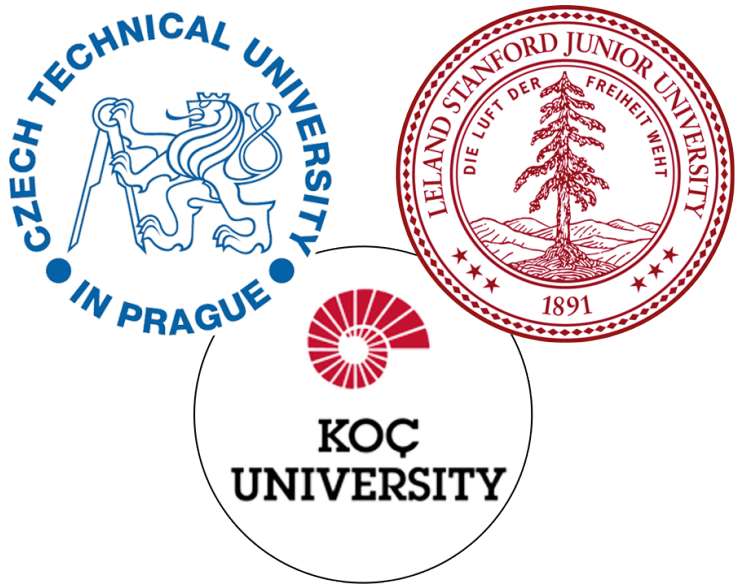
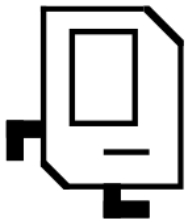
CS Bridge, Lecture 1

Welcome to CS Bridge!



Lecture Plan

- Welcome to CS Bridge!
- Meet Karel the Robot
- For Loops



CS Bridge on Zoom

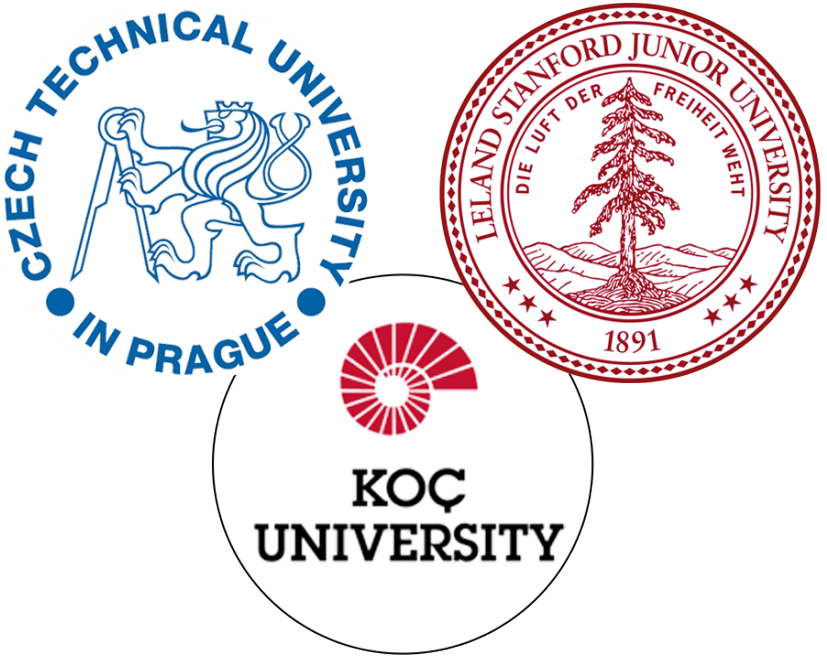
- You are encouraged to share video!
- Post questions/comments/followups in the Zoom chat. We'll take periodic "question breaks" to address them, and have teachers monitoring the chat during the lecture. Everyone is muted by default.

Lecture Plan

- **Welcome to CS Bridge!**
- Meet Karel the Robot
- For Loops

Welcome to CS Bridge!

- We are here to share with you our love for programming!
- An exciting experiment about online teaching
- A joint effort
- **Our goal:** form a community of people to learn and teach programming together



Course Website

online.csbridge.org

Ed Forum

- We are using a website called Ed for asking and answering questions.
- You can post questions there, and we or other students can answer them.
- If you haven't already, check your email for your invite to join Ed.
- **2 Ed Groups:**
  - **CS Bridge (main group)**
  - **Your Section**

Bridging The World

Nice to meet you!

Hi! I’m Nick.

Nice to meet you!

Nick T.

Ondra

Ayca

Chris

Ceren

Nice to meet you!


Nick T.


Ondra

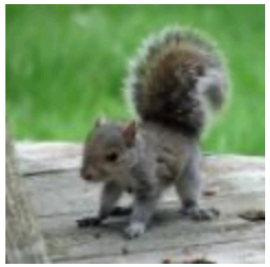

Ayca


Chris


Nick M.


Josef


Yosefa


Burcu


Lisa


Ceren


Ahmet


Anna

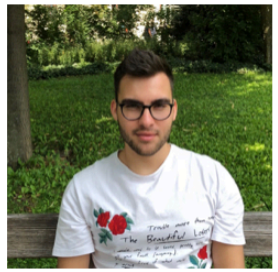

Demet


Eddie


Eliška


Emilia


Emirhan


Eren

Nice to meet you!


Ezgi


Gaby


Gül Sena


Honza


Jan


Jaroslav


Jillian

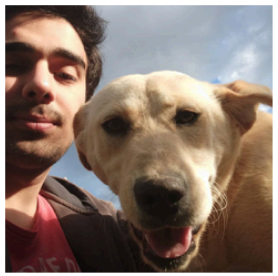

Julie


Melis

Necla


Okan


Oya


Ozan


Pinar

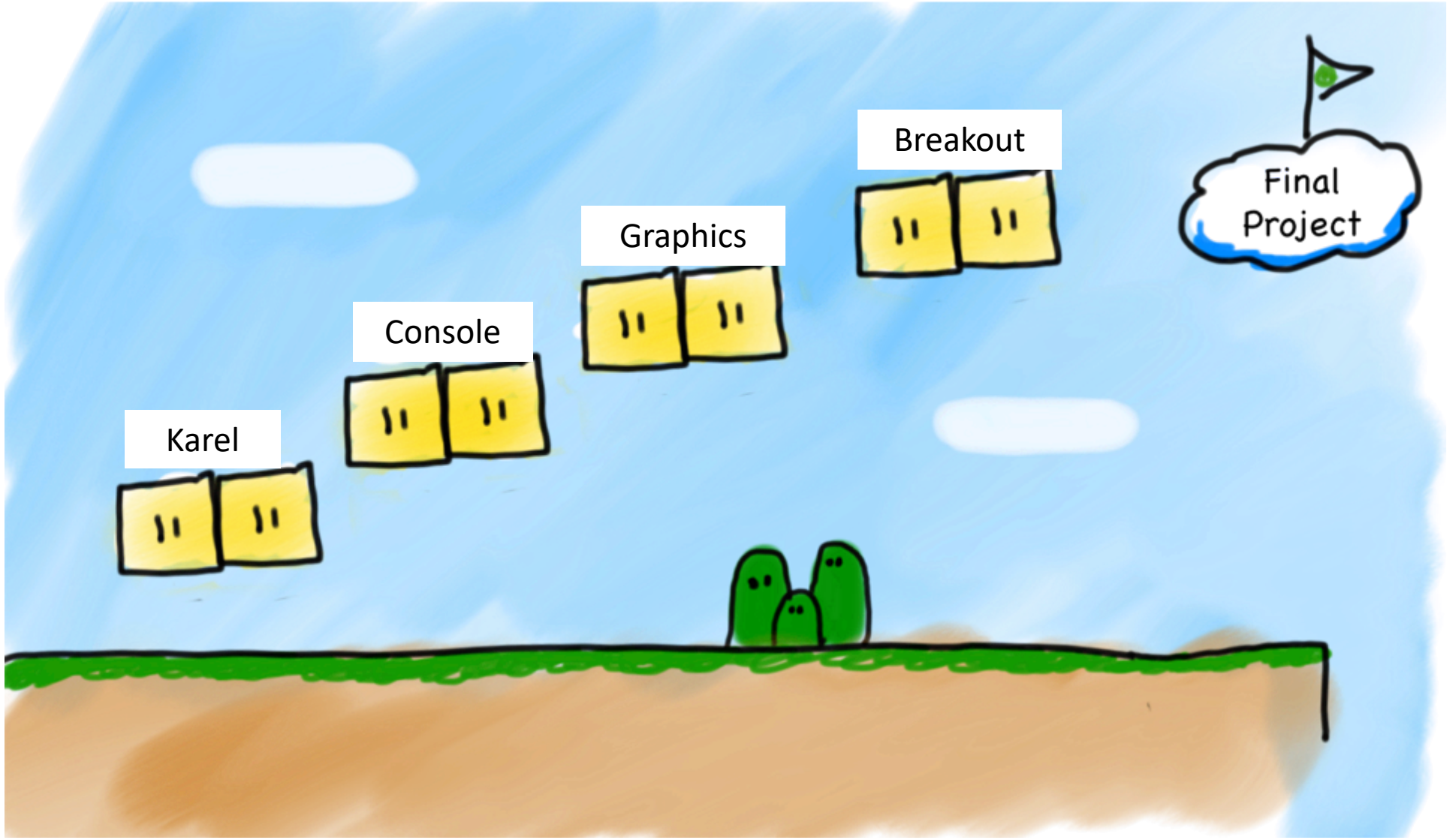

Radek

Section Leader

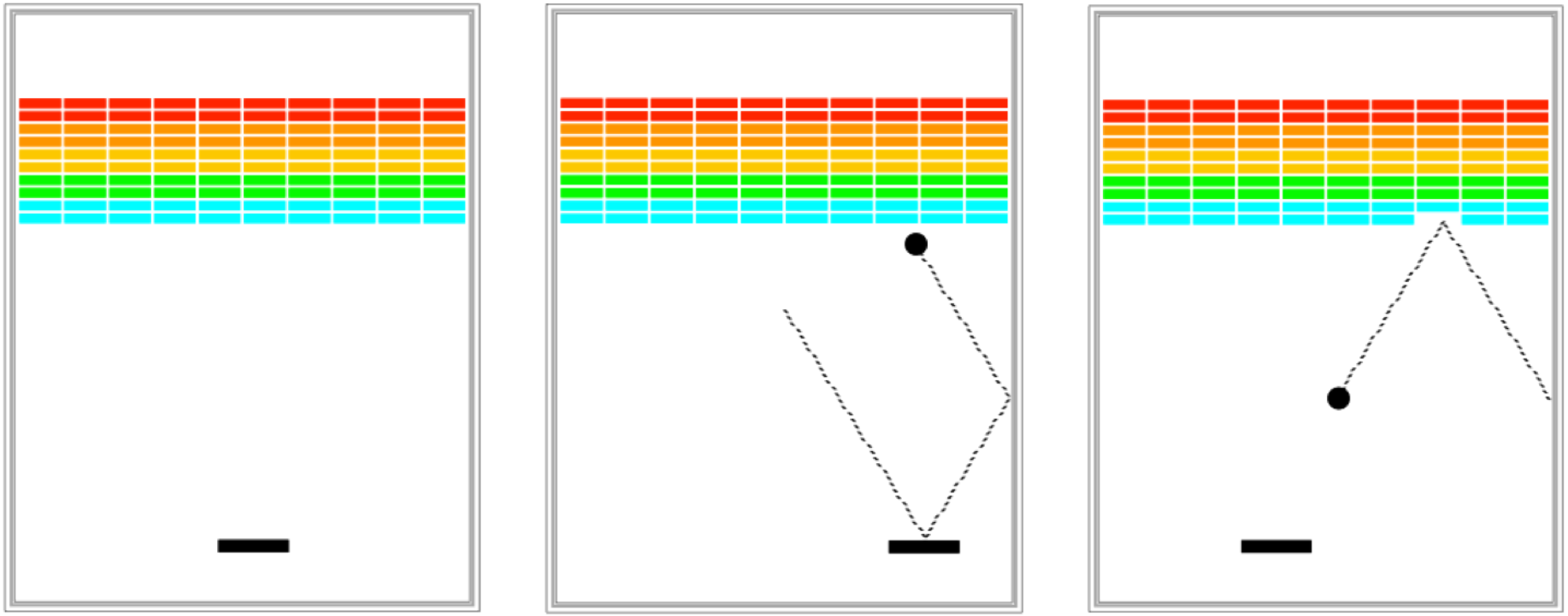
Section Leaders will…

- Lead your daily section and quickstart times
- Help you when you have questions in Office Hours and on Ed
- And much more…

What Will We Learn?



