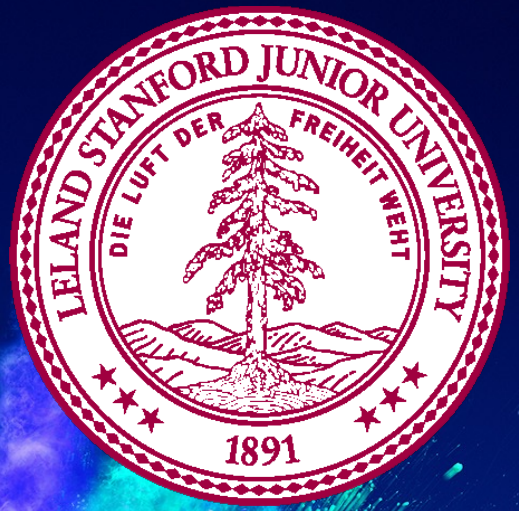


Graphics

Chris Piech and Mehran Sahami
CS106A, Stanford University

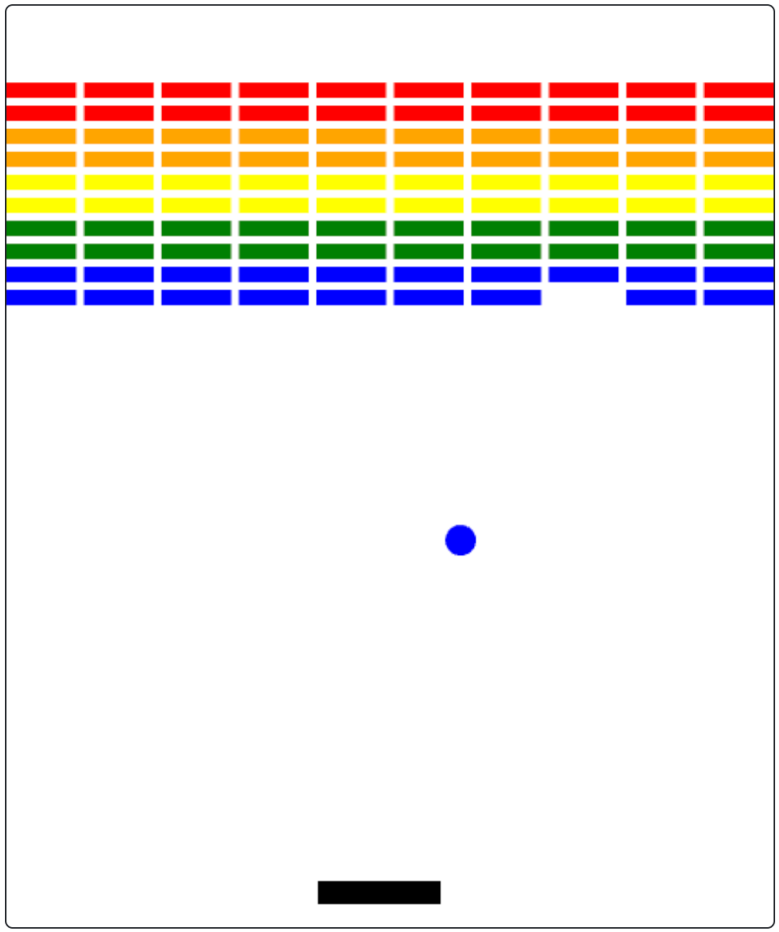
Breakout Demo

Created by Chris Piech

```
Hello! I made this for CS106A!
```

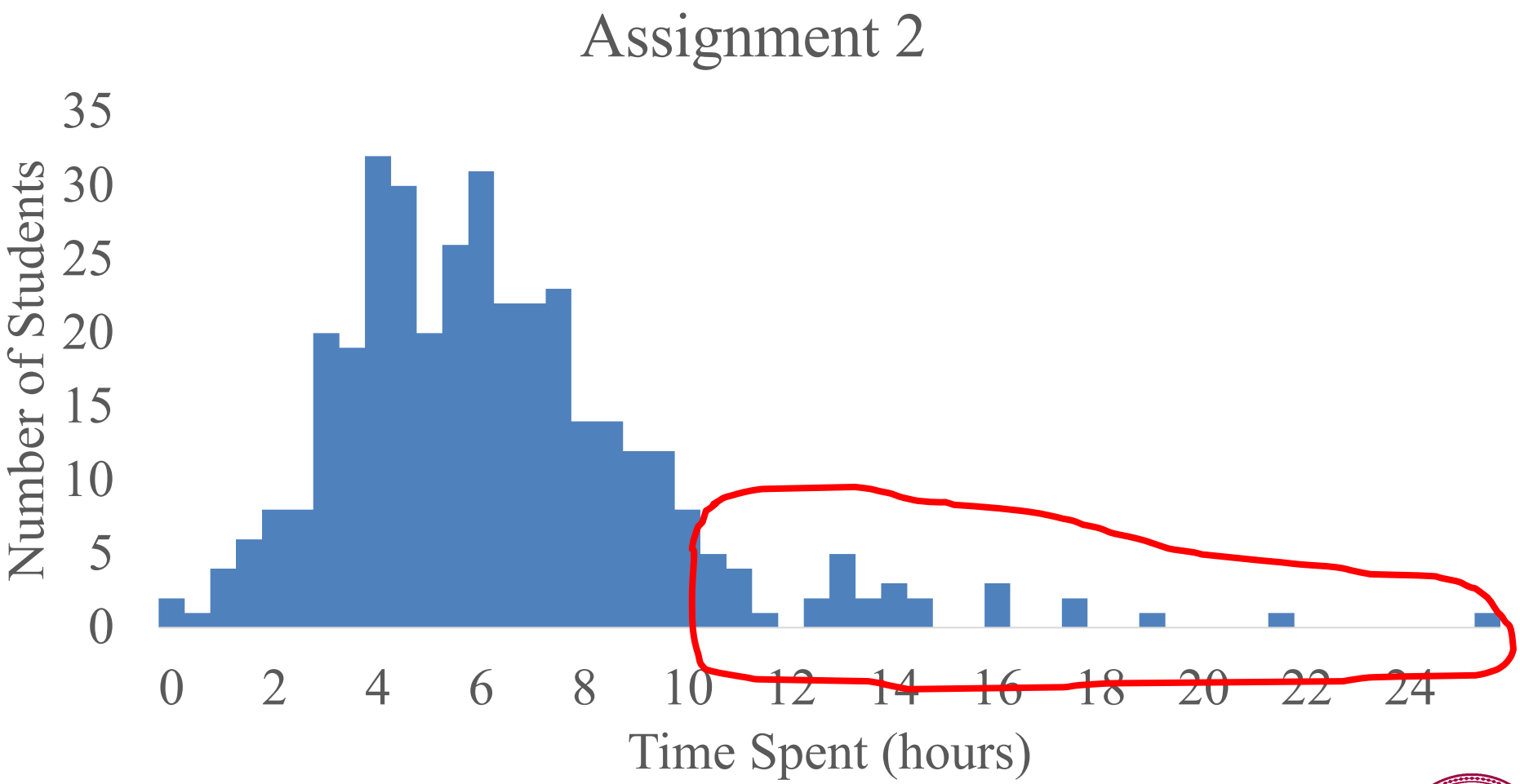
Editable (Owner)

