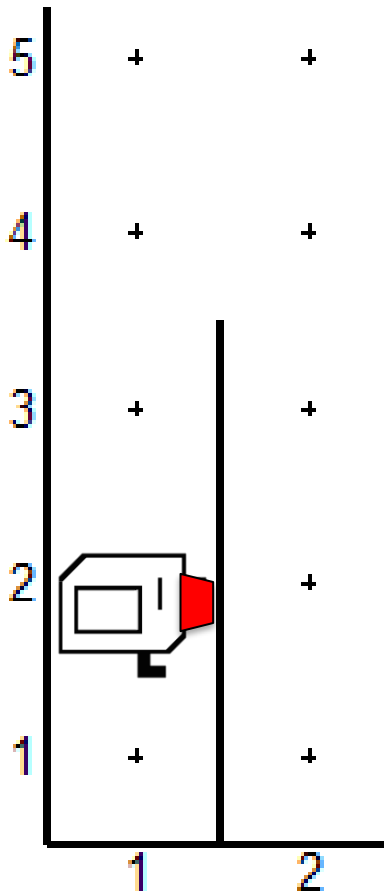
```
while right_is_blocked():  
    move()
```



# Focus on One Steeple

```
turn_left()
```

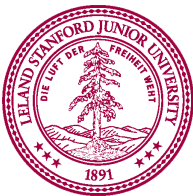
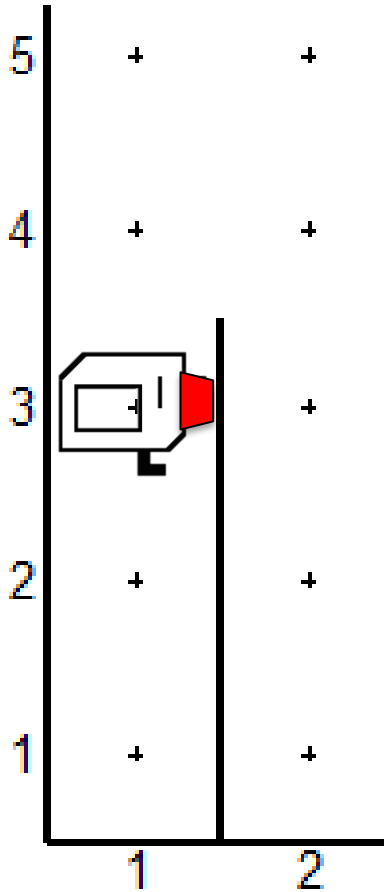
```
while right_is_blocked():  
    move()
```



# Focus on One Steeple

```
turn_left()
```

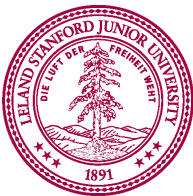
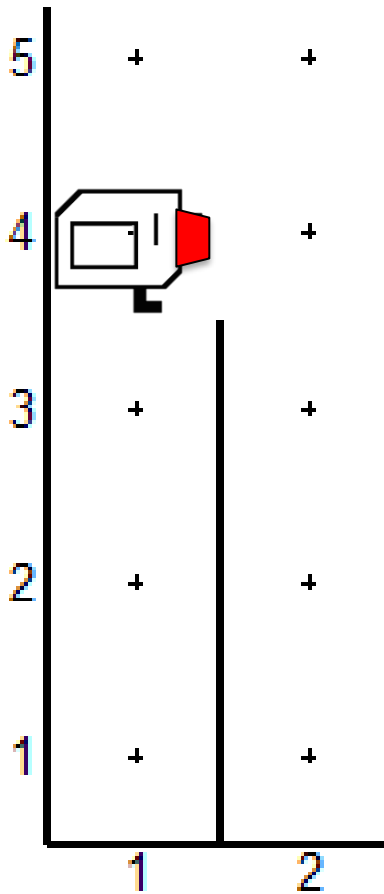
```
while right_is_blocked():  
    move()
```



