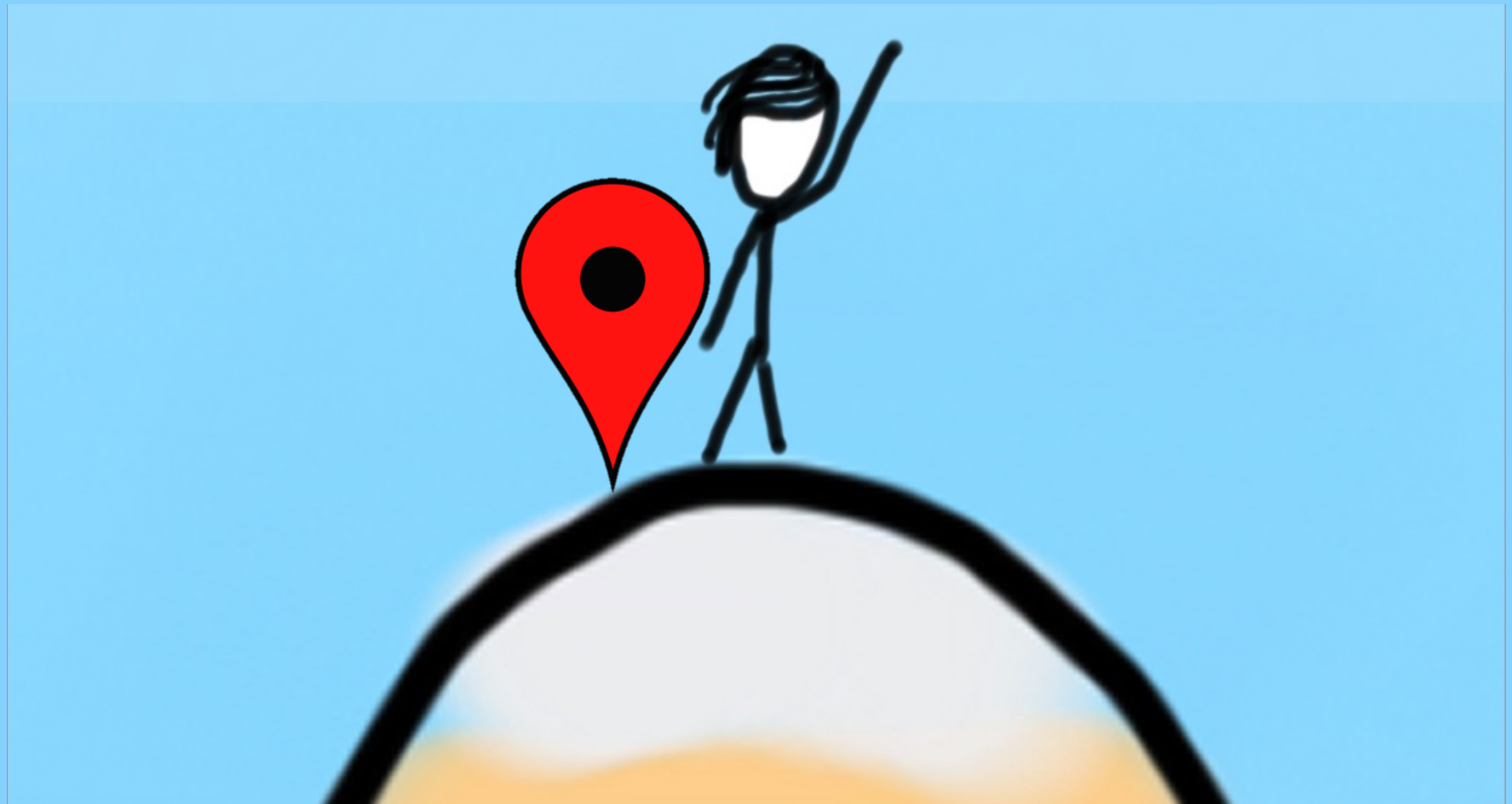
Art of Problem Solving

Chris Piech and Mehran Sahami
CS106A, Stanford University

First, a cool program

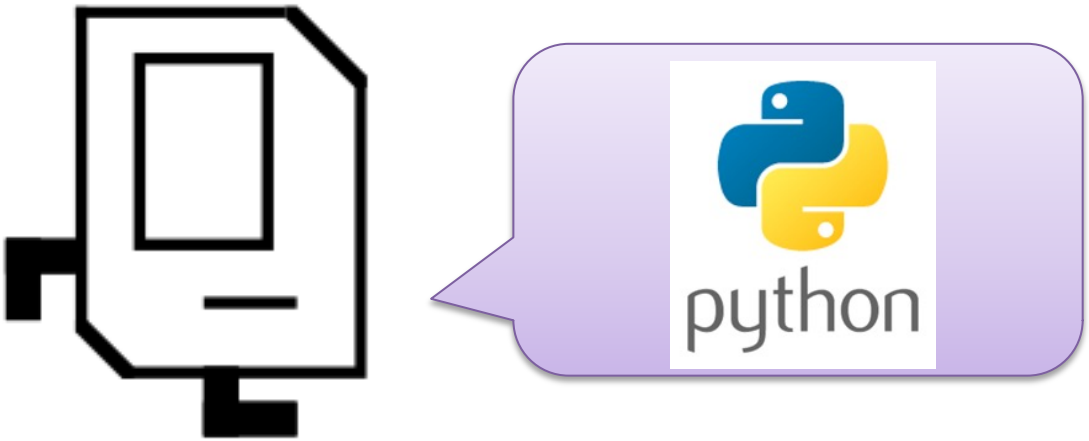
Today's Goal

1. Decomposition: Be able to approach a problem "top down"
2. Stepwise Refinement: Balance testing while decomposing



Quick review

Karel the Robot



Functions

```
def main():
    go_to_moon()


def go_to_moon():
    build_spaceship()
    # a few more steps


def build_spaceship():
    # todo
    put_beeper()
```


For Loops

```
def main():

    # repeats the body 99 times
    for i in range(99):
        # the "body"
        put_beeper()
```

While Loops

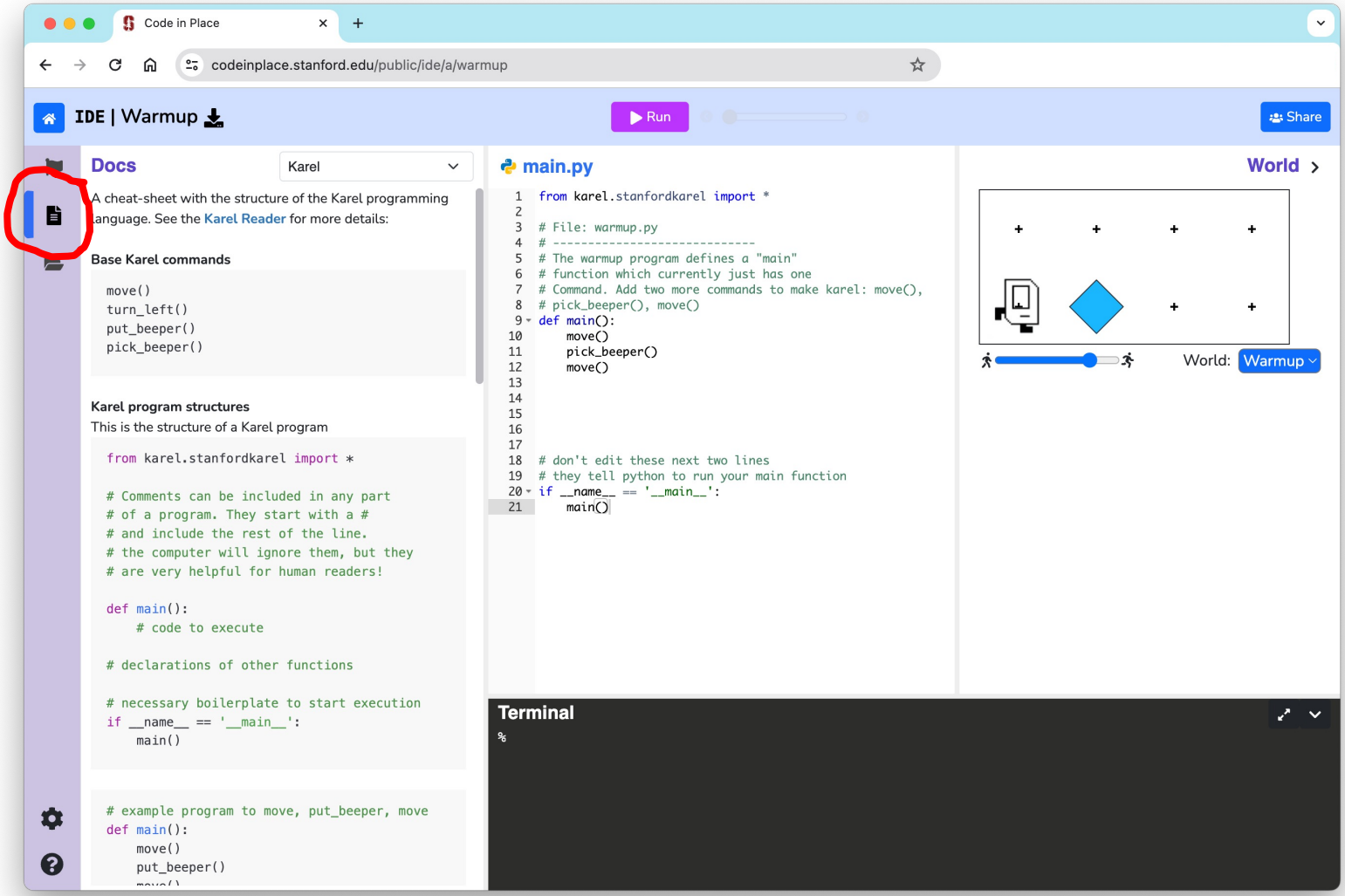
```
def main():

    # while condition holds runs body
    # checks condition after body completes
    while front_is_clear():
        move()
```