Run

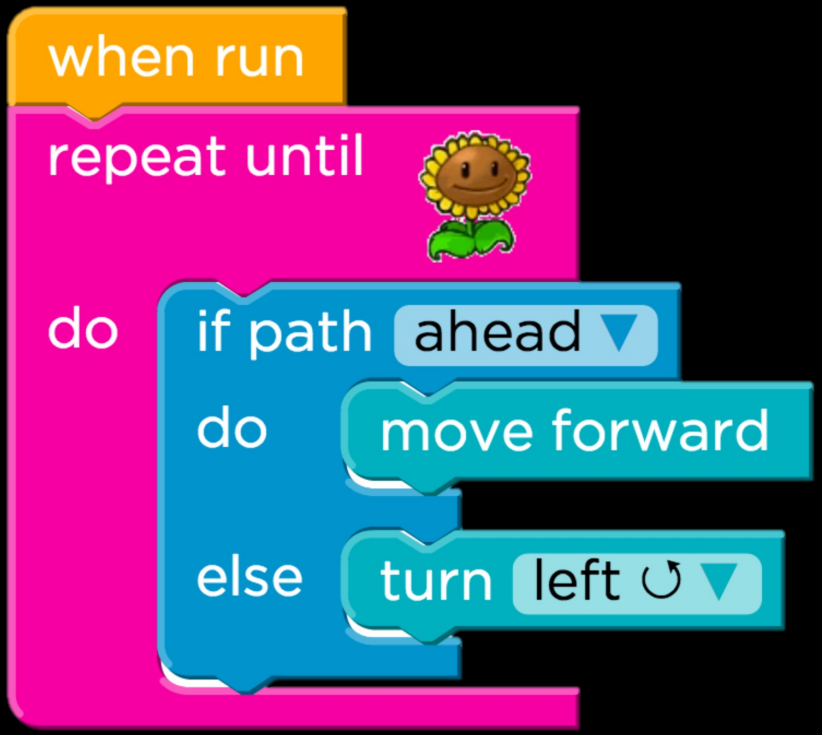


Get a little meta…

Learn by Doing



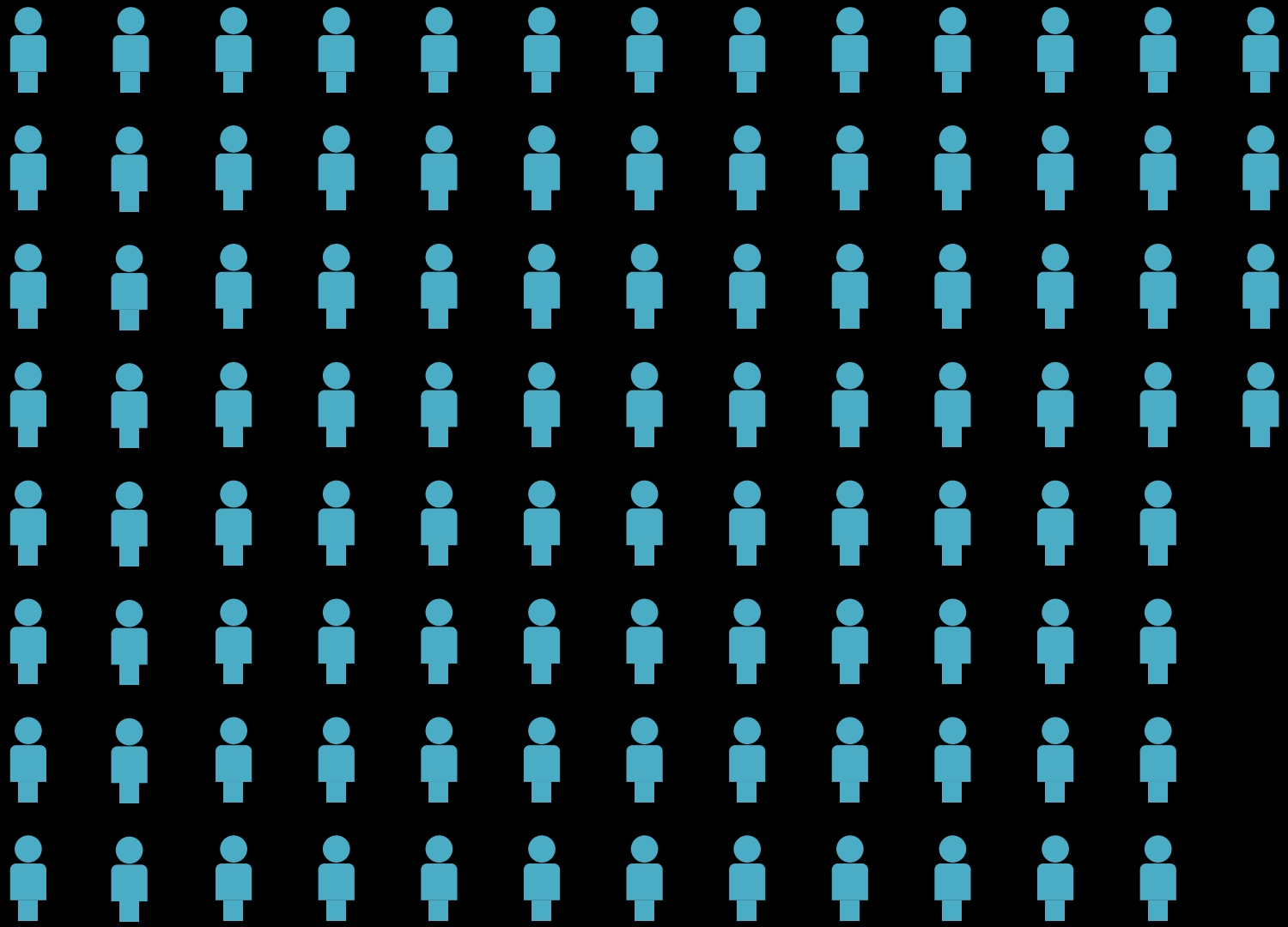
Learning to Program on the Internet



Task

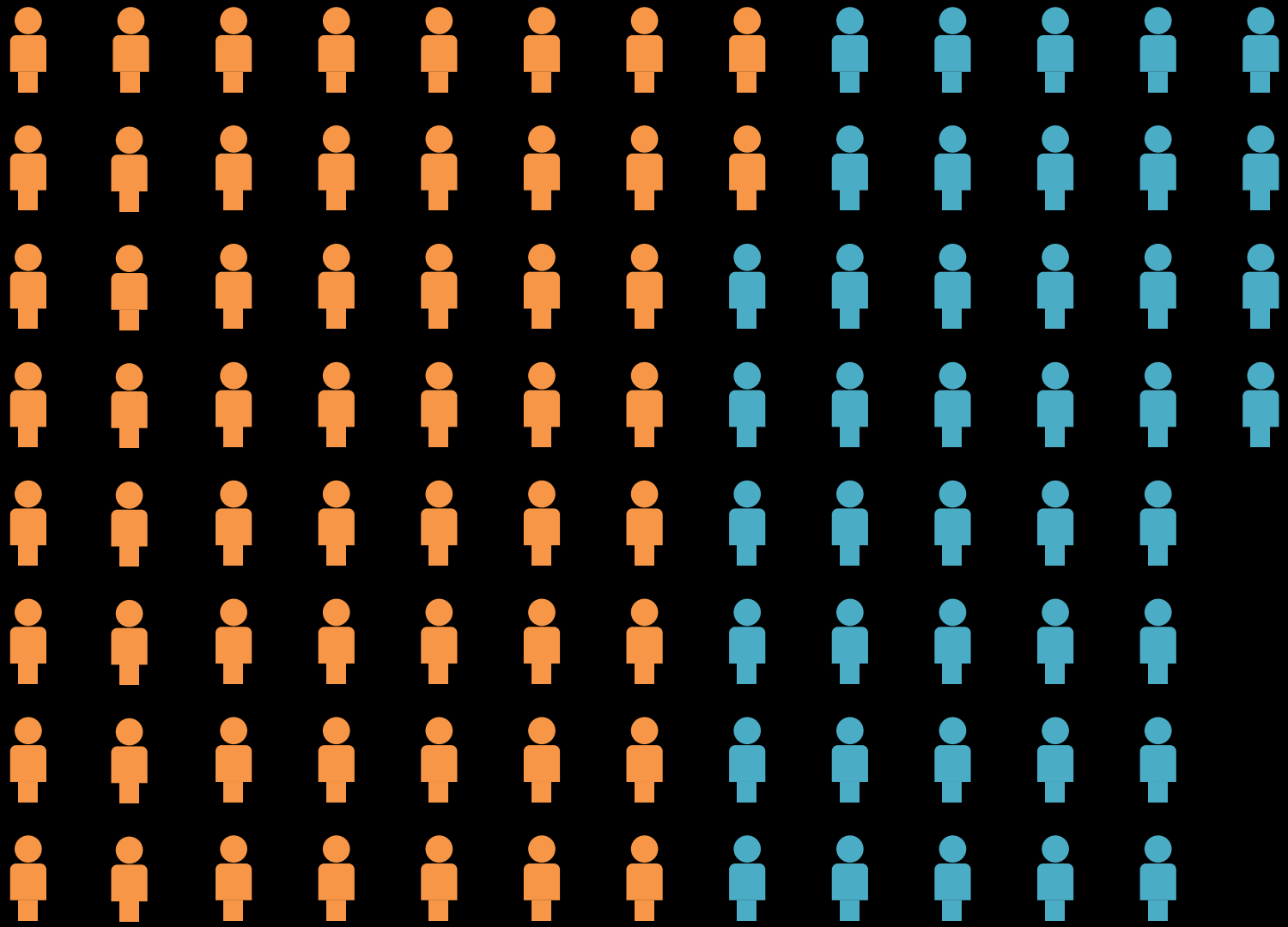
Almost a hundred thousand unique solutions