Breakout



Prerequisites

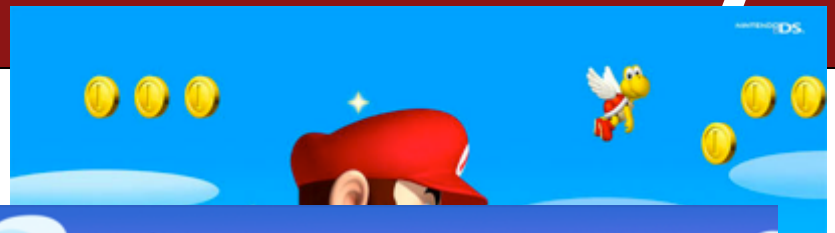
- None! We assume you have never programmed before.

Art Of Computer Science

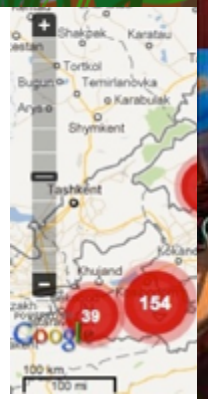
- CS Bridge covers software engineering principles – much more than just programming
- Writing a program is like writing an essay
  - Need a language (Python)
  - But just knowing the language doesn't make you a good essay writer
- Programming is a tool you can use to do amazing things!

The Joy Of Building

The Joy C




ANGRY BIRDS
ROVIO


Shakpak
Karatau
Tortkol
Temirlanovka
Karabulak
Shymkent
Tashkent
Khujand
39
154

Walt Disney Pictures
presents
TOY
STORY

Strive For Everyone To Succeed

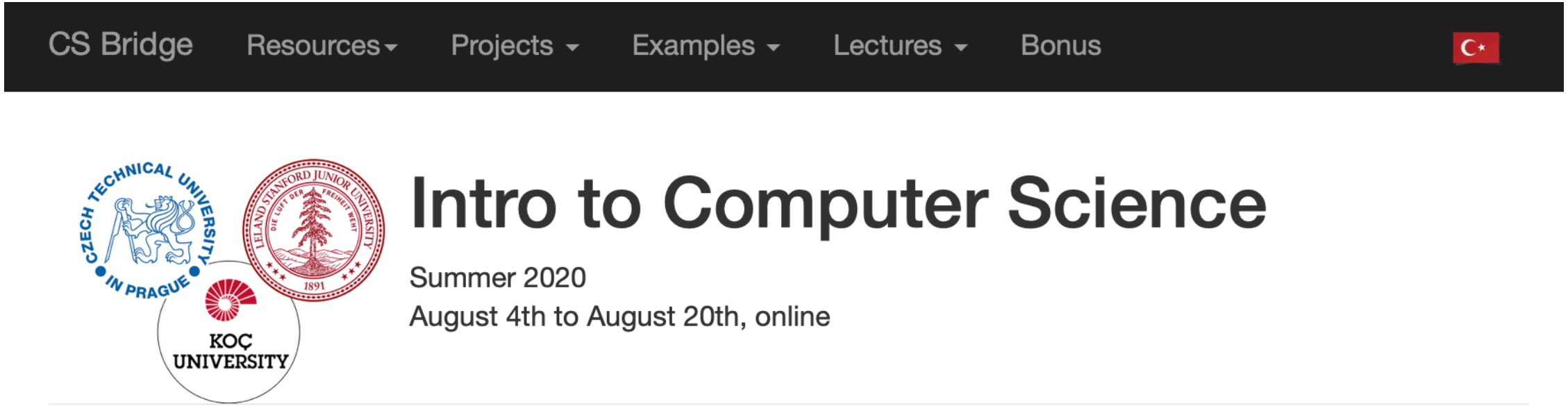
- We want all of you to be successful!
- You are not competing against anyone but yourself.
- Learning programming is hard! We are here to help you. Please post on Ed or talk to your Section Leader if you are having trouble.
- **Your most important task**: show up!

CS Bridge Structure – Learn By Doing

- **Lecture** – Where we introduce new concepts!
- **Quickstart / Tea Time** – Right after each lecture with your section group. Time for you to play around with the material on your own computers and mingle with other students.
- **Section** – Right after evening quickstart, with your section group. You will work as a group with your Section Leader to solve practice problems.
- **Office Hours / Work** – Right after the morning quickstart and the evening section. Your own time to work on the projects and sign up for office hours with the Section Leaders if you need help.

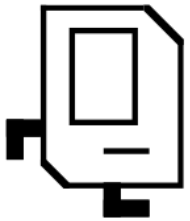
At the end of the morning and evening segments, submit your work!

The Website Tells You Where To Go!



Lecture Plan

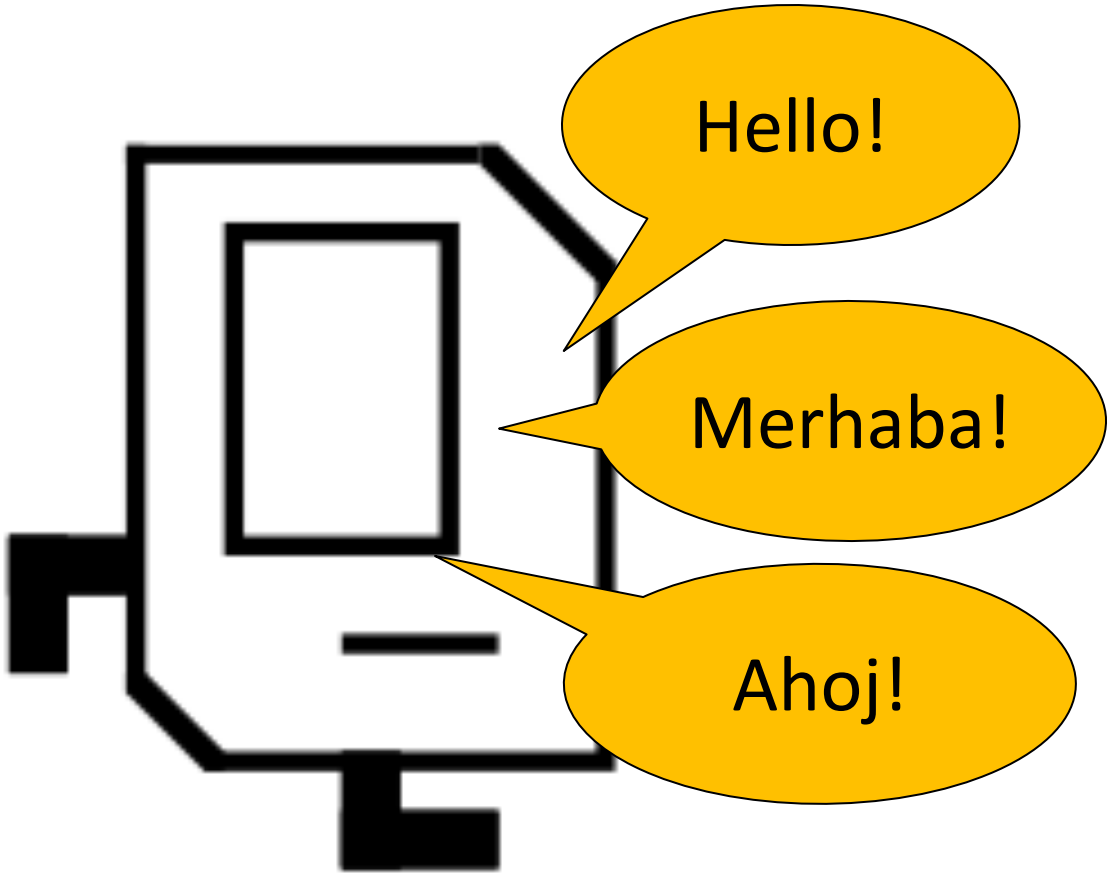
- Welcome to CS Bridge!
- **Meet Karel the Robot**



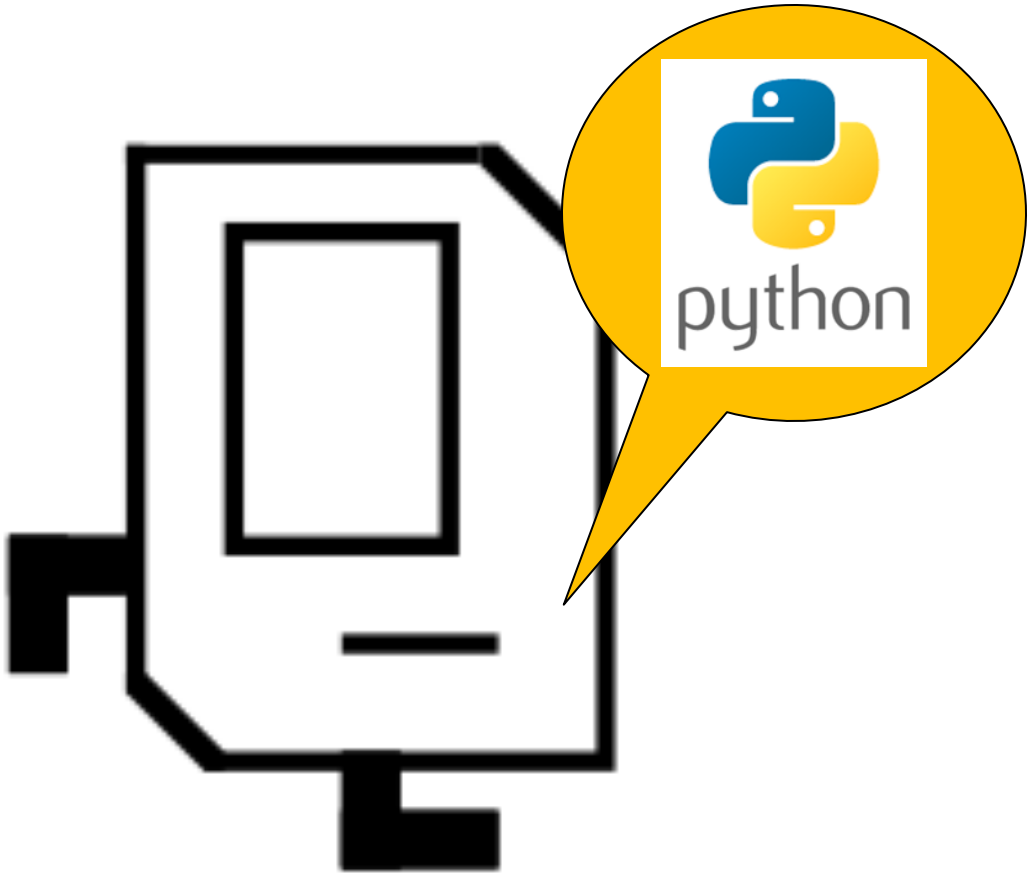
- For Loops

Meet Karel the Robot!