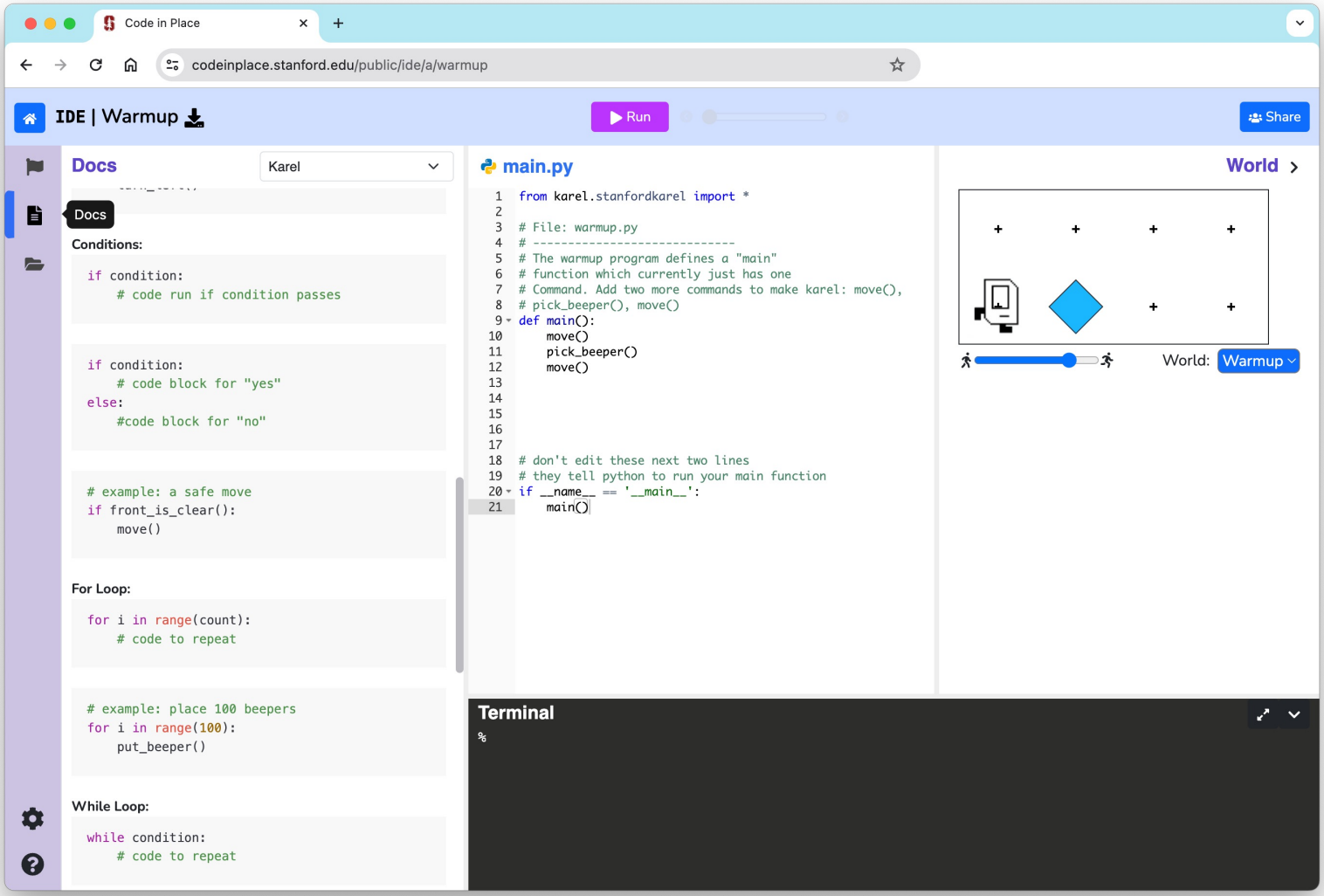

If Statement

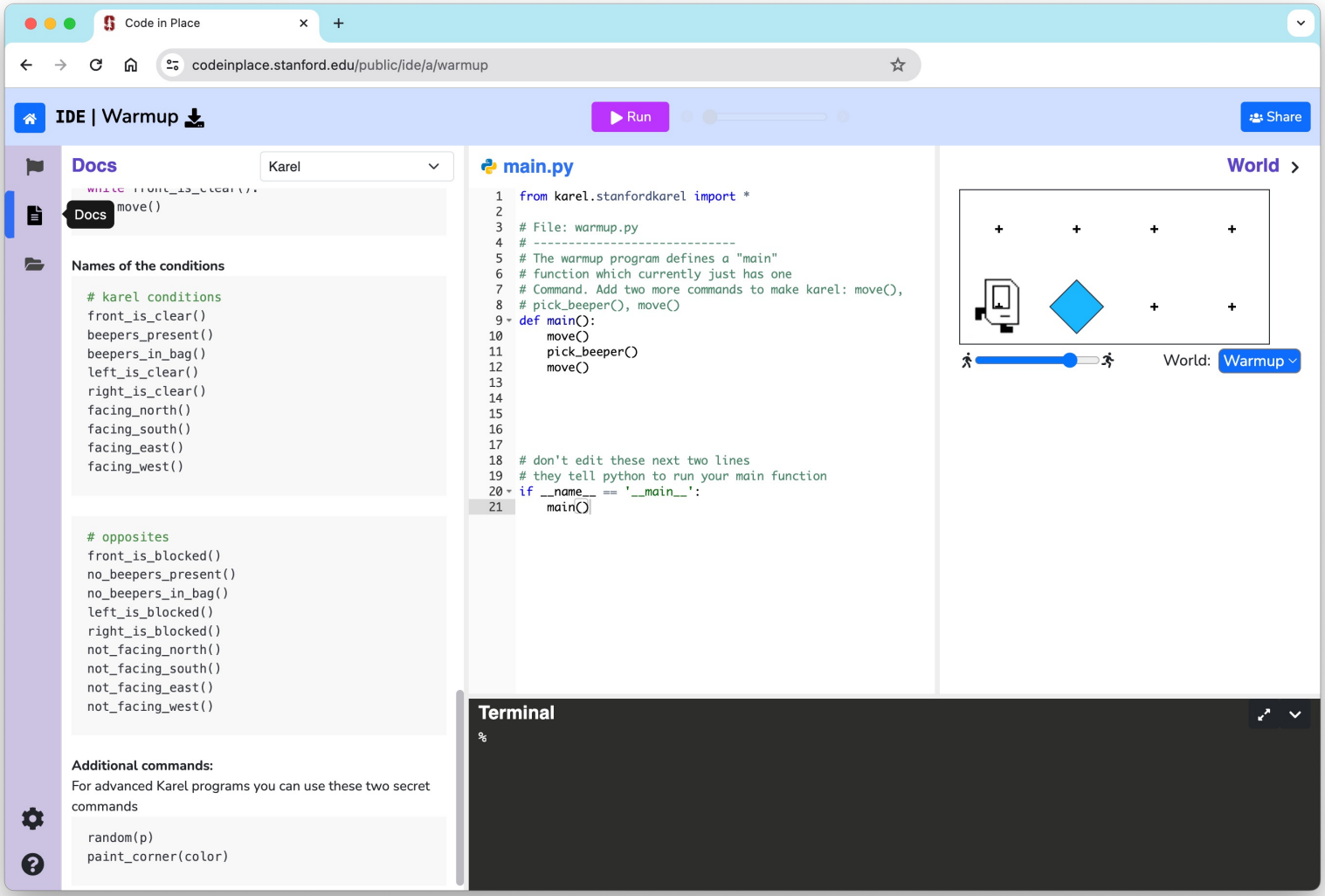
```
def main():

    # If the condition holds, runs body
    if front_is_clear():
        move()
```


