Week 1

CS106R

Sabri **Eyuboglu** & Geoffrey **Angus**

What is
Computer Science?

This is

Computer Science.

This is

Computer Science.

Google

San Francisco

All Flights Images Maps Videos More Settings Tools

About 1,270,000,000 results (1.04 seconds)

San Francisco - Wikipedia

https://en.wikipedia.org/wiki/San_Francisco

San Francisco officially the City and County of **San Francisco**, is the cultural, commercial, and financial center of Northern California. It covers an area of about ...

San Francisco Bay Area · List of cities and towns · List of nicknames for San ...

Things to do in San Francisco

Golden Gate Bridge

Iconic art deco span opened in 1937

Fisherman's Wharf

Crab stands, souvenir shops & sea lions

Alcatraz Island

Notorious prison/historical

Golden Gate Park

Gardens, trails, museums & festivals

San Francisco travel guide

San Francisco Travel | Visitor Information

www.sftravel.com/

San Francisco is home to a little bit of everything. Whether you're a first time visitor or a long-time local, **San Francisco's** Golden Gates welcome all. This is the ...

San Fra

City in California

San Francisco, in n
peninsula surround
known for its year-r
colorful Victorian ho
is its most distinctiv
the notorious forme

Local time: Wedne

Weather: 13°C, Wi

Population: 870,8

Minimum wage: 1

Plan a trip

San Franc

Google

San Francisco

All Flights Images Maps

About 1,270,000,000 results (1.04 second

San Francisco - Wikipedia

https://en.wikipedia.org/wiki/San_Franc

San Francisco officially the City and Cou

financial center of Northern California. It c

San Francisco Bay Area · List of cities an

All Flights Images M

About 1,270,000,000 results (1.04

San Francisco - Wikipedi

https://en.wikipedia.org/wiki/San

San Francisco, officially the City a

Images Maps Videos

0 results (1.04 seconds)

- Wikipedia

a.org/wiki/San_Francisco

What are we going to learn in this class?

Programming Methodology

The tools you need to solve problems with a computer

CS106R Topics

Computer Science

- Algorithms
- Human Computer Interaction
- Programming Methodology
- Artificial Intelligence
- Graphics
- Data Science
- Theoretical CS
- Systems

CS106R Topics

Computer Science

Artificial Intelligence

Human Computer Interaction

Data Science

Graphics

Graphics

Theoretical CS

Systems

Algorithms

Programming Methodology

Programming Methodology

Programming Concepts

- Functions
- Variables
- Conditionals
- Loops
- Objects
- Classes

Programming Style

- Decomposition
- Naming
- Architecture
- OOD
- Documentation

What are the objectives of this course?

CS106R Objectives

1) Give you a technical **foundation** in Computer Science

2) Increase your **understanding** of Computer Science practice in the real world

CS106R Logistics

Class

Monday **13:30 - 15:50**

Lecture + Exercises

Homework

≤ 2 hours per week

Reading + Finish Exercises + Projects

Grading

Credit

1) Attend every class

(If you have to miss class for a legitimate reason, send us an email, and we'll meet with you one-on-one to catch up)

2) Complete **90%** of the mini-projects

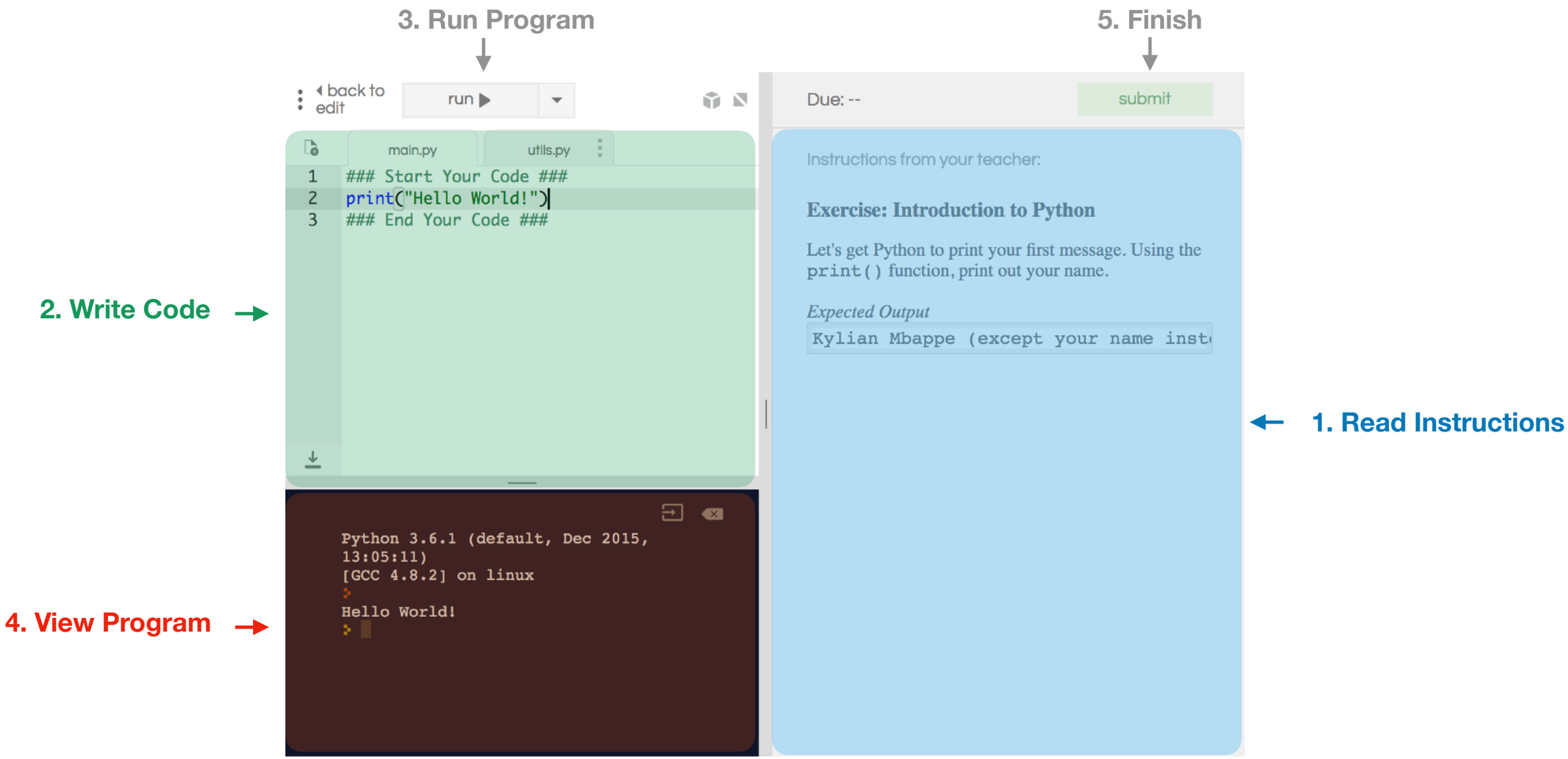
(There will be extra-credit projects you can use to boost your completion rate if needed)