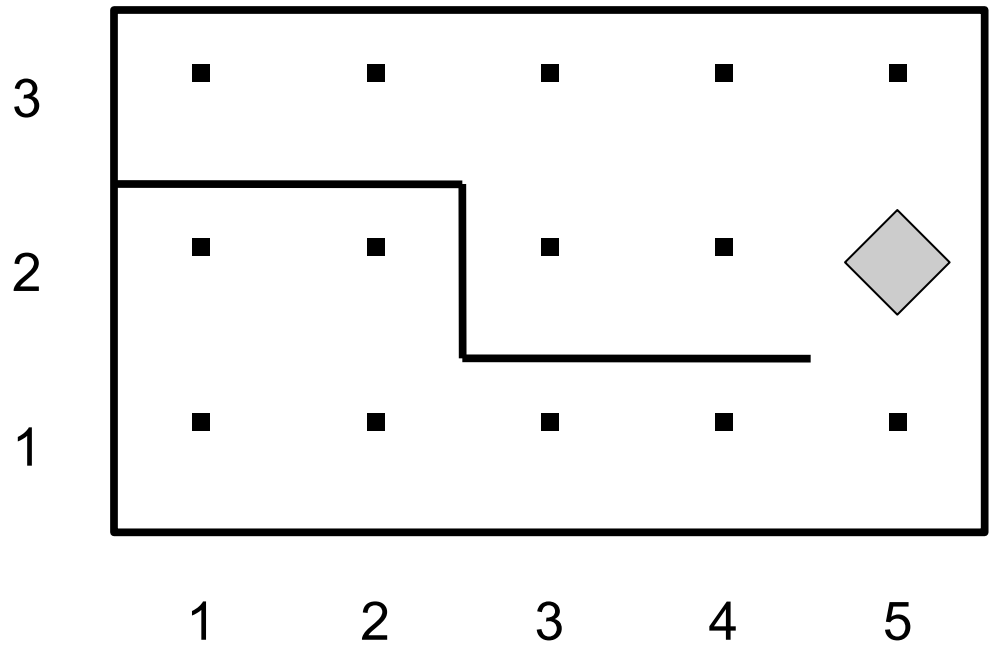
Meet Karel the Robot!



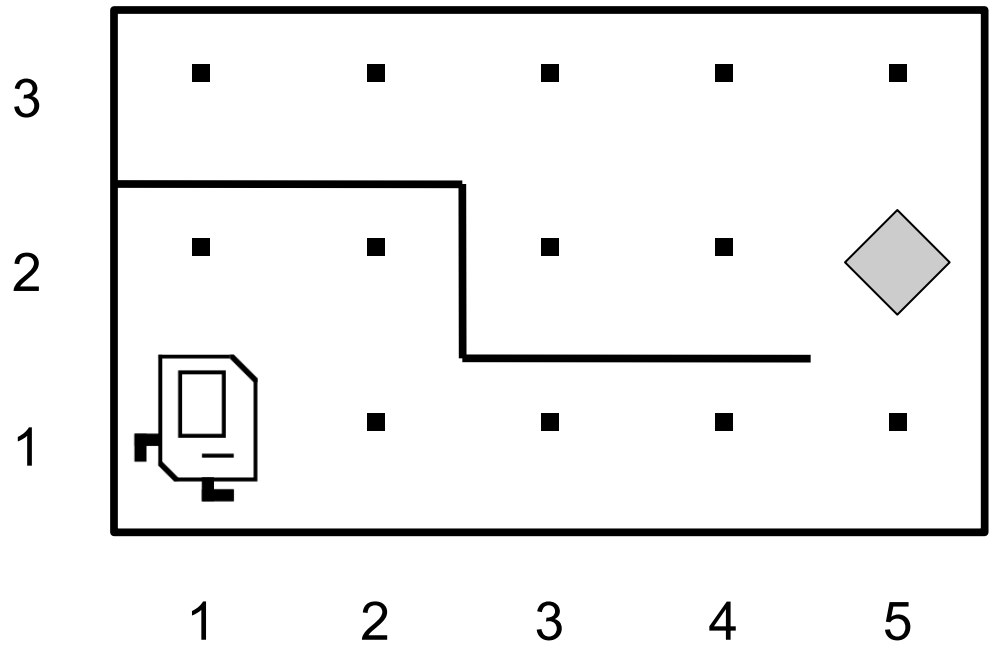
Meet Karel the Robot!



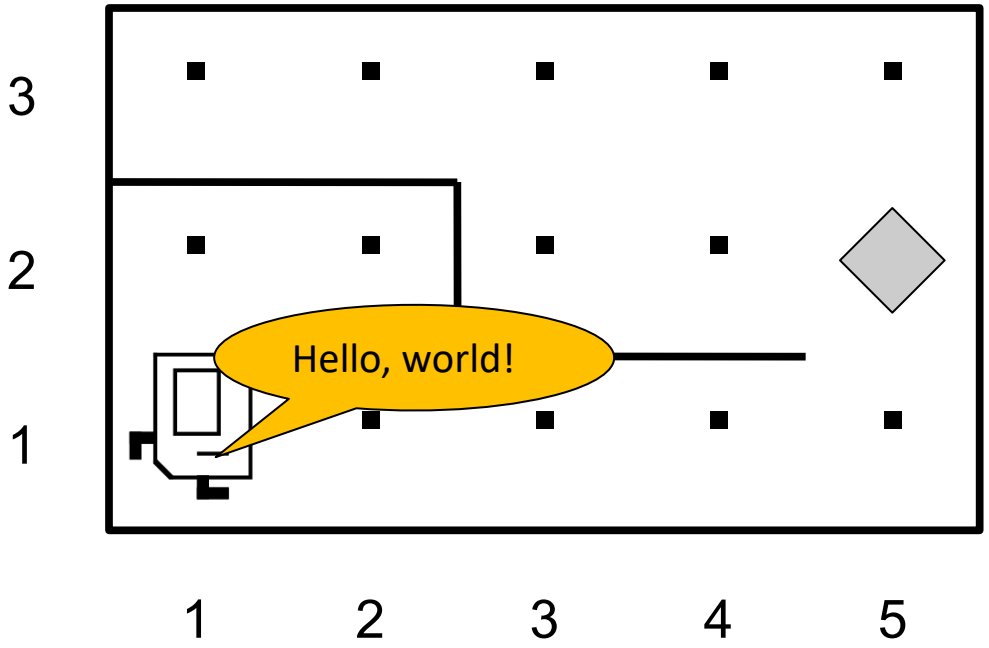
Karel's World



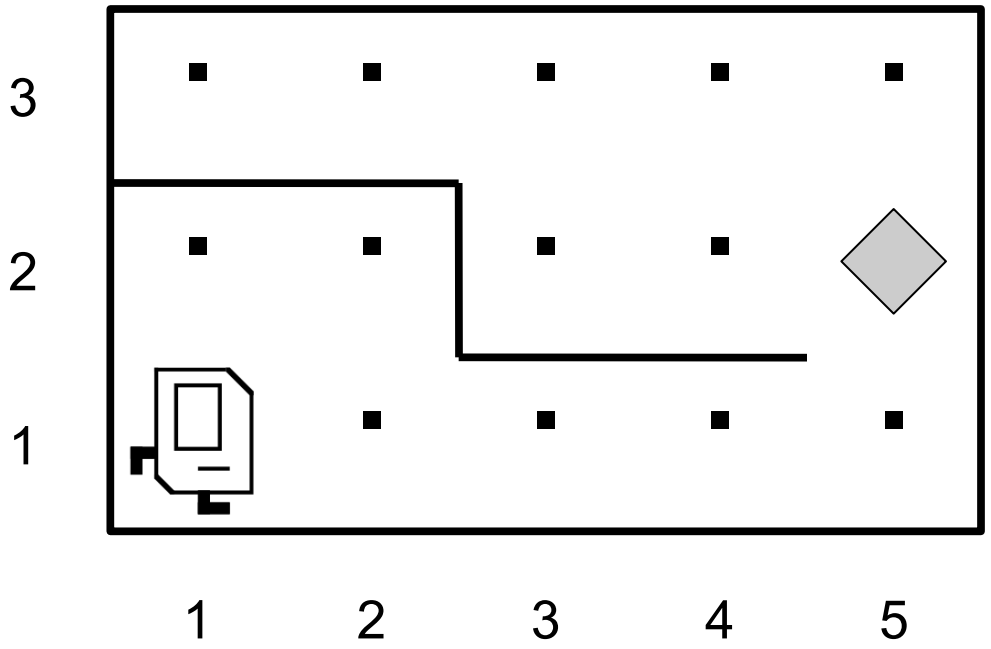
Karel's World



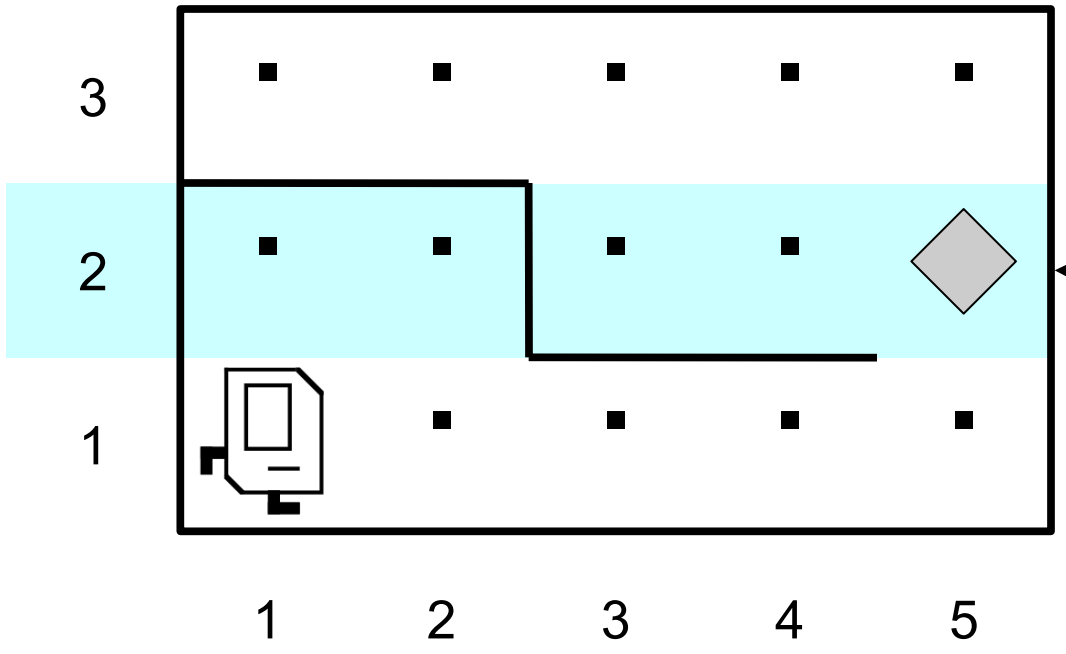
Karel's World



Karel's World

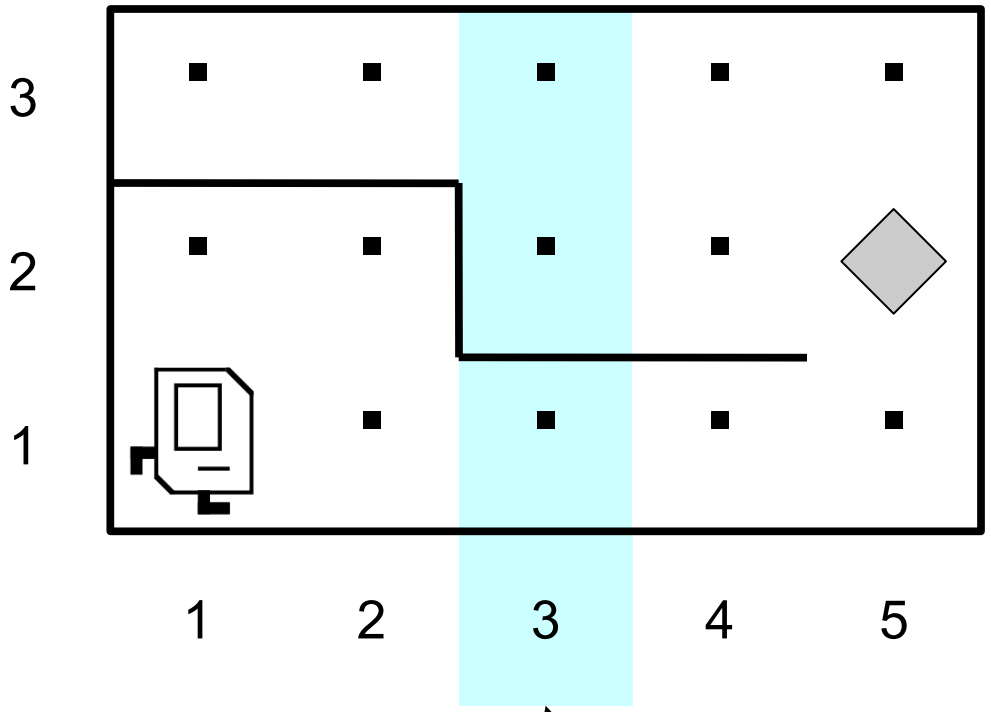


Streets (rows)