If / Else Statement

```
def main():

    # If the condition holds,
    if beepers_present():
        # do this
        pick_beeper()
    else :
        # otherwise, do this
        put_beeper()
```



Code in Place
codeinplace.stanford.edu/public/ide/a/warmup
IDE | Warmup
Run
Share
Docs
Karel
A cheat-sheet with the structure of the Karel programming language. See the Karel Reader for more details:
Base Karel commands
move()
turn_left()
put_beeper()
pick_beeper()
Karel program structures
This is the structure of a Karel program
from karel.stanfordkarel import *
# Comments can be included in any part
# of a program. They start with a #
# and include the rest of the line.
# the computer will ignore them, but they
# are very helpful for human readers!
def main():
# code to execute
# declarations of other functions
# necessary boilerplate to start execution
if __name__ == '__main__':
main()
# example program to move, put_beeper, move
def main():
move()
put_beeper()
main.py
1 from karel.stanfordkarel import *
2
3 # File: warmup.py
4 # ----------------------------
5 # The warmup program defines a "main"
6 # function which currently just has one
7 # Command. Add two more commands to make karel: move(),
8 # pick_beeper(), move()
9 def main():
10 move()
11 pick_beeper()
12 move()
13
14
15
16
17
18 # don't edit these next two lines
19 # they tell python to run your main function
20 if __name__ == '__main__':
21 main()
World
World: Warmup
Terminal
%


Code in Place
codeinplace.stanford.edu/public/ide/a/warmup
IDE | Warmup
Run
Share
Docs
Karel
Docs
Conditions:
if condition:
# code run if condition passes
if condition:
# code block for "yes"
else:
#code block for "no"
# example: a safe move
if front_is_clear():
move()
For Loop:
for i in range(count):
# code to repeat
# example: place 100 beepers
for i in range(100):
put_beeper()
While Loop:
while condition:
# code to repeat
main.py
1 from karel.stanfordkarel import *
2
3 # File: warmup.py
4 # ----------------------------
5 # The warmup program defines a "main"
6 # function which currently just has one
7 # Command. Add two more commands to make karel: move(),
8 # pick_beeper(), move()
9 def main():
10 move()
11 pick_beeper()
12 move()
13
14
15
16
17
18 # don't edit these next two lines
19 # they tell python to run your main function
20 if __name__ == '__main__':
21 main()
World
World: Warmup
Terminal
%


Code in Place
codeinplace.stanford.edu/public/ide/a/warmup
IDE | Warmup
Run
Share
Docs
Karel
Docs
move()
Names of the conditions
# karel conditions
front_is_clear()
beepers_present()
beepers_in_bag()
left_is_clear()
right_is_clear()
facing_north()
facing_south()
facing_east()
facing_west()
# opposites
front_is_blocked()
no_beepers_present()
no_beepers_in_bag()
left_is_blocked()
right_is_blocked()
not_facing_north()
not_facing_south()
not_facing_east()
not_facing_west()
Additional commands:
For advanced Karel programs you can use these two secret commands
random(p)
paint_corner(color)
main.py
1 from karel.stanfordkarel import *
2
3 # File: warmup.py
4 # ----------------------------
5 # The warmup program defines a "main"
6 # function which currently just has one
7 # Command. Add two more commands to make karel: move(),
8 # pick_beeper(), move()
9 def main():
10 move()
11 pick_beeper()
12 move()
13
14
15
16
17
18 # don't edit these next two lines
19 # they tell python to run your main function
20 if __name__ == '__main__':
21 main()
World
World: Warmup
Terminal
%

End review

```
def art_of_problem_solving():
    # lesson plan
    decomposition()
    mountain_karel()
    rhoomba_karel()
    if extra_time():
        word_search_karel()
```

```
def art_of_problem_solving():
    # lesson plan
    decomposition()
    mountain_karel()
    rhoomba_karel()
    if extra_time():
        word_search_karel()
```

```
def art_of_problem_solving():
    # lesson plan
    decomposition()
    mountain_karel()
    rhoomba_karel()
    if extra_time():
        word_search_karel()
```

