Welcome!

We'll get started shortly …

Welcome to Section!

Week Two

Surajit Bose

Agenda

- Review of lecture concepts
  - Control Flow: `for`, `while`, `if`, `if-else`
  - Functions
- Worked Example
- Problem: Hospital Karel

How to get hold of me / get help+

- The class forum. Feel free not only to ask, but also to answer questions!
- Surajit's office hours:
  - Fridays 12 noon–1p, directly after section
  - By appointment on Zoom
- Lane's office hours
- Canvas inbox or Pronto inbox for Lane
- Canvas inbox (preferred) or Pronto for Surajit
- Sina's support section, Fridays 2p–3p on Zoom
- Email bosesurajit@fhda.edu, 24 hr turnaround
- Online or in-person tutoring via the STEM center (Room 4213)
- The section GitHub repo has lecture and section slides and solutions

Lecture Review: Control Flow

Control Flow: Loops, Conditionals

`for` loop:

- Performs some block of code a specific number of times.

`while` loop:

- Repeatedly performs a block of code until a given condition is evaluated to `False`

`if` statement:

- Performs a block of code only when a condition is `True`, and only once.

`if-else` statement:

- Performs a block of code when a condition is `True`, or a different block when the condition is `False`. Either block is performed only once.

Python Loops

`for` Loop

- An example **`for`** loop that you may see and use with Karel:

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

- **`i`** is a conventional name for the counter, from integer
- But you can call the counter anything you like, even **`x`** or **`bob`** or **`emma`**
- This loop is also called a *definite loop* because we know where it ends, when **`i`** reaches 3.

`while` Loop

- An example `while` loop that you may see and use with Karel:

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

- This loop is also called an *indefinite loop* because it will run until the associated condition becomes `False`.

Python Conditionals

`if` Statement

- An example `if` statement that you may see and use with Karel:

```
def safe_move():
    if front_is_clear():
        move()
```

- An `if` statement runs code indented inside of it when the associated condition evaluates to `True`.

`if-else` Statement

- An `if-else` statement runs one of two blocks of code:
  - Either the code inside the `if` block when the associated condition evaluates to `True`,
  - Or the code inside the `else` block when the condition evaluates to `False`.

```
def safe_move_or_turn():
    if front_is_clear():
        move()
    else:
        turn_left()
```

Any Questions?

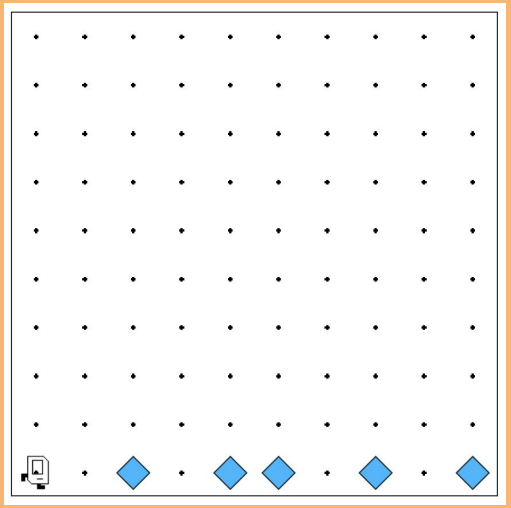
Lecture Review: Functions

Functions

- The process of breaking down a problem into smaller, self-contained building blocks is **decomposition**
- These smaller building blocks are **functions**
  - A function is a sequence of steps that achieves a specific outcome
  - Any set of steps that needs to be repeated could be made a function
  - So could any logically self-contained portion of the problem
- How to decompose a problem and write functions:
  - Identify the building blocks
  - Assemble blocks in `main()` to solve the big problem
  - Assume the building blocks are done (use the `pass` keyword)
  - Implement each building block!

Functions

Example: Karel needs to walk from the first corner `[1,1]` to the end of the row. Every time Karel lands on a beeper, it needs to spin 360º .



