Functions + Control Flow

CS106AP Lecture 2
Today’s questions

How do we teach Karel new things?
How do we handle different worlds?
Today’s topics

1. Karel review
2. Karel functions
3. Control flow
   Conditionals (if)
   Loops (while)
4. What’s next?
Remember Karel?
This is Karel’s world

corners
(intersections)

walls

streets
(rows)

beepers

avenues
(columns)
What Karel can do

- move()
- turn_left()
- put_beeper()
- pick_beeper()
What Karel can do

**move()**

Makes Karel move forward one square in the direction it is facing.

Errors if there is a wall in front of Karel.
What Karel can do

`turn_left()`

Makes Karel turn left.
What Karel can do

**put_beeper()**

Makes Karel place a beeper on the corner where it is currently standing.

You can place multiple beepers on any given corner.
What Karel can do

`pick_beeper()`

Makes Karel pick up a beeper from the corner where it is currently standing.

Errors if there are no beepers present on the corner.
Karel demo!
Karel Goes Home

begin

end
Karel Goes Home
Karel Goes Home

```
turn_left()
```
Karel Goes Home

move()
Karel Goes Home

```
turn_left()
turn_left()
turn_left()
```

turning right

```
turn_left()
turn_left()
turn_left()
```
Karel Goes Home

```
move()
```
Karel Goes Home

turn_left()
turn_left()
turn_left()
Karel Goes Home

move()
Karel Goes Home

turn_left()
Karel Goes Home

move()
Karel Goes Home

pick_beeper()
def main():
    # Climb wall
    turn_left()
    turn_left()
    turn_left()
    move()
    turn_left()
    turn_left()
    move()
    turn_left()
    turn_left()
    turn_left()
    move()

    # Go home
    turn_left()
    move()
def main():
    # Climb wall
    turn_left()
    move()
    turn_left()
    turn_left()
    move()
    turn_left()
    turn_left()
    turn_left()
    move()

    # Go home
    turn_left()
    move()
How do we teach Karel new commands?
How do we teach Karel new commands?

Functions!
**Definition**

**function**
A smaller part of your program that solves a specific sub-problem
Functions

● Smaller parts of your program that solve specific sub-problems
  ○ By decomposing your program into many small functions, your code becomes easier to read.
Functions

- Smaller parts of your program that solve a specific sub-problems

- A function has two parts:
  - The function definition
  - One or more function calls
Functions

- Smaller parts of your program that solve a specific sub-problems

- A function has two parts:
  - The function definition
    - What does the new Karel command do?
  - One or more function calls
Functions

● Smaller parts of your program that solve a specific sub-problems

● A function has two parts:
  ○ The function definition
    What does the new Karel command do?
  ○ One or more function calls
    Where you tell Karel to actually do the command
Let’s revisit KarelGoesHome.py
Turning right

def turn_right():
    """
    Makes Karel turn right.
    """
    turn_left()
    turn_left()
    turn_left()
Turning right

def turn_right():
    """
    Makes Karel turn right.
    """
    turn_left()
    turn_left()
    turn_left()
    turn_left()

def main():
    ...
    turn_right()
    ...

Turning right

```python
def turn_right():
    ""
    Makes Karel turn right.
    """
    turn_left()
    turn_left()
    turn_left()
```

```python
def main():
    ...
    turn_right()
    ...
```

Function definition
What does the new Karel command do?
Turning right

```python
def turn_right():
    """
    Makes Karel turn right.
    """
    turn_left()
    turn_left()
    turn_left()
    turn_left()

def main():
    ...
    turn_right()
    ...
```

Function call

*Where you tell Karel to actually do the command*
def turn_right():
    
    """
    Makes Karel turn right.
    """
    turn_left()
    turn_left()
    turn_left()

def main():
    ...
    turn_right()
    ...

“Go find the function definition, do the commands inside, and come back when done.”

Function call
Where you tell Karel to actually do the command
General structure of functions

[whiteboard]
The structure of a function

def my_function_name():
    
    """
    Write a good description
    for your function here.
    """
    
    # Put other Karel commands here

...

def caller_function():
    my_function_name() # Function call
The structure of a function

def my_function_name():
    
    """
    Write a good description for your function here.
    """
    
    # Put other Karel commands here

    ...

def caller_function():
    my_function_name() # Function call
The structure of a function

```python
def my_function_name():
    """
    Write a good description for your function here.
    """
    # Put other Karel commands here

...  
def caller_function():
    my_function_name()  # Function call
```

**Definition**

**inline comments**
Comments that describe a specific, complex block of code using `#`
The structure of a **function**

```python
def my_function_name():
    
    """
    Write a good description for your function here.
    """

    # Put other Karel commands here

    ...

def caller_function():
    my_function_name()  # Function call
```

Make sure lines inside each function are indented.
The structure of a **function**

```python
def my_function_name():
    
    """
    Write a good description for your function here.
    """

    # Put other Karel commands here

    ...

def caller_function():
    my_function_name()  # Function call
```

**Style note**

**indentation**

Each level of indentation should be four spaces.