Make Pasta from Scratch

Make Pasta from Scratch

My helper is a 3 year old. Like Karel, needs very clear instructions

We are going to need to decompose the task

Make Pasta from Scratch

`make_dough()`

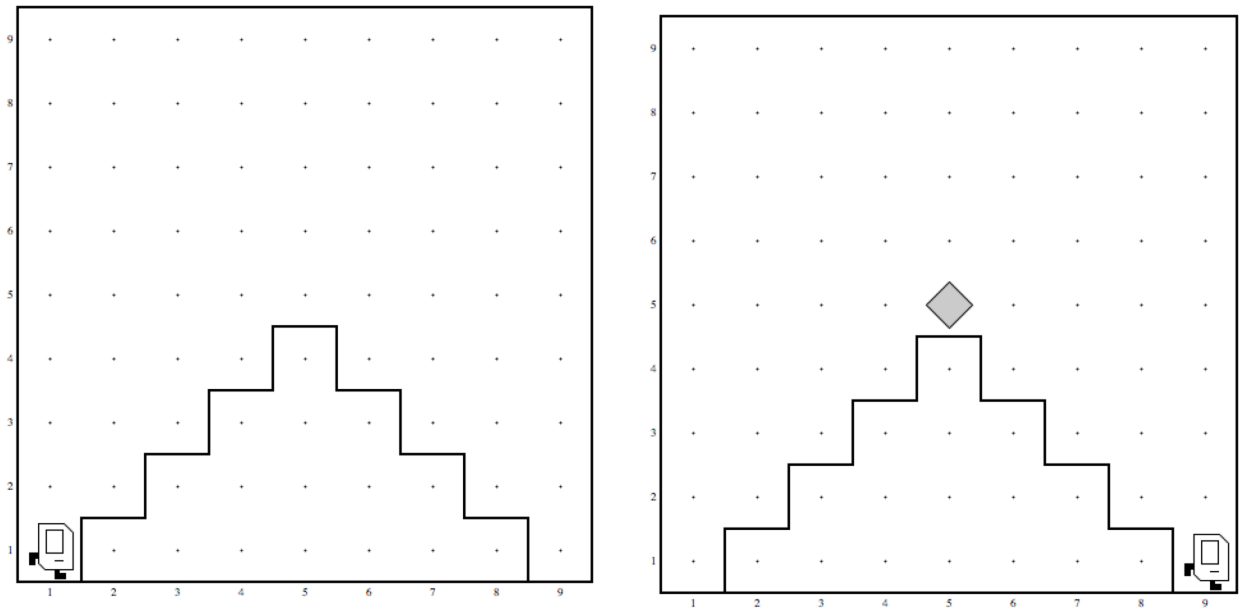
`shape_pasta()`

`cook_pasta()`

Make Pasta from Scratch

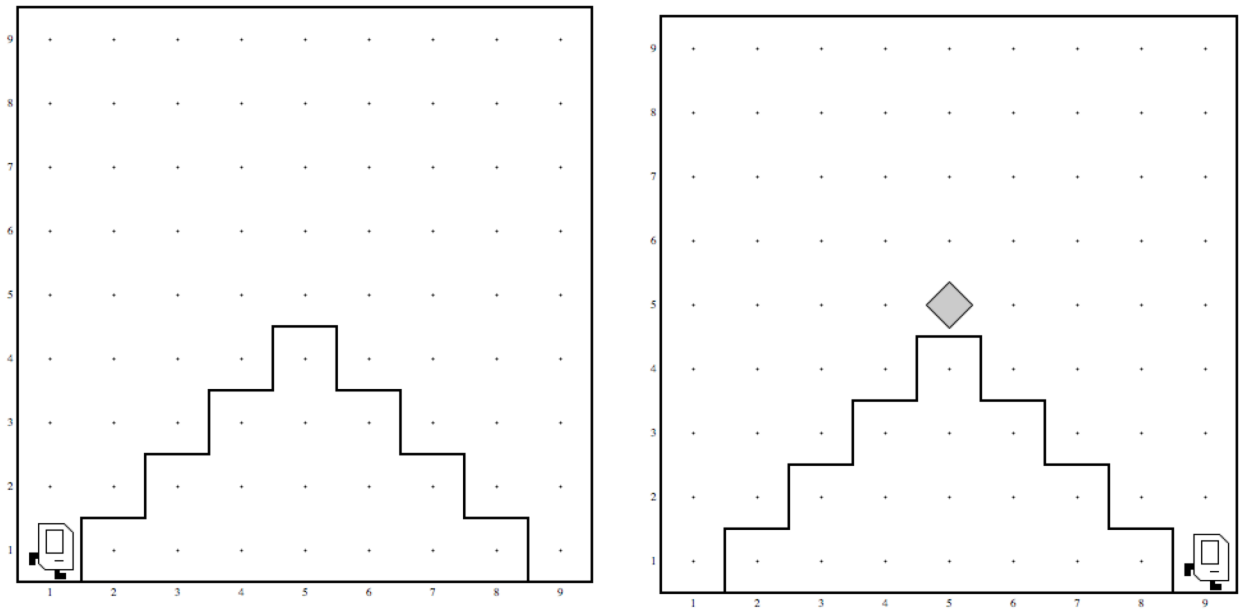
```
1  def main():
2      """
3      This program is meant to show how an everyday
4      task (like cooking) can be approached using
5      the art of stepwise refinement.
6      """
7      make_dough()
8      shape_pasta()
9      cook_pasta()
10
11 def make_dough():
12     # TODO: implement
13     pass
14
15 def shape_pasts():
16     # TODO: implement
17     pass
18
19 def cook_pasta():
20     # TODO: implement
21     pass
```

Mountain Karel

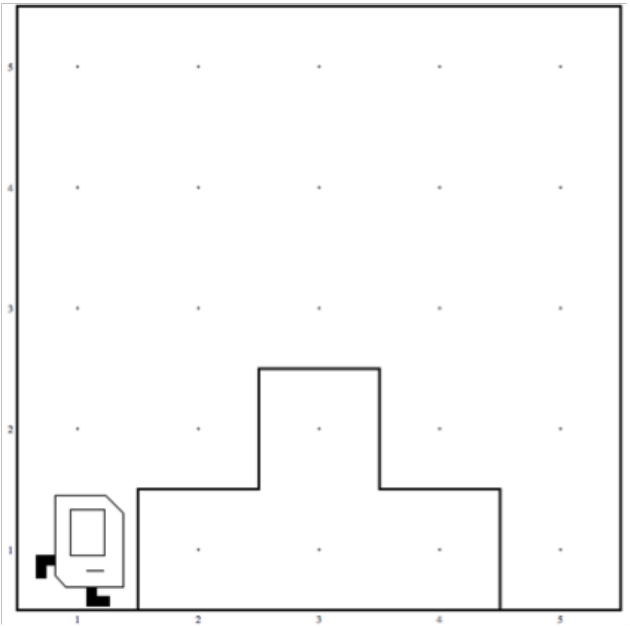


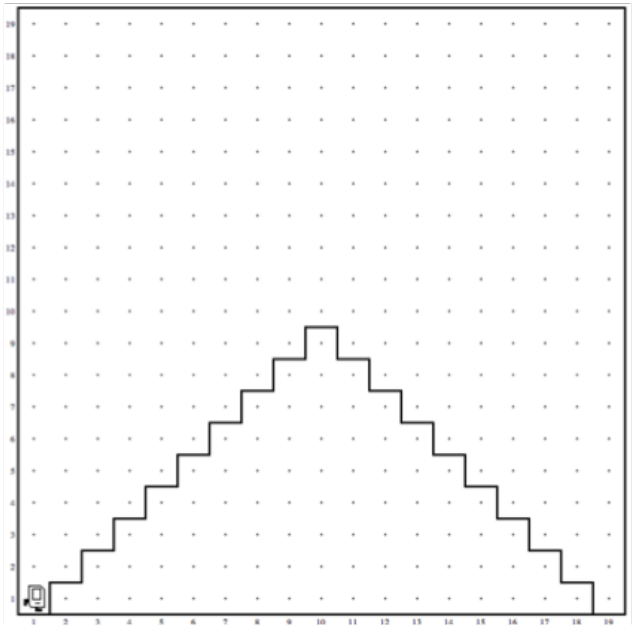
```
def art_of_problem_solving():
    # lesson plan
    decomposition()
    mountain_karel()
    rhoomba_karel()
    if extra_time():
        word_search_karel()
```

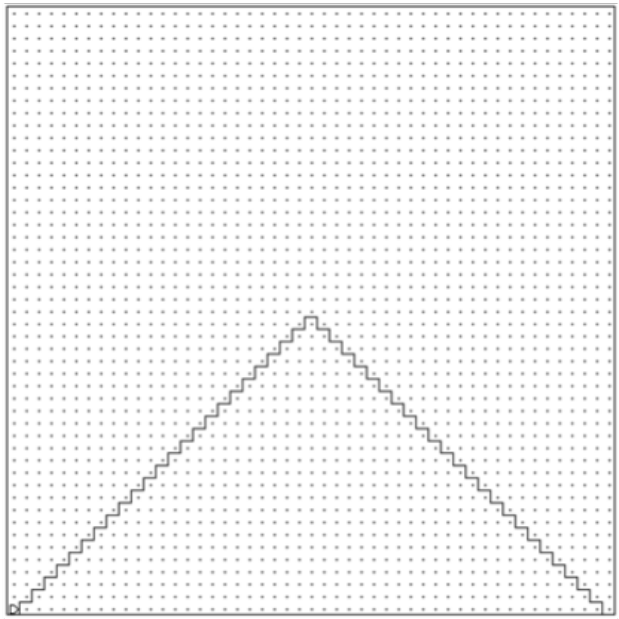
Mountain Karel



Mountain Karel







Muhammed ibn
Musa Al Kwarizmi


DO
YOUR
THING

Pro Tips

A good function should do ”one conceptual thing”

Know what it does by looking at its name

Less than 10 lines, 3 levels of indentation

Reusable and easy to modify

Well commented

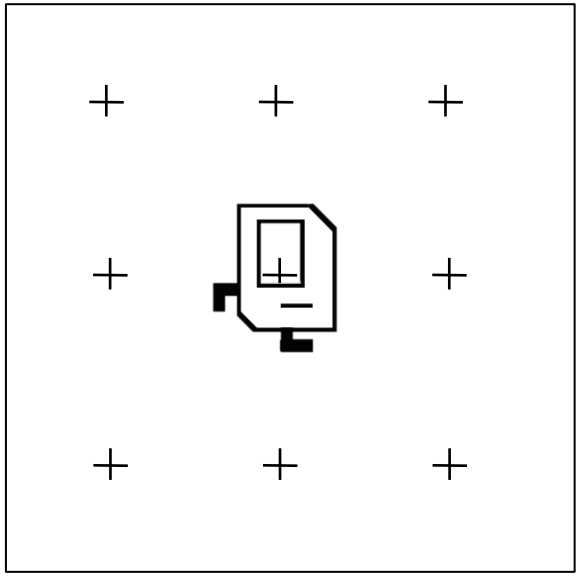
There are two types of programs.
One is so complex, there is nothing obvious wrong with it.
One is so clear, that this obviously nothing wrong with it.

