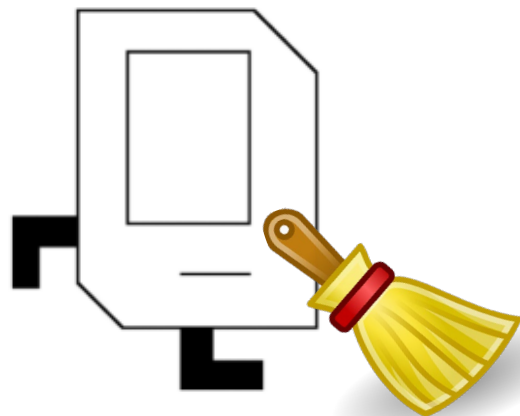


Control Flow

Chris Gregg and Mehran Sahami
CS106A, Stanford University

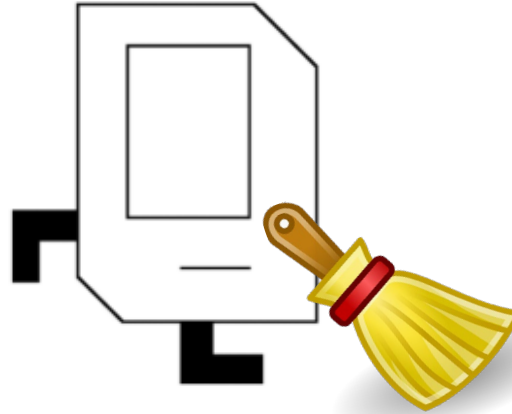
Housekeeping I



- 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., Apr. 3 at 5pm; end Sun., Apr. 6 at 5pm
- Assignment #0 still open (over 150 responses already)
 - About 65% taking class for “Fun and Enlightenment”
 - About 65% have 0-10 hours coding experience
- Assignment #1 released, due April 11 at 11:59pm



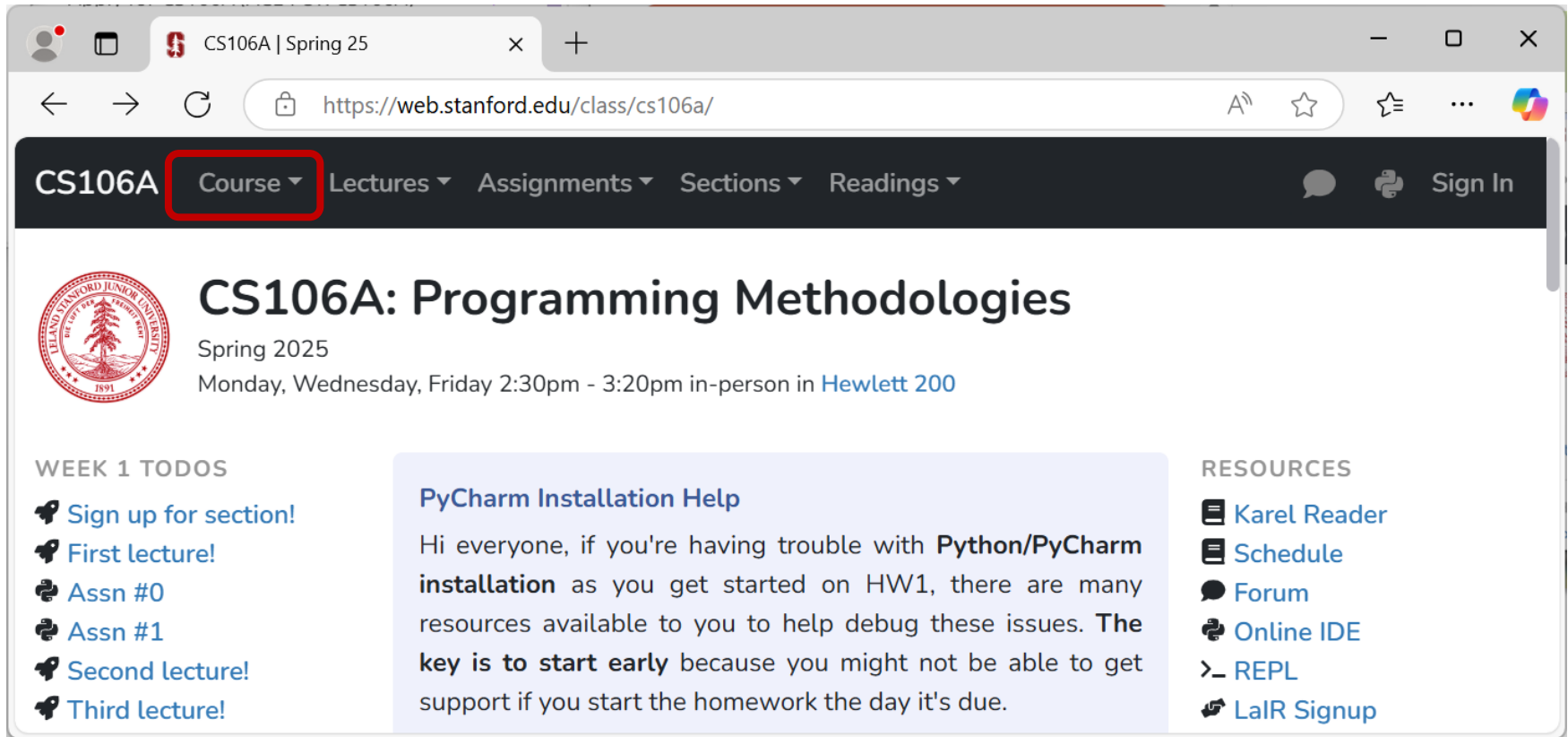
Housekeeping II



- Please send OAE letters to Ngoc (Head TA)
- Application open for CS100A (link on CS106A Ed **Forum**)
 - 1-unit supplementary section for stronger foundation
- “Forum” for questions/discussion
 - Link on top right corner of CS106A class web page
- LaIR Helper Hours start this Sunday (April 6)
 - Located in Durand Building, 3rd floor



Install PyCharm




The screenshot shows a web browser window with the URL <https://web.stanford.edu/class/cs106a/>. The page title is "CS106A: Programming Methodologies" for Spring 2025. The navigation bar includes "Course", "Lectures", "Assignments", "Sections", and "Readings". The "Course" link is highlighted with a red box. The main content area features the Stanford University seal, the course title, and the semester. Below this, there are three sections: "WEEK 1 TODOS" with links to sign up for the section, the first lecture, and assignments; "PyCharm Installation Help" with a message about debugging issues; and "RESOURCES" with links to Karel Reader, Schedule, Forum, Online IDE, REPL, and LaIR Signup.







CS106A | Spring 25

<https://web.stanford.edu/class/cs106a/>

CS106A Course Lectures Assignments Sections Readings Sign In

 **CS106A: Programming Methodologies**
Spring 2025
Monday, Wednesday, Friday 2:30pm - 3:20pm in-person in [Hewlett 200](#)







WEEK 1 TODOS

-  [Sign up for section!](#)
-  [First lecture!](#)
-  [Assn #0](#)
-  [Assn #1](#)
-  [Second lecture!](#)
-  [Third lecture!](#)

PyCharm Installation Help

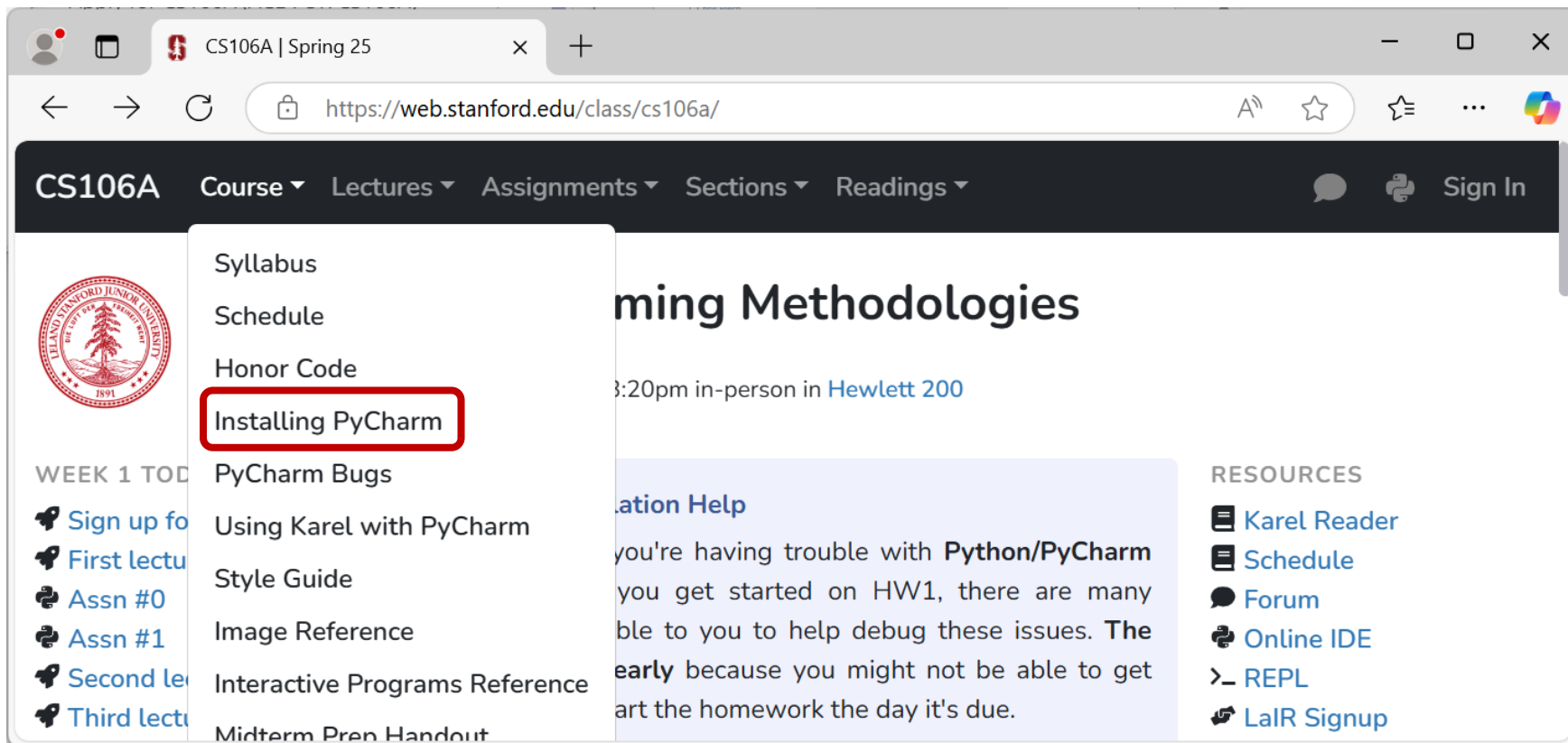
Hi everyone, if you're having trouble with **Python/PyCharm installation** as you get started on HW1, there are many resources available to you to help debug these issues. **The key is to start early** because you might not be able to get support if you start the homework the day it's due.

RESOURCES

-  [Karel Reader](#)
-  [Schedule](#)
-  [Forum](#)
-  [Online IDE](#)
-  [REPL](#)
-  [LaIR Signup](#)



Install PyCharm



The screenshot shows the CS106A course website. The navigation menu is open, and the 'Installing PyCharm' link is highlighted with a red box. The website header includes the course name 'CS106A' and navigation links for 'Course', 'Lectures', 'Assignments', 'Sections', and 'Readings'. The main content area displays 'Timing Methodologies' and a list of resources including 'Karel Reader', 'Schedule', 'Forum', 'Online IDE', 'REPL', and 'LaIR Signup'.

CS106A | Spring 25

<https://web.stanford.edu/class/cs106a/>

CS106A Course ▾ Lectures ▾ Assignments ▾ Sections ▾ Readings ▾

Sign In

Syllabus

Schedule

Honor Code

Installing PyCharm

PyCharm Bugs

Using Karel with PyCharm

Style Guide

Image Reference

Interactive Programs Reference

Midterm Prep Handout

Timing Methodologies

8:20pm in-person in [Hewlett 200](#)

Installation Help

If you're having trouble with **Python/PyCharm** when you get started on HW1, there are many resources available to you to help debug these issues. **The** **early** because you might not be able to get started the homework the day it's due.