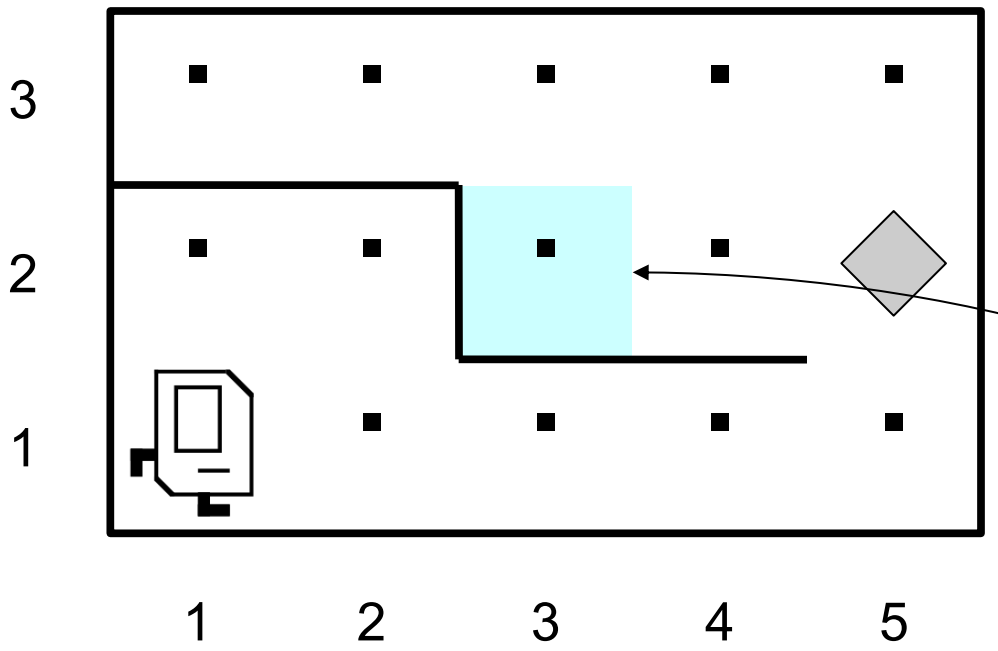
Each row is called a street.

Avenues (columns)



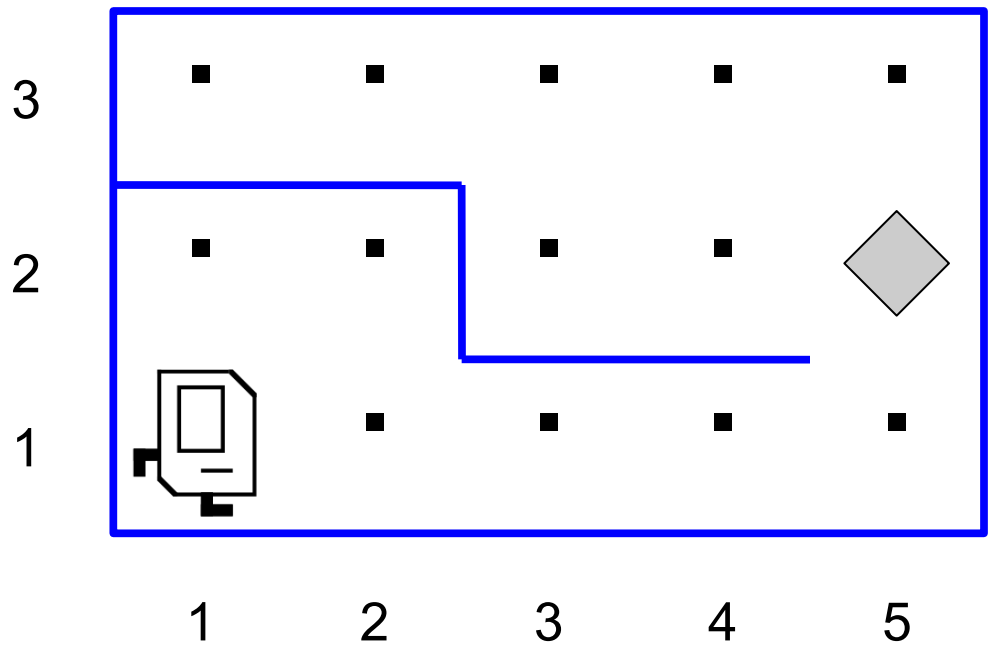
Each column is called an avenue.

Corners (locations)



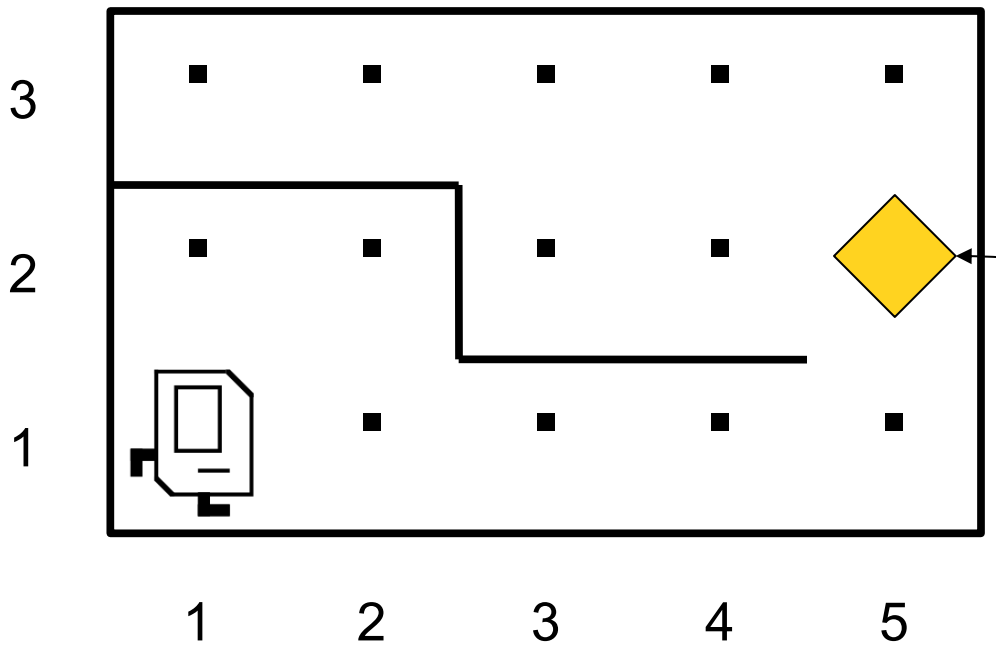
The intersection of a street and an avenue is a corner.

Walls



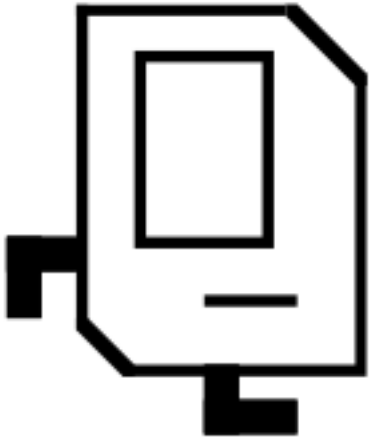
Karel cannot move through walls.

Beepers



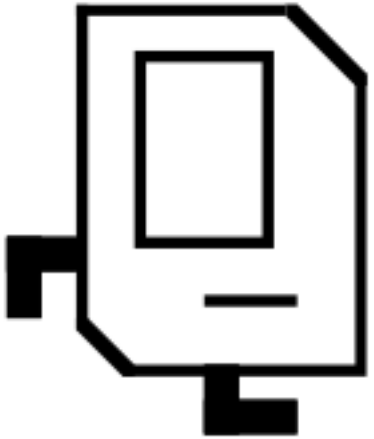
Beepers mark locations in Karel's world. Karel can pick them up and put them down.

Karel Knows 4 Commands



```
move

turn_left

put_beeper

pick_beeper
```

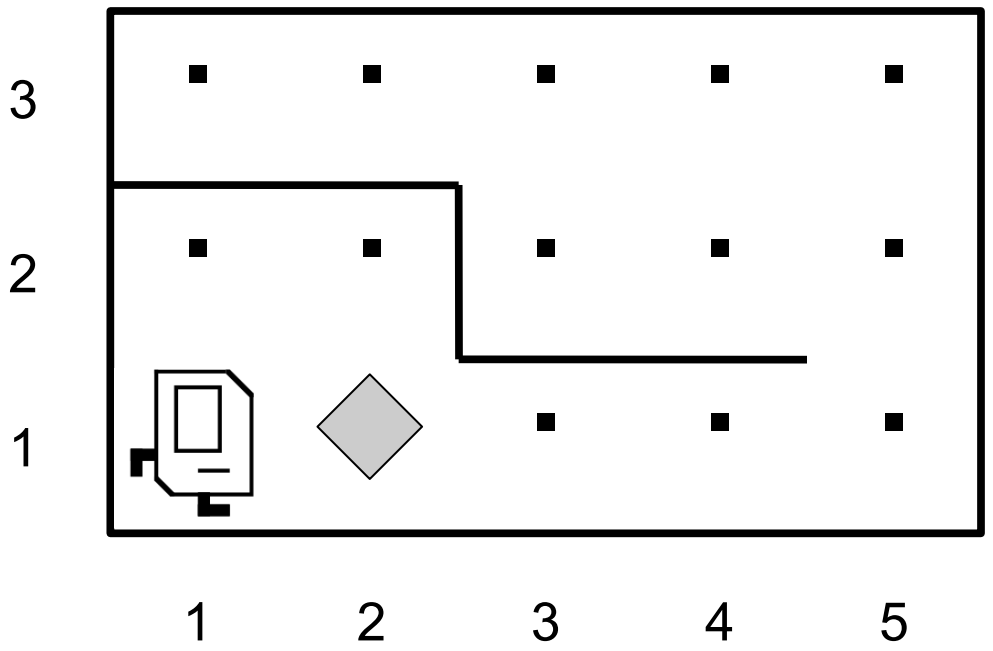
Karel Knows 4 Commands



```
move
turn_left
put_beeper
pick_beeper
```

"functions"