# Focus on One Steeple

```
turn_left()
```

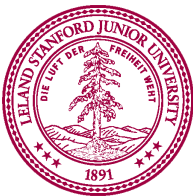
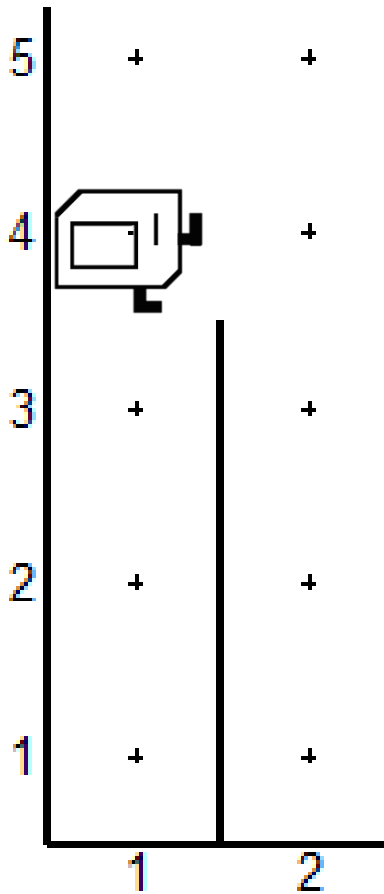
```
while right_is_blocked():  
    move()
```





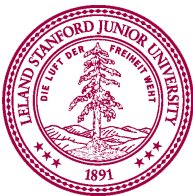
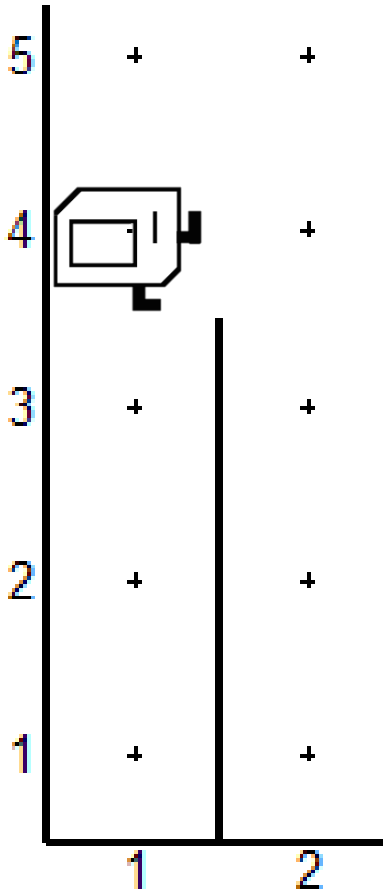
# Focus on One Steeple

```
turn_left()  
while right_is_blocked():  
    move()
```



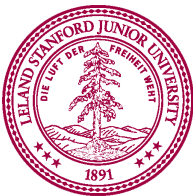
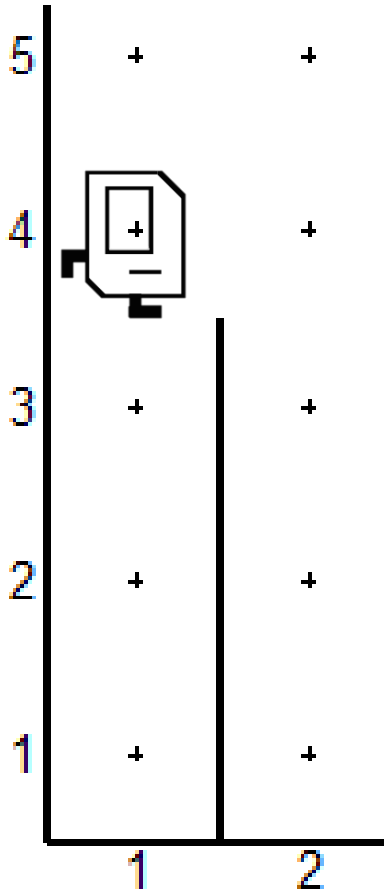
# Focus on One Steeple

```
turn_left()  
while right_is_blocked():  
    move()  
turn_right()
```