RESOURCES

- [Karel Reader](#)
- [Schedule](#)
- [Forum](#)
- [Online IDE](#)
- [REPL](#)
- [LaIR Signup](#)

WEEK 1 TO DO

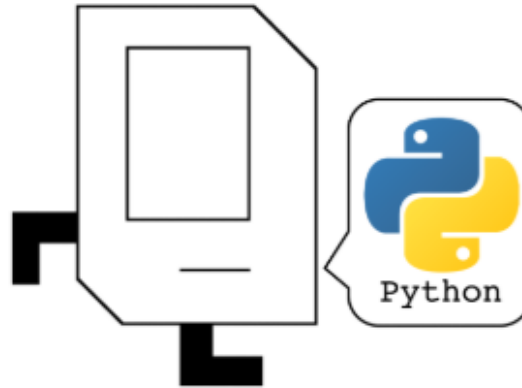
- [Sign up for](#)
- [First lecture](#)
- [Assn #0](#)
- [Assn #1](#)
- [Second lecture](#)
- [Third lecture](#)

Please follow the instructions *closely*!

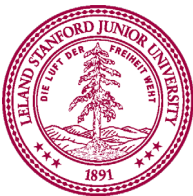
PyCharm installation help session on
Thursday, April 3, 7-8:30pm in CoDa B45



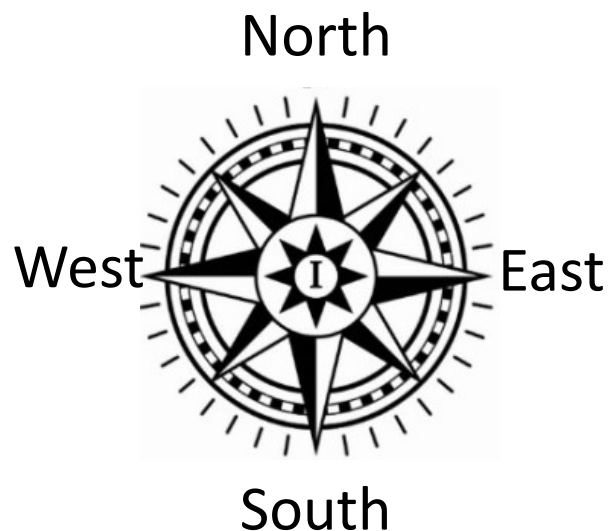
Using Karel and Assignment 1



- Reading: Should read the “Karel Reader” on class website
 - Link on top right corner of CS106A webpage
- Handout: “Honor Code”
- Handout: “Using Karel with PyCharm”
 - Tells you how to get started with writing Karel programs
- Handout: “Assignment 1”
 - Set of Karel programs for you to write
 - Due 11:59pm on Friday, April 11th
- Only use features of Karel in the course reader
 - No other features of Python may be used in Karel programs!



Recall, Karel's World

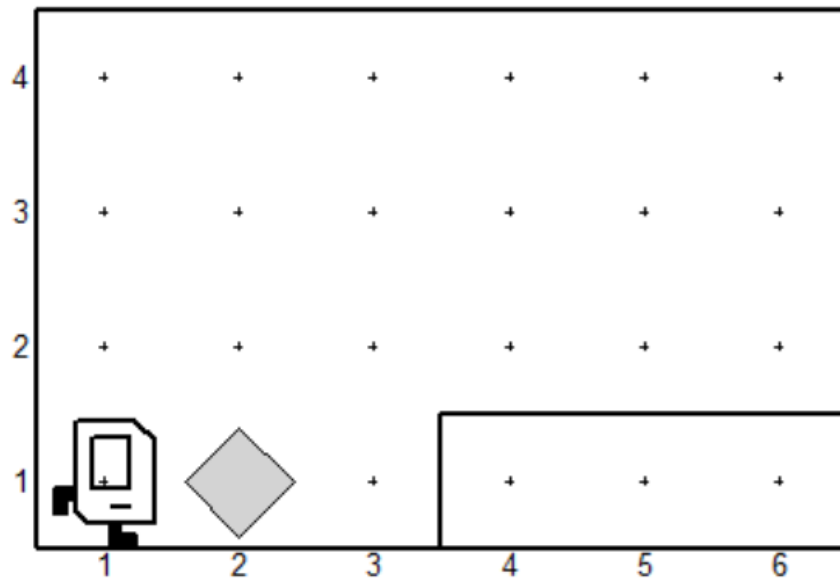


- Grid, where “corner” is intersection of each street/avenue
- Karel is currently on corner (1, 1)
- If Karel moved forward, Karel would be on corner (2, 1)
- Karel’s beeper bag can have 0, 1, or more (up to infinite) beepers

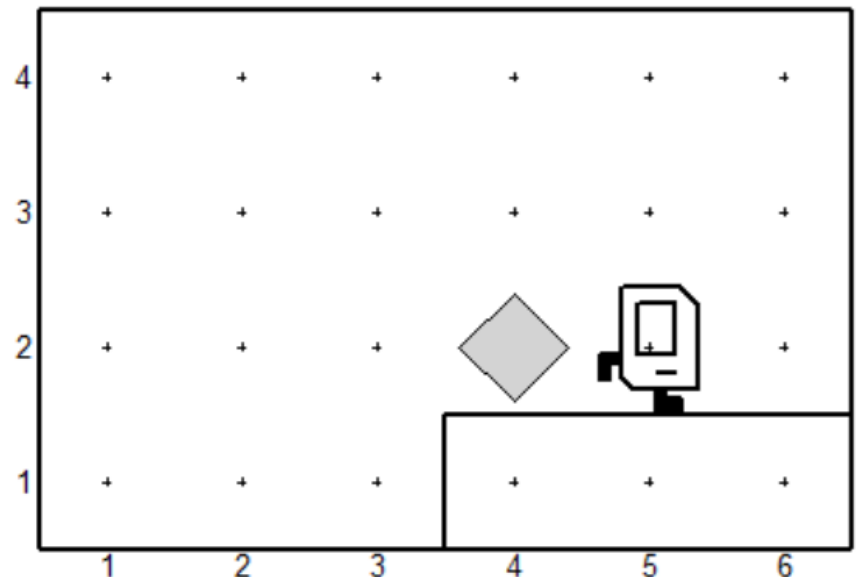
Recall: Step-Up Example

<https://ide.stanford.edu/cs106a/a/stepup>

Starting world:



Ending world:



Anatomy of a Program

Import Packages

Program



Anatomy of a Program

Import Packages

main function

helper functions

start program



Anatomy of a Program

Import Packages

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

helper functions

start program



Anatomy of a Program

Import Packages

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

helper functions

start program



Function Definition

```
def name() :  
    function statements
```

This adds a new
command to Karel's
vocabulary



Anatomy of a Program

Import Packages

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

start program



Anatomy of a Program

Import Packages

```
def main():  
    move()  
    pick_beeper()  
    move()  
    turn_left()  
    move()  
    turn_right()  
    move()  
    put_beeper()  
    move()  
  
def turn_right():  
    turn_left()  
    turn_left()  
    turn_left()  
  
if __name__ == "__main__":  
    run_karel_program()
```



Anatomy of a Program

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

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

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

```
if __name__ == "__main__":  
    run_karel_program()
```



Anatomy of a Program

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

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

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

```
if __name__ == "__main__":  
    run_karel_program()
```



Anatomy of a Program

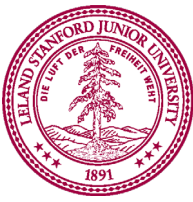
```
from karel.stanfordkarel import *
```

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

This piece of the program's **source code** is called a **function**.

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

```
if __name__ == "__main__":  
    run_karel_program()
```



Anatomy of a Program

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

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

This line of code gives the **name** of the function
(here, the name is: **main**)

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

```
if __name__ == "__main__":  
    run_karel_program()
```



Anatomy of a Program

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

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

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

```
if __name__ == "__main__":  
    run_karel_program()
```

This line of code gives the *name* of the function
(here, the name is: **turn_right**)



Anatomy of a Program

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

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

This is called a **code block**
(Note the indenting)

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

```
if __name__ == "__main__":  
    run_karel_program()
```



Anatomy of a Program

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

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

This is called a **code block**
(Note the indenting)

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

```
if __name__ == "__main__":  
    run_karel_program()
```



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)

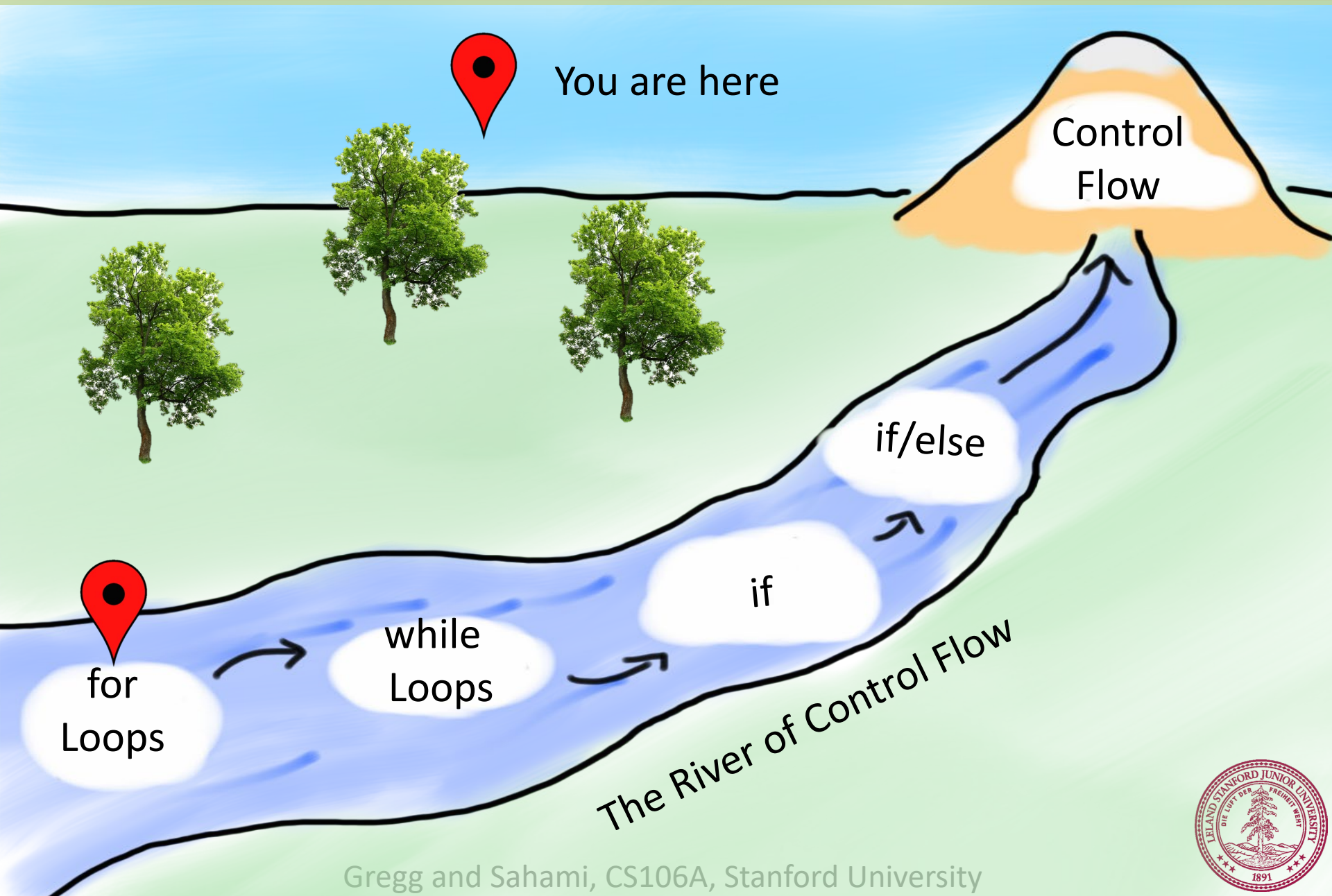


Today's Goal

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
```

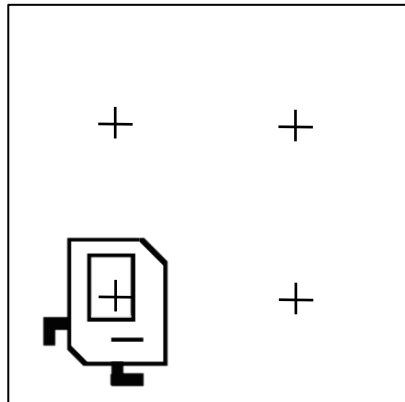