Big picture:

- Karel starts on `[1,1]`
- It moves until it is on a corner with a beeper.
- It spins, then moves forward again.
- This process continues until Karel reaches a wall.

What is the small building block that will be useful?

- What action does Karel not yet know how to do, but will need to do repeatedly?

Functions

```
spin()
```

Any Questions?

Worked Example: Spin on Beeper

Using functions and control flow to make Karel spin when it is on a beeper

https://codeinplace.stanford.edu/cs49-w24/ide/p/iY5ilPhtlQluU3sLpROC

Solve the big problem

```
def main():
    while front_is_clear():
        move()
        if beepers_present():
            spin()

def spin():
    pass     # Placeholder
```

Implement the building blocks

```
def main():
    while front_is_clear():
        move()
        if beepers_present():
            spin()

def spin():
    for i in range(4):
        turn_left()
```

Let's Try It Out!

Did it work?

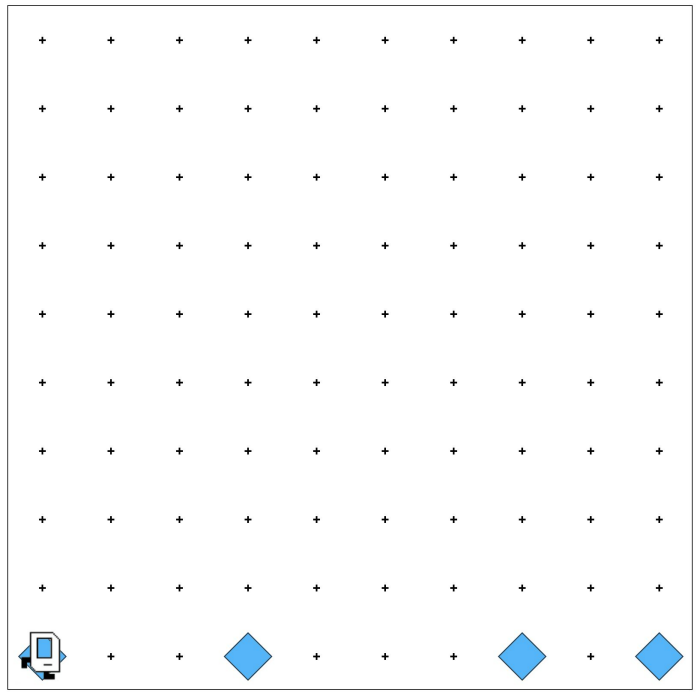
Test and refine the entire solution

```
def main():
    if beepers_present():     #  Fencepost problem: there
        spin()                #  could be a beeper at [1,1]
    while front_is_clear():
        move()
        if beepers_present():
            spin()

def spin():
    for i in range(4):
        turn_left()
```

`spin()` in action

You can see Karel spinning as desired here. The implementation uses the code on the previous slide.



Any Questions?