CS106R Website

cs106r.bomjesus.br

Python

Repl.it


3. Run Program
5. Finish
back to edit
run
Due: --
submit
main.py
utils.py
Start Your Code
print("Hello World!")
End Your Code
Instructions from your teacher:
Exercise: Introduction to Python
Let's get Python to print your first message. Using the print() function, print out your name.
Expected Output
Kylian Mbappe (except your name inst
2. Write Code
1. Read Instructions
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
Hello World!
4. View Program

print( )

```
print("Message goes here")
```

Example

Python Code

```
print("I am a Python program!")
```

Result

```
> I am a Python program!
```

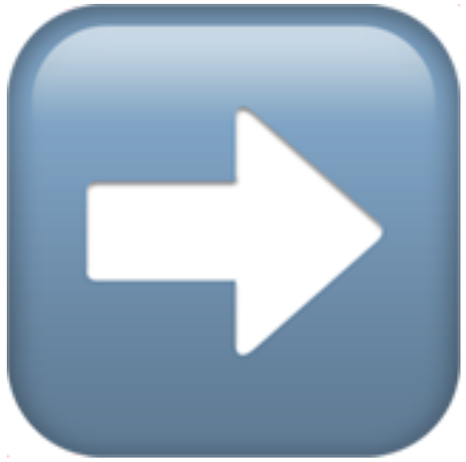
Example: Print Message

Print Name

Harvest Your First Fruit

Turn Left

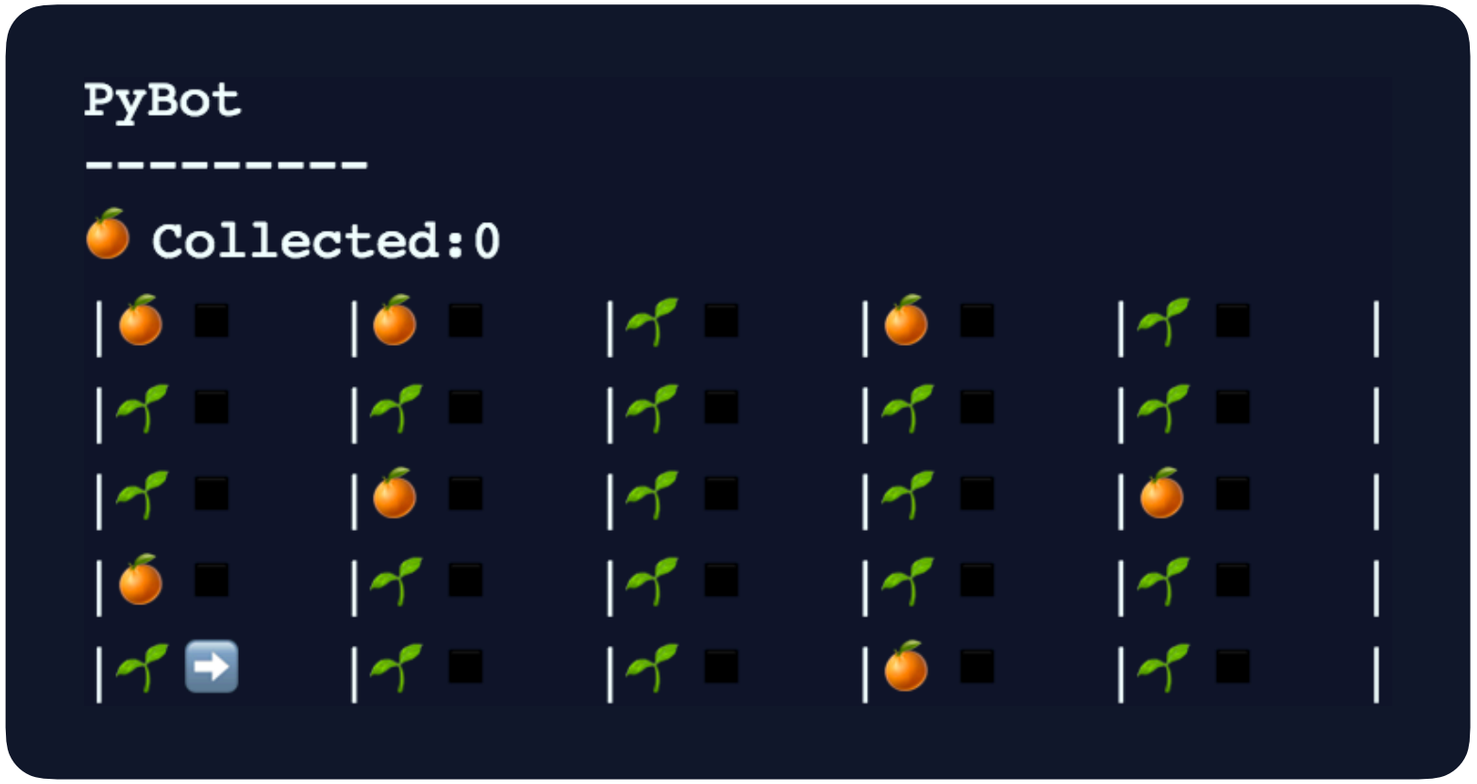
Introducing **PyBot**



PyCountry **Fields**

NORTH

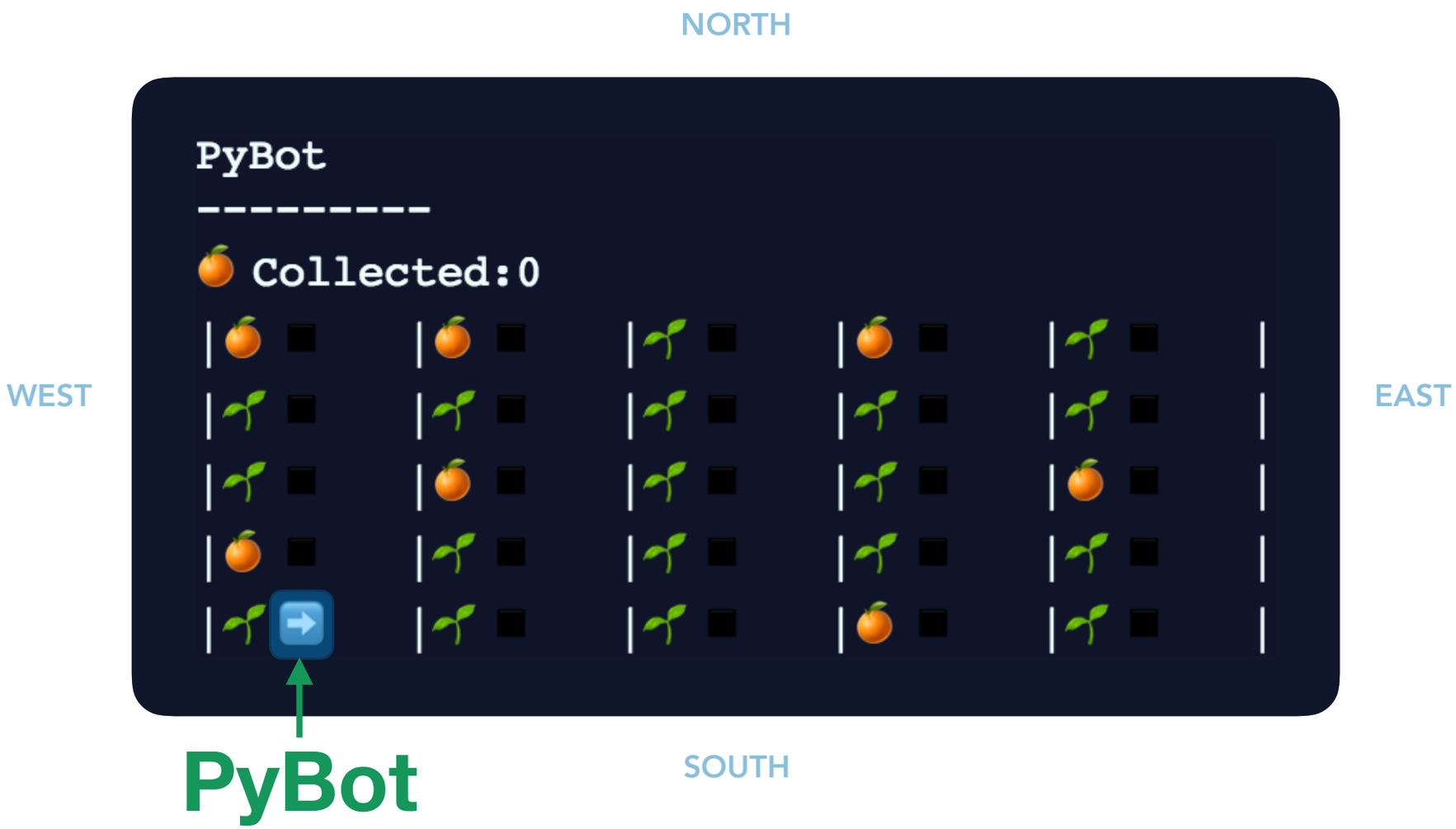
WEST



EAST

SOUTH

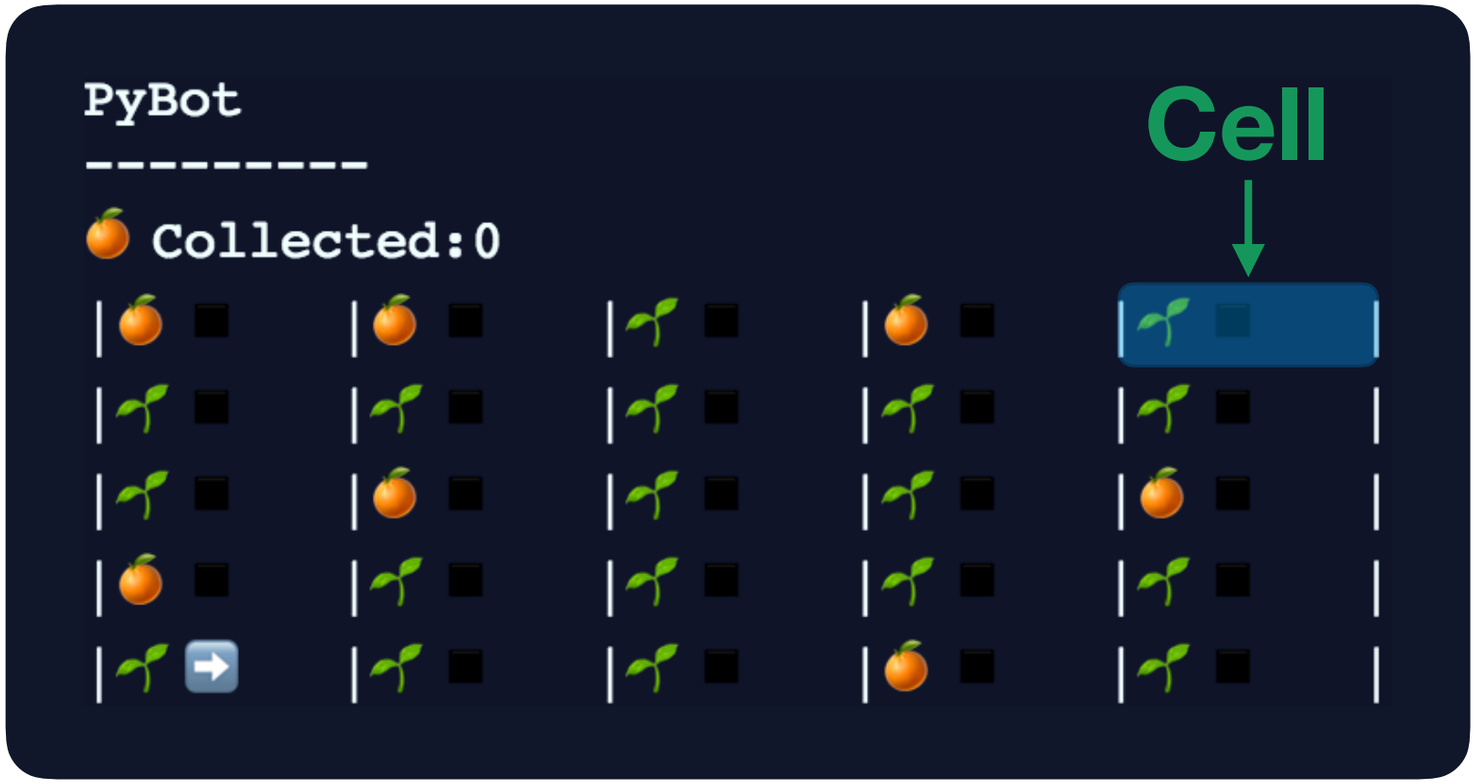
PyCountry **Fields**



PyCountry **Fields**

NORTH

WEST



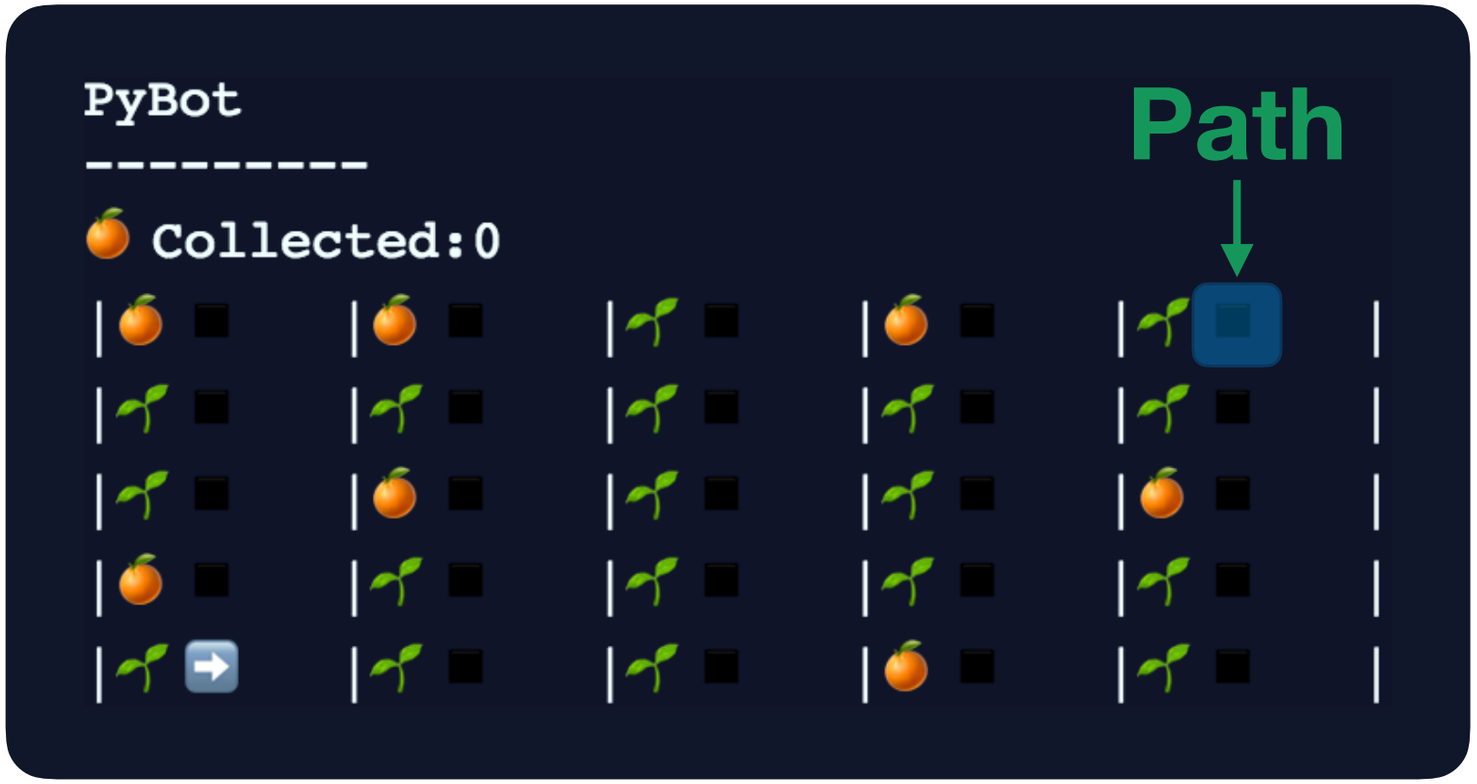
EAST

SOUTH

PyCountry **Fields**

NORTH

WEST



EAST

SOUTH