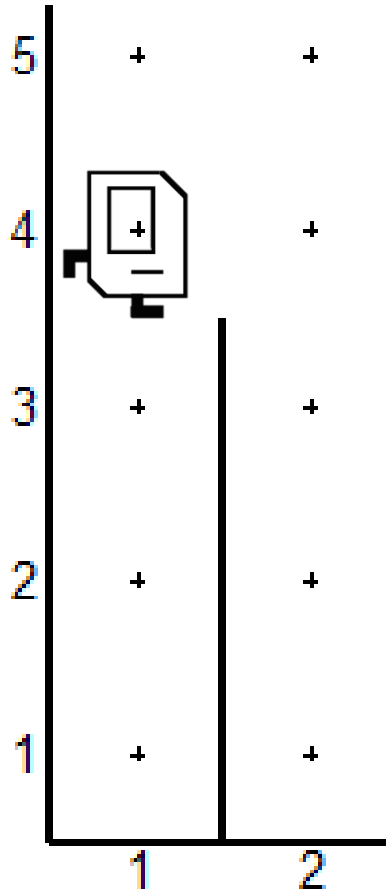


# Focus on One Steeple

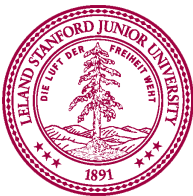
```
turn_left()  
while right_is_blocked():  
    move()  
    turn_right()
```



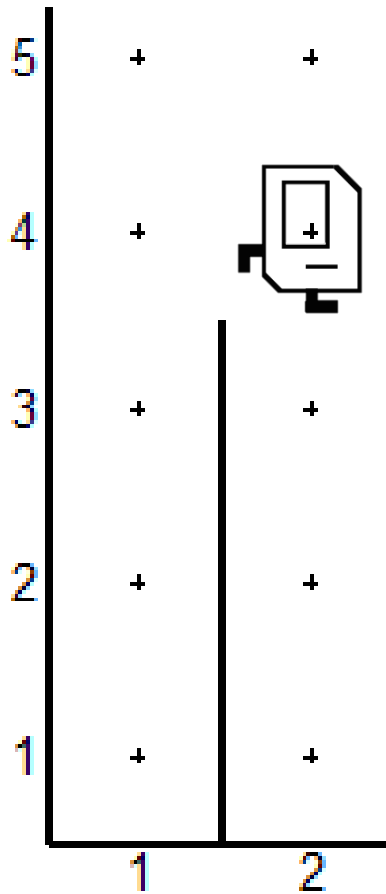
# Focus on One Steeple



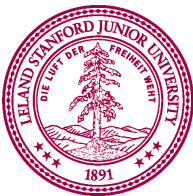
```
turn_left()  
while right_is_blocked():  
    move()  
turn_right()  
move()
```



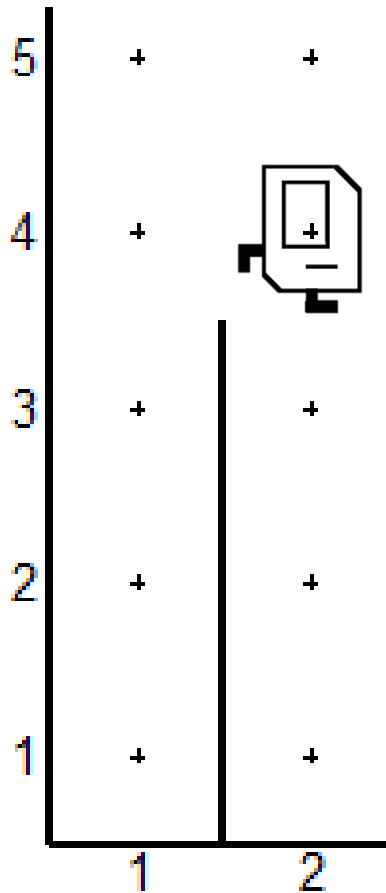
# Focus on One Steeple



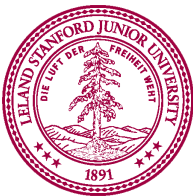
```
turn_left()  
while right_is_blocked():  
    move()  
turn_right()  
move()
```