```
def turn_right():  
    for i in range(3):  
        turn_left()          # note indenting
```



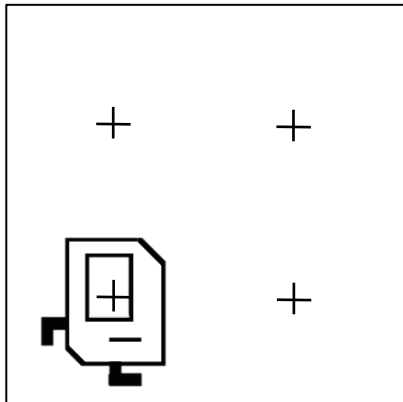
Place Beeper Square

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



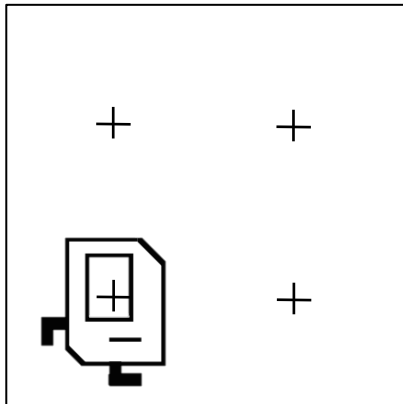
Place Beeper Square

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



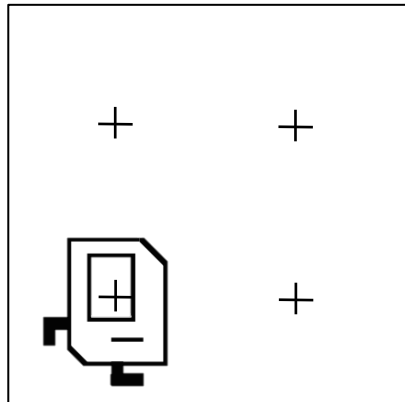
Place Beeper Square

```
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()
```

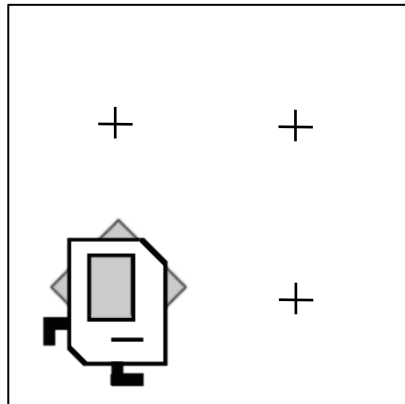


First time
through the
loop



Place Beeper Square

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

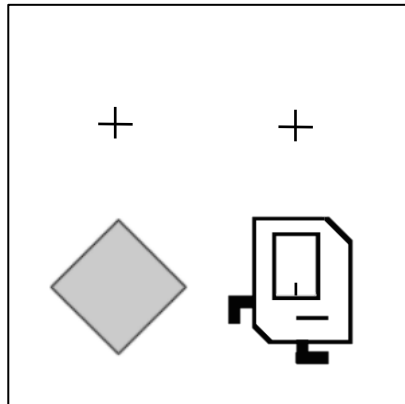


First time
through the
loop



Place Beeper Square

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

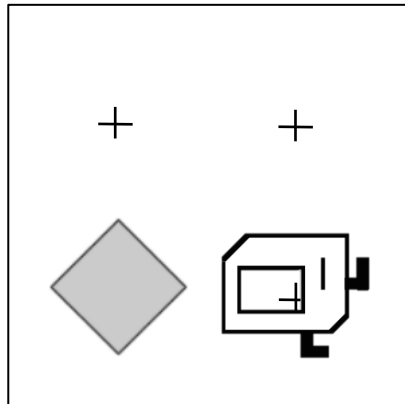


First time
through the
loop



Place Beeper Square

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

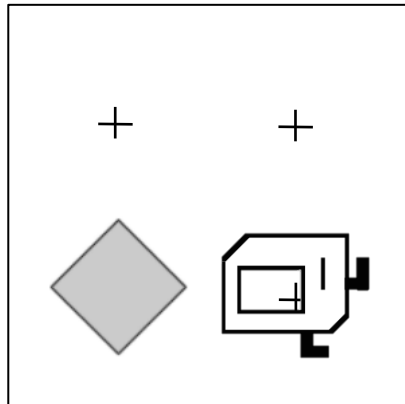


First time
through the
loop



Place Beeper Square

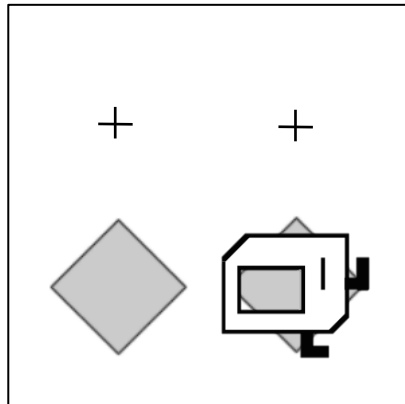
```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```



Second time
through the
loop

Place Beeper Square

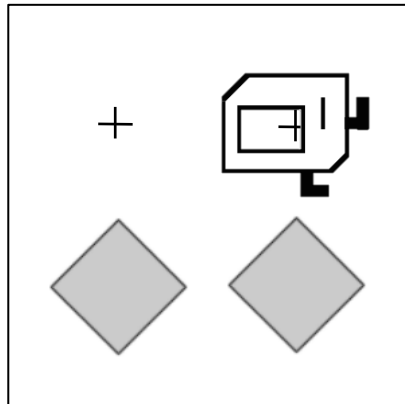
```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```



Second time
through the
loop

Place Beeper Square

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

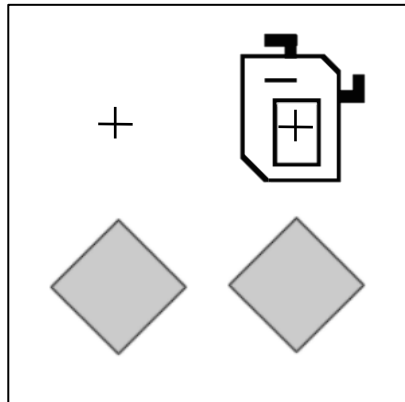


Second time
through the
loop



Place Beeper Square

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

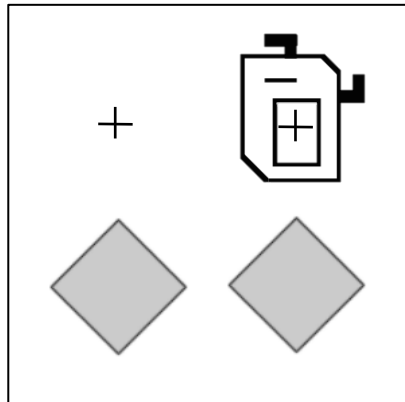


Second time
through the
loop



Place Beeper Square

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

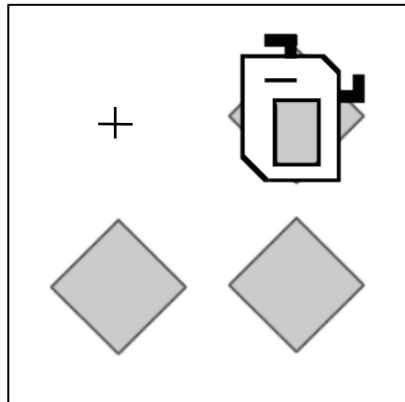


Third time
through the
loop



Place Beeper Square

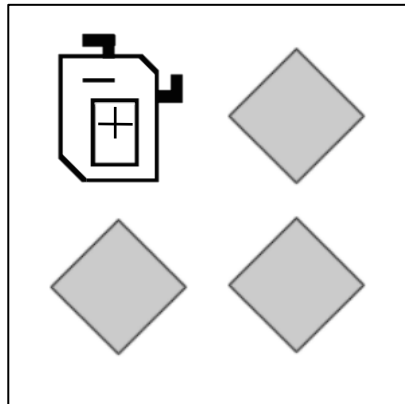
```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```



Third time
through the
loop

Place Beeper Square

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

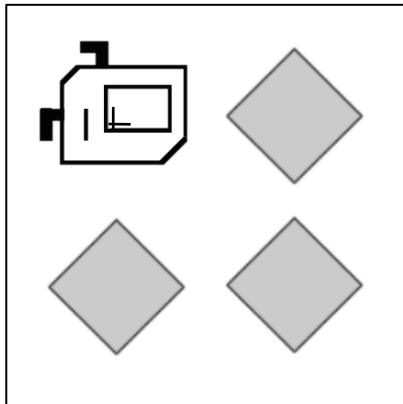


Third time
through the
loop



Place Beeper Square

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

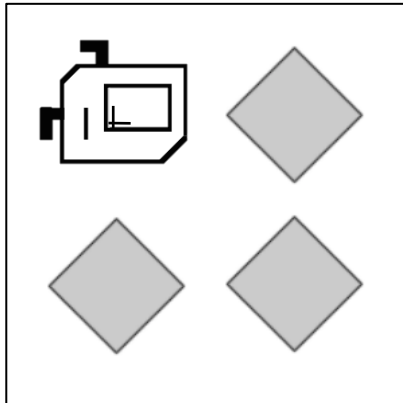


Third time
through the
loop



Place Beeper Square

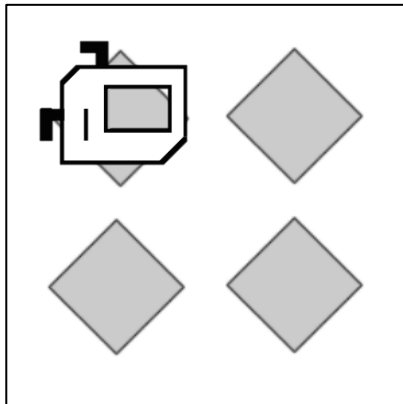
```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```



Fourth time
through the
loop

Place Beeper Square

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

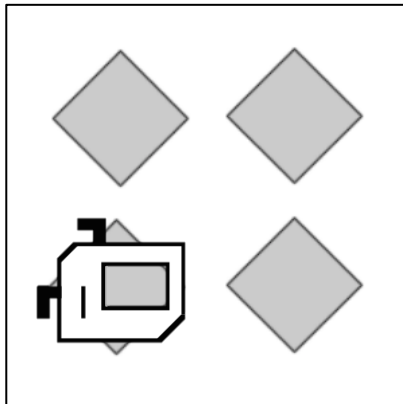


Fourth time
through the
loop



Place Beeper Square

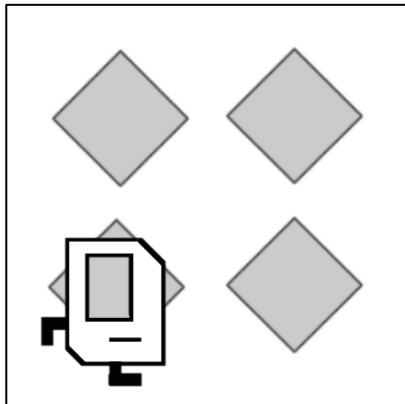
```
def main():  
    for i in range(4):  
        put_beeper()  
        move()  
        turn_left()
```



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



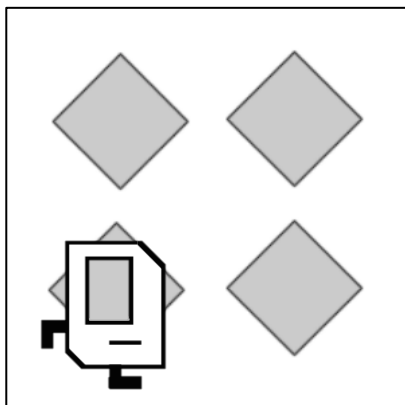
Place Beeper Square

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

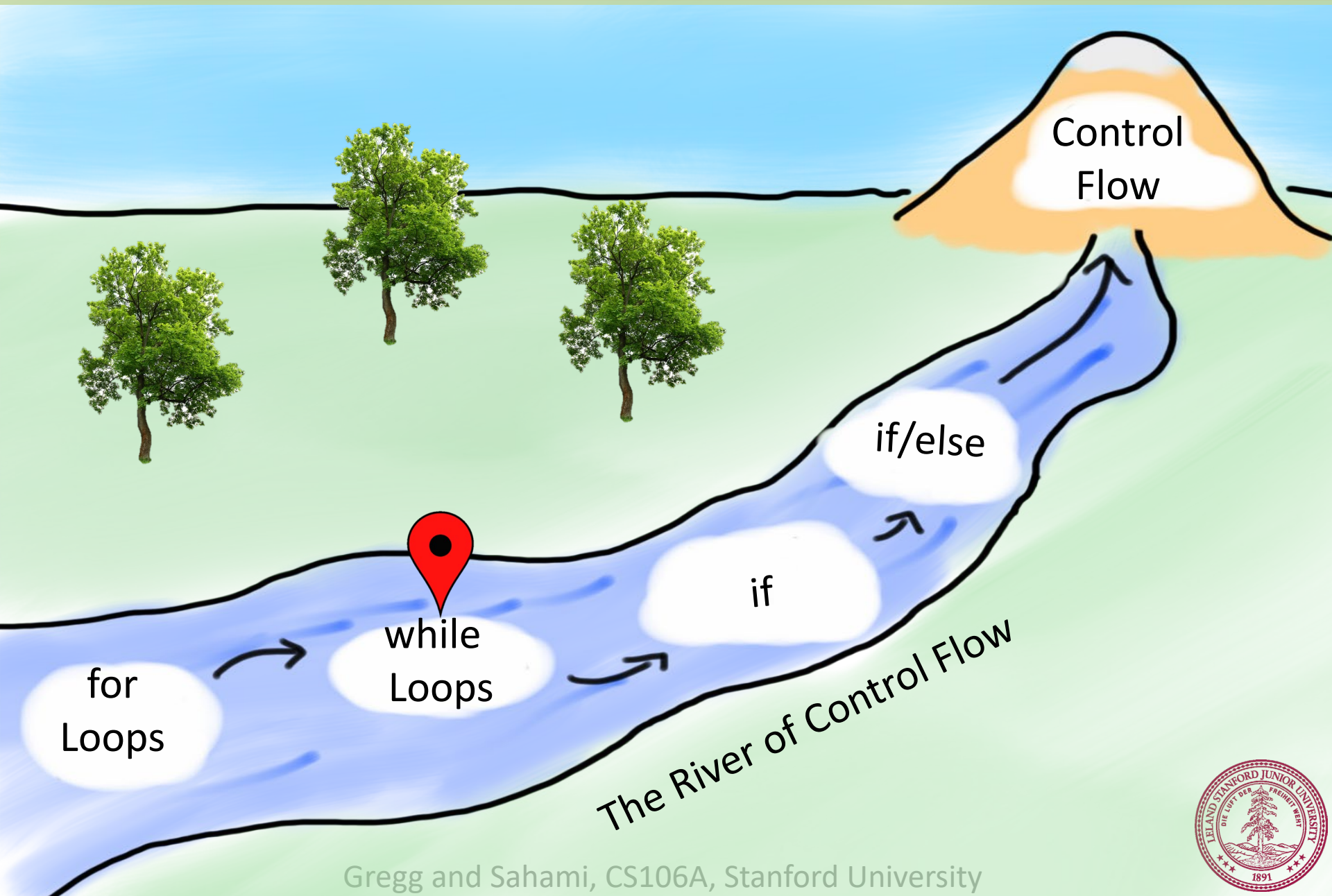


You often want the **postcondition** of a loop to match the **precondition**

Done!



Today's Route



while loop