Commands: move



Karel Commands

move

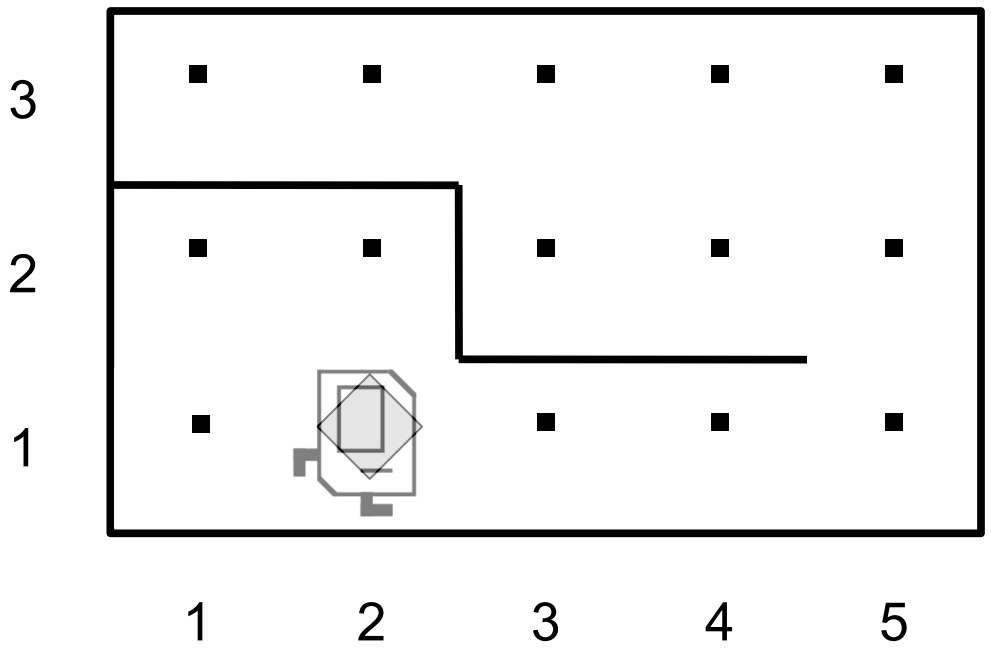
turn_left

pick_beeper

put_beeper

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

Commands: move



Karel Commands

move

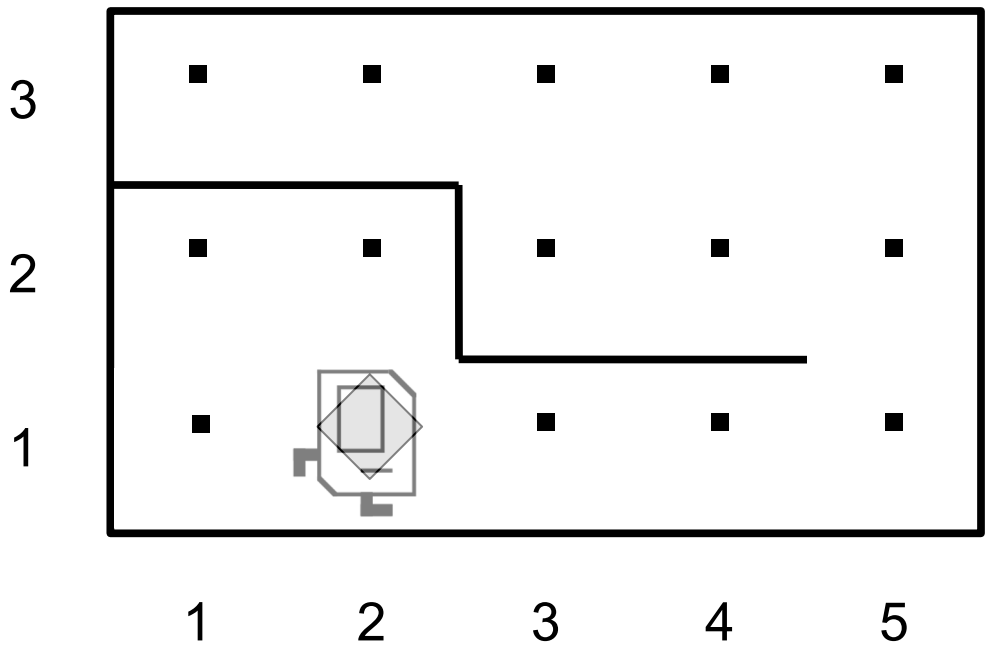
turn_left

pick_beeper

put_beeper

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

Commands: turn_left

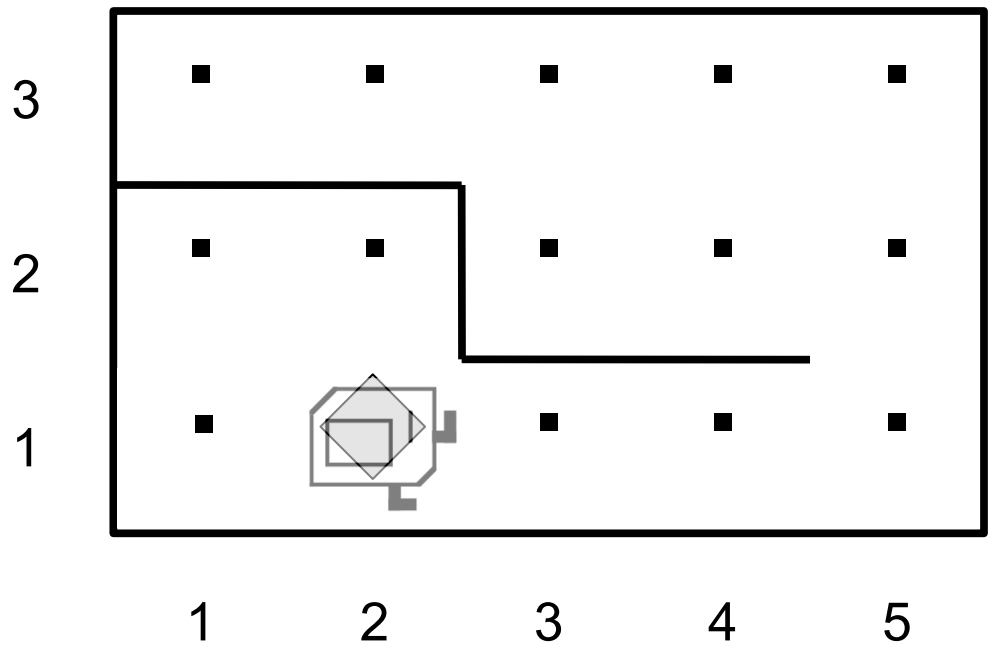


Karel Commands

```
move
turn_left
pick_beeper
put_beeper
```

- `turn_left` makes Karel rotate 90° counter-clockwise.
- There is no `turn_right` command. (Why not?)

Commands: turn_left

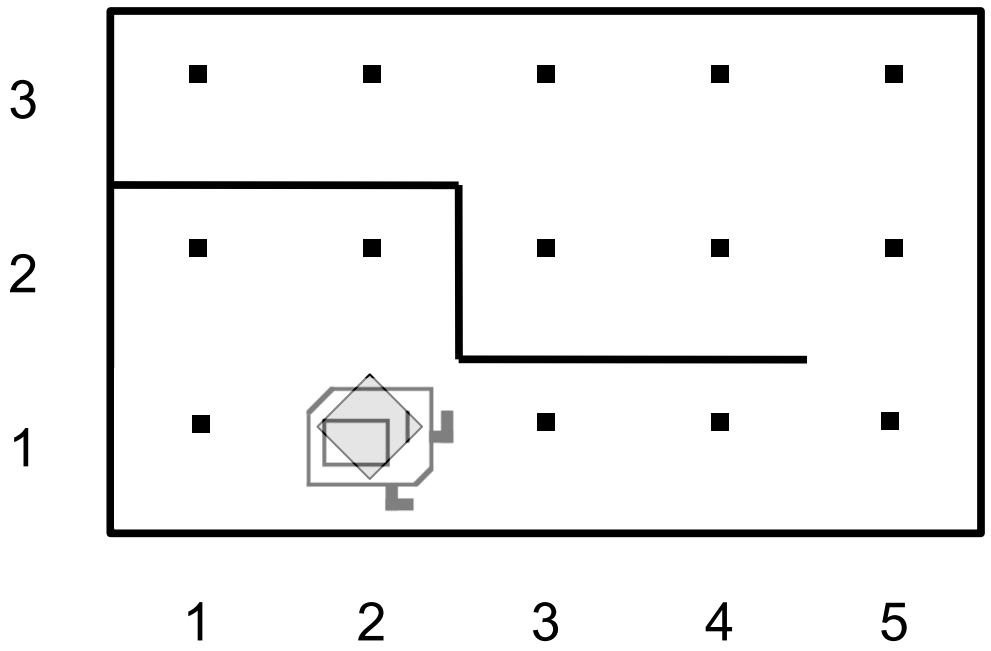


Karel Commands

```
move
turn_left
pick_beeper
put_beeper
```

- `turn_left` makes Karel rotate 90° counter-clockwise.
- There is no `turn_right` command. (Why not?)

Commands: pick_beeper

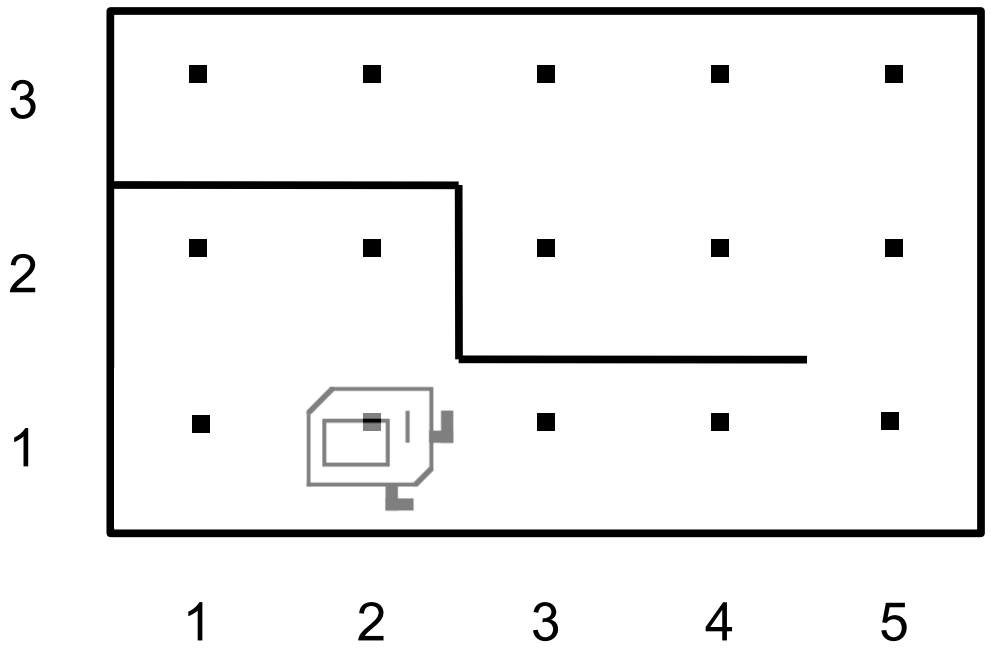


Karel Commands

move
turn_left
pick_beeper
put_beeper

`pick_beeper` makes Karel pick up the beeper at the current corner. Karel can hold multiple beepers at a time in its "beeper bag".

Commands: pick_beeper

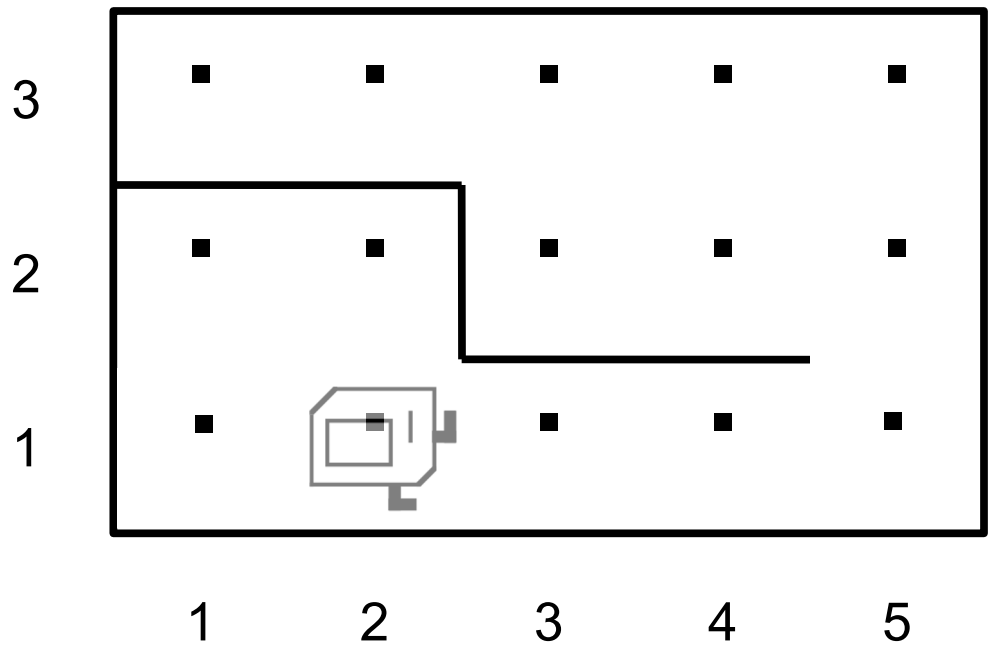


Karel Commands

```
move
turn_left
pick_beeper
put_beeper
```

`pick_beeper` makes Karel pick up the beeper at the current corner. Karel can hold multiple beepers at a time in its "beeper bag".