Make sure lines inside each function are indented
The structure of a function

```python
def my_function_name():
    """
    Write a good description for your function here.
    """
    # Put other Karel commands here

...

def caller_function():
    my_function_name()  # Function call
```

**Style note**

*function names*

Separate words with underscores and use verbs.
The structure of a function

```python
def my_function_name():
    """
    Write a good description for your function here.
    """
    # Put other Karel commands here

... def caller_function():
    my_function_name() # Function call
```

Note: The caller function doesn’t have to be main()!
Think/Pair/Share:

Write a function `turn_around()` that makes Karel rotate 180 degrees.
Common issues
[demo]
“Syntax” errors (i.e. typos)

- Forgetting `()` or `:` in the function definition
  - Line number shows up in the error!

- Forgetting `()` in the function call
  - You get a warning (yellow underlining) instead of an error here
“Syntax” errors (i.e. typos)

- Forgetting `()` or `:` in the function definition
  - Line number shows up in the error!

- Forgetting `()` in the function call
  - You get a warning (yellow underlining) instead of an error here

These are super common, even for professional programmers!
Common issues
Common bugs

issues
**Definition**

debugging
The process of finding and fixing errors (syntactical or logical) in your code.
debugging

The process of finding and fixing errors (syntactical or logical) in your code.

**Definition**

Most of your time will be spent doing this!
How do we handle different worlds?
How do we handle different worlds?

Control flow!
Definition

**control flow structures**
Code features that affect the order, or flow, in which the lines of code in a program happen
Conditionals
(if statements)
If statements

if ___________:
    # Do something
If statements

if Condition:

# Do something
If statements

\[
\text{Condition} \quad (\text{any boolean expression})
\]

if \[\quad :\]

# Do something
If statements

if ___________:
    # Do something

Condition
(any boolean expression)

Definition

boolean expression
A code snippet that evaluates to True or False
If statements

if ___________:
    # Do something

Condition

True

False
If statements

```python
if ___________:
    # Do something
```

Condition

- True
- False
If statements

```
if ___________:
    # Do something
```

Condition

True  False
If statements

if ___________:
    # Do something
If statements

```python
if ___________
    # Do something
```
If statements

if ___________:
    # Do something
If statements

if the light is red:
   Stop
Let’s bring it back to Karel: 

`move_safely()`

[demo]
General structure of if statements

[whiteboard]
The structure of an **if** statement

```python
if boolean expression:
    # Do something when condition is True
```
The structure of an **if statement**

```python
if boolean expression:
    # Do something when condition is True
```

Make sure lines inside the if statement are indented!
Boolean expressions in Karel
Karel functions that evaluate to boolean expressions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>front_is_clear()</td>
<td>Is there no wall in front of Karel?</td>
</tr>
<tr>
<td>left_is_clear()</td>
<td>Is there no wall to Karel’s left?</td>
</tr>
<tr>
<td>right_is_clear()</td>
<td>Is there no wall to Karel’s right?</td>
</tr>
<tr>
<td>on_beeper()</td>
<td>Is there a beeper on the corner where Karel is standing?</td>
</tr>
<tr>
<td>facing_north()</td>
<td>Is Karel facing north?</td>
</tr>
<tr>
<td>facing_south()</td>
<td>Is Karel facing south?</td>
</tr>
<tr>
<td>facing_east()</td>
<td>Is Karel facing east?</td>
</tr>
<tr>
<td>facing_west()</td>
<td>Is Karel facing west?</td>
</tr>
</tbody>
</table>
Karel functions that evaluate to boolean expressions

- These built-in Karel functions evaluate to:
  - True if the answer to the corresponding question is “yes”
  - False if the answer to the corresponding question is “no”

- You can find all of these in the Karel Reference Guide on the course website.
Loops

(while loops)
Before we had if statements...

```python
if ____________:
    # Do something
```
Before we had if statements...

```python
if __________ :
    # Do something
```

What if we want something to happen multiple times?
While loops

while ____________:

    # Do something
While loops

while ________:  
# Do something

Condition
While loops

```
while ___________:
    # Do something
```

- **Condition**
  - **True**
  - **False**
While loops

```python
while ___________:
    # Do something
```

Condition

- True
- False

repeat if true
While loops

while ___________:
    # Do something

Condition

True

False
While loops

while ___________:
    # Do something
While loops

while road is clear:

Drive forward
Let’s bring it back to Karel: travel_row()
General structure of

while loops
The structure of a **while loop**

```python
while boolean expression:
    # Do something while condition is True
```
The structure of a **while loop**

```python
while boolean expression:
    # Do something while condition is True
```

Make sure lines inside the if statement are indented!
Let’s bring it back to Karel: `travel_row()`
Let’s bring it back to Karel:

travel_row()
Let’s bring it back to Karel: `make_beeper_row()`
[demo]
Definition

fencepost problem (off-by-one bug)
An error in which your code performs a task one too many or too few times.
fencepost problem (off-by-one bug)
An error in which your code performs a task one too many or too few times.

Why?
What if some corners already have beepers?
What is some corners already have beepers?
Your turn to try!

`make_beeper_row()` (V2)

(tomorrow)
What if some corners already have beepers?

- Combine what we know about `if` and `while`!

- Potentially useful:
  - `on_beeper()`
  - `not`
While loop summary (and misconceptions)
while loop summary

- All lines inside the while loop will run if the condition is true
while loop summary

- **All lines** inside the while loop will run if the condition is true.

- Lines inside the while loop happen **one at a time** and **in order**.
while loop summary

- **All lines** inside the while loop will run if the condition is true.
- Lines inside the while loop happen **one at a time** and **in order**.
- The condition is checked again **only after all of the lines have run**.
while loop summary

- **All lines** inside the while loop will run if the condition is true

- Lines inside the while loop happen **one at a time** and **in order**

- The condition is checked again **only after all of the lines have run**

*It’s a very common misconception to expect the while loop to stop in the middle when the condition is no longer true!*
while loop summary

- All lines inside the while loop will run if the condition is true
- Lines inside the while loop happen one at a time and in order
- The condition is checked again only after all of the lines have run
- It is possible for the while loop body to run 0 times!
  - Just like with if statements
Today’s topics

1. Karel review
2. Karel functions
3. Control flow
   - Conditionals (if)
   - Loops (while)
4. What’s next?
What’s next?
Roadmap

Programming Basics
- Variables
- Control Flow
- Decomposition + Functions
- Karel the Robot

The Console

Images

Graphics

Data structures

Midterm

Object-Oriented Programming

Everyday Python

Life after CS106AP!
Tomorrow...

- Focus on writing good functions and good programs
- Practice breaking down complex Karel problems
  - Good homework practice
  - Bring your laptops!
- More examples of how Karel can represent real-world problems
Extra Slides

(will cover tomorrow!)
Your turn to try!
make_beeper_row() (V2)
What if some corners already have beepers?

- Combine what we know about `if` and `while`

- Potentially useful:
  - `on_beeper()`
  - `not`
What if some corners already have beepers?

- Combine what we know about `if` and `while`!
- Potentially useful:
  - `on_beeper()`
  - `not`

Write your code in the `MakeBeeperRowKarel.py` file in the `pycharm_intro` folder you downloaded.
Common bugs
Common bugs

- Forgetting not
Common bugs

● Forgetting not

● Forgetting an if statement around the first `put_beeper()`
Common bugs

- Forgetting **not**

- Forgetting an if statement around the first **put_beeper()**

- Forgetting to call your function!
Common bugs

- Forgetting not
- Forgetting an if statement around the first `put_beeper()`
- Forgetting to call your function!
- Getting stuck in an infinite loop
Common bugs

- Forgetting not
- Forgetting an if statement around the first `put_beeper()`
- Forgetting to call your function!
- Getting stuck in an infinite loop

**Definition**

**infinite loop**
When your code gets stuck inside a loop because the condition never evaluates to false
Common bugs

● Forgetting *not*

● Forgetting an if statement around the first `put_beeper()`

● Forgetting to call your function!

● Getting stuck in an infinite loop
While loop summary (and misconceptions)
while loop summary

● **All lines** inside the while loop will run if the condition is true

● Lines inside the while loop happen **one at a time** and **in order**

● The condition is checked again **only after all of the lines have run**

● It is possible for the while loop body to run 0 times!
  ○ Just like with if statements

● Beware of infinite loops!