PyCountry **Fields**

NORTH

WEST

EAST

SOUTH

PyCountry **Fields**

NORTH

WEST

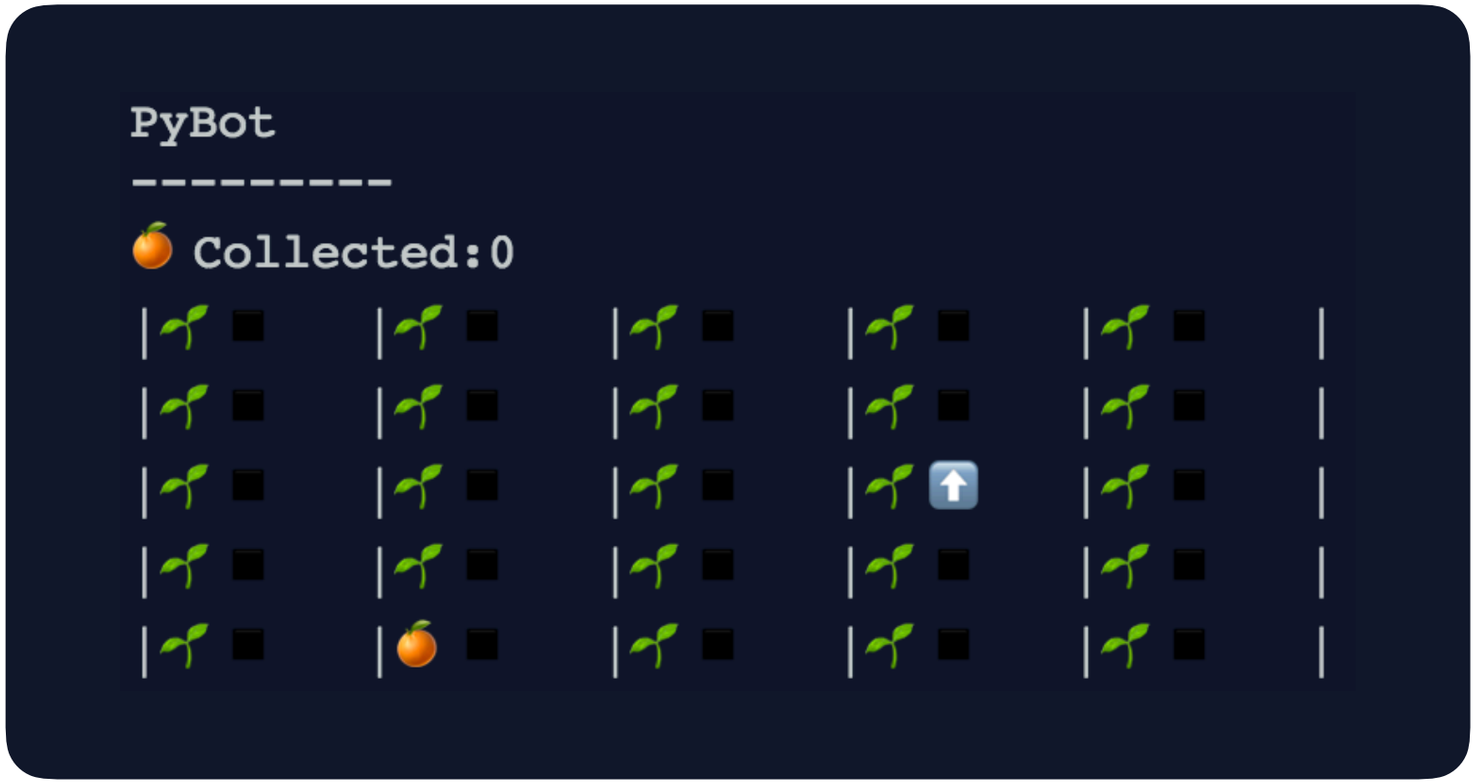
EAST

SOUTH

PyBot **Position**

NORTH

WEST



EAST

SOUTH

What can **PyBot** do?

Move!

PyBot moves forward one cell in the direction she is facing.

BEFORE

Move!

PyBot moves forward one cell in the direction she is facing.

AFTER

Pick Oranges!

PyBot picks the fruit in the current cell.

BEFORE

Pick Oranges!

PyBot picks the fruit in the current cell.

AFTER

Turn Right!

PyBot rotates 90 degrees to the right, facing a new direction.