Aside: Common Errors

Lather,
Rinse,
Repeat

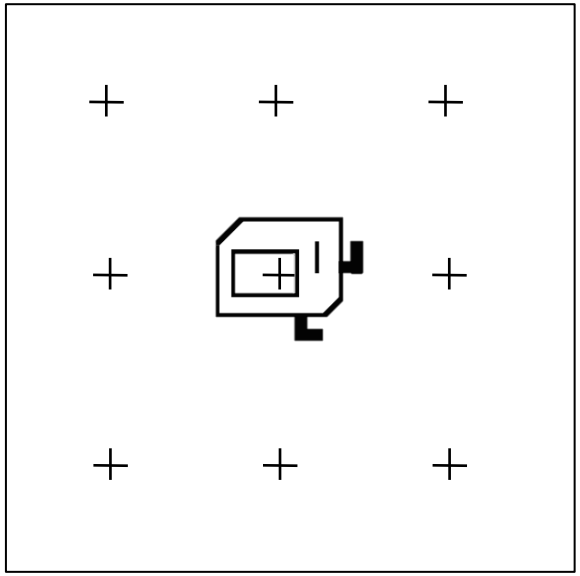
Infinite Loop

```
def turn_to_wall():
    while left_is_clear():
        turn_left()
```



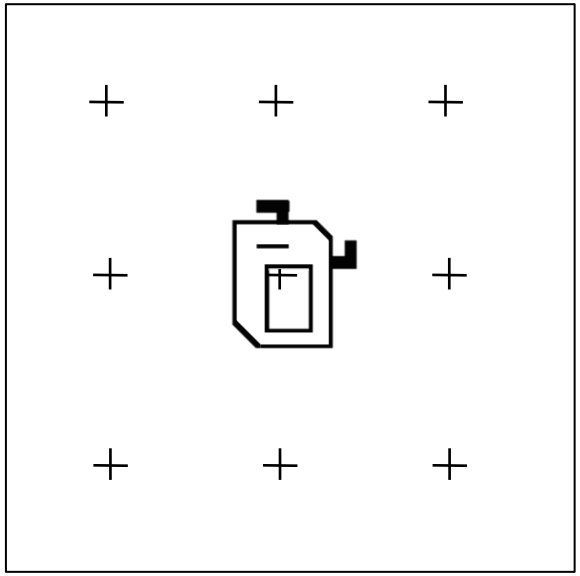
Infinite Loop

```
def turn_to_wall():
    while left_is_clear():
        turn_left()
```



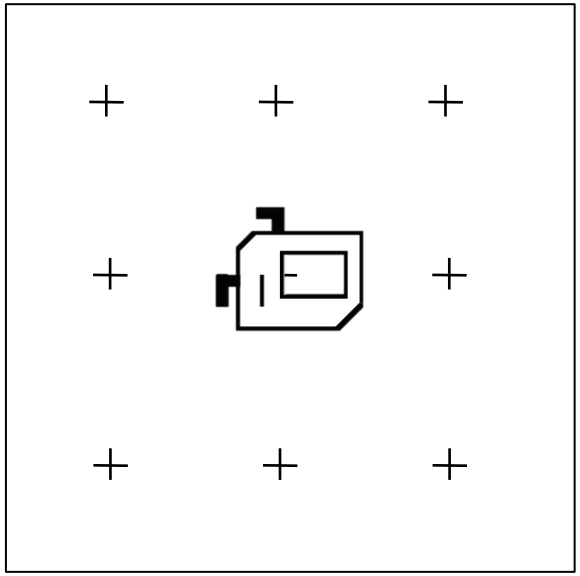
Infinite Loop

```
def turn_to_wall():
    while left_is_clear():
        turn_left()
```



Infinite Loop

```
def turn_to_wall():
    while left_is_clear():
        turn_left()
```



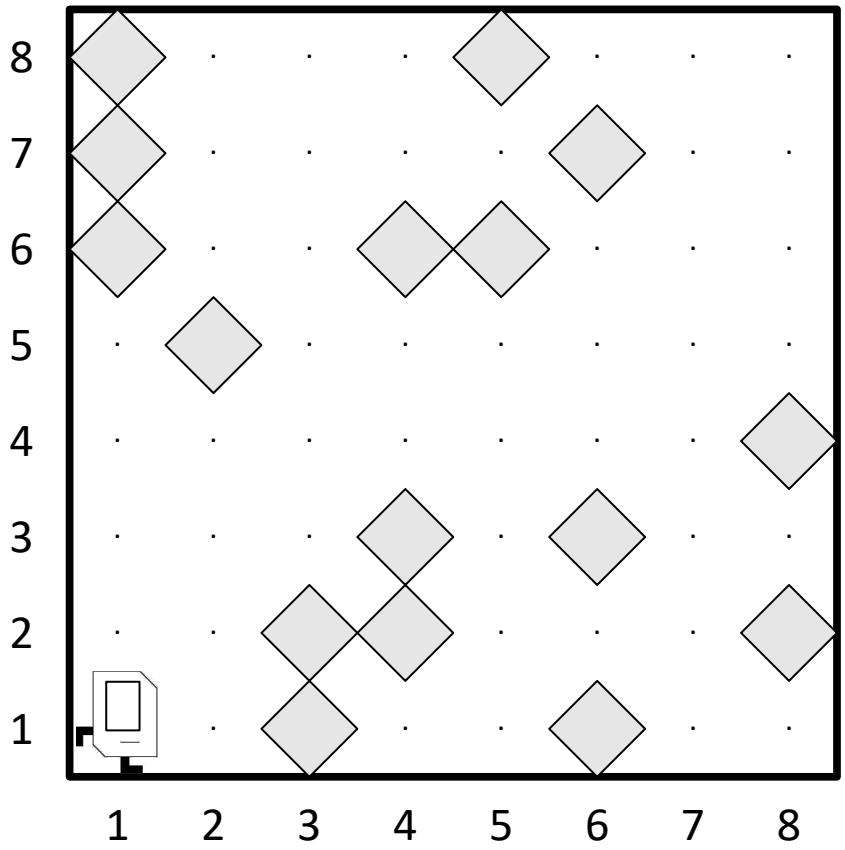
Pre/Post that Don't Match

```
def climb_mountain():
    while front_is_clear():
        # step up
        move()
        turn_left()
        move()
```

What do you assume here?

Does the end condition of the loop match the start assumptions?

Rhoomba Karel

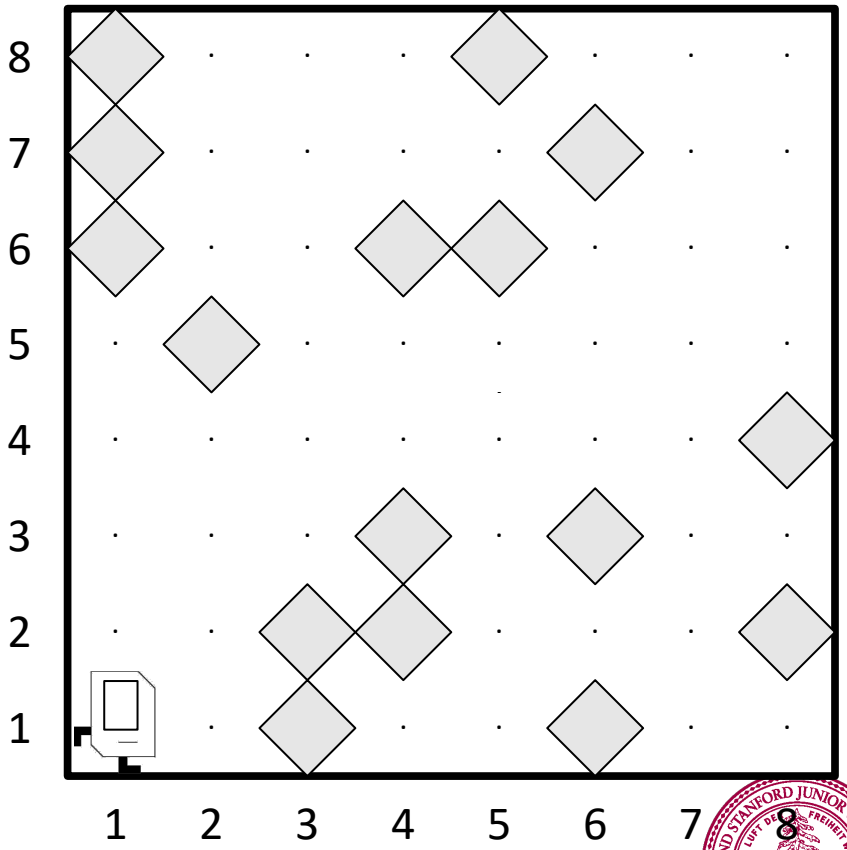


An actual Rhoomba Robot

```
def art_of_problem_solving():
    # lesson plan
    decomposition()
    mountain_karel()
    rhoomba_karel()
    if extra_time():
        word_search_karel()
```

