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!
• 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
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 if you have any questions or concerns!
Install PyCharm

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

<table>
<thead>
<tr>
<th>North</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>South</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>2</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>1</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 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
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.
"""

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

# Karel turns to the right

def turn_right():
    turn_left()
    turn_left()
    turn_left()
Today’s Goal

1. Code using loops and conditions
2. Trace programs that use loops and conditions
Today's Route

You are here

Control Flow

The River of Control Flow

for Loops

while Loops

if/else

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

def turn_right():
    for i in range(3):
        turn_left()  # note indenting
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()

You often want the postcondition of a loop to match the precondition.
Today’s Route

The River of Control Flow

- for Loops
- while Loops
- if
- if/else

Control Flow
while **condition**:

    statements

    # note indenting

---

def move_to_wall():

    while front_is_clear():

        move()  # note indenting
### Conditions Karel Can Check For

<table>
<thead>
<tr>
<th>Test</th>
<th>Opposite</th>
<th>What it checks</th>
</tr>
</thead>
<tbody>
<tr>
<td>front_is_clear()</td>
<td>front_is_blocked()</td>
<td>Is there a wall in front of Karel?</td>
</tr>
<tr>
<td>left_is_clear()</td>
<td>left_is_blocked()</td>
<td>Is there a wall to Karel’s left?</td>
</tr>
<tr>
<td>right_is_clear()</td>
<td>right_is_blocked()</td>
<td>Is there a wall to Karel’s right?</td>
</tr>
<tr>
<td>beepers_present()</td>
<td>no_beepers_present()</td>
<td>Are there beepers on this corner?</td>
</tr>
<tr>
<td>beepers_in_bag()</td>
<td>no_beepers_in_bag()</td>
<td>Any there beepers in Karel’s bag?</td>
</tr>
<tr>
<td>facing_north()</td>
<td>not_facing_north()</td>
<td>Is Karel facing north?</td>
</tr>
<tr>
<td>facing_east()</td>
<td>not_facing_east()</td>
<td>Is Karel facing east?</td>
</tr>
<tr>
<td>facing_south()</td>
<td>not_facing_south()</td>
<td>Is Karel facing south?</td>
</tr>
<tr>
<td>facing_west()</td>
<td>not_facing_west()</td>
<td>Is Karel facing west?</td>
</tr>
</tbody>
</table>

This is in Chapter 10 of the online Karel course reader
Task: Place Beeper Line

Before

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

Not in while loop

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

for Loop

Don’t know how many times (indefinite loop)

while Loop
Actual Bug from Marc II

1100
1525
1545

Started Cosine Tape (Sine -
Started Multi Adder Test.

First actual case of bug

1630
1700

1630 and angulal started.
closed down.
Grace Hopper
Today’s Route

The River of Control Flow

- for Loops
- while Loops
- if
- if/else

Control Flow

Sahami, CS106A, Stanford University
if condition:
    statements  # note indenting

def safe_pick_up():
    if beepers_present():
        pick_beeper()  # note indenting
Today’s Route

The River of Control Flow

Control Flow

if/else

if

while Loops

for Loops
if-else statement

```python
if condition:
    statements  # note indenting
else:
    statements  # note indenting

def invert_beepees():
    if beepers_present():
        pick_beeper()  # note indenting
    else:
        put_beepeer()  # 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

```python
turn_left()
while right_is_blocked():
    move()
```
Focus on One Steeple

turn_left()
while right_is_blocked():
    move()
Focus on One Steeple

code:

```
turn_left()
while right_is_blocked():
    move()
```
Focus on One Steeple

turn_left()
while right_is_blocked():
    move()
turn_left()
while right_is_blocked():
    move()
turn_left()
while right_is_blocked():
    move()
turn_right()
```python
turn_left()
while right_is_blocked():
    move()
    turn_right()
```
Focus on One Steeple

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

turn_left()
while right_is_blocked():
    move()

turn_right()
move()

turn_right()
move_to_wall()

---

def move_to_wall():
    while front_is_clear():
        move()
```
def move_to_wall():
    while front_is_clear():
        move()
    turn_right()
    move()
    turn_right()
    move_to_wall()
def move_to_wall():
    while front_is_clear():
        move()
    turn_right()
    move()
focus on one steeple

turn_left()
while right_is_blocked():
    move()
    turn_right()
move()
def move_to_wall():
    while front_is_clear():
        move()

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()
turn_left()
```

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

ascend_hurdle()

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

turn_right()
move_to_wall()
turn_left()

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

ascend_hurdle()

move()

turn_right()
move_to_wall()
turn_left()

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

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