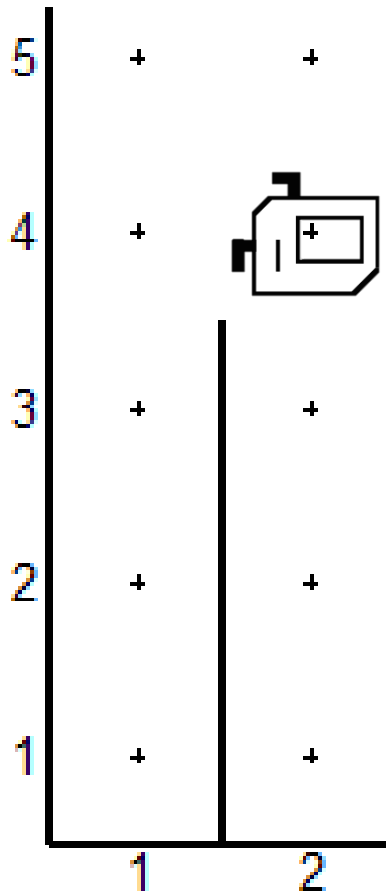
# Focus on One Steeple



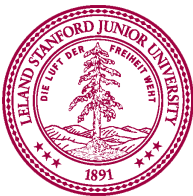
```
turn_left()  
while right_is_blocked():  
    move()  
turn_right()  
move()  
turn_right()
```



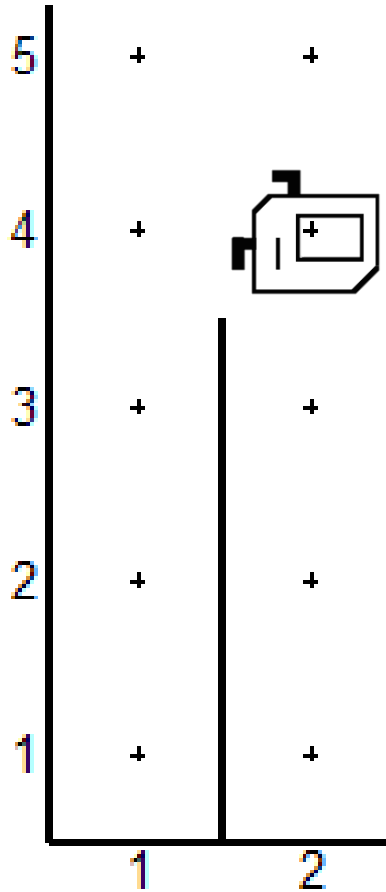
# Focus on One Steeple



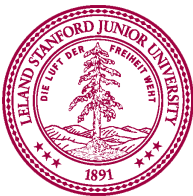
```
turn_left()  
while right_is_blocked():  
    move()  
turn_right()  
move()  
turn_right()
```