```
while condition:
```

```
    statements
```

```
# note indenting
```

```
def move_to_wall():
```

```
    while front_is_clear():
```

```
        move()
```

```
# note indenting
```

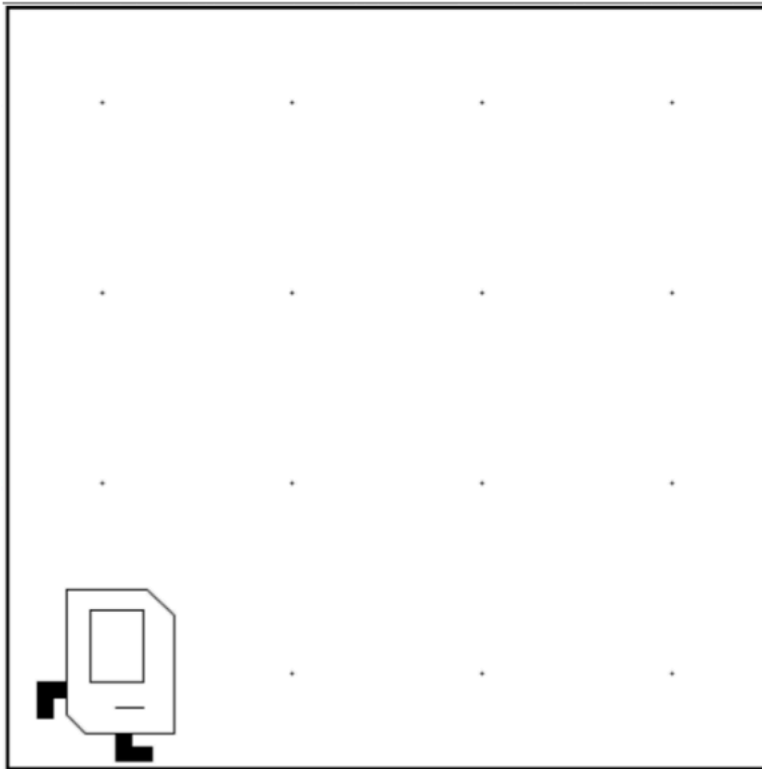
Conditions Karel Can Check For

<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_beepers_present()</code>	Are there beepers on this corner?
<code>beepers_in_bag()</code>	<code>no_beepers_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?

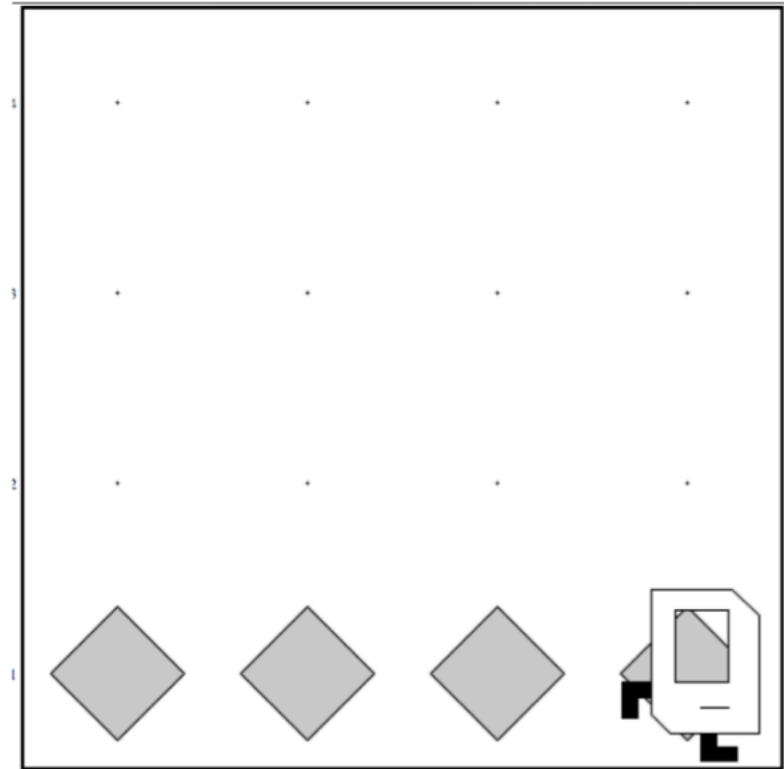
This is in Chapter 10 of the online Karel course reader

Task: Place Beeper Line

Before



After



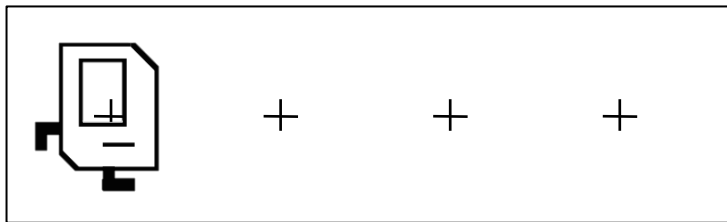
Place Beeper Line

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



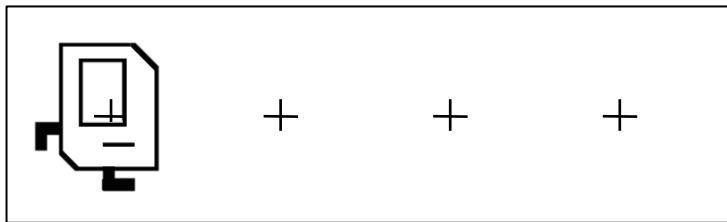
Place Beeper Line

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



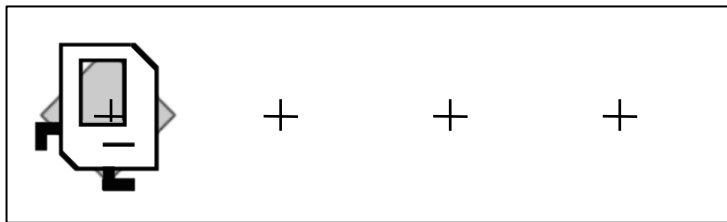
Place Beeper Line

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



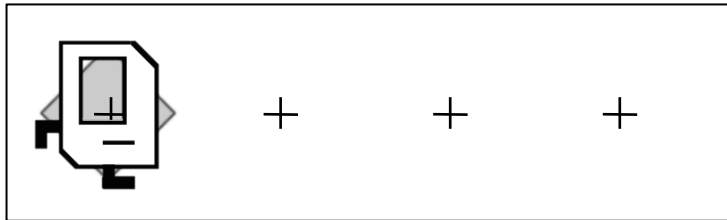
Place Beeper Line

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



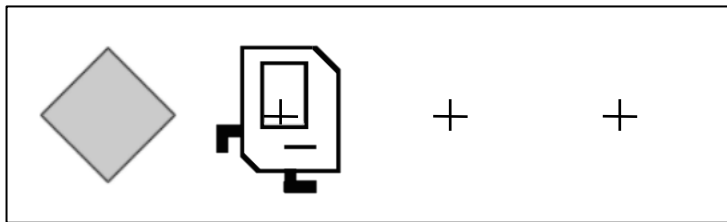
Place Beeper Line

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



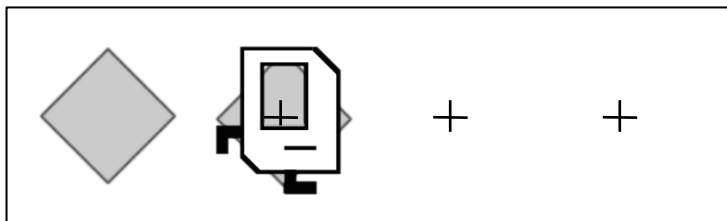
Place Beeper Line

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



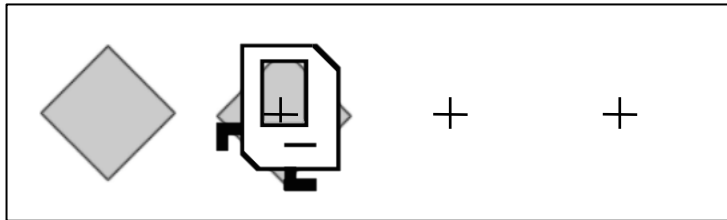
Place Beeper Line

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



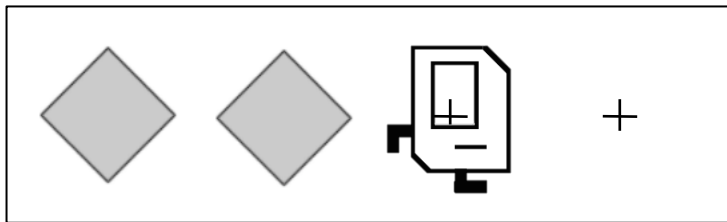
Place Beeper Line

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



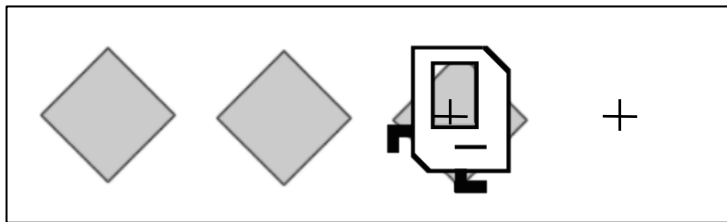
Place Beeper Line

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



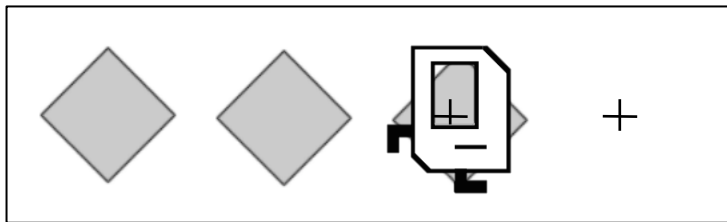
Place Beeper Line

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



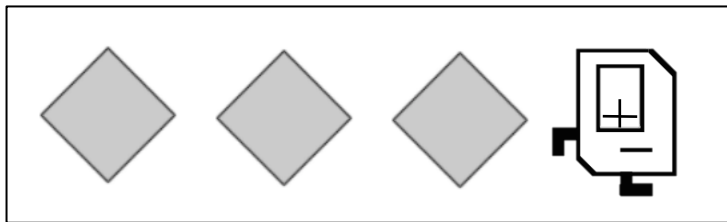
Place Beeper Line

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



Place Beeper Line

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



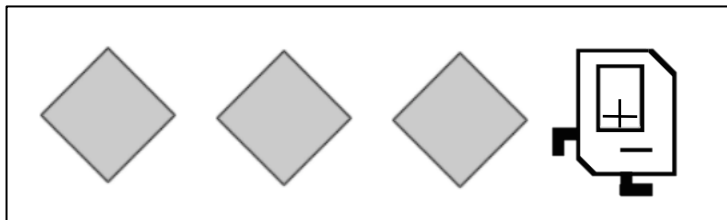
Place Beeper Line

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



BUGGY!

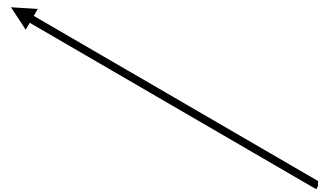
Done!



Place Beeper Line

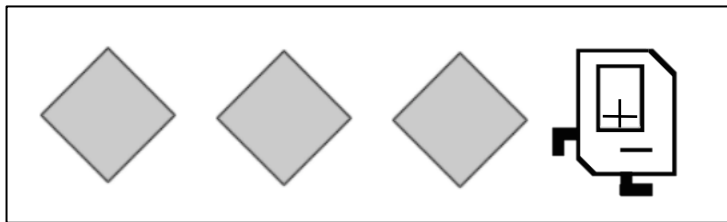