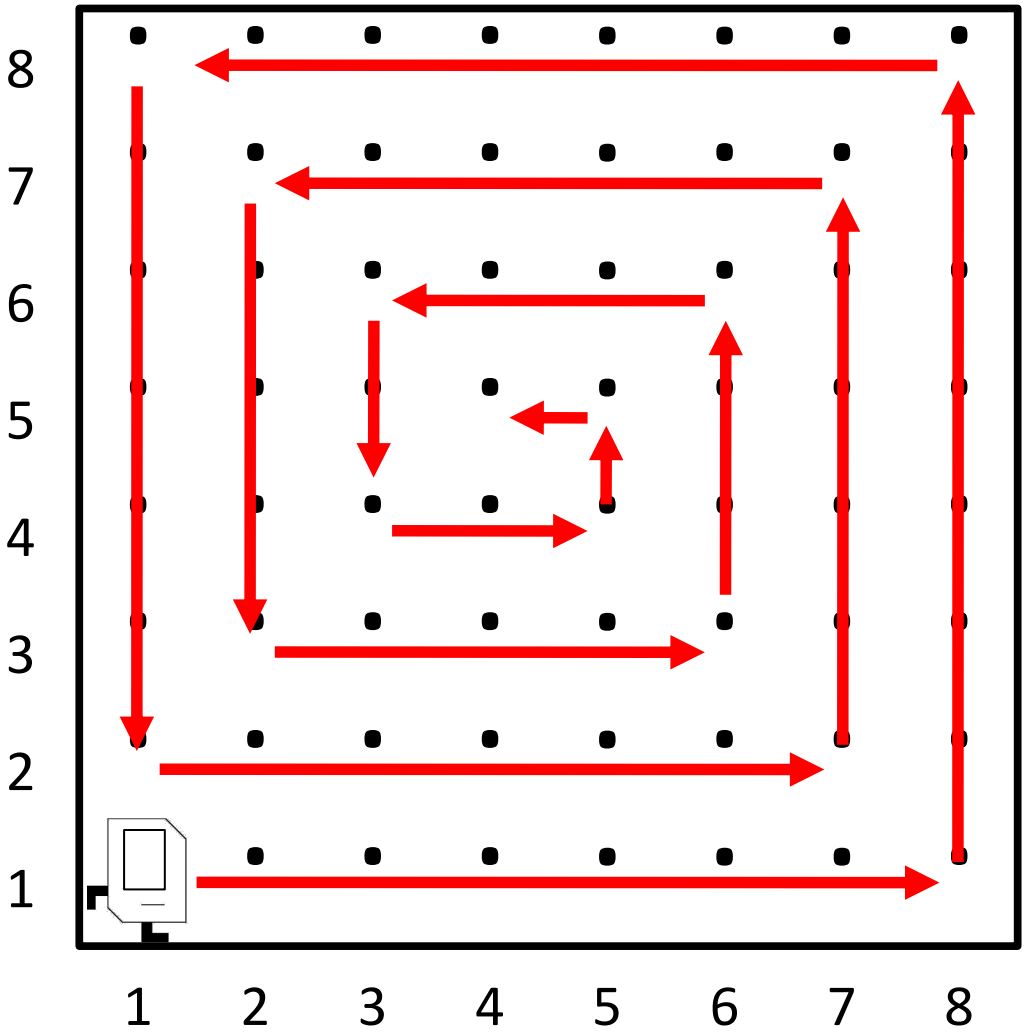
Rhoomba Karel

- Write a **Roomba** Karel that sweeps the entire world of all beepers.
  - Karel starts at (1,1) facing East.
  - The world is rectangular, and some squares contain beepers.
  - There are no interior walls.
  - When the program is done, the world should contain 0 beepers.
  - Karel's ending location does not matter.

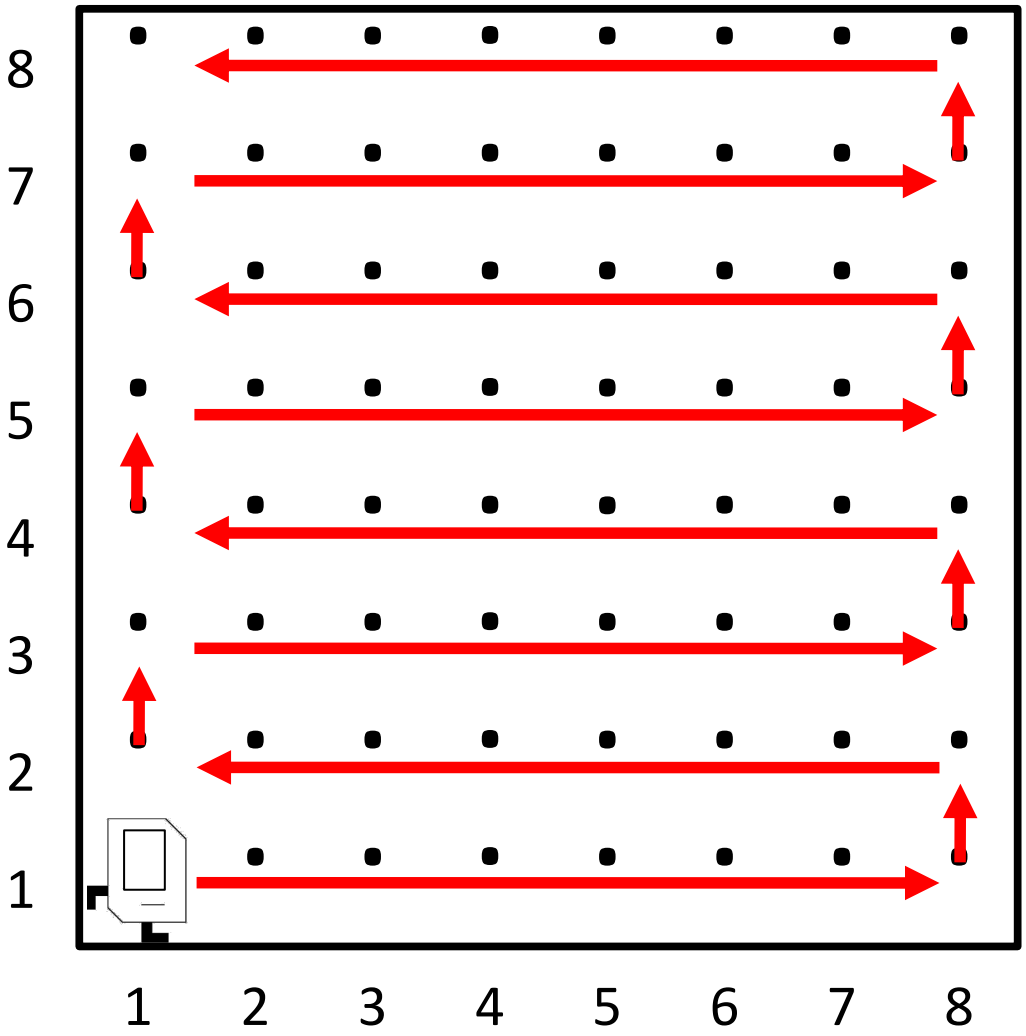


- How should we approach this tricky problem?