Commands: put_beeper



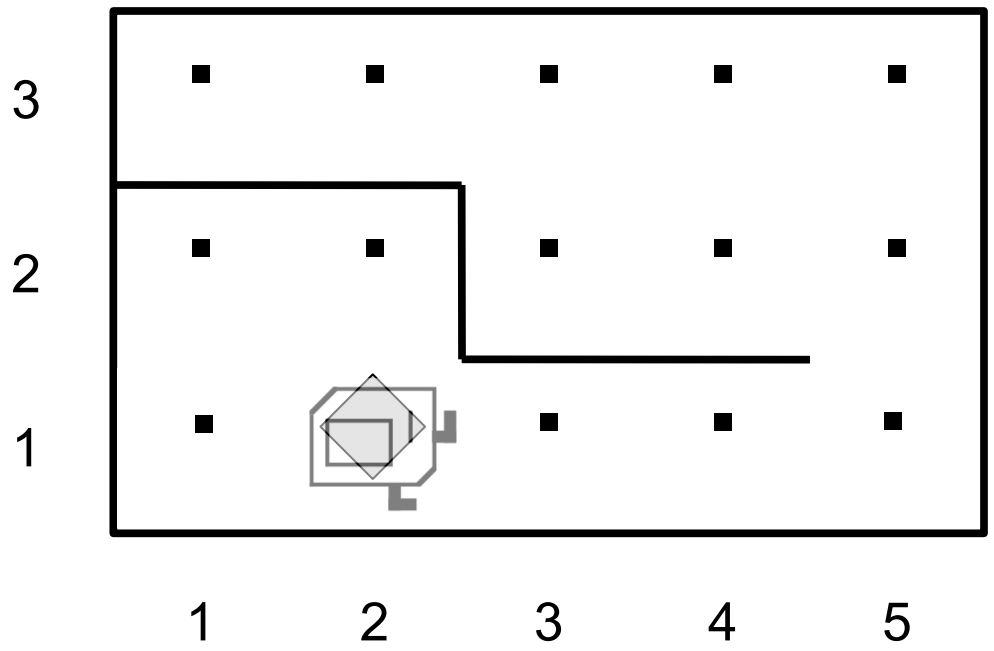
Karel Commands

```
move
turn_left
pick_beeper
put_beeper
```

`put_beeper` makes Karel put a beeper down at its current location.

- `pick_beeper` and `put_beeper` are used to move beepers around.

Commands: put_beeper



Karel Commands

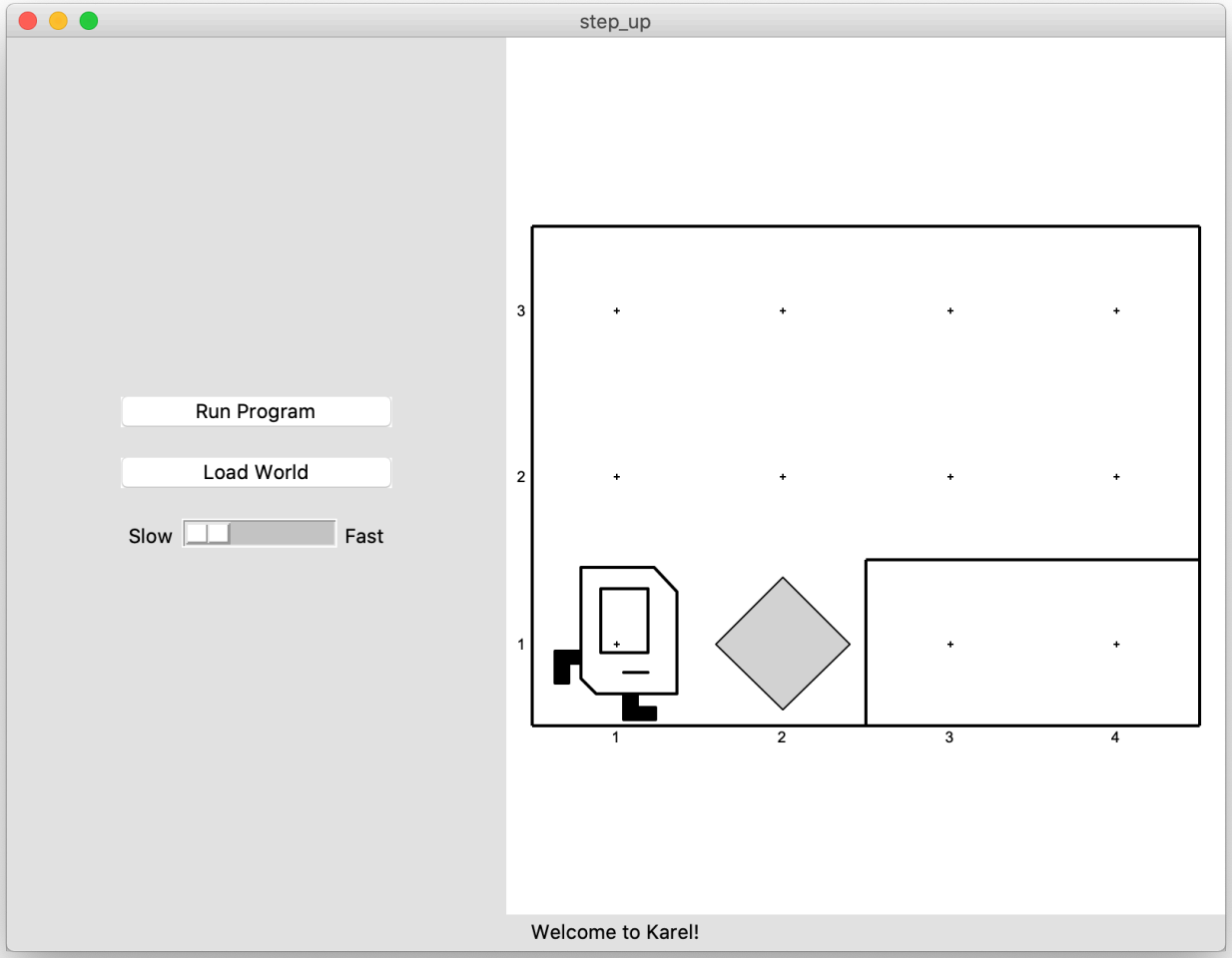
```
move
turn_left
pick_beeper
put_beeper
```

`put_beeper` makes Karel put a beeper down at its current location.

- `pick_beeper` and `put_beeper` are used to move beepers around.

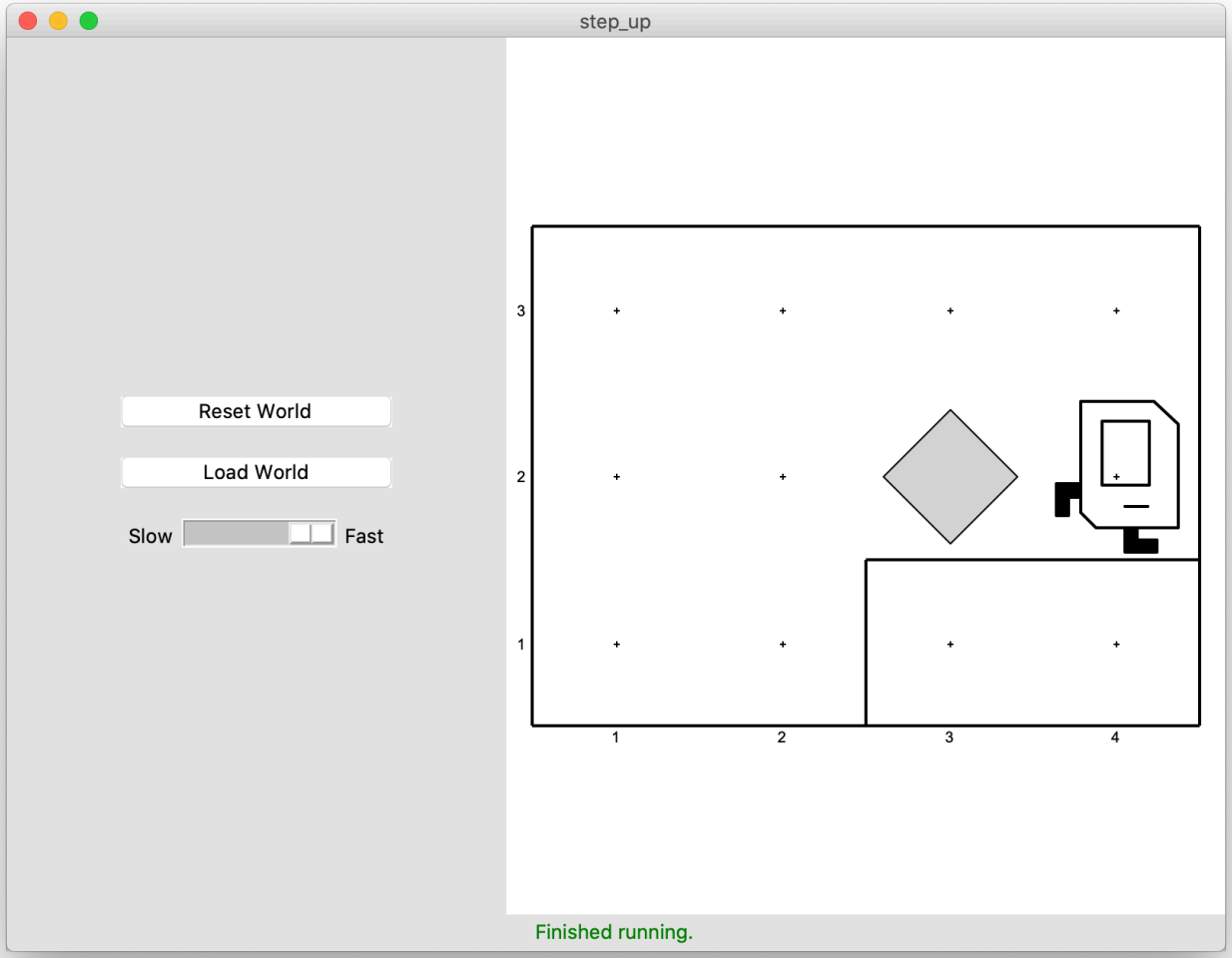
Our First Karel Program

Before:



Our First Karel Program

After:



Demo

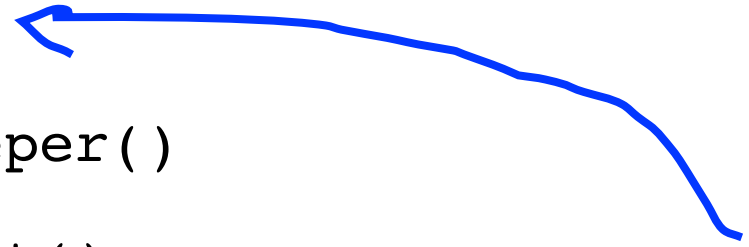
Anatomy Of A Program

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


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

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

Anatomy Of A Program

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


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



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

Anatomy Of A Program

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

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

This is called a ***code block.*** Python requires that code blocks be indented one level.