```
def main():  
    while front_is_clear():  
        put_beeper()  
        move()  
    put_beeper()           # add final put_beeper
```



Not in **while** loop

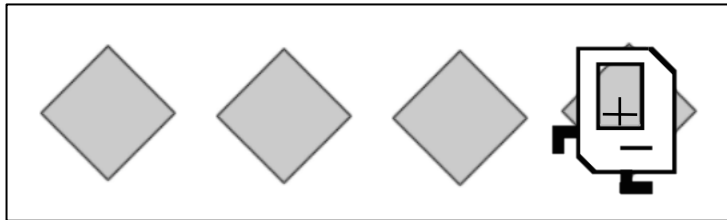
Fixed!



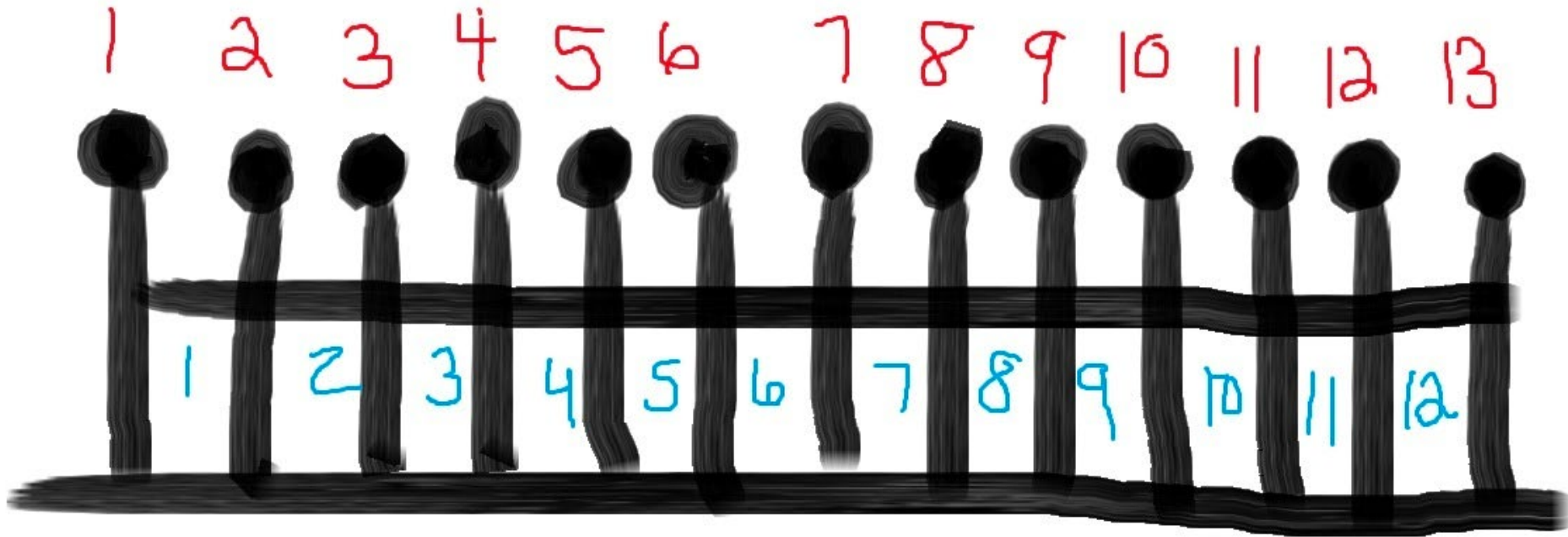
Place Beeper Line

```
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”

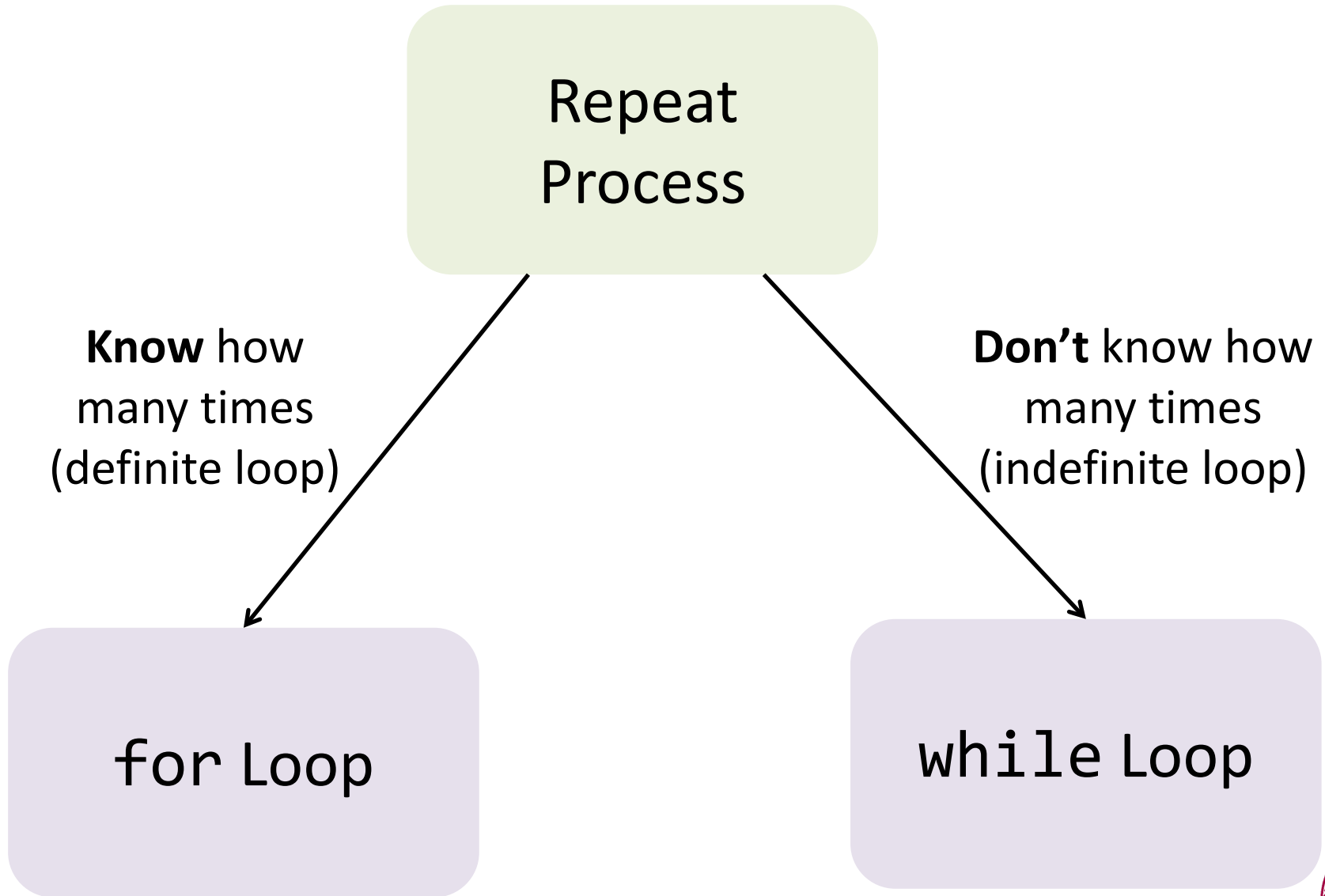


A program executes one line at a time.

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



Which Loop



Actual Bug from Marc II

9/9


0800 Andam started
 1000 " stopped - andam ✓

1300 (032) MP - MC ~~1.982647000~~
 (033) PRO 2 2.130476415 ~~(23)~~ 4.615925059(-2)
 coned 2.130676415

Relays 6-2 in 033 failed special speed test
 in relay " 10.000 test -

Relays changed

1700 Started Cosine Tape (Sine check)
 1525 Started Multi Adder Test.

1545  Relay #70 Panel F
 (moth) in relay.

First actual case of bug being found.

1630 Andam started.
 1700 closed down.

Relay 2145
 Relay 3370

Image: First Computer Bug, 1945.jpg, Wikimedia, Public Domain,
https://en.wikipedia.org/wiki/Grace_Hopper#/media/File:First_Computer_Bug,_1945.jpg



Admiral Grace Hopper



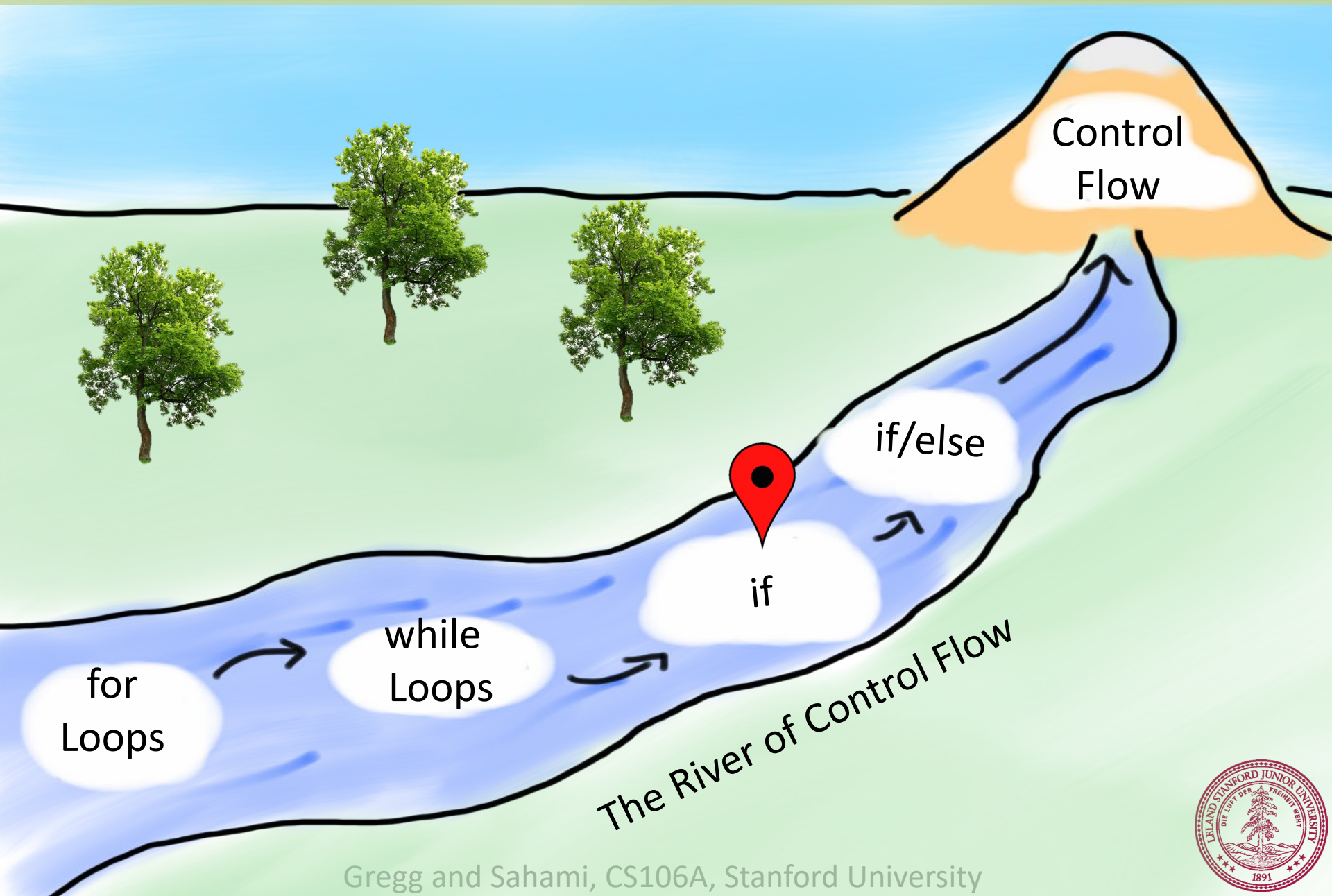
"The most important thing I've accomplished, other than building the compiler," she said, "is training young people."