US K-12 Students

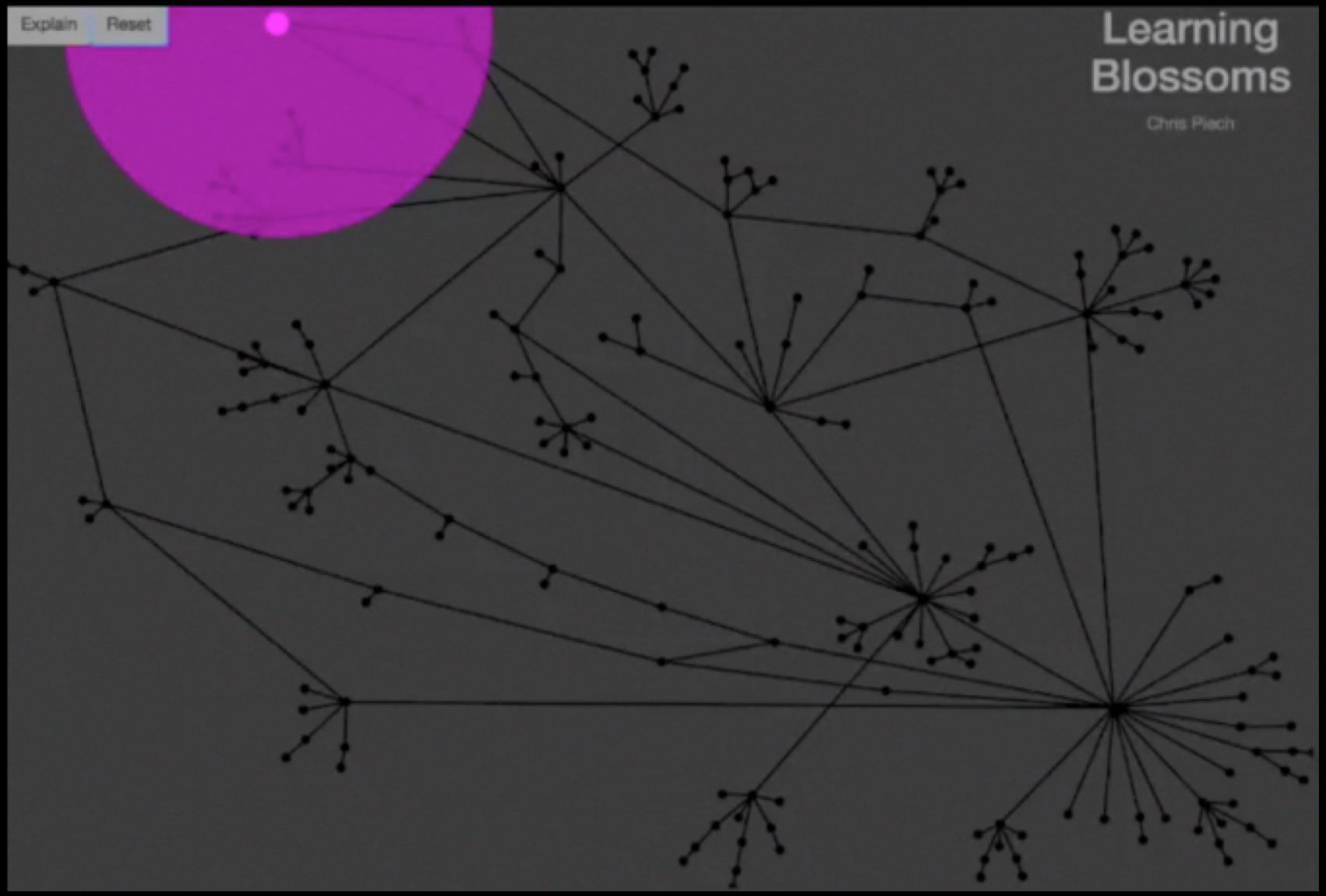


= 500,000 learners

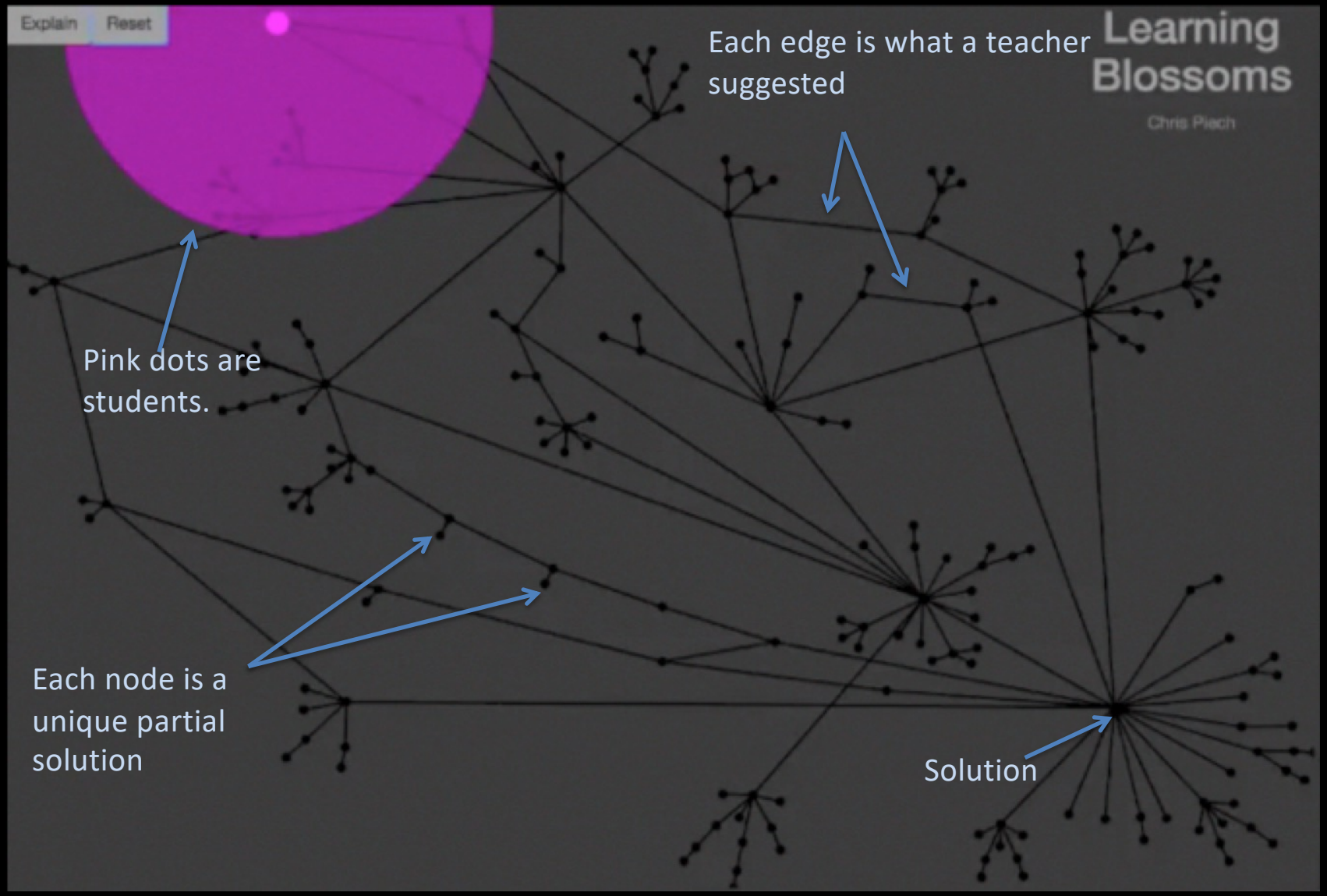
Code.org Students



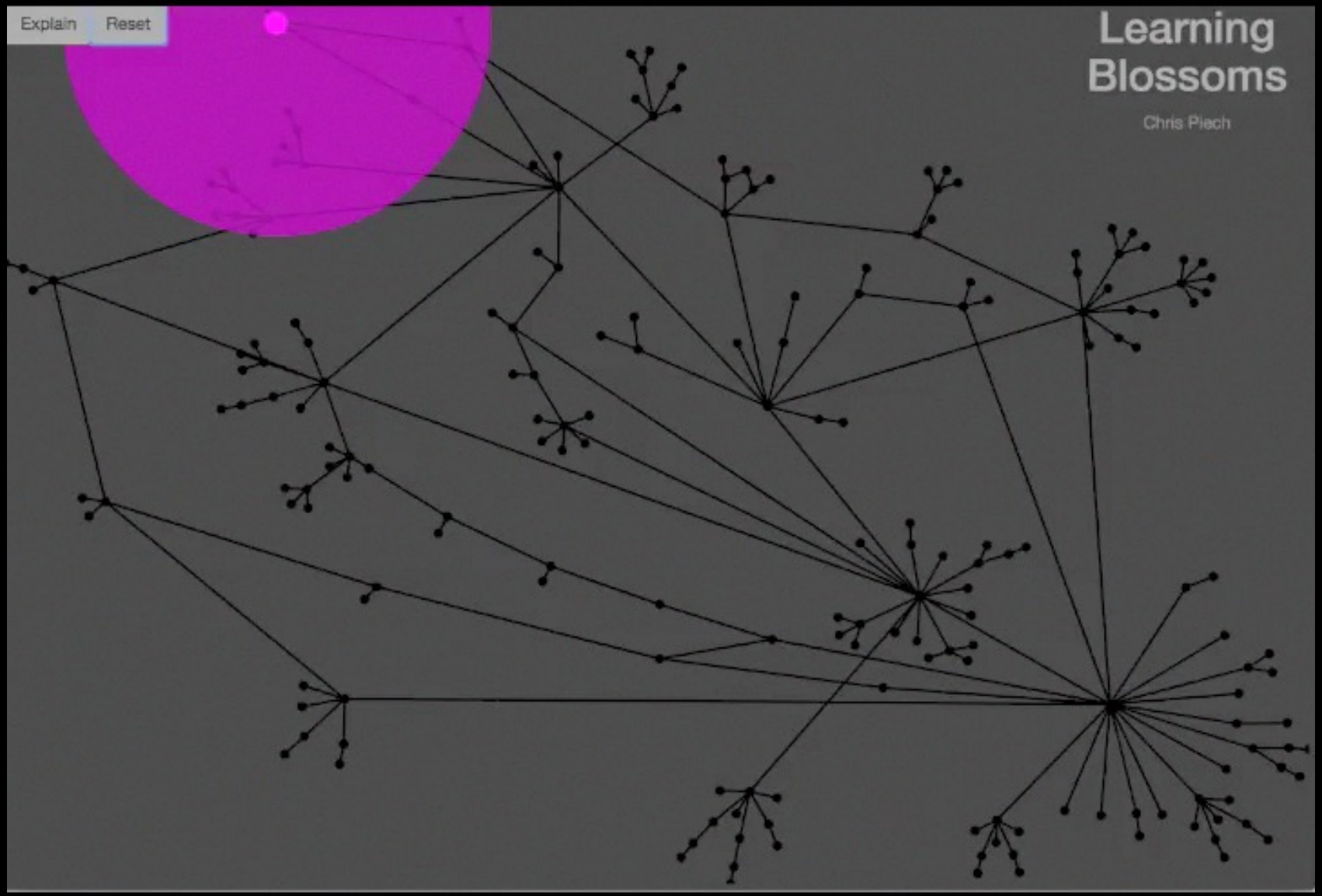
= 500,000 learners



Autonomously Generating Hints by Inferring Problem Solving Policies - Piech, Sahami et al.



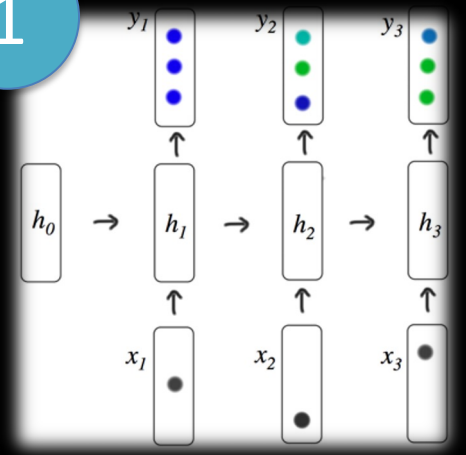
Autonomously Generating Hints by Inferring Problem Solving Policies - Piech, Sahami et al.



Autonomously Generating Hints by Inferring Problem Solving Policies - Piech, Sahami et al.