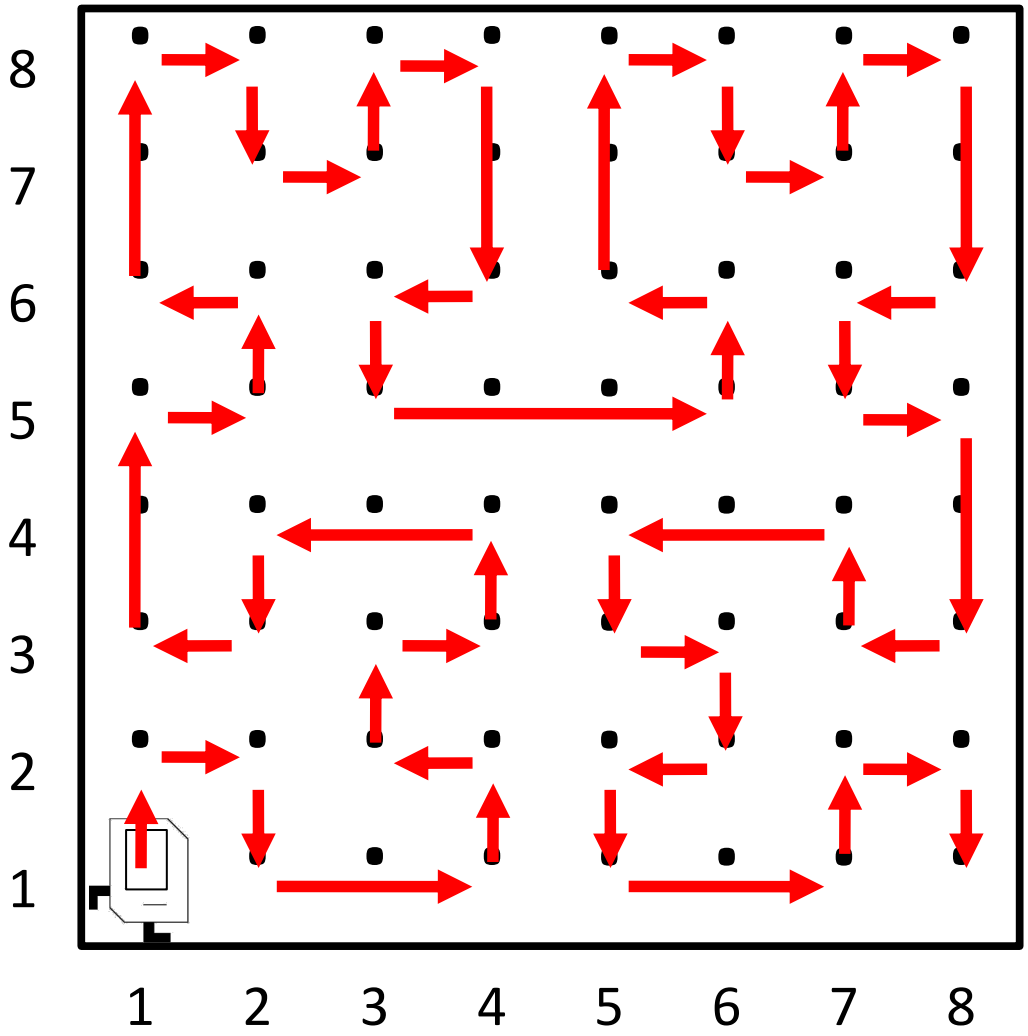
Possible Algorithm 1



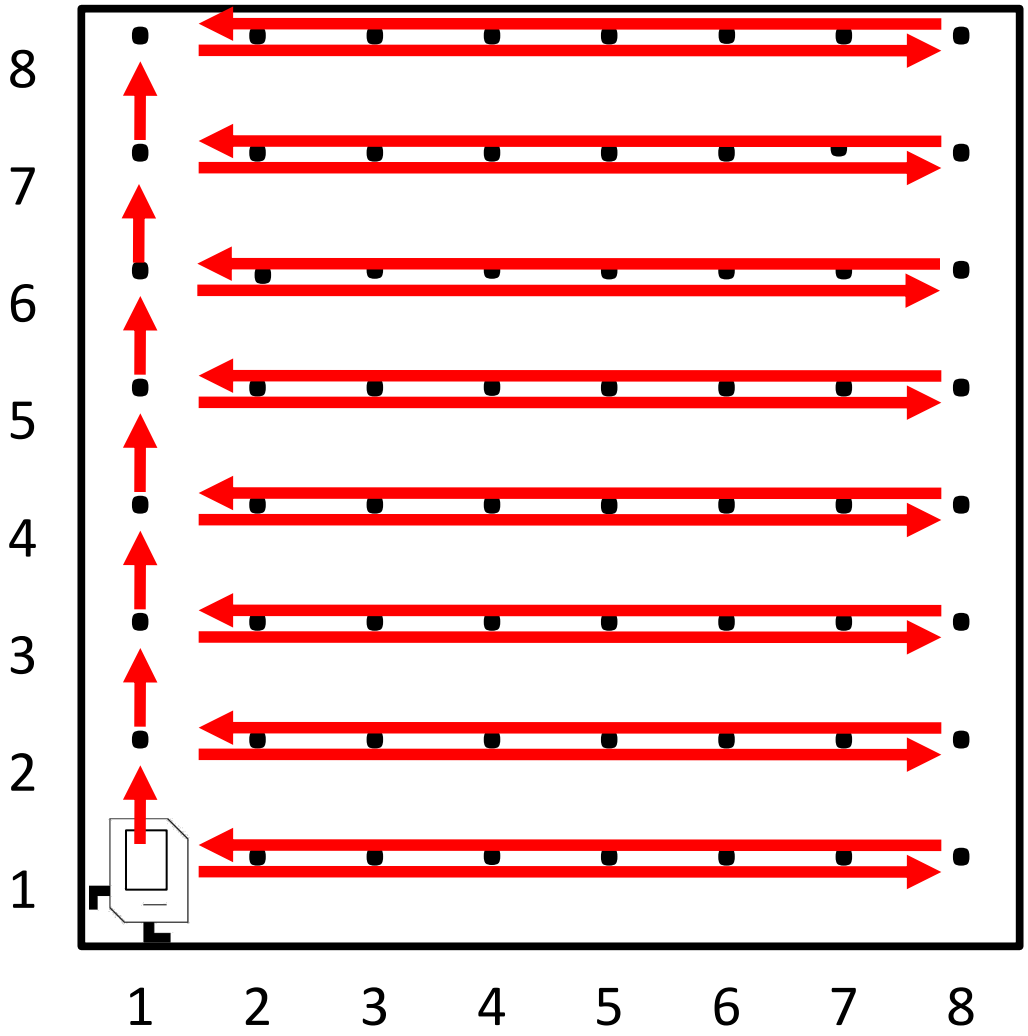
Possible Algorithm 2



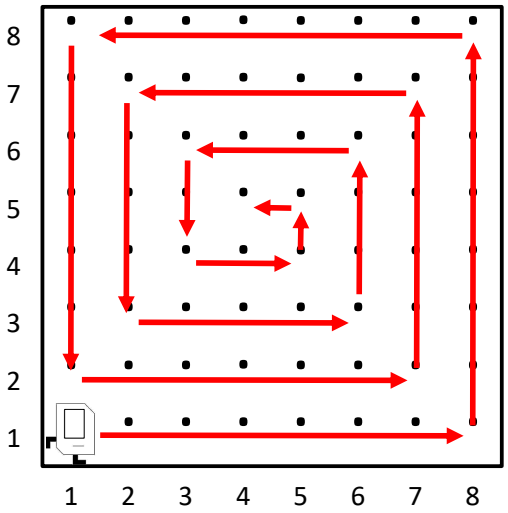
Possible Algorithm 3

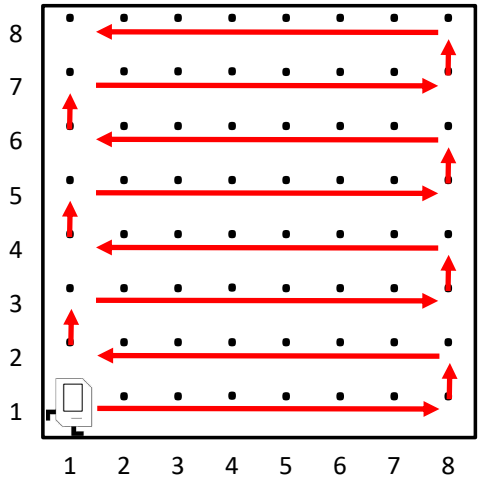


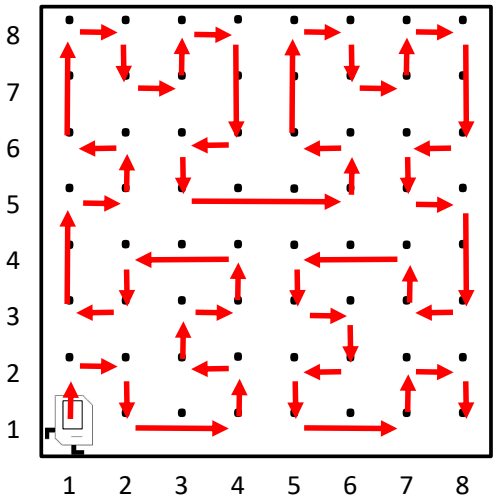
Possible Algorithm 4

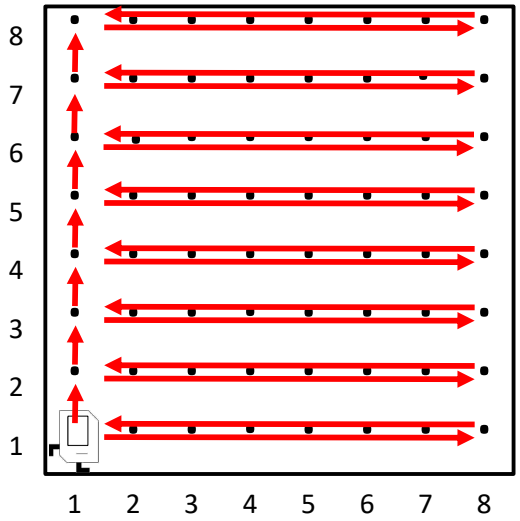


Which Algorithm???

