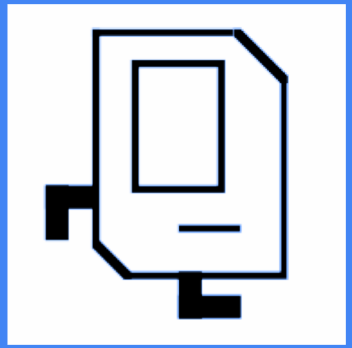
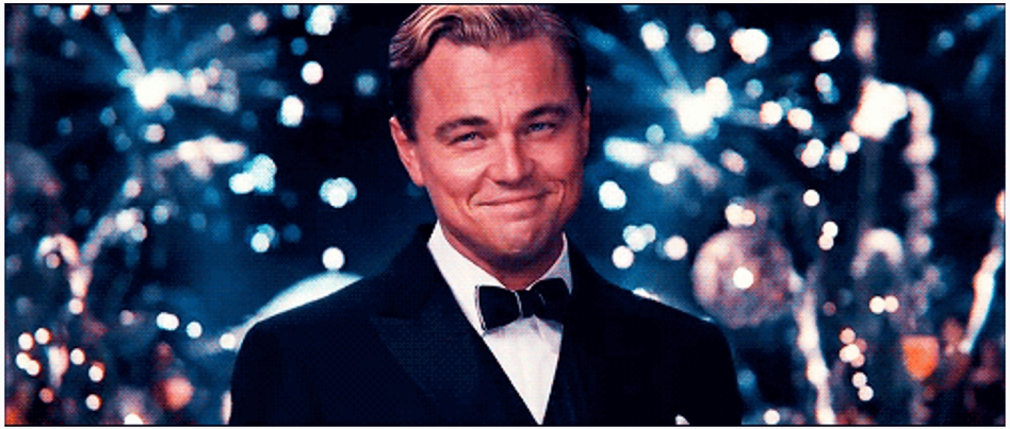
Monty's Coders Section

Week 1 [April 25, 2024]



Welcome to the section!

Congratulations!



Introduction

Let’s get to know each other!

Materials

- Course Materials: https://github.com/mkrdip/codeinplace-2024
- Week 1: https://github.com/mkrdip/codeinplace-2024/tree/main/week1
- Section Leader: Mrinal (Mree-nal) Roy
- Email: mkrdip@gmail.com

Why are we here?

Section is what makes Code in Place different.

```
section != lecture
```

Building a community of learners.

How can you get the most out of this time?

This is your time.

Finishing problem isn't important.

If we run over, no pressure to stay.

Naming this section

Naming Things is Hard

> There are two hard things in computer science: cache invalidation, naming things, and off-by-one errors.

Naming this section

Monty's Coder Section
Suggested by Arthur

There are two hard things in computer science: cache invalidation, naming things, and off-by-one errors.

Approaching a problem

Approaching a problem

Approaching a problem

1) Understand
    a) Write stuff down.
    b) Explain problem out loud to someone.
    c) Draw a picture.

Approaching a problem

1) Understand
    a) Write stuff down.
    b) Explain problem out loud to someone.
    c) Draw a picture.
2) Strategize
    a) Walk through an example or two.
    b) Look for patterns.
    c) Think about what things you'll need: loops, conditions, helper functions, etc.

Approaching a problem

1) Understand
    a) Write stuff down.
    b) Explain problem out loud to someone.
    c) Draw a picture.
2) Strategize
    a) Walk through an example or two.
    b) Look for patterns.
    c) Think about what things you'll need: loops, conditions, helper functions, etc.
3) Translate
    a) First into **pseudocode**.
    b) Then into Python.

Control Flow

Control flow

`if`

`if/else`

`for`

`while`

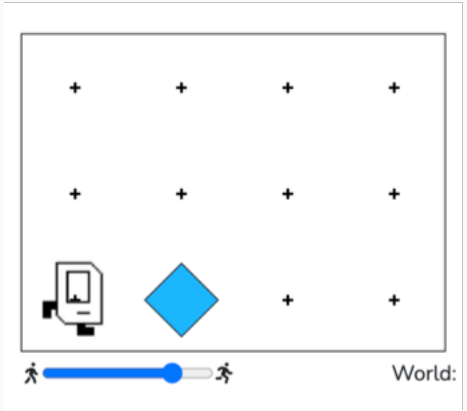
Control flow

`if` Do something when a condition is met.