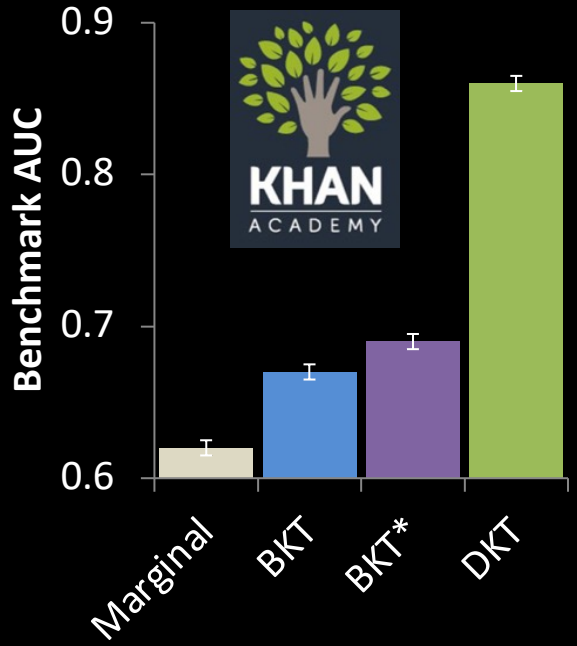
Deep Learning Algorithms

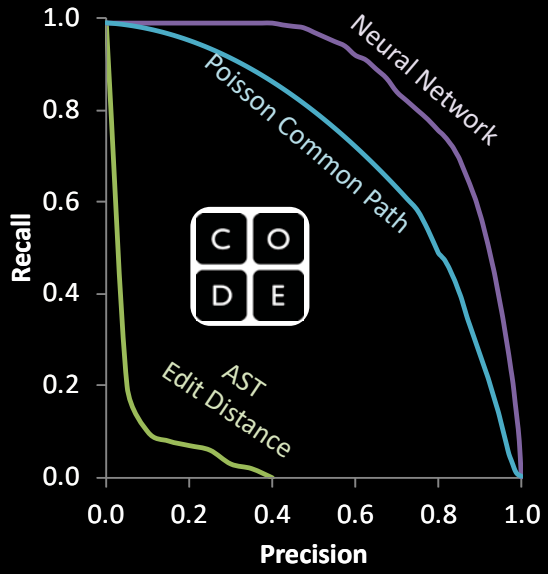
1



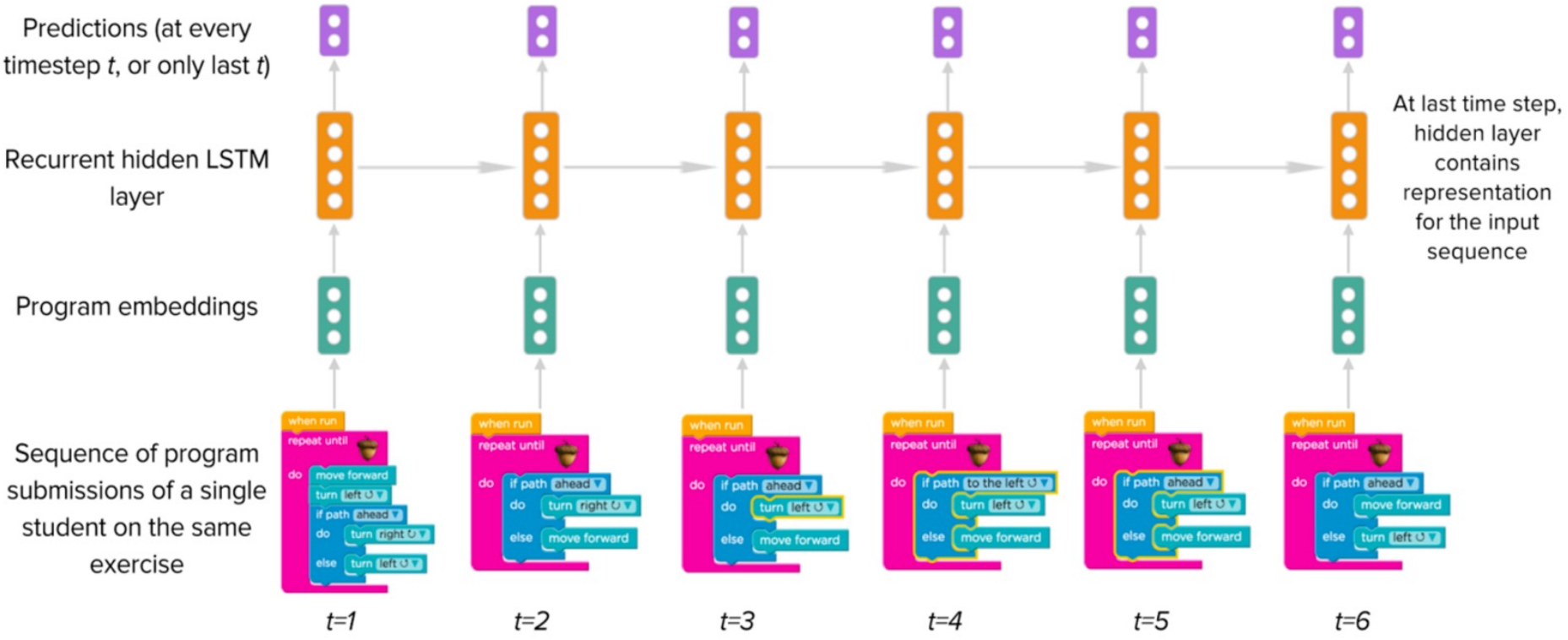
2

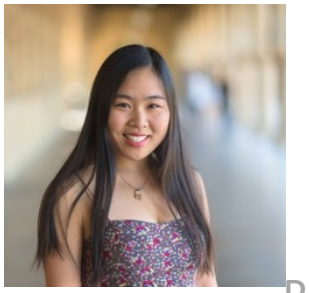
Program $\rightarrow$ $\mathbb{R}^n$





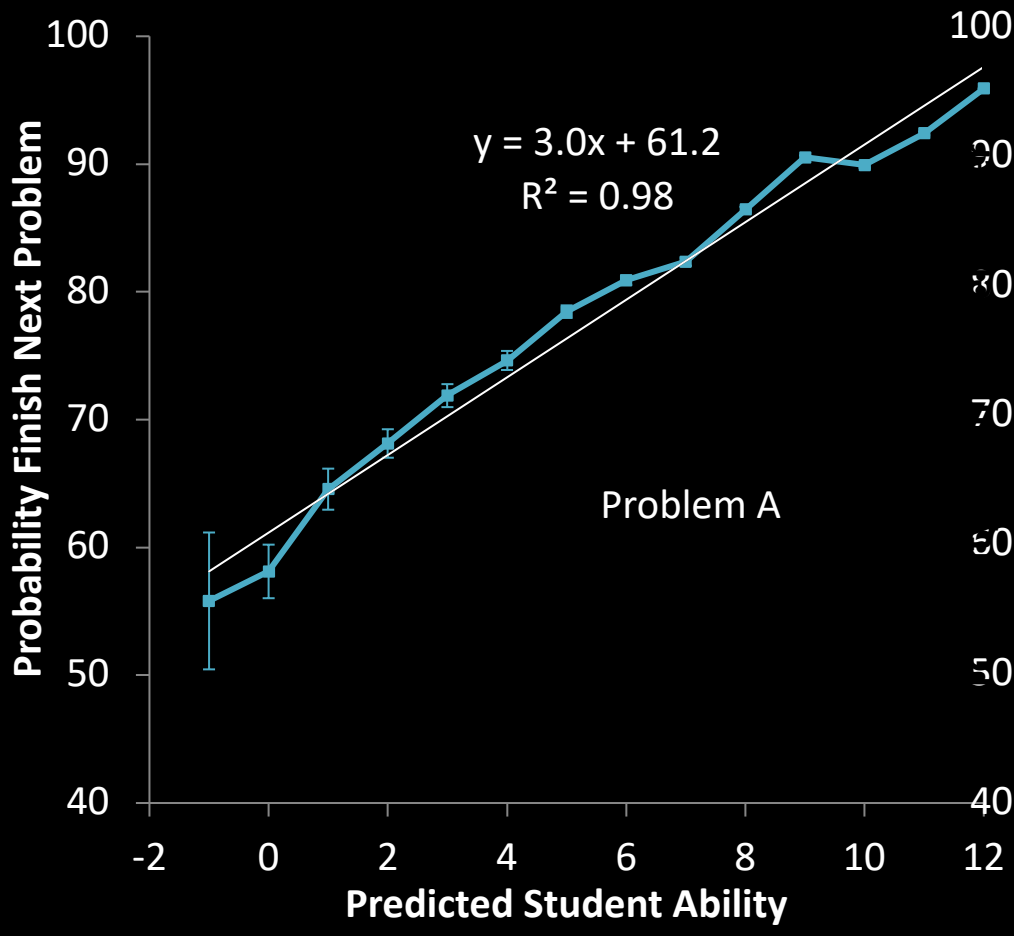
Deep Learning on Trajectories

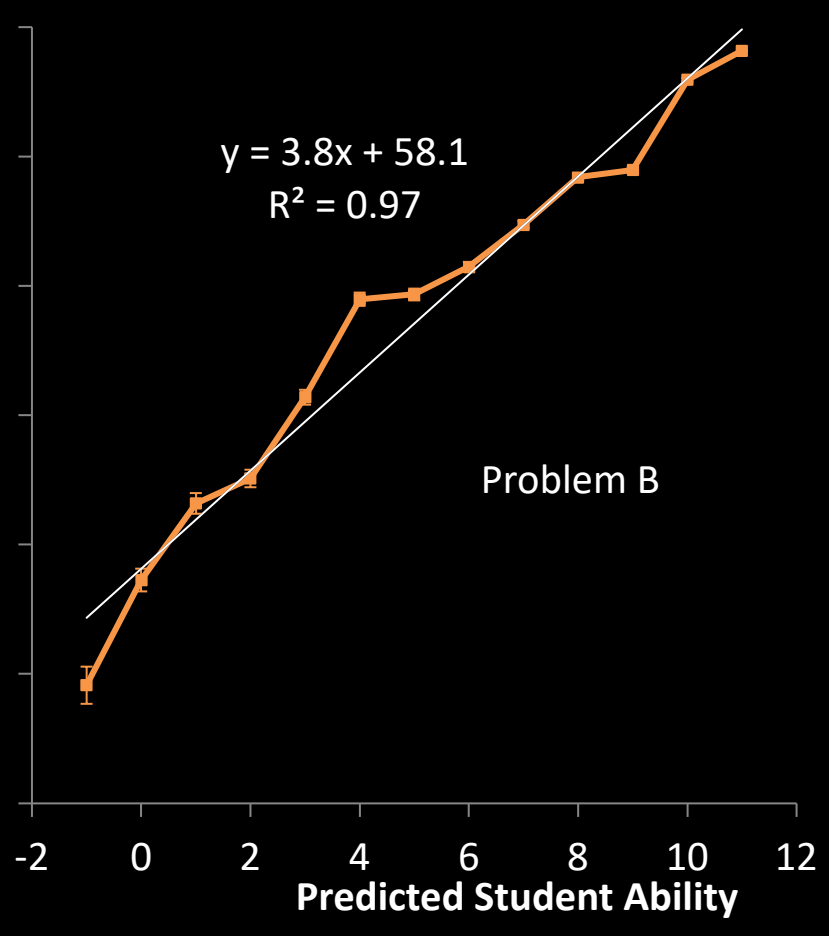




Research in collaboration with Lisa Wang

Predicts Future Success

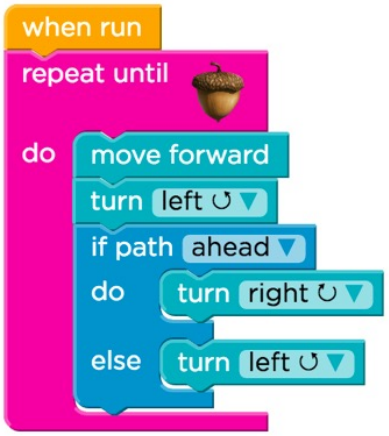




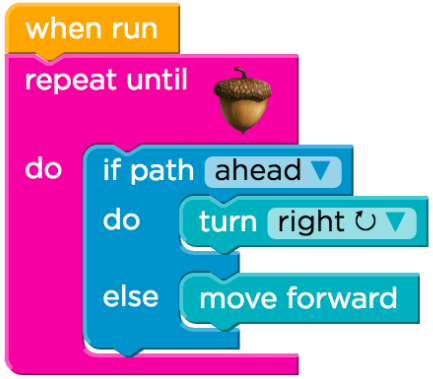
Most likely to succeed?