Grace Hopper college at Yale University was renamed in her honor.

Image: Commodore Grace M. Hopper, USN (covered), Wikimedia, Public Domain,
[https://en.wikipedia.org/wiki/Grace_Hopper#/media/File:Commodore_Grace_M._Hopper,_USN_\(covered\).jpg](https://en.wikipedia.org/wiki/Grace_Hopper#/media/File:Commodore_Grace_M._Hopper,_USN_(covered).jpg)



Today's Route



if statement

```
if condition:
```

```
    statements
```

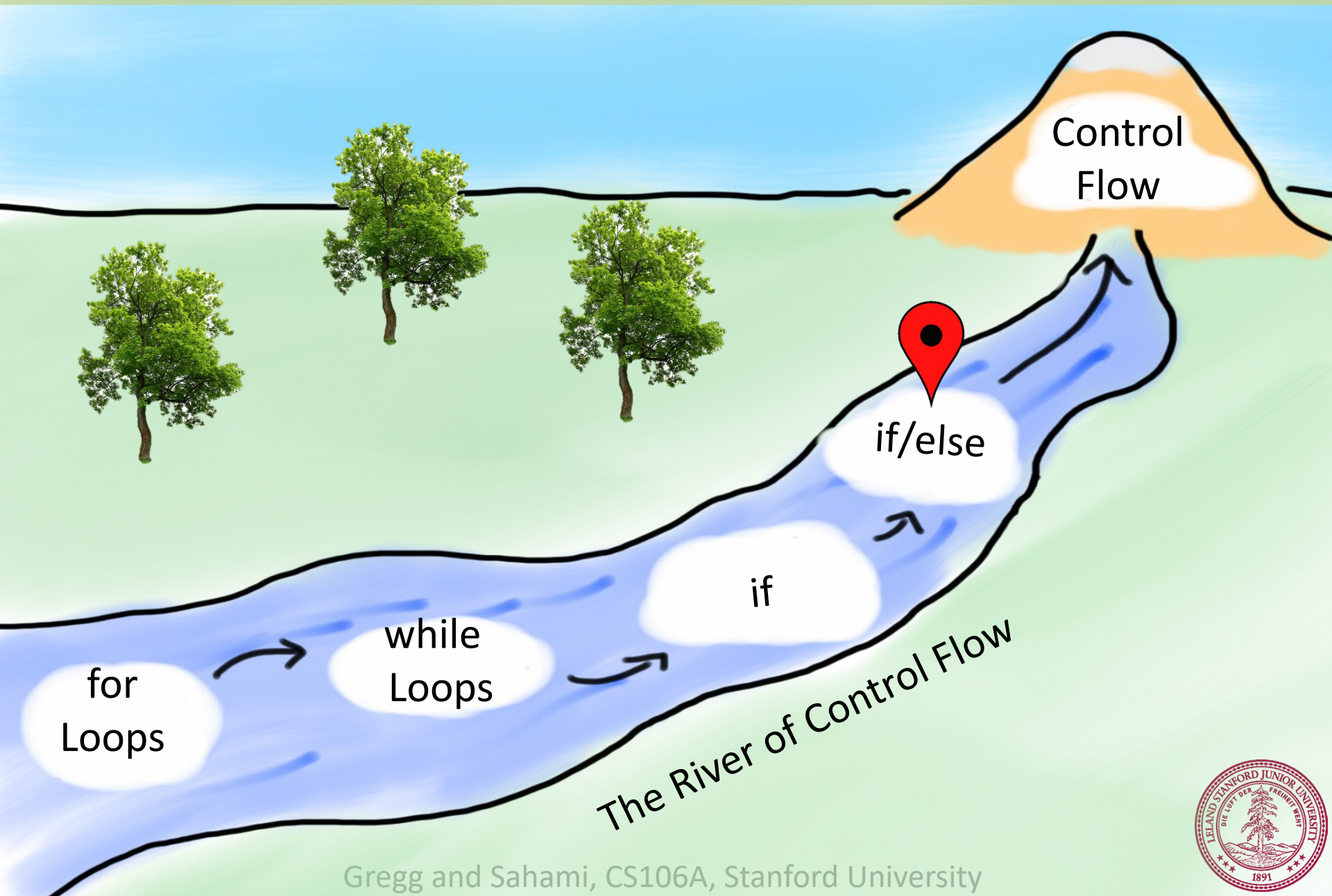
```
# note indenting
```

```
def safe_pick_up():
```

```
    if beepers_present():
```

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

Today's Route

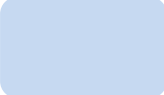


if-else statement


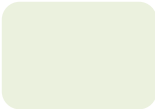
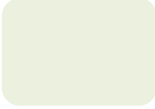
```
if condition:
```

```
     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

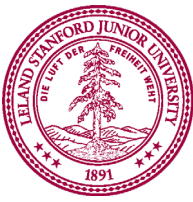
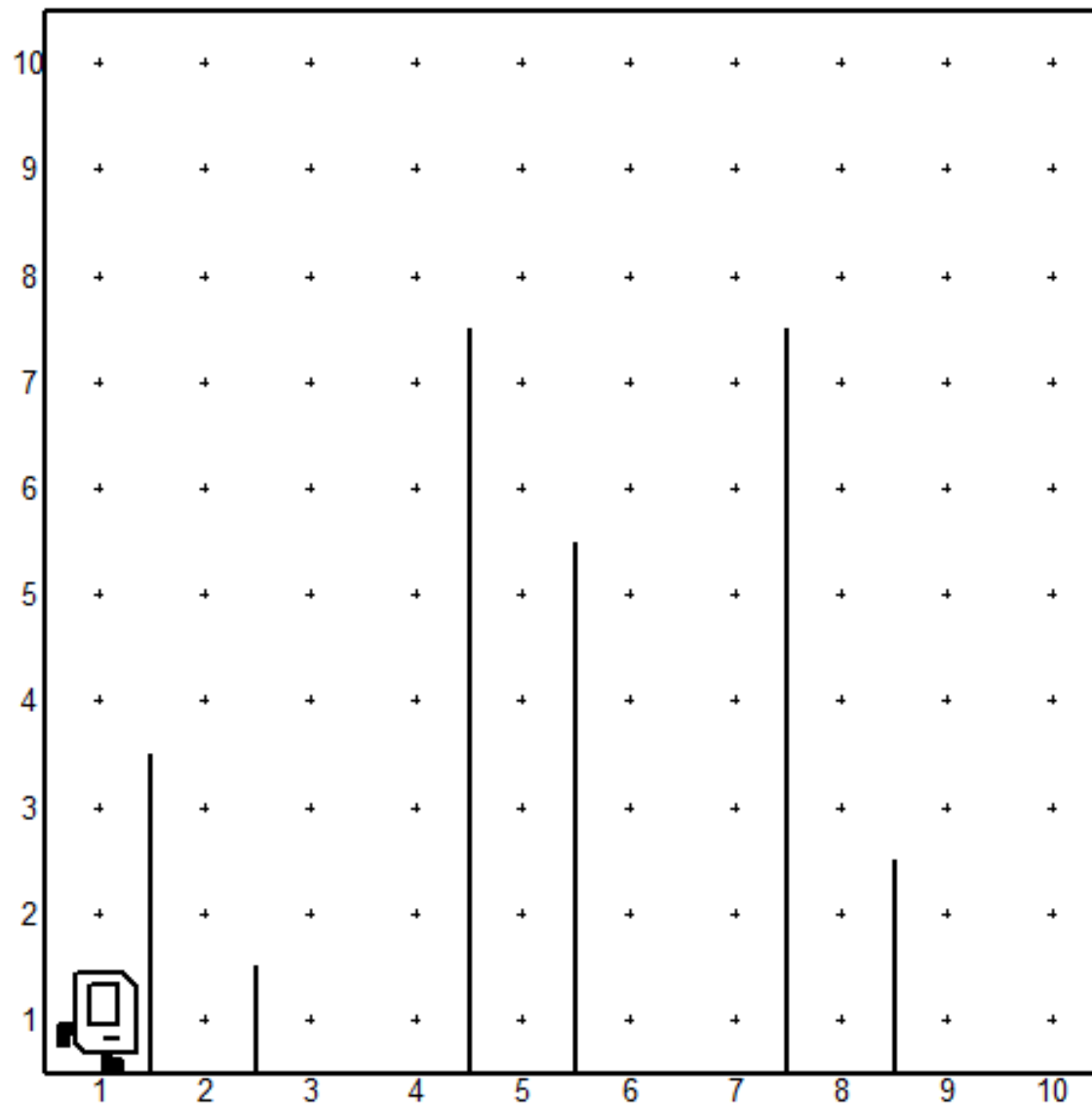
1. Code using loops and conditions
2. Trace programs that use loops and conditions



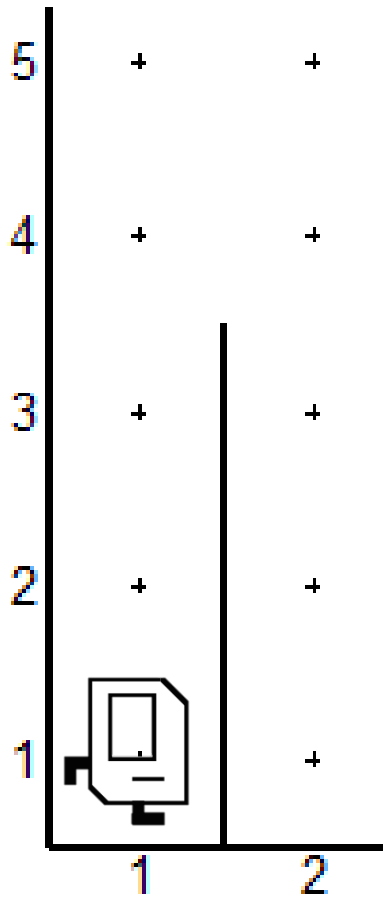


Putting it all together
SteepChaseKarel.py

Steeple Chase

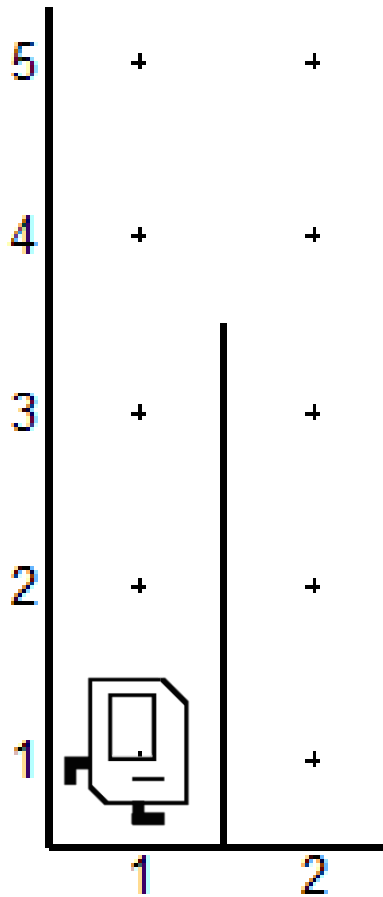


Focus on One Steeple



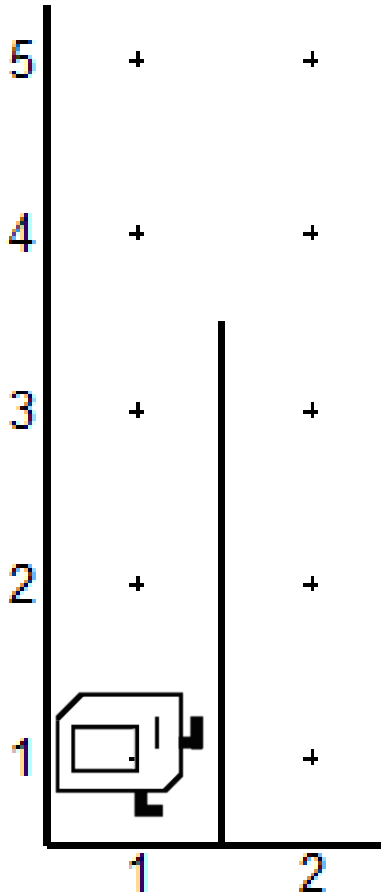
Focus on One Steeple

`turn_left()`



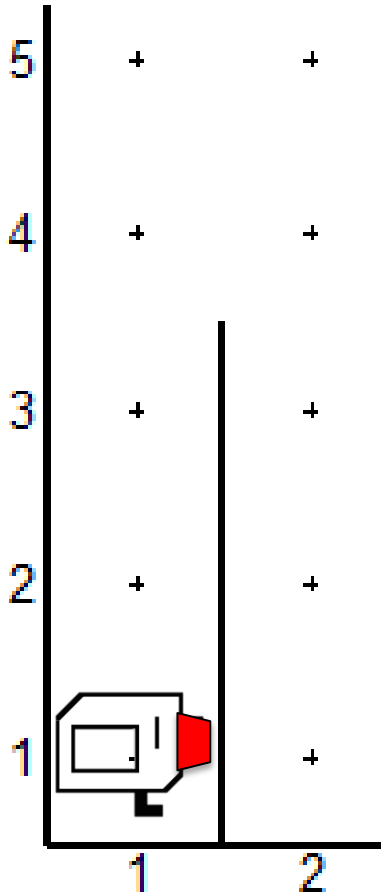
Focus on One Steeple

`turn_left()`



Focus on One Steeple

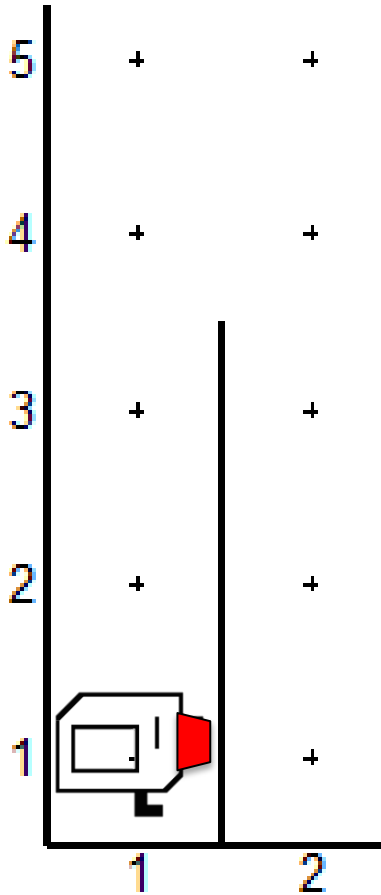
```
turn_left()  
while right_is_blocked():  
    move()
```



Focus on One Steeple