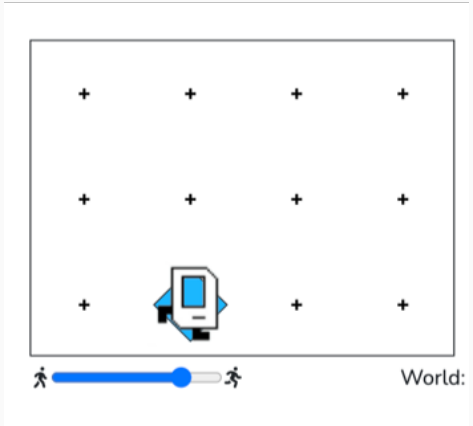
`if/else`

`for`

`while`




```
if front_is_clear():
    move()
```



Control flow

`if`

`if/else`

`for`

`while`

Control flow

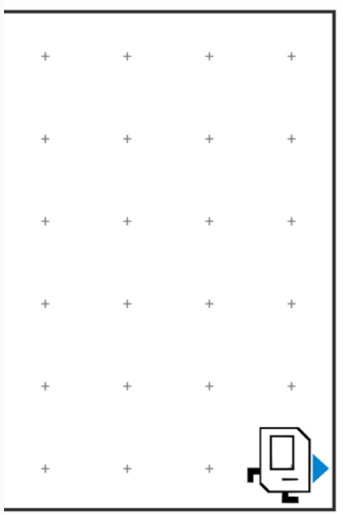
`if`

`if/else`

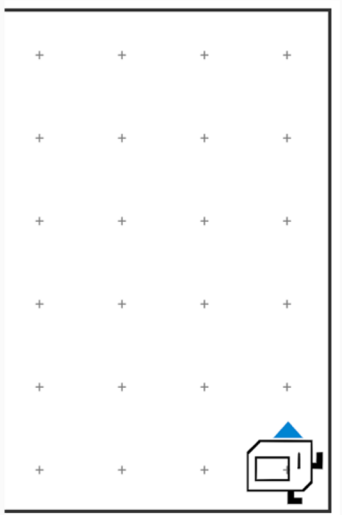
`for`

`while`

Also do something else when a condition is not met.



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



Control flow

`if`

`if/else`

`for`

`while`

Control flow

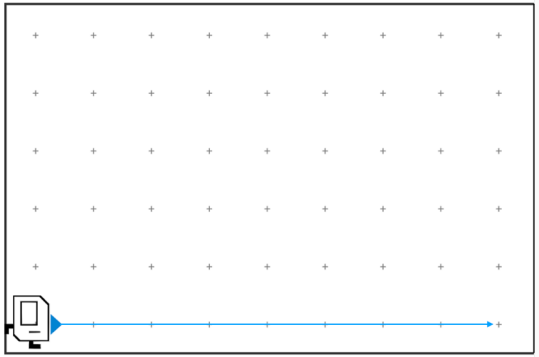
`if`

`if/else`

`for`

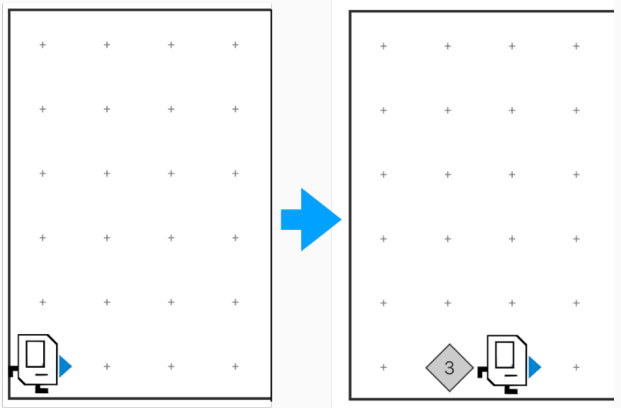
`while`

Do something a set number of times.