Highly Rates Grit

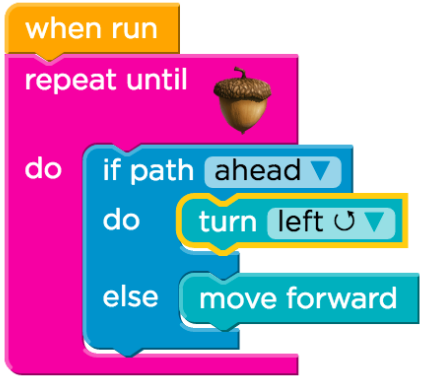
1. Two compound errors



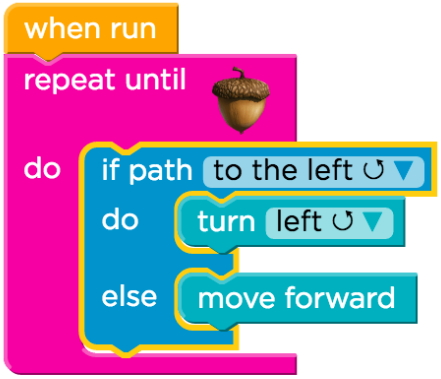
2. Solves first error



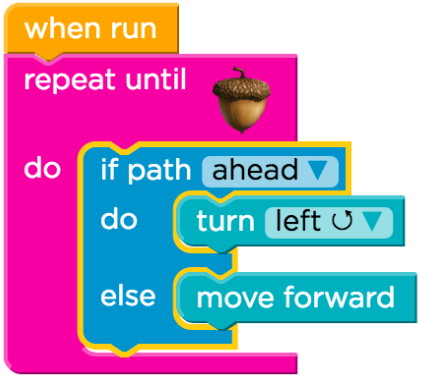
3. Starts reasonable attempt



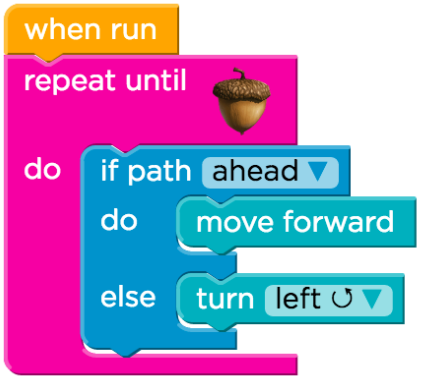
4. Completes attempt



5. Backtracks

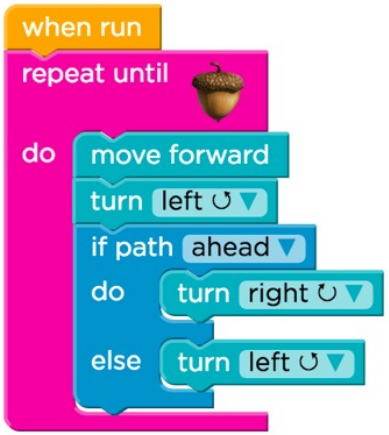


6. Finds solution



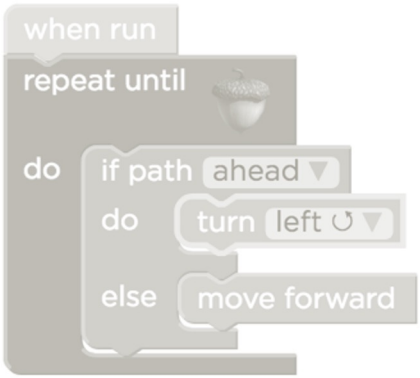
Highly Rates Grit

1. Two compound errors



2. Solves first error

3. Starts reasonable attempt



4. Completes attempt

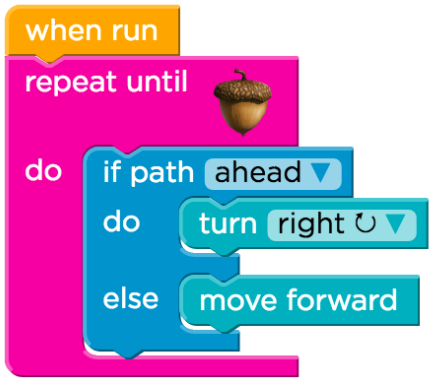
5. Backtracks

6. Finds solution

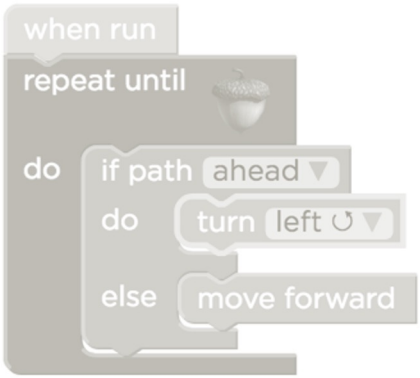
Highly Rates Grit

1. Two compound errors

2. Solves first error



3. Starts reasonable attempt



4. Completes attempt

5. Backtracks

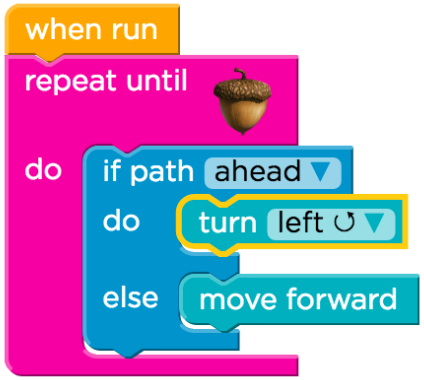
6. Finds solution

Highly Rates Grit

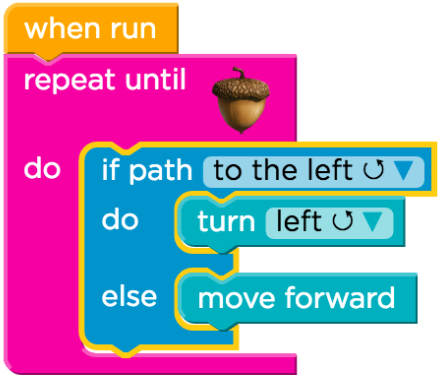
1. Two compound errors

2. Solves first error

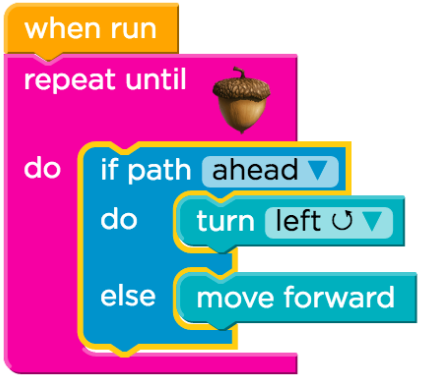
3. Starts reasonable attempt



4. Completes attempt



5. Backtracks