Defining New Commands

We can make new commands (or **functions**) for Karel. This lets us *decompose* our program into smaller pieces that are easier to understand.

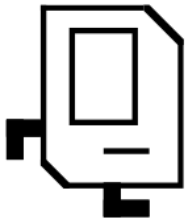
```
def name():
    statement
    statement
    ...
```

For example:

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

Lecture Plan

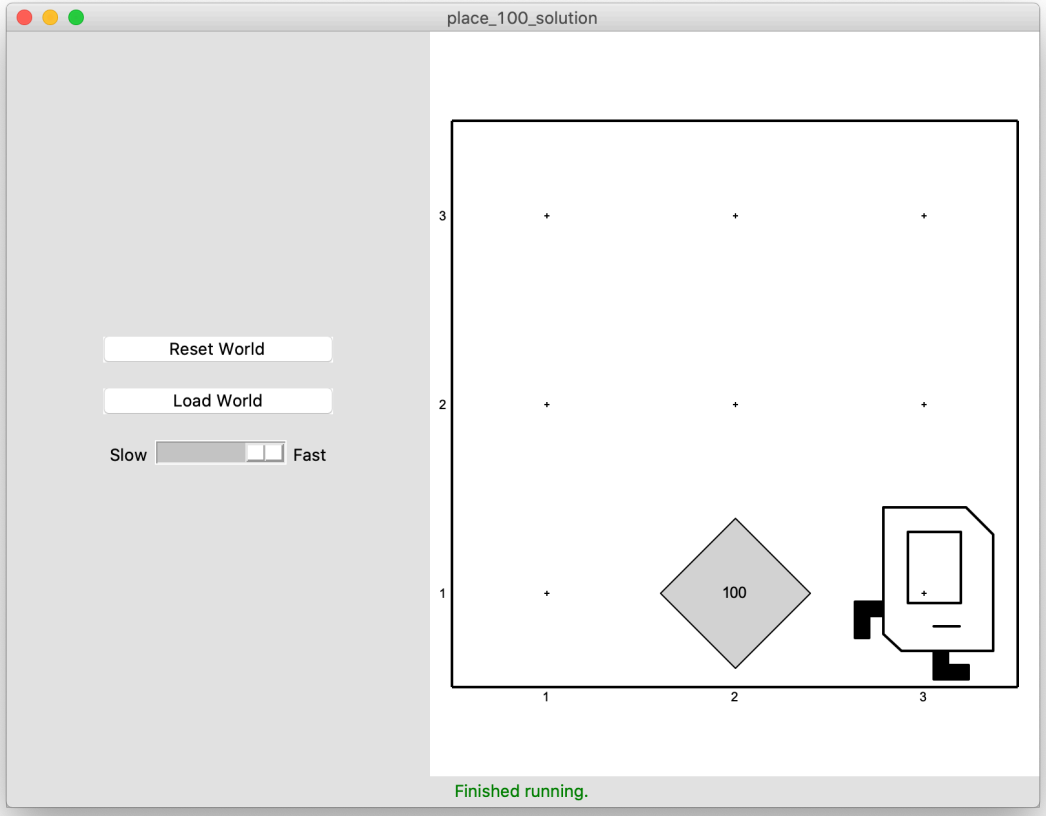
- Welcome to CS Bridge!
- Meet Karel the Robot



- **For Loops**

For Loops

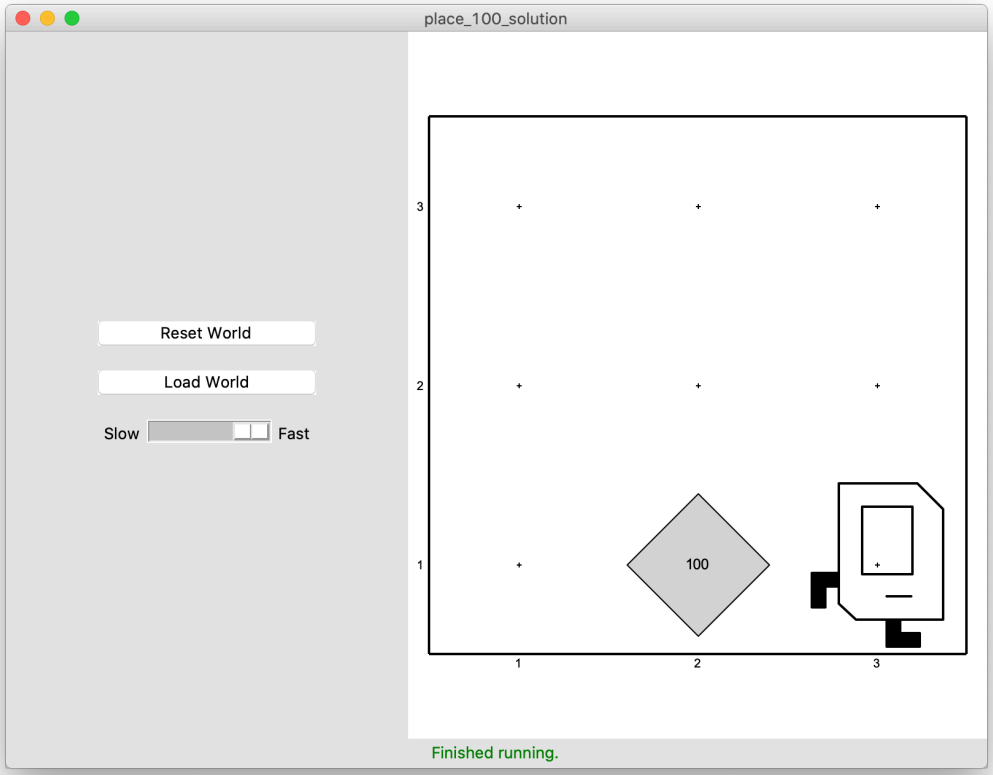
I want to make Karel put 100 beepers down on a corner. How do I do this?



For Loops

Can’t just say:

```
move()
put_beeper()
put_beeper()
put_beeper()
...
move()
```



This is too repetitive! Plus, it’s difficult to change (e.g. to 25 beepers).

For Loops

Instead, use a **for** loop:

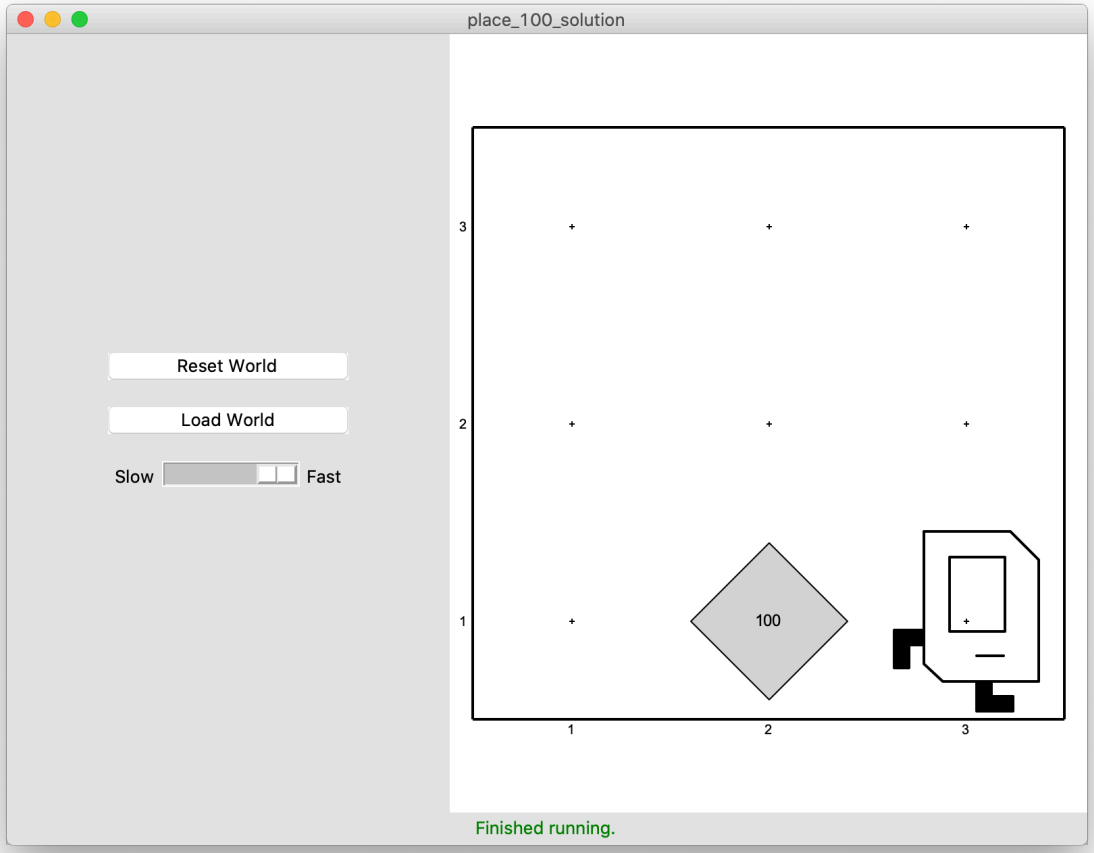
```
for i in range(count):
    statement
    statement
    ...
```

Repeats the statements in the body ***count*** times.

For Loops

Now we can say:

```
move()
for i in range(100):
    put_beeper()
move()
```



This is less repetitive and is easier to change (e.g. to 25 beepers).

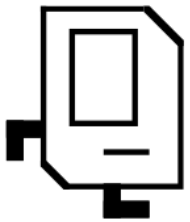
For Loops and Indentation

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

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

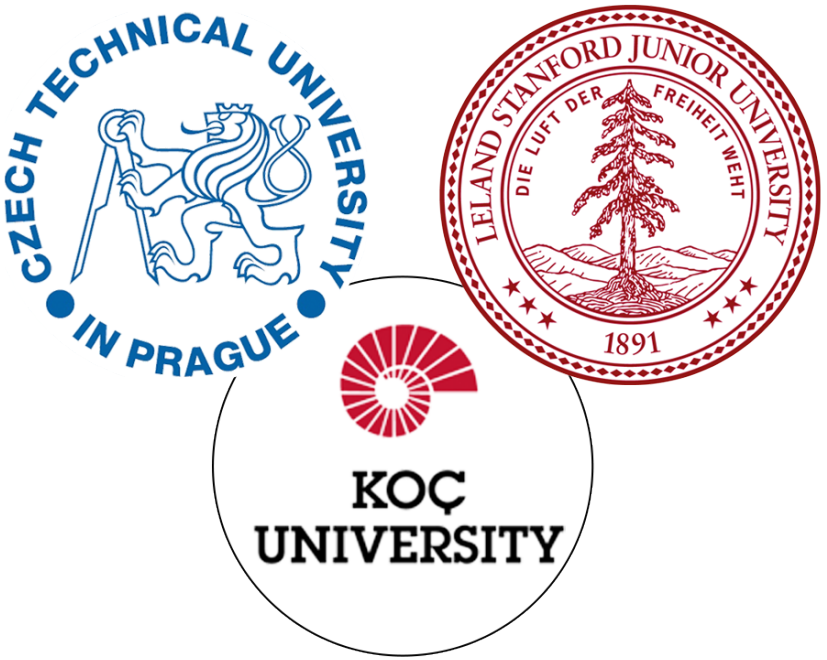
Lecture Recap

- Welcome to CS Bridge!
- Meet Karel the Robot
- For Loops



Welcome to CS Bridge!

- We are here to share with you our love for programming!
- An exciting experiment about online teaching
- A joint effort
- **Our goal:** form a community of people to learn and teach programming together



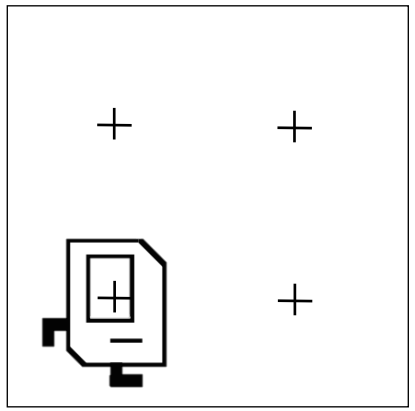
What's Next?