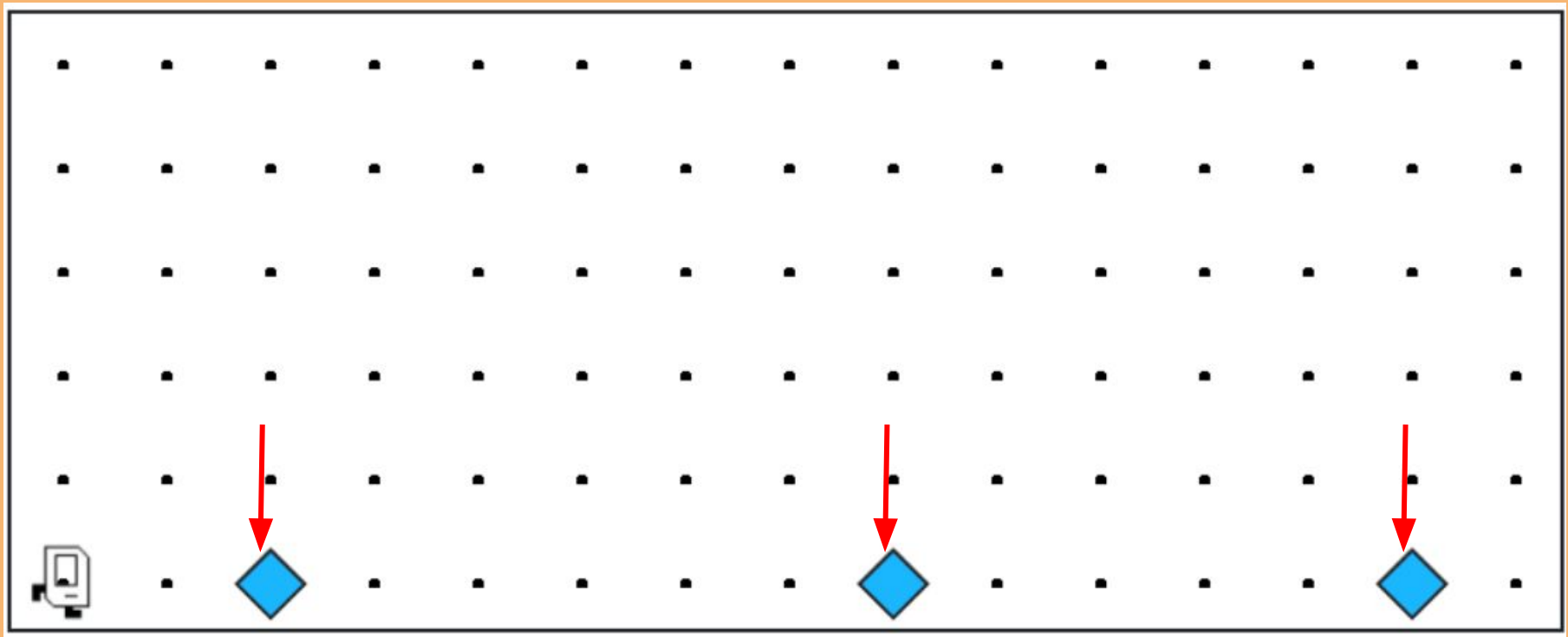
Section Problem: Hospital Karel

https://codeinplace.stanford.edu/cs49-w24/ide/a/hospital

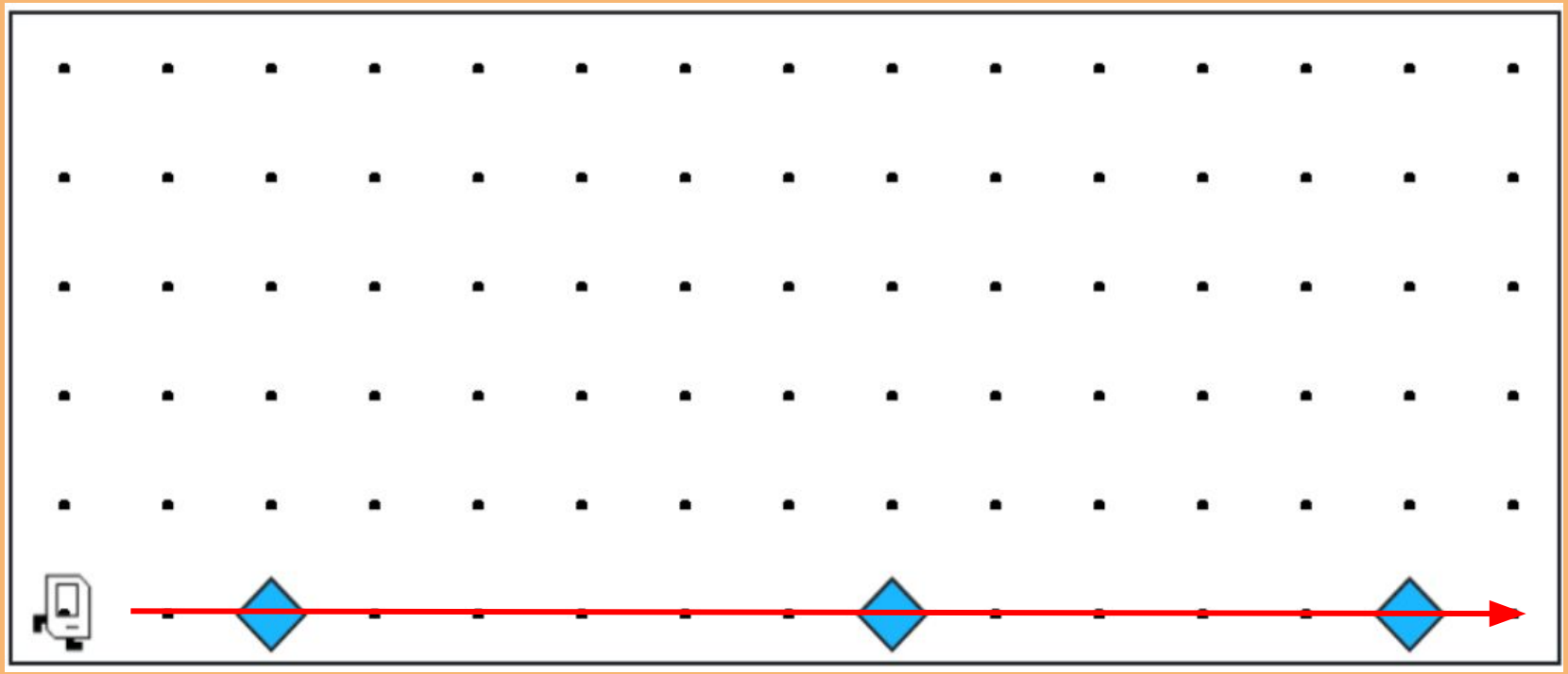
Setting Context

Countries around the world are dispatching hospital-building robots to make sure anyone who gets sick can be treated. They have decided to enlist Karel robots. Your job is to program those robots.

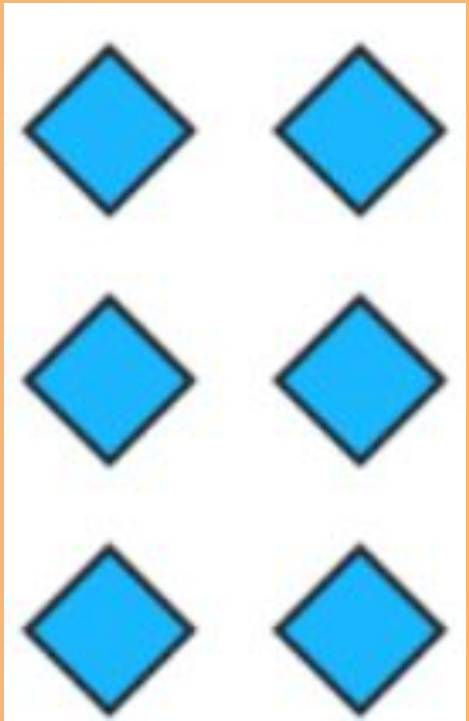
Each beeper in the figure represents a location