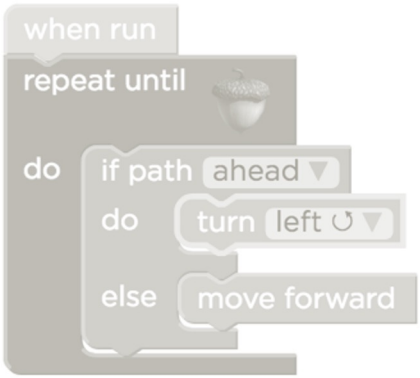
6. Finds solution

Highly Rates Grit

1. Two compound errors

2. Solves first error

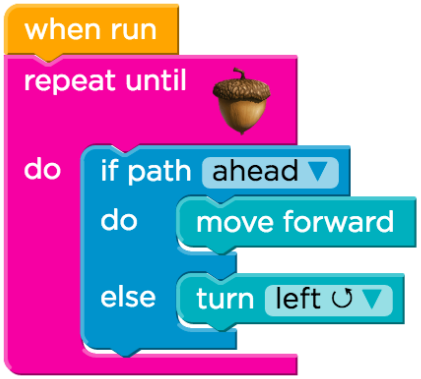
3. Starts reasonable attempt



4. Completes attempt

5. Backtracks

6. Finds solution

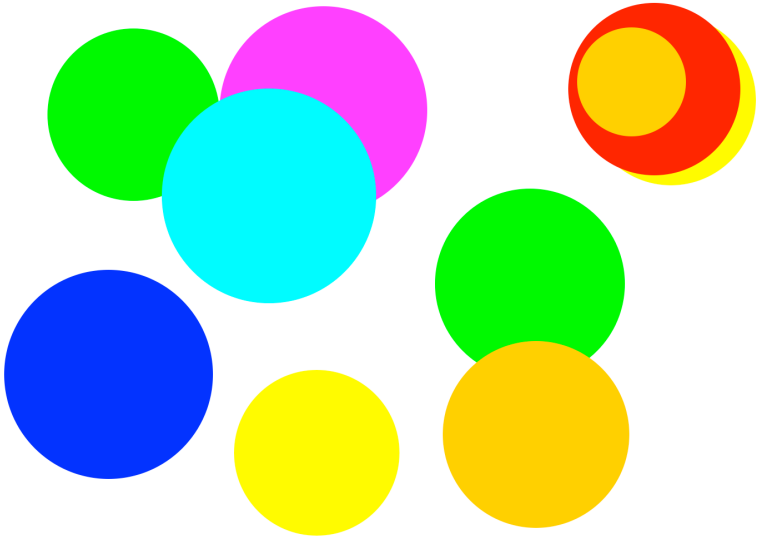


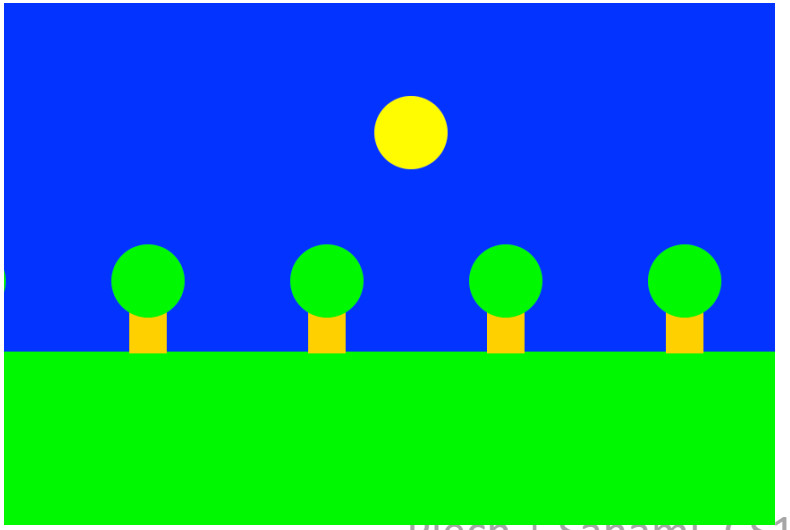
Coding is a new way to think
Keep challenging yourself.

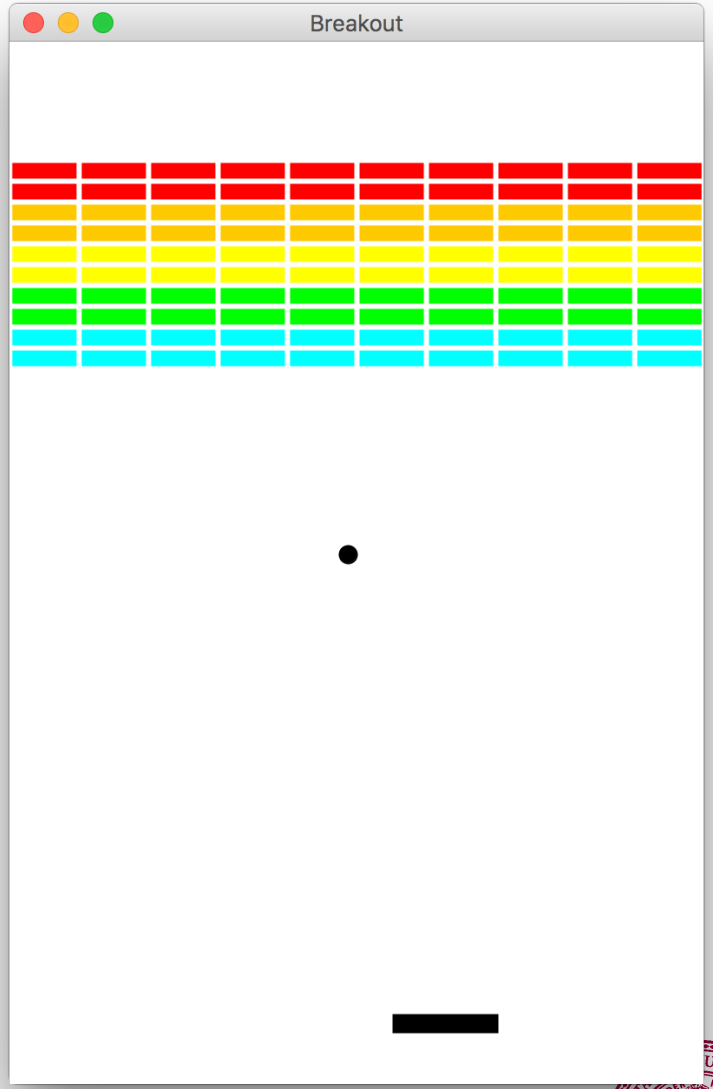
Today's Goal

1. How do I draw things?

Graphics Programs



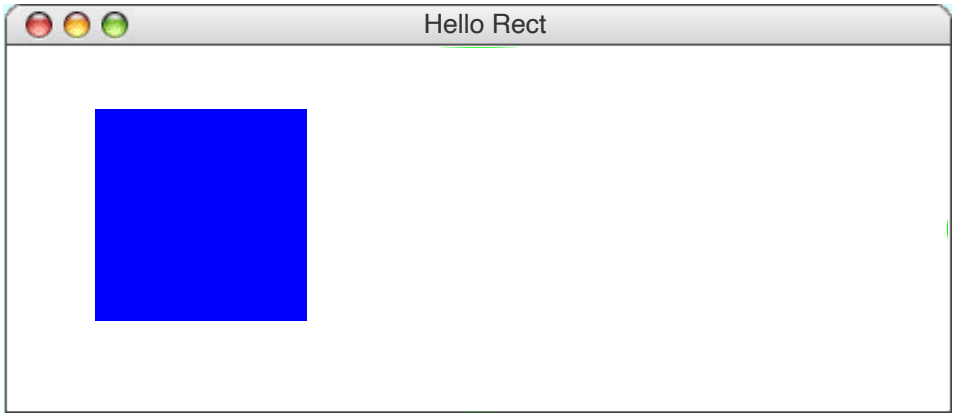




Draw a Rectangle

the following `main` method displays a blue square

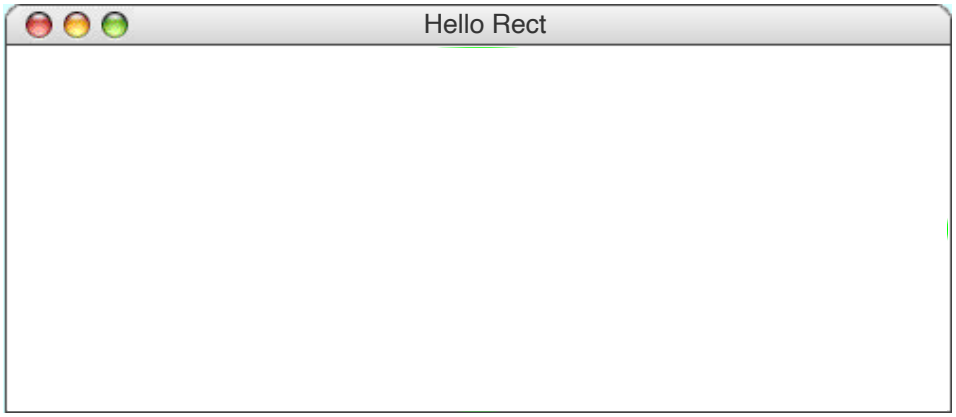
```
def main():
    canvas = Canvas(800, 200)
    canvas.create_rectangle(20, 20, 100, 100, "blue")
```



Draw a Rectangle

the following `main` method displays a blue square

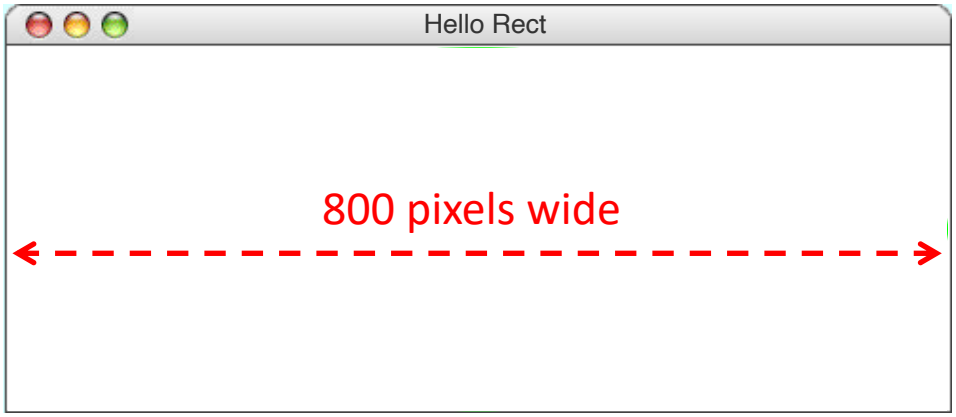
```
def main():
    canvas = Canvas(800, 200)
```



Draw a Rectangle

the following `main` method displays a blue square

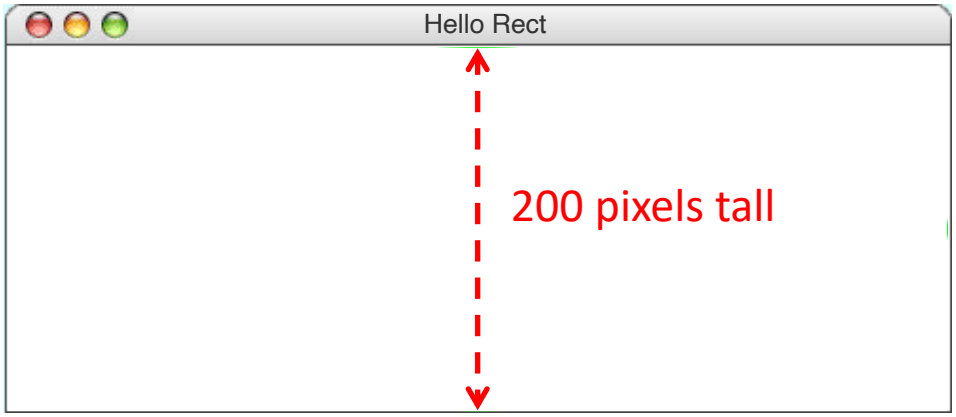
```
def main():
    canvas = Canvas(800, 200)
```