BEFORE

Turn Right!

PyBot rotates 90 degrees to the right, facing a new direction.

AFTER

PyBot **Actions**

PyBot can't turn left!

PyBot **Crashes**

1) **If** PyBot moves off the edge of the board, **she crashes**

```
PYBOT
---------
🍊 Collected:0
|🌱■   |🌱■   |🌱■   |🌱■   |🌱■   |
|🌱■   |🌱■   |🌱■   |🌱■   |🌱■   |
|🌱■   |🌱■   |🌱■   |🌱■   |🌱■   |
|🌱■   |🌱■   |🌱■   |🌱■   |🌱■   |
|🌱■   |🌱■   |🌱■   |🌱■   |🌱➡️  |
Crash: You moved off the edge of the world!
```

PyBot **Crashes**

2) **If** PyBot tries to pick a fruit where there is no fruit, **she crashes**

```
PYBOT
---------
🍊 Collected:0
|🌱 ■    |🌱 ■    |🌱 ■    |🌱 ■    |🌱 ■    |
|🌱 ■    |🌱 ■    |🌱 ■    |🌱 ■    |🌱 ■    |
|🌱 ■    |🌱 ■    |🌱 ■    |🌱 ■    |🌱 ■    |
|🌱 ■    |🌱 ⬆️   |🌱 ■    |🌱 ■    |🌱 ■    |
|🌱 ■    |🌱 ■    |🌱 ■    |🌱 ■    |🌱 ■    |
Crash: You tried to pick up a fruit where there was no fruit!
```

True or False

Questions you can ask **PyBot!**

Is there an orange?

Does PyBot's current cell have an orange in it?

FALSE

TRUE

Facing a wall?

Is PyBot facing the edge of the field?

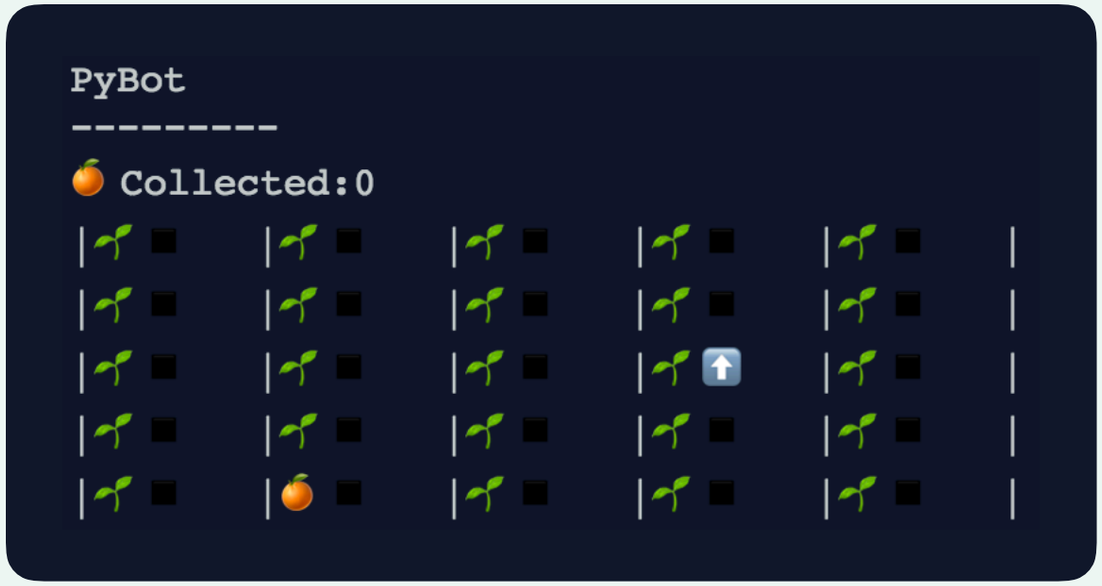
FALSE

TRUE

Facing north?