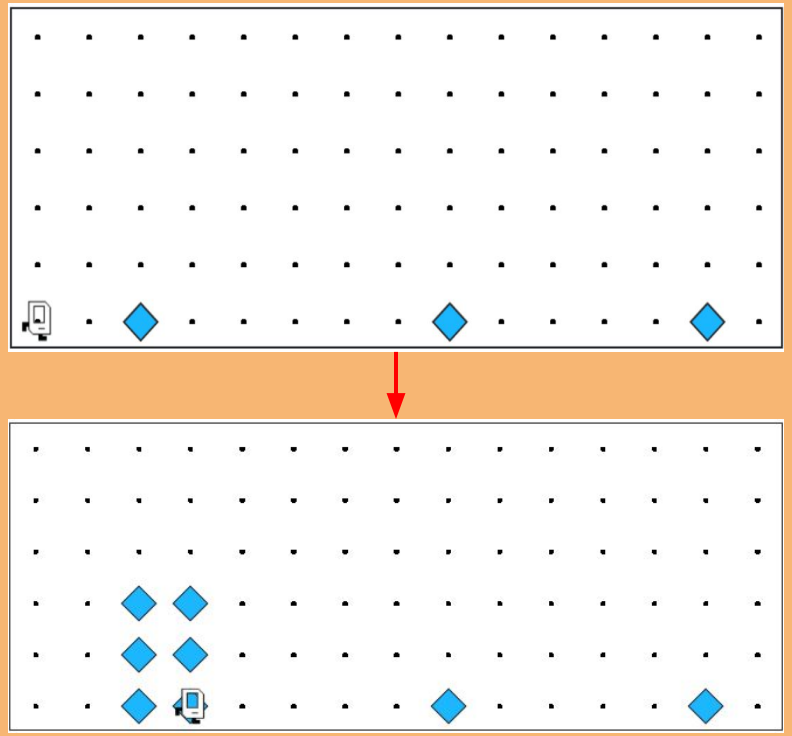
Karel must walk the row and build a hospital at each location



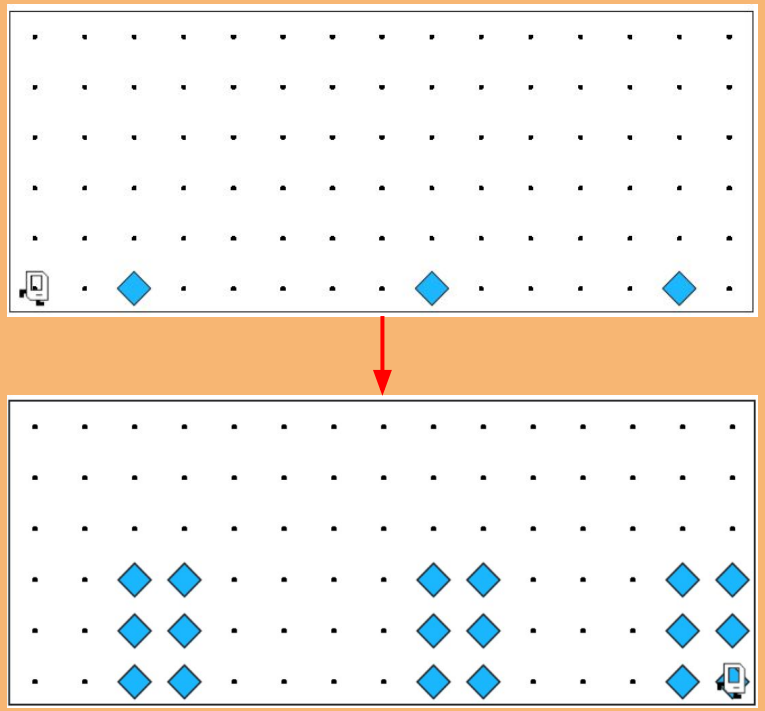
Hospitals look like this: a 3x2 rectangle of beepers!



Here is the state after Karel has built the first hospital



At the end of the run, Karel should be at the end of the row having created a set of hospitals. For the initial conditions shown, the result would look like this:



Notes to Keep in Mind

- Karel starts facing east at `[1, 1]` with an infinite number of beepers in its beeper bag
- The beepers indicating the positions at which hospitals should be built will be spaced so that there is room to build the hospitals without overlapping or hitting walls
- There will be no supplies left on the last column
- Karel should not run into a wall if it builds a hospital that extends into that final corner
- Remember to decompose the problem: identify the building blocks!

Questions Before We Begin?

Let's get to work!