# Focus on One Steeple

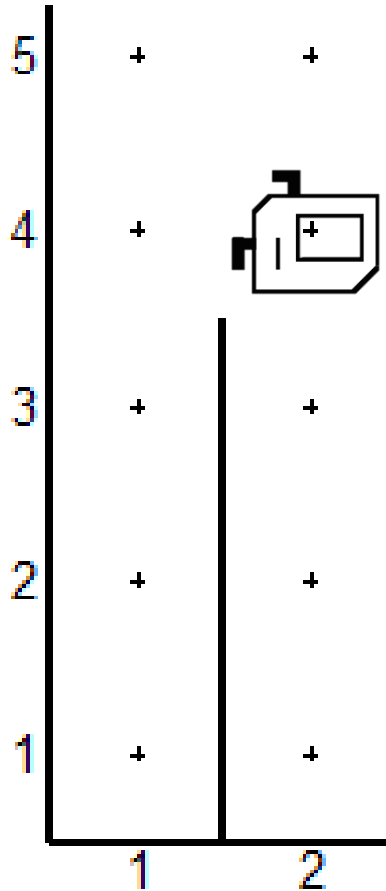


```
turn_left()  
while right_is_blocked():  
    move()  
turn_right()  
move()  
turn_right()  
move_to_wall()
```



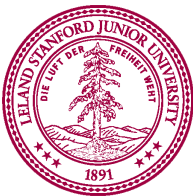


# Focus on One Steeple

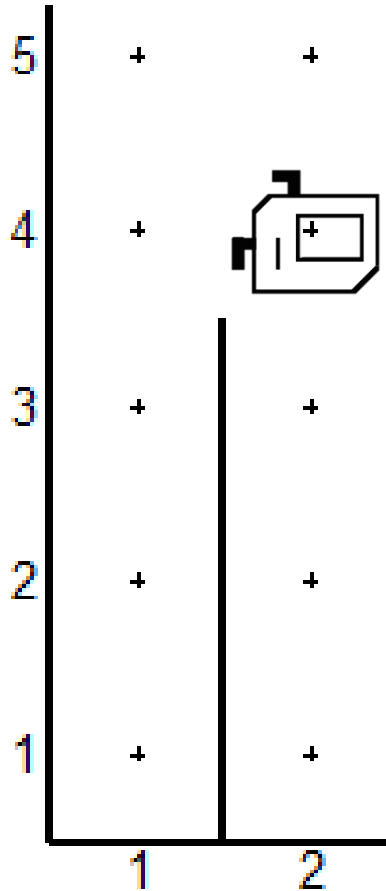


```
turn_left()
while right_is_blocked():
    move()
turn_right()
move()
turn_right()
move_to_wall()
```

```
def move_to_wall():
    while front_is_clear():
        move()
```

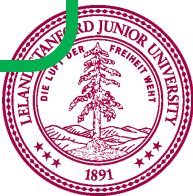


# Focus on One Steeple

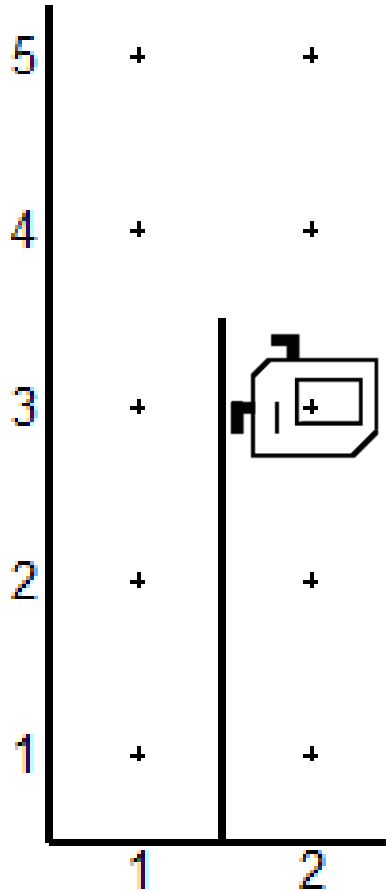


```
turn_left()  
while right_is_blocked():  
    move()  
turn_right()  
move()  
turn_right()  
move_to_wall()
```

```
def move_to_wall():  
    while front_is_clear():  
        move()
```