Draw a Rectangle

the following `main` method displays a blue square

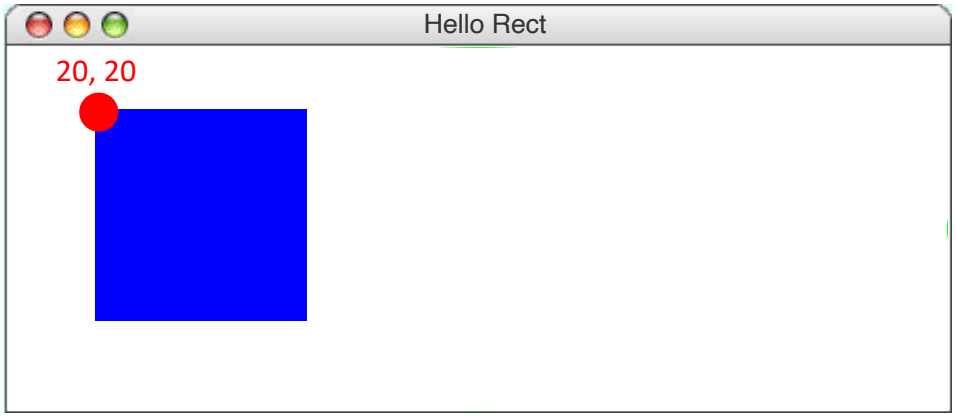
```
def main():
    canvas = Canvas(800, 200)
```



Draw a Rectangle

the following `main` method displays a blue square

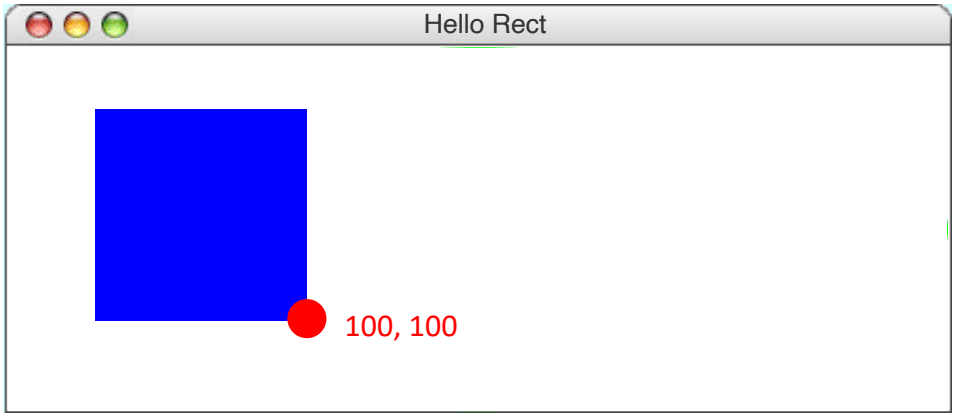
```
def main():
    canvas = Canvas(800, 200)
    canvas.create_rectangle(20, 20, 100, 100, "blue")
```



Draw a Rectangle

the following `main` method displays a blue square

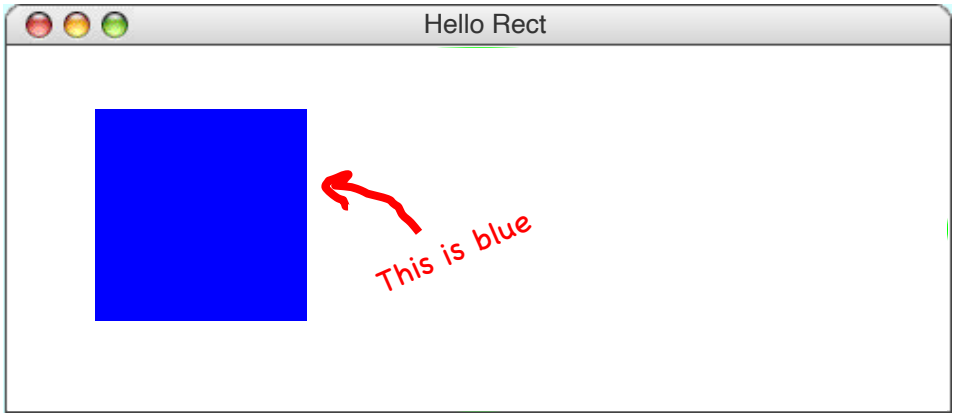
```
def main():
    canvas = Canvas(800, 200)
    canvas.create_rectangle(20, 20, 100, 100, "blue")
```



Draw a Rectangle

the following `main` method displays a blue square

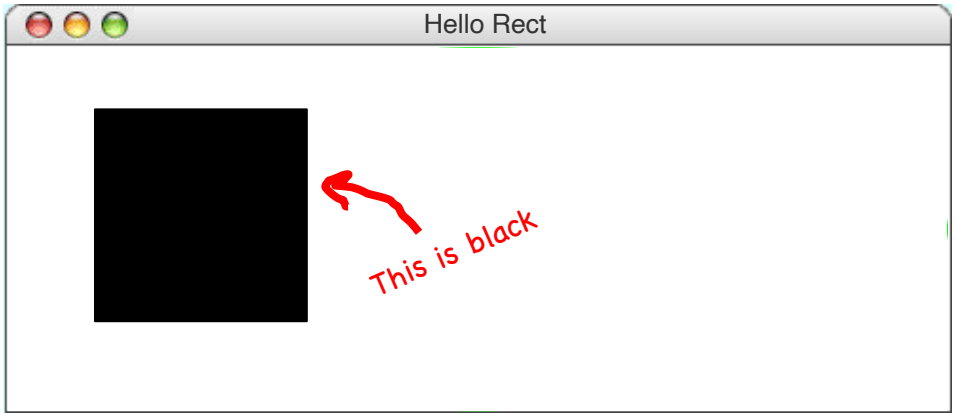
```
def main():
    canvas = Canvas(800, 200)
    canvas.create_rectangle(20, 20, 100, 100, "blue")
```



Draw a Rectangle

the following `main` method displays a blue square

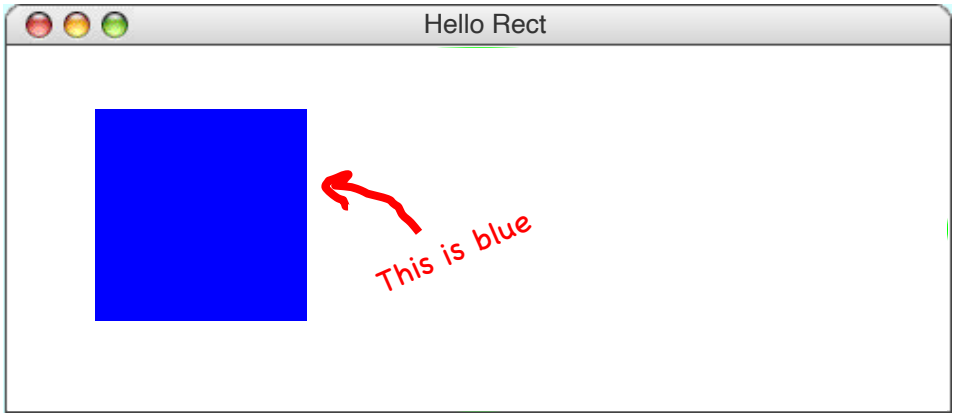
```
def main():
    canvas = Canvas(800, 200)
    canvas.create_rectangle(20, 20, 100, 100)
```



Draw a Rectangle

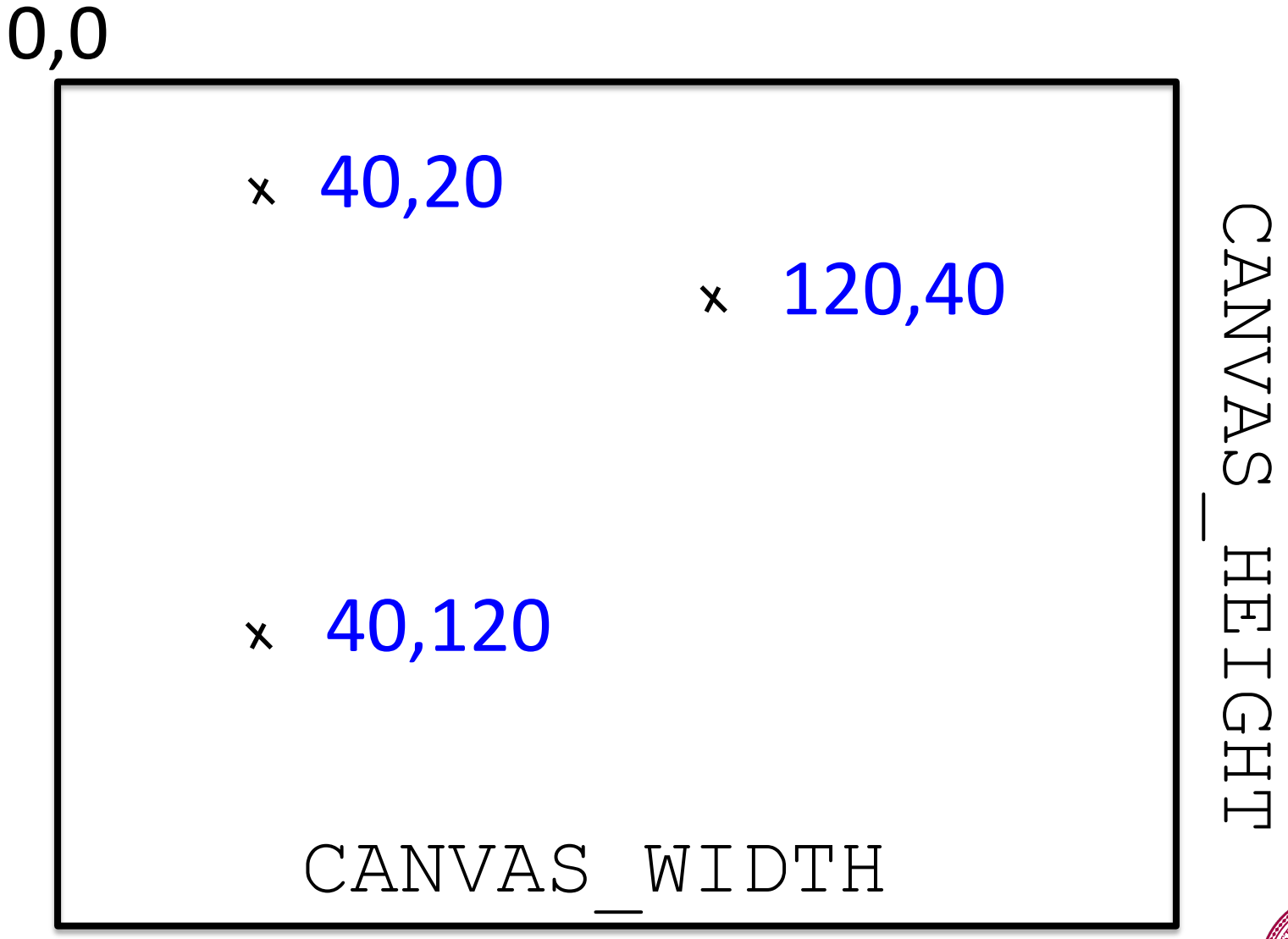
the following `main` method displays a blue square

```
def main():
    canvas = Canvas(800, 200)
    canvas.create_rectangle(20, 20, 100, 100, "blue")
```