Is PyBot facing north ?

FALSE



TRUE

We can program PyBot using Python **functions**

PyBot **Functions**

!

`move()` PyBot moves forward one cell in the direction she is facing.

BEFORE

AFTER

PyBot **Functions**

!

`turn_right()` PyBot rotates 90 degrees to the right, facing a new direction.

BEFORE

AFTER

PyBot **Functions**

!

`pick_fruit()` PyBot picks the fruit in the current cell.

BEFORE

AFTER

PyBot **Functions**

?

`has_fruit()` Returns **True** if PyBot's current cell has an orange.

FALSE

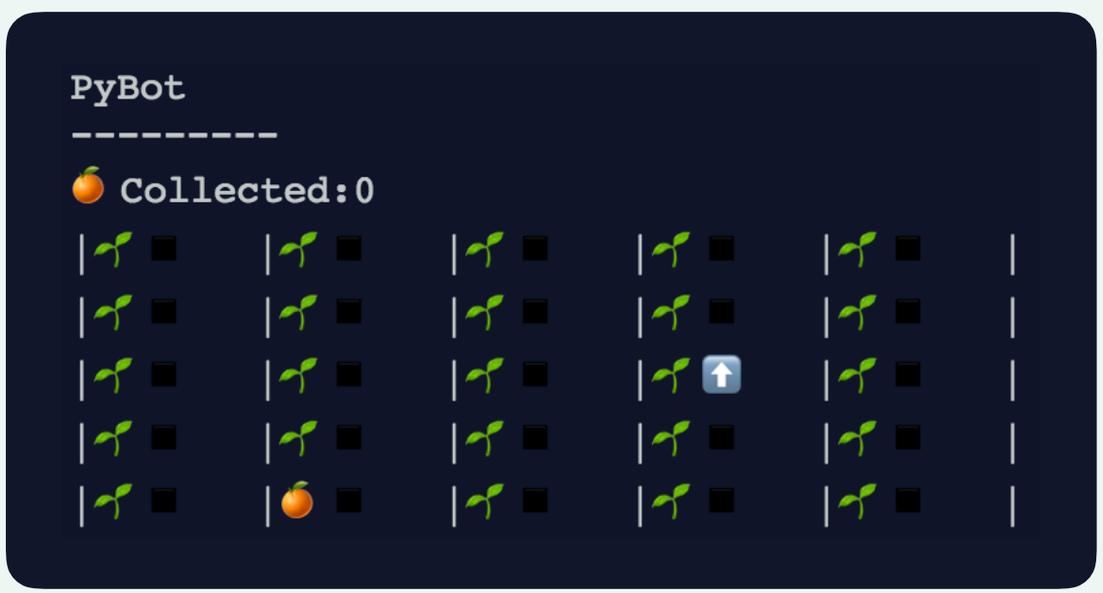
TRUE

PyBot **Functions**

?

`is_facing_north()` Returns **True** if PyBot is facing north.

FALSE



TRUE

Example: Harvest a Fruit

Print Name

Harvest Your First Fruit

Turn Left

What are **functions**?

```
move()
turn_right()
pick_fruit()
has_fruit()
is_facing_north()
is_facing_east()
is_facing_south()
is_facing_west()
```

These are **functions**.

Definition

Function - *Code that is grouped together and packaged under a name, so it can be* ***executed*** *in one line.*

Definition

Execute - *To make the computer do something.*

Function **Structure**

The "**def**" keyword

```
def this_is_a_function():
    """
    This is an example function for the class notes.
    """
    if not front_is_blocked():
        move()
    turn_right()
    turn_right()
    move()
    move()
```

Function **Structure**

The "**def**" keyword

The function **name** + " **( )** " + " **:** "

```
def this_is_a_function():
    """
    This is an example function for the class notes.
    """
    if not front_is_blocked():
        move()
    turn_right()
    turn_right()
    move()
    move()
```

Function **Structure**

The "**def**" keyword

The function **name** + " **( )** " + " **:** "

```
def this_is_a_function():
    """
    This is an example function for the class notes.
    """
    if not front_is_blocked():
        move()
    turn_right()
    turn_right()
    move()
    move()
```

The function **body**

Function **Structure**

The "**def**" keyword

The function **name** + " **( )** " + " **:** "

```
def this_is_a_function():
    """
    This is an example function for the class notes.
    """
    if not front_is_blocked():
        move()
    turn_right()
    turn_right()
    move()
    move()
```

The function **body**

Function **Structure**

The "**def**" keyword

The function **name** + " **( )** " + " **:** "

```
def this_is_a_function():
    """
    This is an example function for the class notes.
    """
    if not front_is_blocked():
        move()
    turn_right()
    turn_right()
    move()
    move()
```

*Functions can be **called** from other functions!*

The function **body**

Definition

Call - *To execute the code packaged within a function.*

Function **Structure**

I "**called**" `this_is_a_function()`

is the same thing as…

I "**executed**"

```
if not front_is_blocked():
    move()
turn_right()
turn_right()
move()
move()
```

Example: Pick and Move Function

Function **Structure**

Print Name

Harvest Your First Fruit

Turn Left

Function **Implementation**

We **implement** functions in order to **decompose** our code.