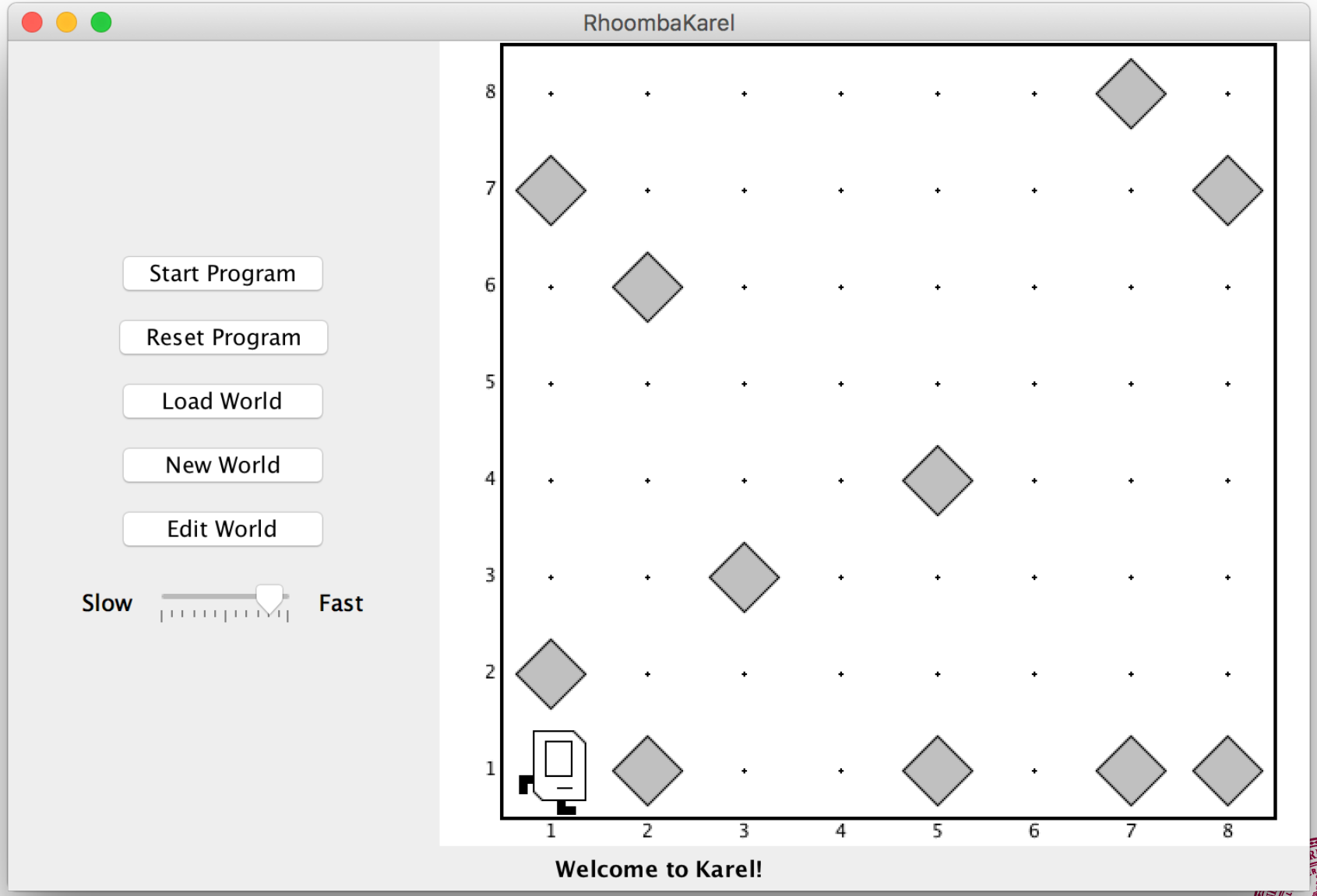


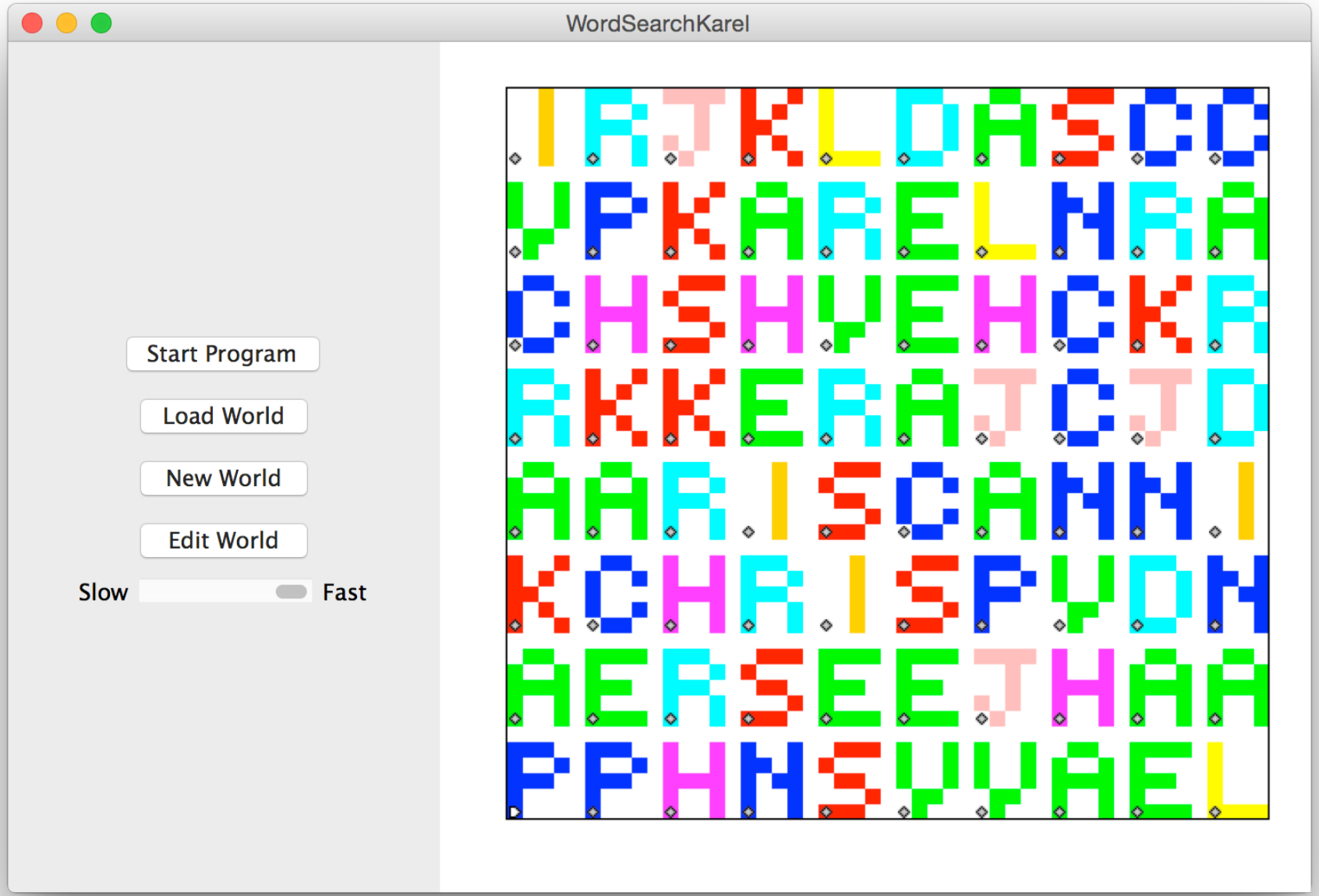






Rhoomba Karel




WordSearchKarel
Start Program
Load World
New World
Edit World
Slow
Fast

```
def art_of_problem_solving():
    # lesson plan
    decomposition()
    mountain_karel()
    rhoomba_karel()
    if extra_time():
        word_search_karel()
```

```
def art_of_problem_solving():
    # lesson plan
    decomposition()
    mountain_karel()
    rhoomba_karel()
    if extra_time():
        word_search_karel()
```