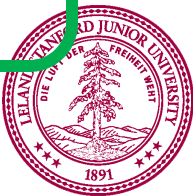


# Focus on One Steeple

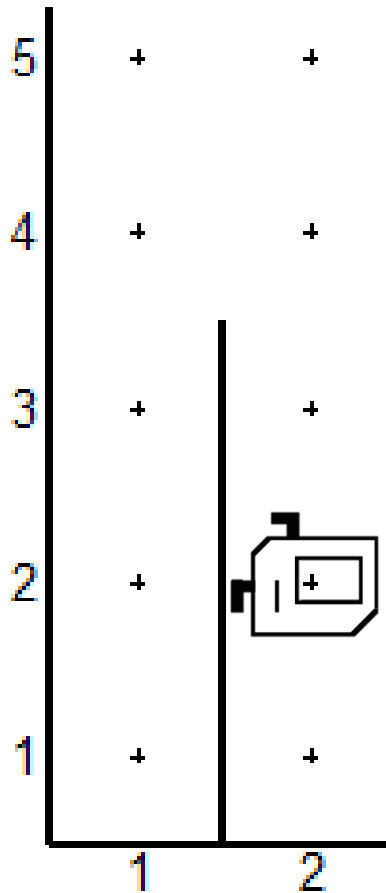


```
turn_left()
while right_is_blocked():
    move()
turn_right()
move()
turn_right()
move_to_wall()
```

```
def move_to_wall():
    while front_is_clear():
        move()
```

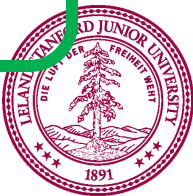


# Focus on One Steeple

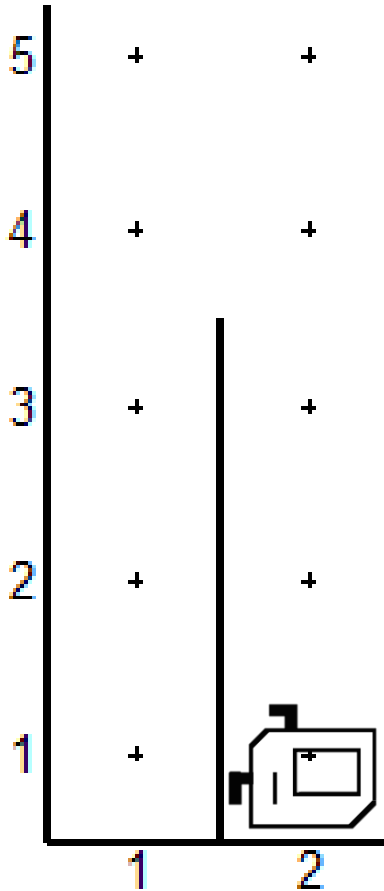


```
turn_left()  
while right_is_blocked():  
    move()  
turn_right()  
move()  
turn_right()  
move_to_wall()
```

```
def move_to_wall():  
    while front_is_clear():  
        move()
```

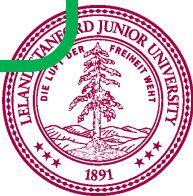


# Focus on One Steeple



```
turn_left()  
while right_is_blocked():  
    move()  
turn_right()  
move()  
turn_right()  
move_to_wall()
```

```
def move_to_wall():  
    while front_is_clear():  
        move()
```



# Focus on One Steeple



```
turn_left()
while right_is_blocked():
    move()
turn_right()
move()
turn_right()
move_to_wall()
```

```
def move_to_wall():
    while front_is_clear():
        move()
```

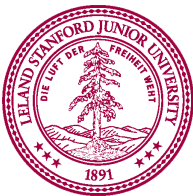


# Focus on One Steeple



```
turn_left()
while right_is_blocked():
    move()
turn_right()
move()
turn_right()
move_to_wall()
turn_left()
```

```
def move_to_wall():
    while front_is_clear():
        move()
```