Move forward 8 squares

```
for i in range(8):
```



Place 3 beepers

```
for i in range(3):
```

Control flow

`if`

`if/else`

`for`

`while`

Control flow

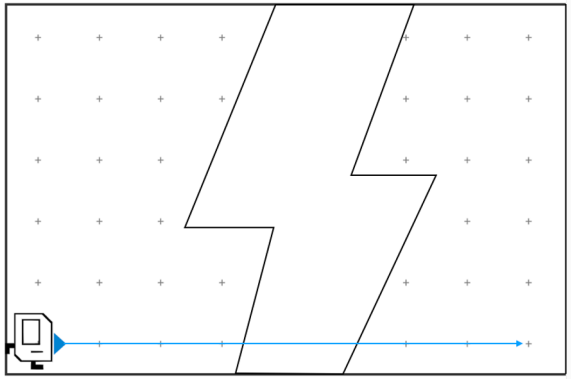
`if`

`if/else`

`for`

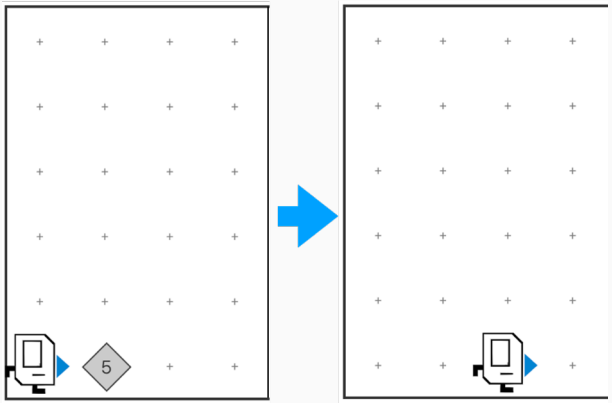
`while`

Do something until condition is no longer true.