- Time for your section's quickstart time!
- Check your section's Ed group for more information
- We hope you will be able to get set up if you aren't already, and complete at least collect newspaper Karel. Have fun!

Extra: Beeper Square

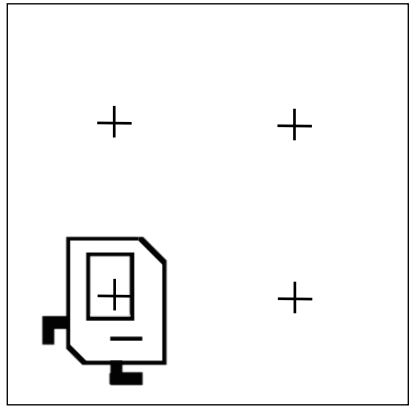
For Loop Example: Beeper Square

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



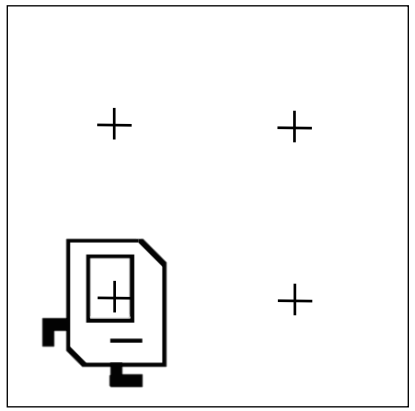
For Loop Example: Beeper Square

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



For Loop Example: Beeper Square

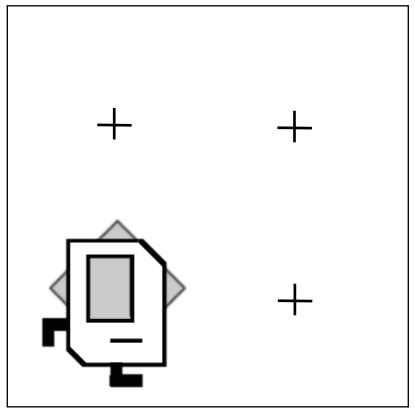
```
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
```



First time through the loop

For Loop Example: Beeper Square

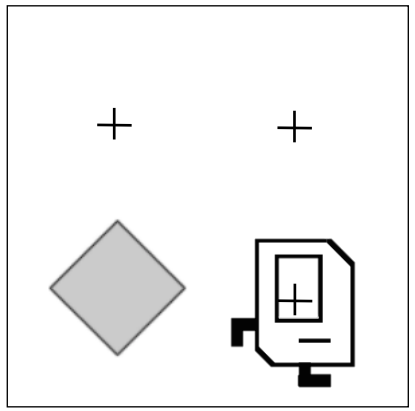
```
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
```



First time through the loop

For Loop Example: Beeper Square

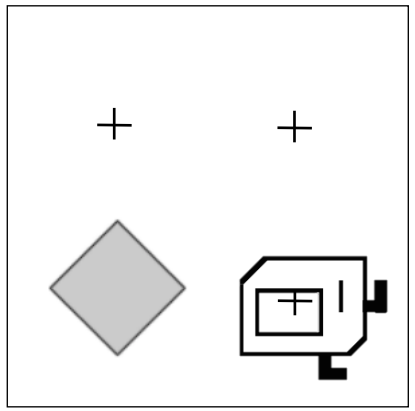
```
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
```



First time through the loop

For Loop Example: Beeper Square

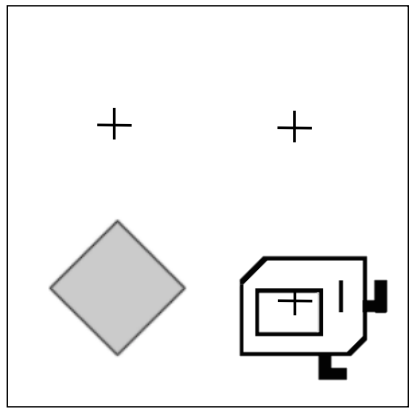
```
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
```



First time through the loop

For Loop Example: Beeper Square

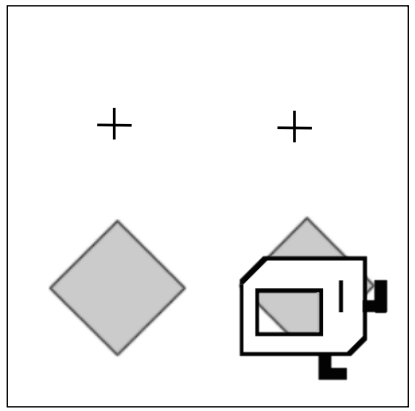
```
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
```



Second time through the loop

For Loop Example: Beeper Square

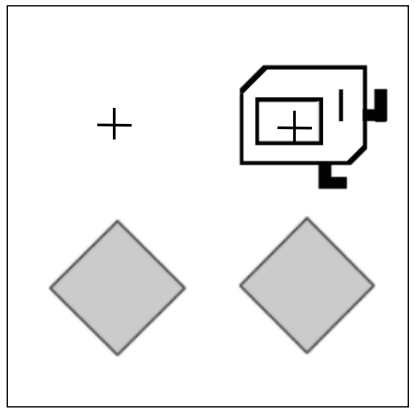
```
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
```



Second time through the loop

For Loop Example: Beeper Square

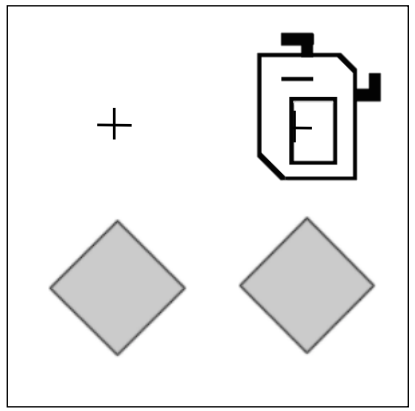
```
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
```



Second time through the loop

For Loop Example: Beeper Square

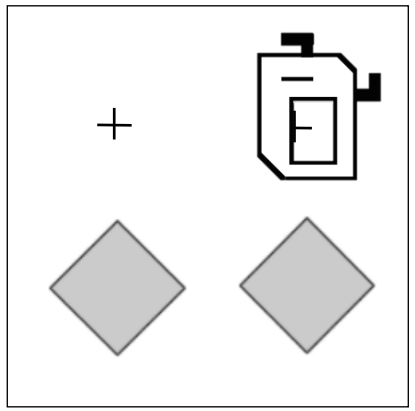
```
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
```



Second time through the loop

For Loop Example: Beeper Square

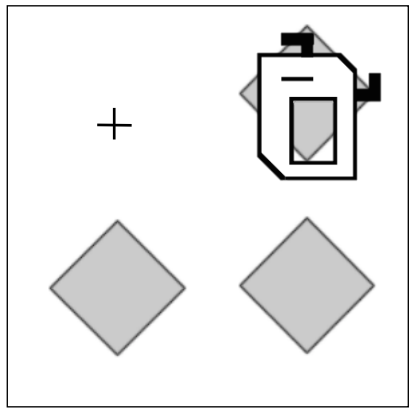
```
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
```



Third time through the loop

For Loop Example: Beeper Square

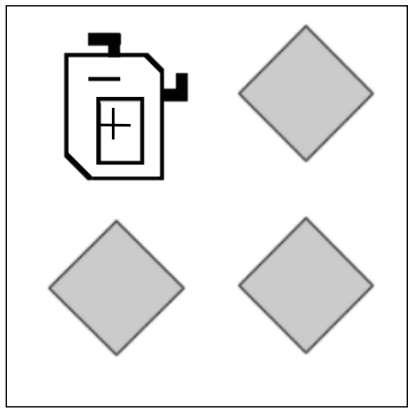
```
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
```



Third time through the loop

For Loop Example: Beeper Square

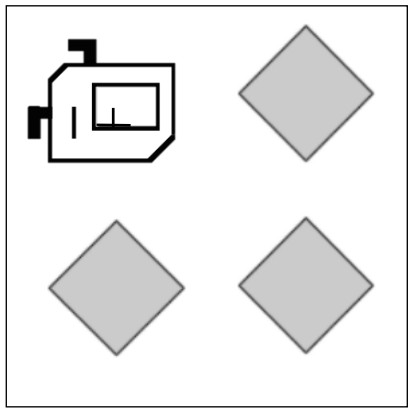
```
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
```



Third time through the loop

For Loop Example: Beeper Square

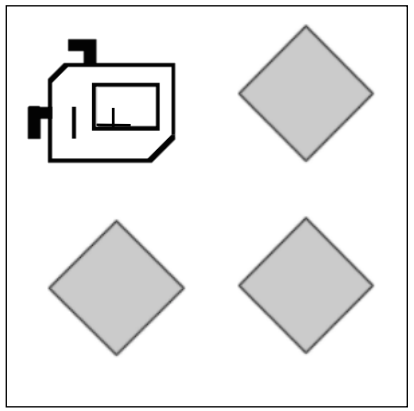
```
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
```



Third time through the loop

For Loop Example: Beeper Square

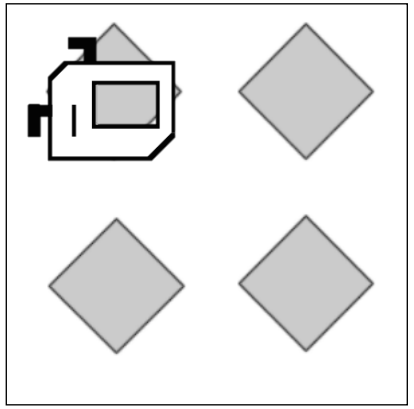
```
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
```



Fourth time through the loop

For Loop Example: Beeper Square

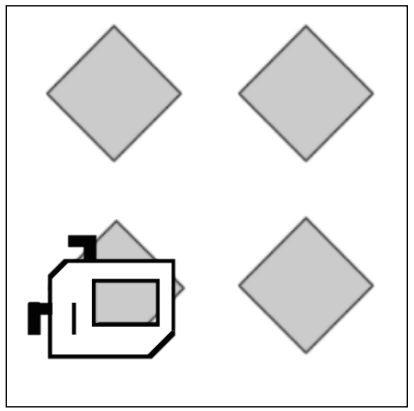
```
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
```



Fourth time through the loop

For Loop Example: Beeper Square

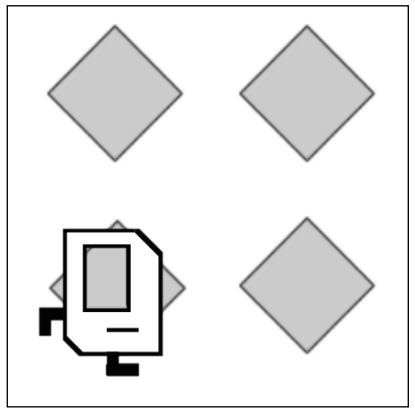
```
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
```



Fourth time through the loop

For Loop Example: Beeper Square

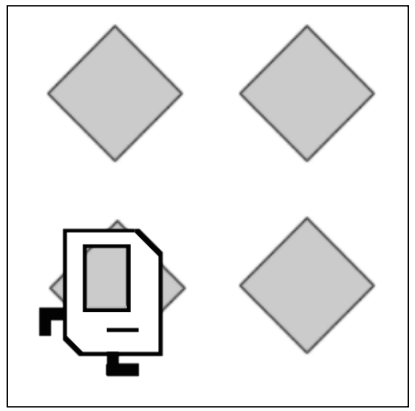
```
def main():
    for i in range(4):
        put_beeper()
        move()
        turn_left()
```



Fourth time through the loop

For Loop Example: Beeper Square

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



Done!