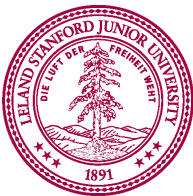


# Focus on One Steeple



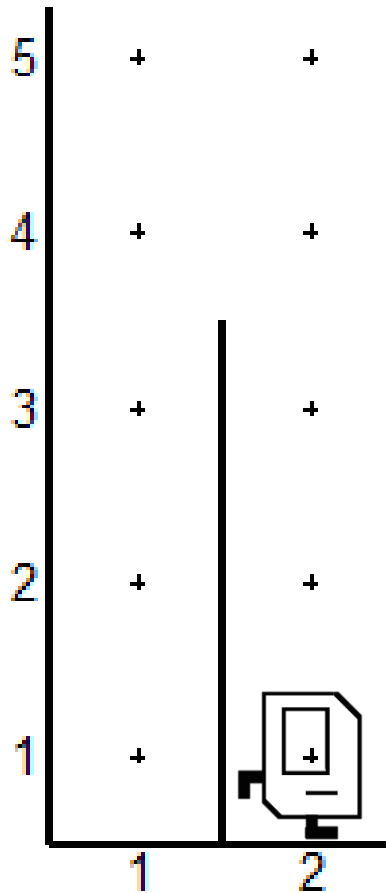
```
turn_left()  
while right_is_blocked():  
    move()  
turn_right()  
move()  
turn_right()  
move_to_wall()  
turn_left()
```

```
def move_to_wall():  
    while front_is_clear():  
        move()
```





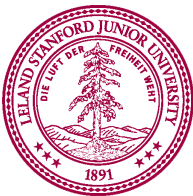
# Focus on One Steeple



```
turn_left()  
while right_is_blocked():  
    move()  
turn_right()  
move()  
turn_right()  
move_to_wall()  
turn_left()
```

You want the **postcondition** of a loop to match the **precondition**

```
def move_to_wall():  
    while front_is_clear():  
        move()
```



# Focus on One Steeple

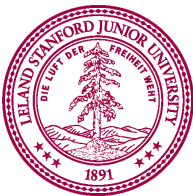


```
turn_left()  
while right_is_blocked():  
    move()  
turn_right()  
move()
```

```
turn_right()  
move_to_wall()  
turn_left()
```

`ascend_hurdle()`

`descend_hurdle()`



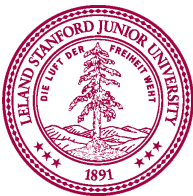
# Focus on One Steeple

```
turn_left()  
while right_is_blocked():  
    move()  
turn_right()  
move()
```

```
turn_right()  
move_to_wall()  
turn_left()
```

`ascend_hurdle()`

`descend_hurdle()`



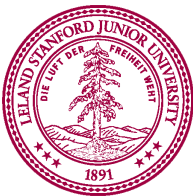
# Focus on One Steeple

```
def ascend_hurdle():  
    turn_left()  
    while right_is_blocked():  
        move()  
    turn_right()
```

```
ascend_hurdle()  
move()
```

```
turn_right()  
move_to_wall()  
turn_left()
```

```
descend_hurdle()
```

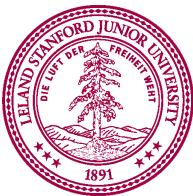


# Focus on One Steeple

```
def ascend_hurdle():
    turn_left()
    while right_is_blocked():
        move()
    turn_right()

def descend_hurdle():
    turn_right()
    move_to_wall()
    turn_left()
```

`ascend_hurdle()`  
`move()`  
`descend_hurdle()`



# Focus on One Steeple

```
def ascend_hurdle():  
    turn_left()  
    while right_is_blocked():  
        move()  
    turn_right()
```

```
def descend_hurdle():  
    turn_right()  
    move_to_wall()  
    turn_left()
```

```
def jump_hurdle():  
    ascend_hurdle()  
    move()  
    descend_hurdle()
```



A Whole Program:  
SteepChaseKarel.py