Move forward as long as the front is clear

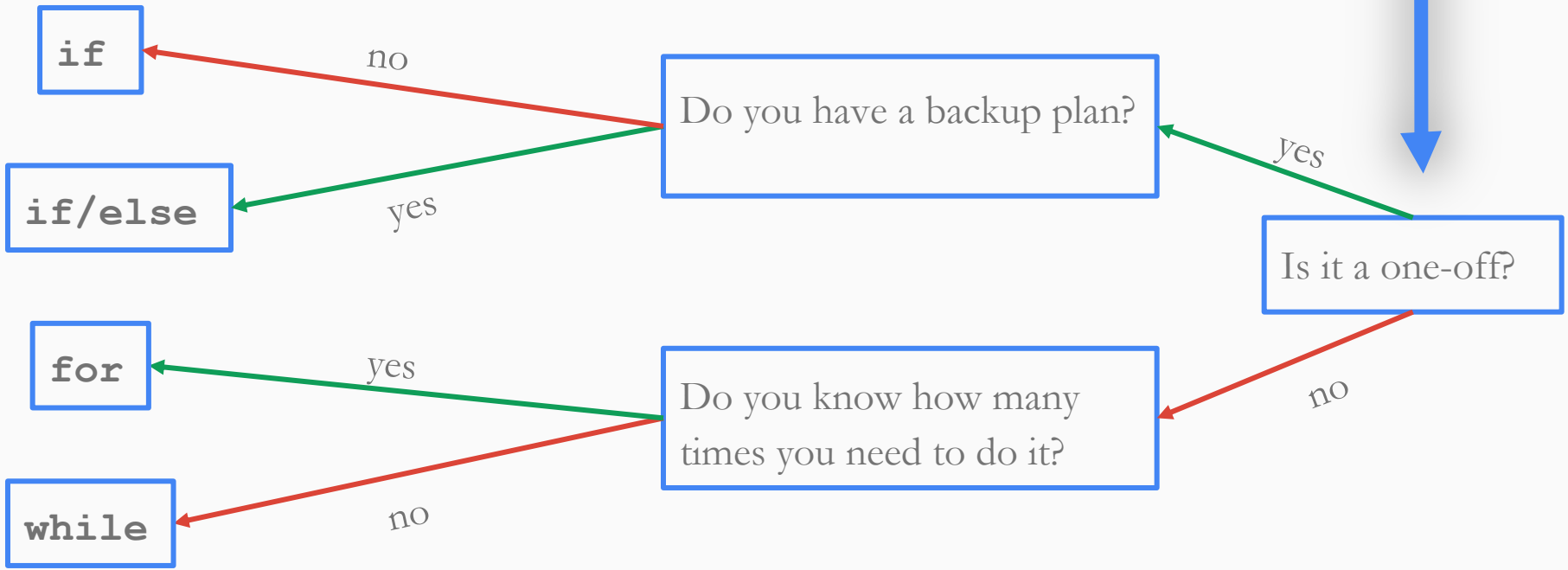
```
while front_is_clear():
```



Pick up beepers while there are still some to pick up

```
while beepers_present():
```

Control flow



Decomposition

Decomposition

Define your function like this:

```
def function_name():
    <write code here>
```

And call it like this:

```
function_name()
```

Decomposition

More of an art than a science.

Functions should be short and read like English.

If you repeat things (or find yourself hitting copy/paste), take a step back.

Top-down programming & the leap of faith.

Decomposition

```
def main():
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
```

Decomposition

```
def main():
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
    turn_left()
```

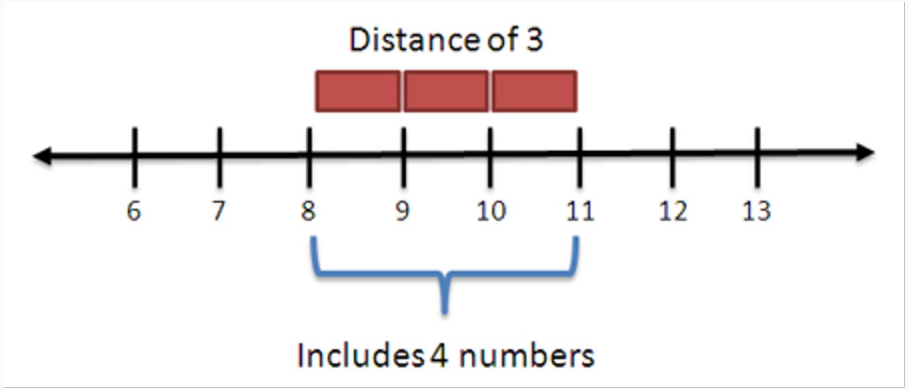
```
def main():
    spin()
    spin()
    spin()

def spin():
    turn_left()
    turn_left()
    turn_left()
    turn_left()
```

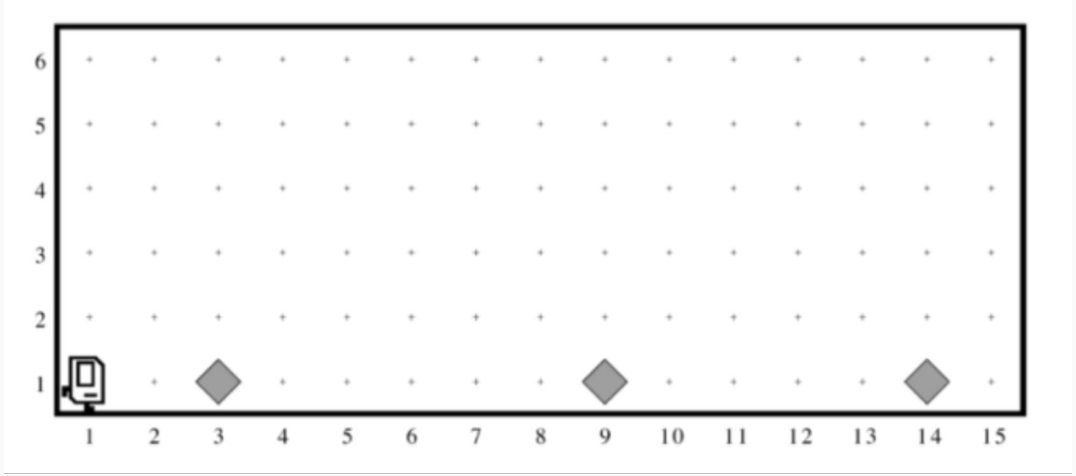
Pre/Post conditions

Each helper function defines a contract.