Definition

Implement - To ~write~ code! The word for a specific instance of written code is "*implementation*."

Definition

Decompose - To break apart code into small, reusable pieces.

Function **Implementation**

```python
def main():
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    turn_right()
    turn_right()
    turn_right()
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    turn_right()
    turn_right()
    turn_right()

if __name__ == '__main__':
    main()
```

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

def pick_and_move():
    pick_fruit()
    move()

def pick_fruit_across():
    pick_and_move()
    pick_and_move()
    pick_and_move()
    pick_and_move()

def main():
    pick_fruit_across()
    turn_left()
    pick_fruit_across()
    turn_left()

if __name__ == '__main__':
    main()
```

Function **Implementation**

```
def main():
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    turn_right()
    turn_right()
    turn_right()
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    turn_right()
    turn_right()
    turn_right()

if __name__ == '__main__':
    main()
```

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

def pick_and_move():
    pick_fruit()
    move()

def pick_fruit_across():
    pick_and_move()
    pick_and_move()
    pick_and_move()
    pick_and_move()

def main():
    pick_fruit_across()
    turn_left()
    pick_fruit_across()
    turn_left()

if __name__ == '__main__':
    main()
```

These code segments do the *same thing*.

Function **Implementation**

```python
def main():
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    turn_right()
    turn_right()
    turn_right()
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    turn_right()
    turn_right()
    turn_right()

if __name__ == '__main__':
    main()
```

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

def pick_and_move():
    pick_fruit()
    move()

def pick_fruit_across():
    pick_and_move()
    pick_and_move()
    pick_and_move()
    pick_and_move()

def main():
    pick_fruit_across()
    turn_left()
    pick_fruit_across()
    turn_left()

if __name__ == '__main__':
    main()
```

These code segments do the *same thing*.

The one on the right is well *decomposed*.

Function **Implementation**

```
def main():
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    turn_right()
    turn_right()
    turn_right()
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    turn_right()
    turn_right()
    turn_right()

if __name__ == '__main__':
    main()
```

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

def pick_and_move():
    pick_fruit()
    move()

def pick_fruit_across():
    pick_and_move()
    pick_and_move()
    pick_and_move()

def main():
    pick_fruit_across()
    turn_left()
    pick_fruit_across()
    turn_left()

if __name__ == '__main__':
    main()
```

These code segments do the *same thing*.

The one on the right is well *decomposed*.

It is not only *shorter*, but also *easier to read*.

Function **Implementation**

```python
def main():
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    turn_right()
    turn_right()
    turn_right()
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    turn_right()
    turn_right()
    turn_right()

if __name__ == '__main__':
    main()
```

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

def pick_and_move():
    pick_fruit()
    move()

def pick_fruit_across():
    pick_and_move()
    pick_and_move()
    pick_and_move()

def main():
    pick_fruit_across()
    turn_left()
    pick_fruit_across()
    turn_left()

if __name__ == '__main__':
    main()
```

These code segments do the *same thing*.

The one on the right is well *decomposed*.

It is not only *shorter*, but also *easier to read*.

It is also easier to *fix*.

Function **Implementation**

```python
def main():
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    turn_right()
    turn_right()
    turn_right()
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    turn_right()
    turn_right()
    turn_right()

if __name__ == '__main__':
    main()
```

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

def pick_and_move():
    pick_fruit()
    move()

def pick_fruit_across():
    pick_and_move()
    pick_and_move()
    pick_and_move()

def main():
    pick_fruit_across()
    turn_left()
    pick_fruit_across()
    turn_left()

if __name__ == '__main__':
    main()
```

These code segments do the *same thing*.

The one on the right is well *decomposed*.

It is not only *shorter*, but also *easier to read*.

It is also easier to *fix*.

Function **Implementation**

What if we wanted to change `pick_fruit()` to `pick_vegetable()`?

Function **Implementation**

8 lines :(

```python
def main():
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    turn_right()
    turn_right()
    turn_right()
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    pick_fruit()
    move()
    turn_right()
    turn_right()
    turn_right()

if __name__ == '__main__':
    main()
```

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

def pick_and_move():
    pick_fruit()
    move()

def pick_fruit_across():
    pick_and_move()
    pick_and_move()
    pick_and_move()
    pick_and_move()

def main():
    pick_fruit_across()
    turn_left()
    pick_fruit_across()
    turn_left()

if __name__ == '__main__':
    main()
```

1 line :D

Function **Implementation**

Let's start working on this week's project!

Project: Introducing You

Recap

repl.it = Where we will be coding.

PyBot = Your new best friend. Learn her set of commands!

Functions are little packages of code.

Implement functions to *decompose* and *make your life easier.*