```
turn_left()
```

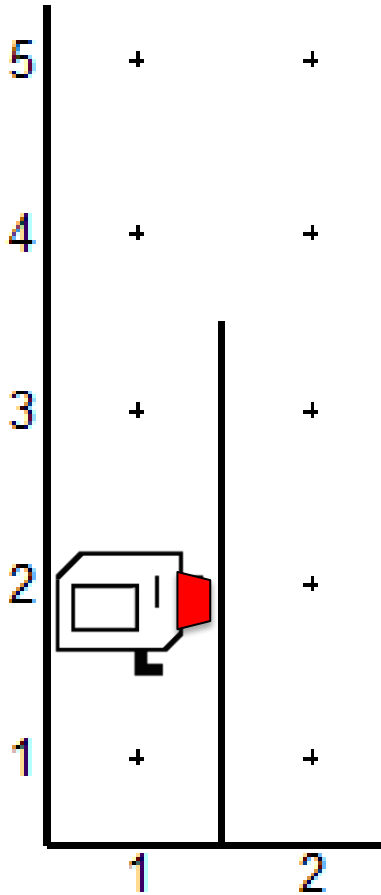
```
while right_is_blocked():  
    move()
```



Focus on One Steeple

```
turn_left()
```

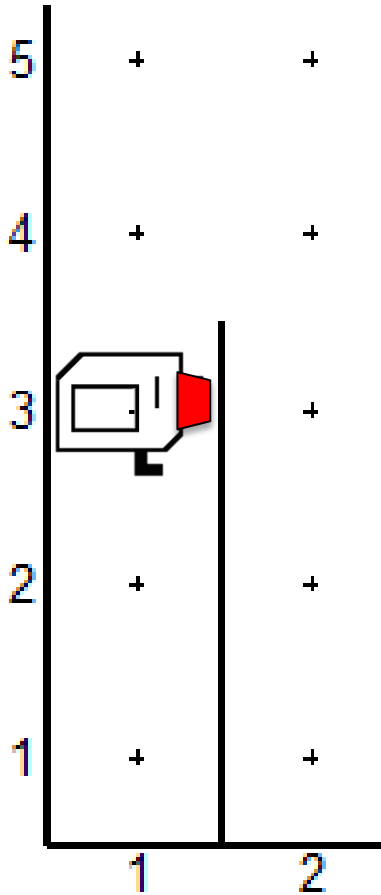
```
while right_is_blocked():  
    move()
```



Focus on One Steeple

```
turn_left()
```

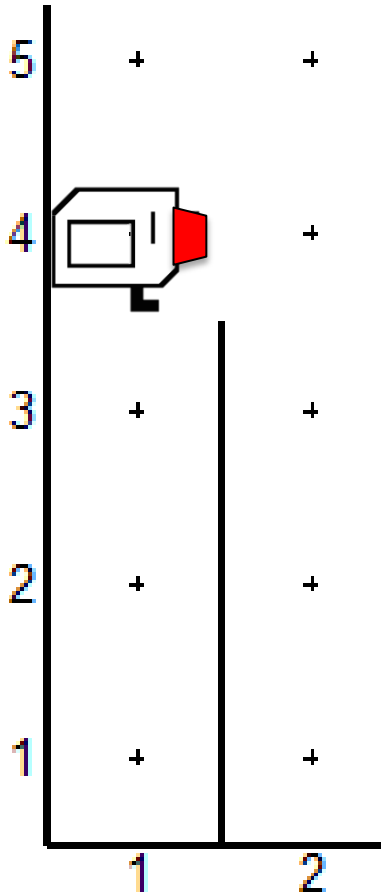
```
while right_is_blocked():  
    move()
```



Focus on One Steeple

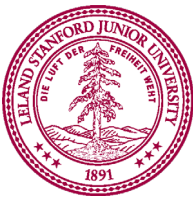
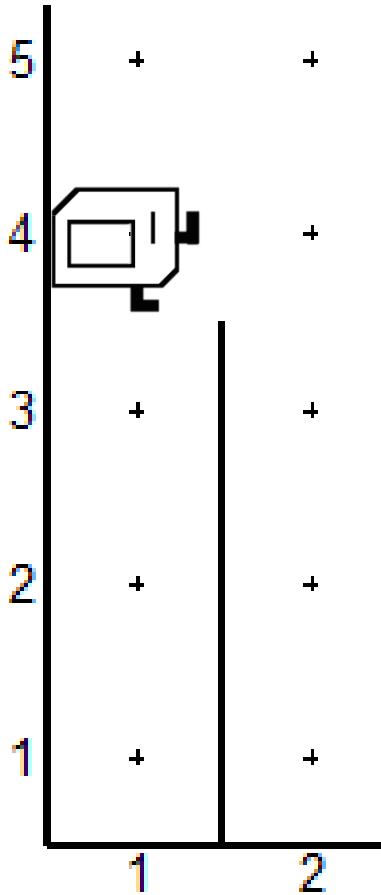
```
turn_left()
```

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



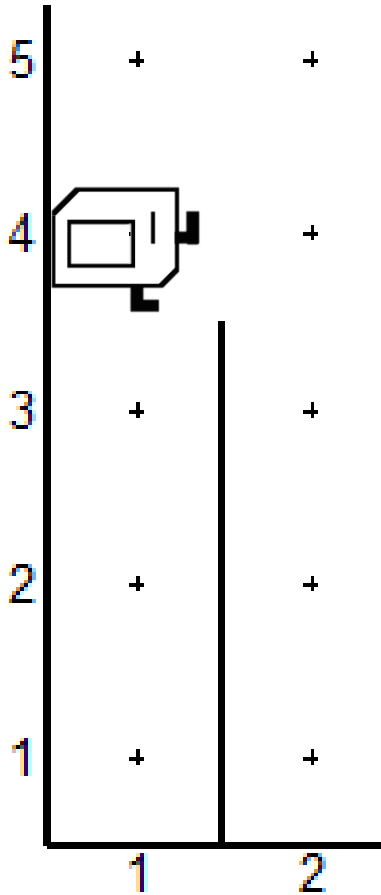
Focus on One Steeple

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



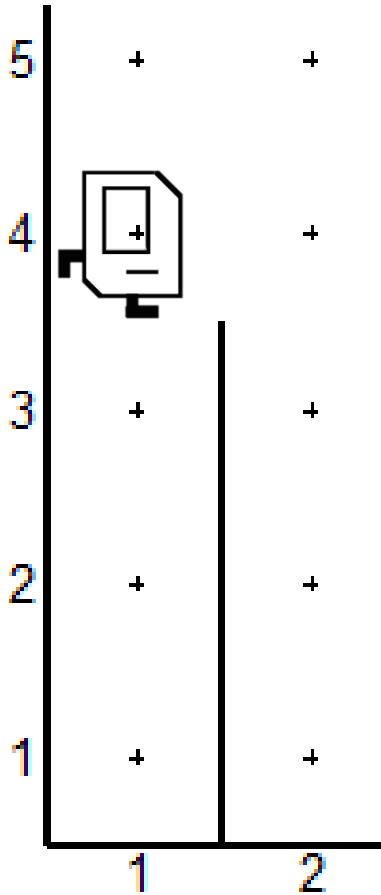
Focus on One Steeple

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



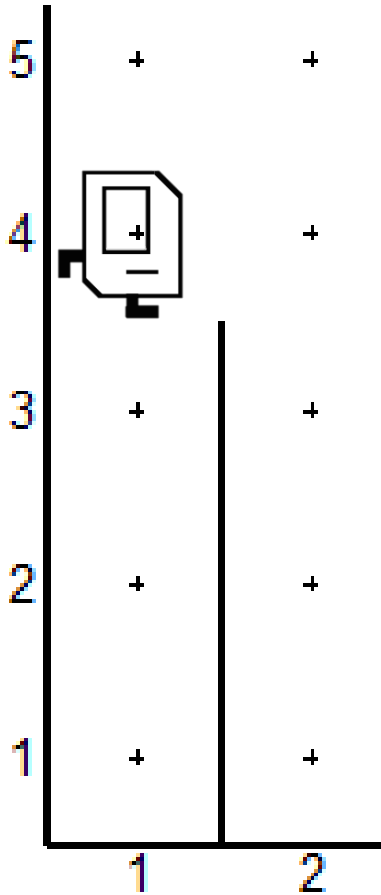
Focus on One Steeple

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

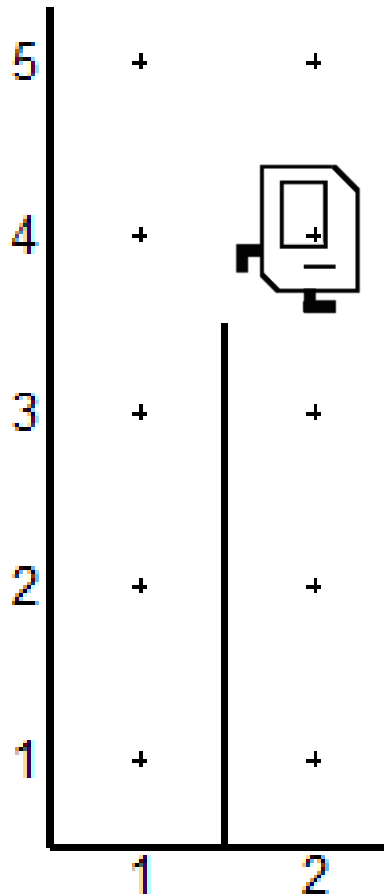


Focus on One Steeple

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



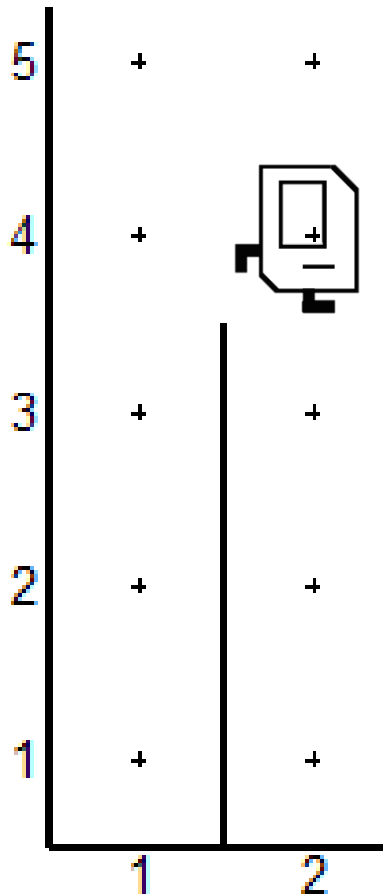
Focus on One Steeple



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



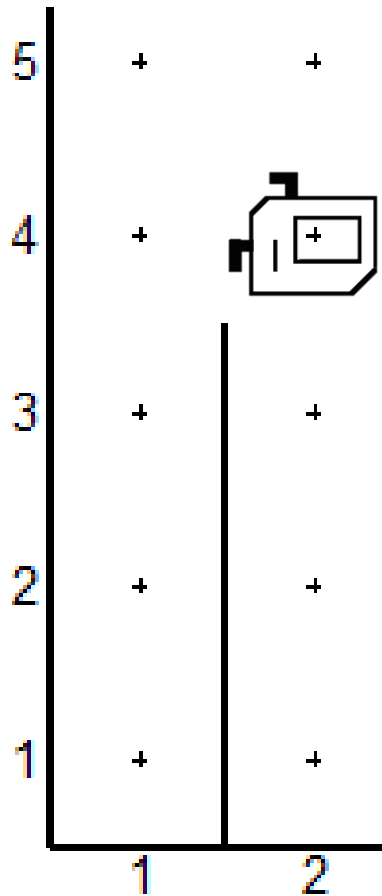
Focus on One Steeple



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



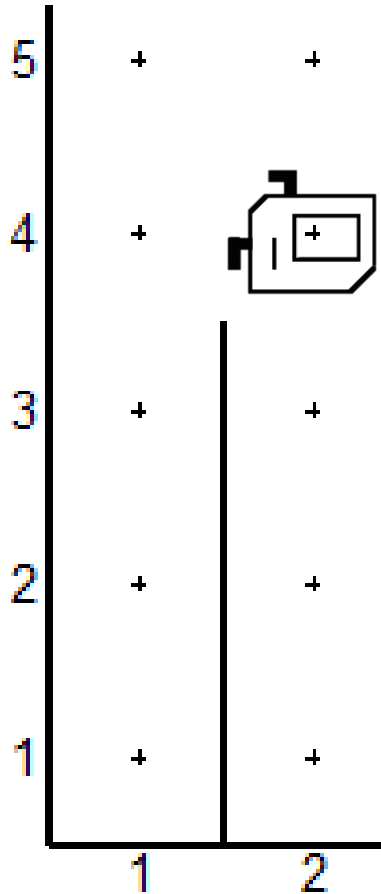
Focus on One Steeple



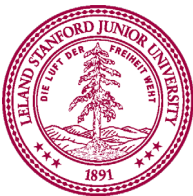
```
turn_left()  
while right_is_blocked():  
    move()  
turn_right()  
move()  
turn_right()
```



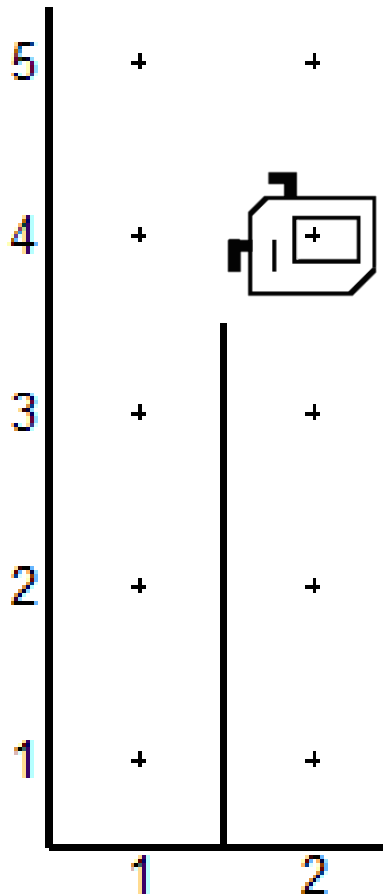
Focus on One Steeple



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



Focus on One Steeple

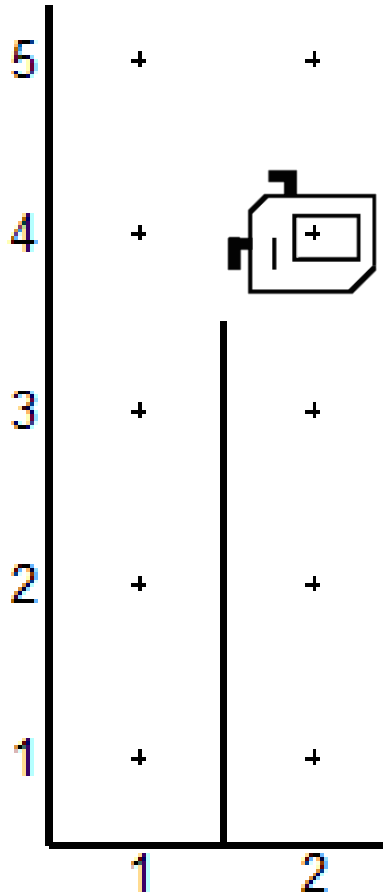