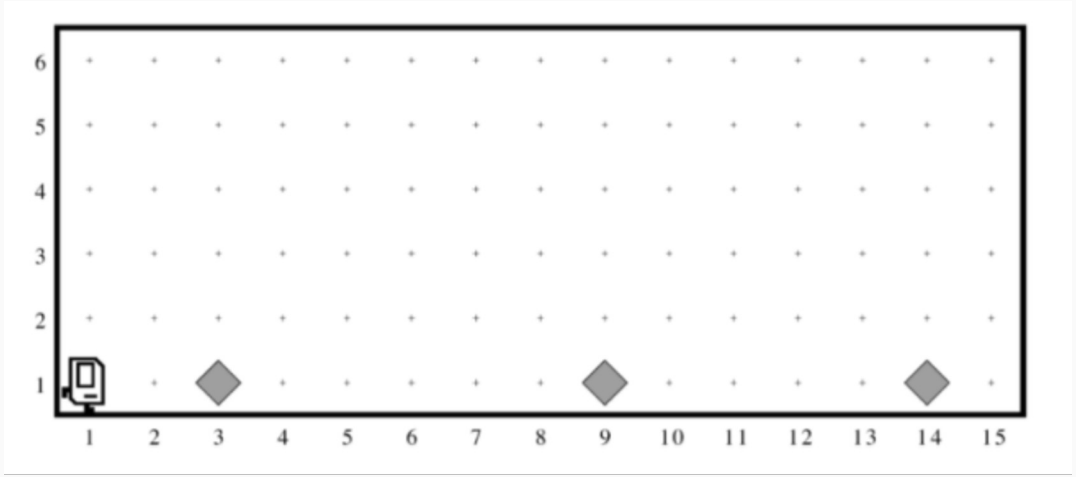
Fencepost problem

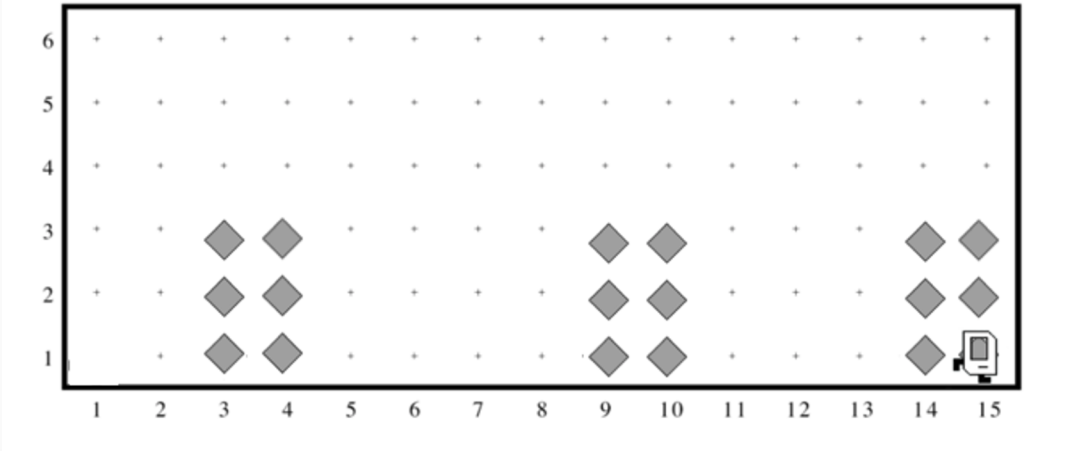


Hospital Karel



Example precondition





Example postcondition

More thoughts

Getting started

Setting up your environment is hard.

You only have to do this once.

Karel

Visual = tangible. Run/watch to understand what you're code is doing.

Constraints because of narrow language → Creativity!

Algorithmically challenging!

Edge conditions.

Follow-ups from questions

Python style

PEP 8 style guide: https://peps.python.org/pep-0008/ is a great resource!

- It recommends **`snake_case`** over **`camelCase.`**
- "Function and variable names should be lowercase, with words separated by underscores as necessary to improve readability."
- e.g., **`def build_hospital()`**