```
turn_left()  
while right_is_blocked():  
    move()  
turn_right()  
move()  
turn_right()  
move_to_wall()
```

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

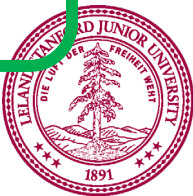


Focus on One Steeple

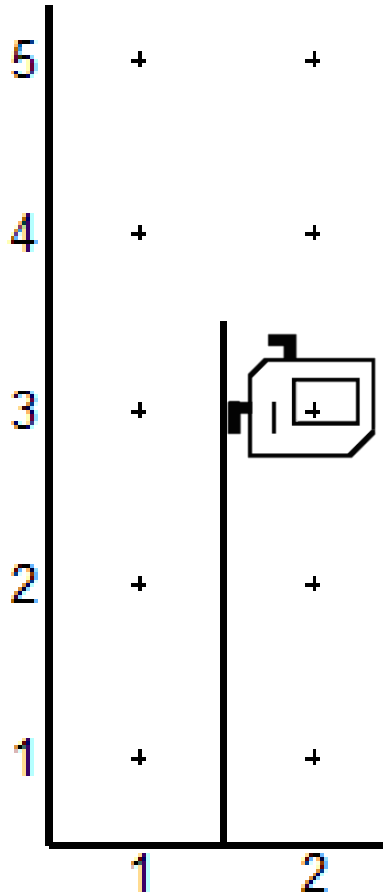


```
turn_left()
while right_is_blocked():
    move()
turn_right()
move()
turn_right()
move_to_wall()
```

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

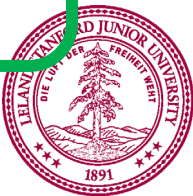


Focus on One Steeple

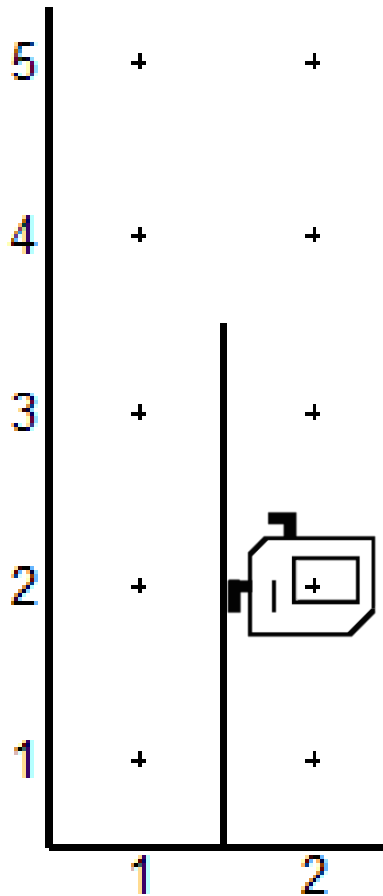


```
turn_left()
while right_is_blocked():
    move()
turn_right()
move()
turn_right()
move_to_wall()
```

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

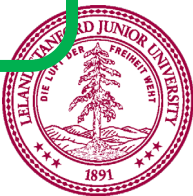


Focus on One Steeple

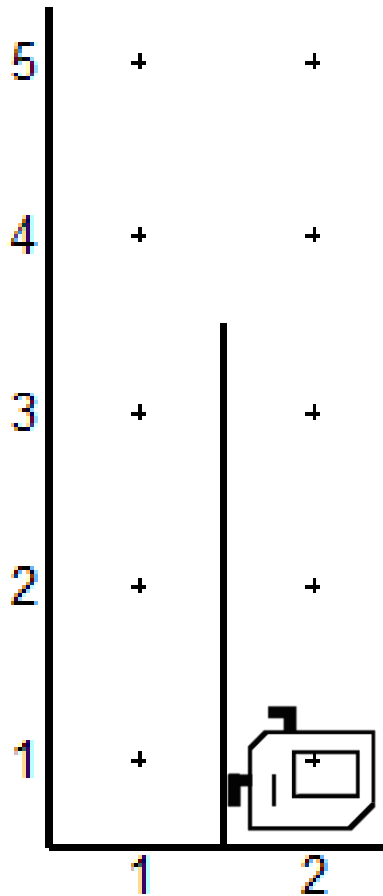


```
turn_left()  
while right_is_blocked():  
    move()  
turn_right()  
move()  
turn_right()  
move_to_wall()
```

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

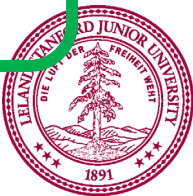


Focus on One Steeple



```
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

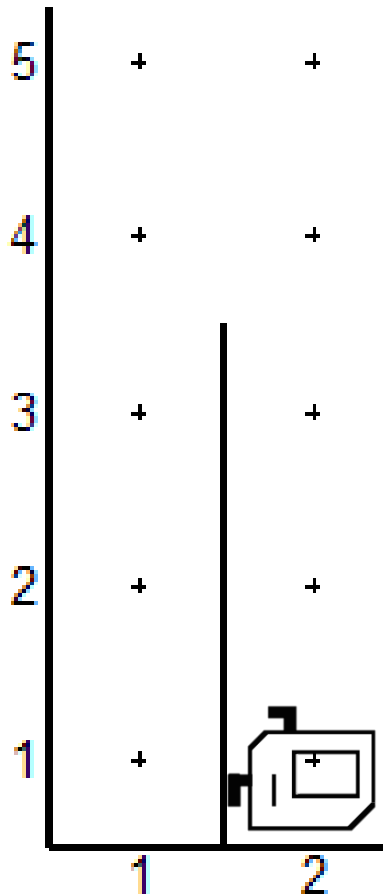


```
turn_left()  
while right_is_blocked():  
    move()  
turn_right()  
move()  
turn_right()  
move_to_wall()
```

```
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

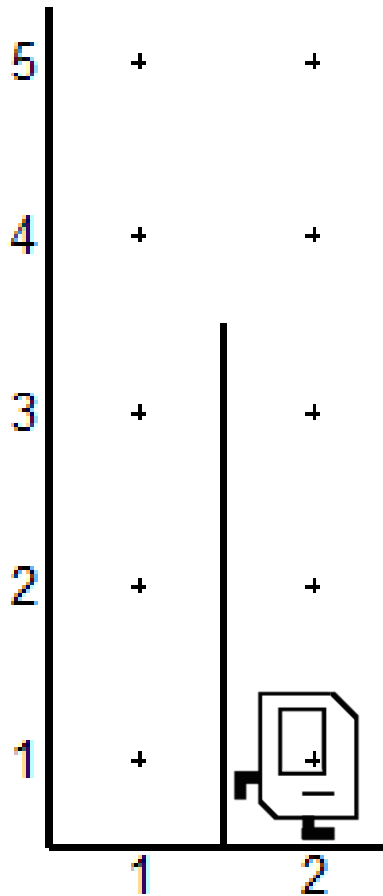


```
turn_left()  
while right_is_blocked():  
    move()  
turn_right()  
move()  
turn_right()  
move_to_wall()  
turn_left()
```

```
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:
SteepleChaseKarel.py