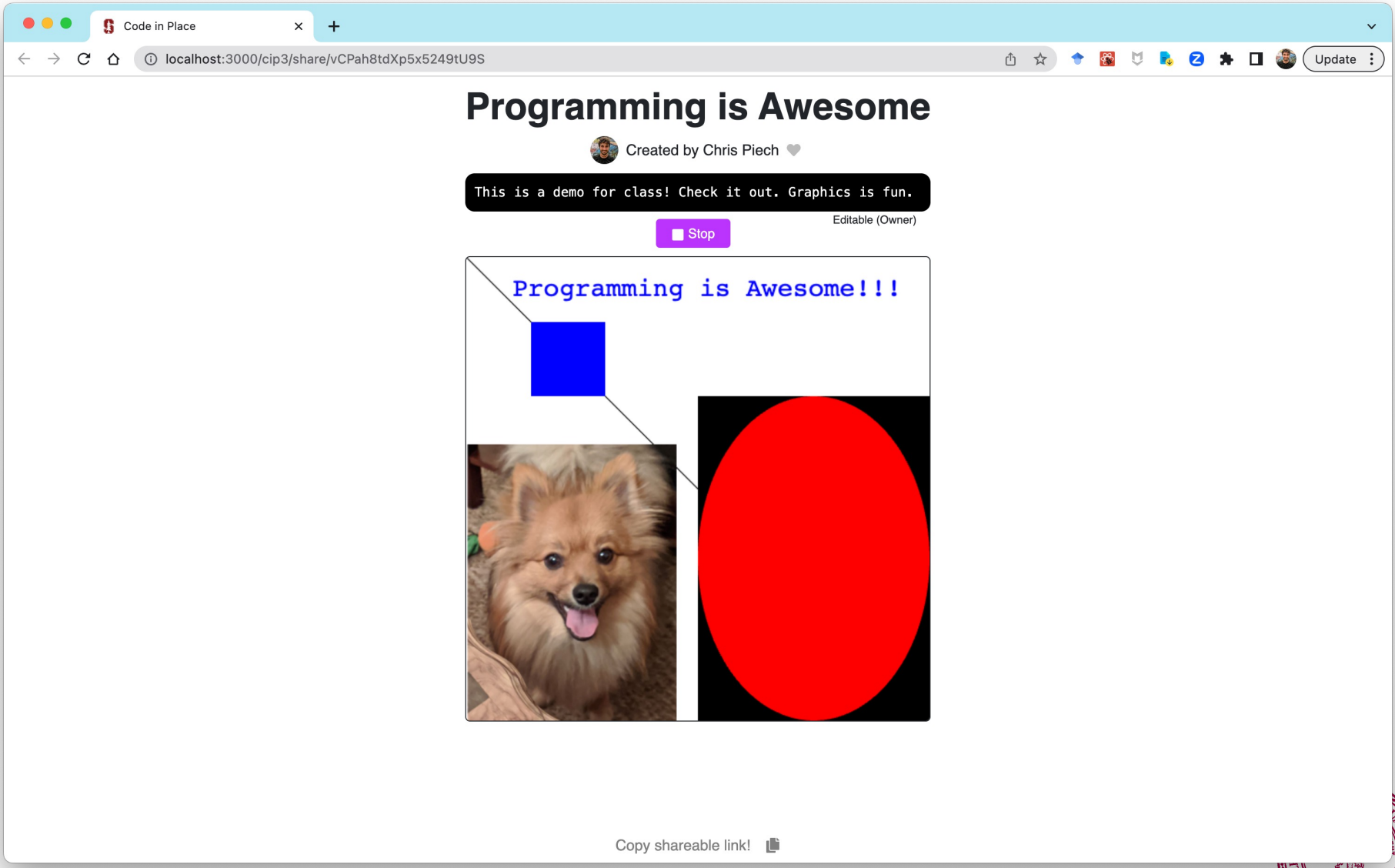


TK Natural Graphics

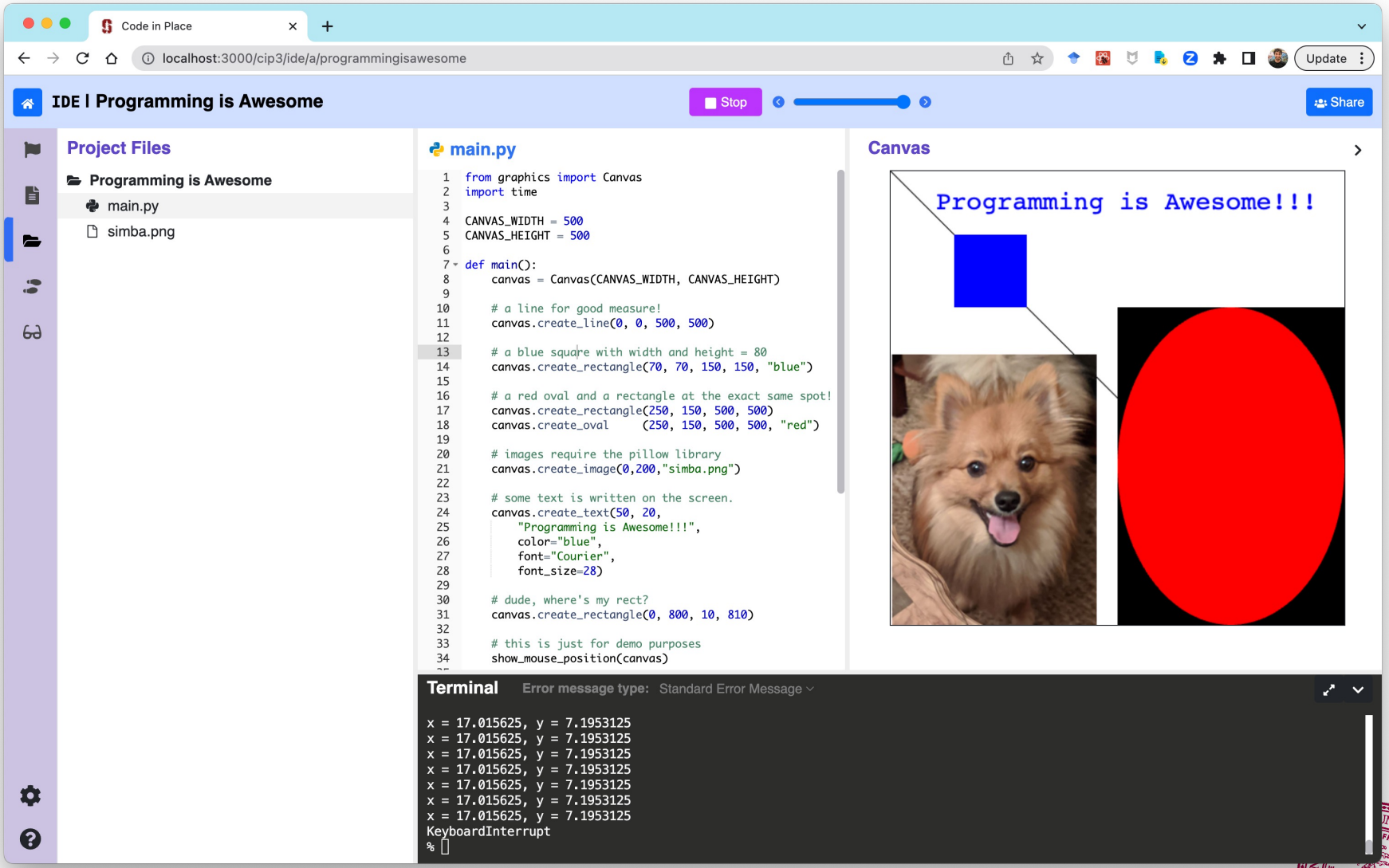
Graphics Coordinates



Rectangles, Ovals, Text



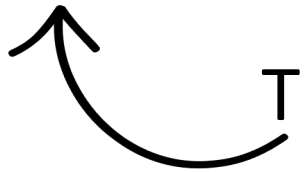
Rectangles, Ovals, Text



- `canvas.create_line()`
- `canvas.create_oval()`
- `canvas.create_text()`

- `canvas.create_line(x1, y1, x2, y2)`
- `canvas.create_oval()`
- `canvas.create_text()`

- `canvas.create_line(x1, y1, x2, y2)`
- `canvas.create_oval()`
- `canvas.create_text()`



The first point of the line is `(x1, y1)`

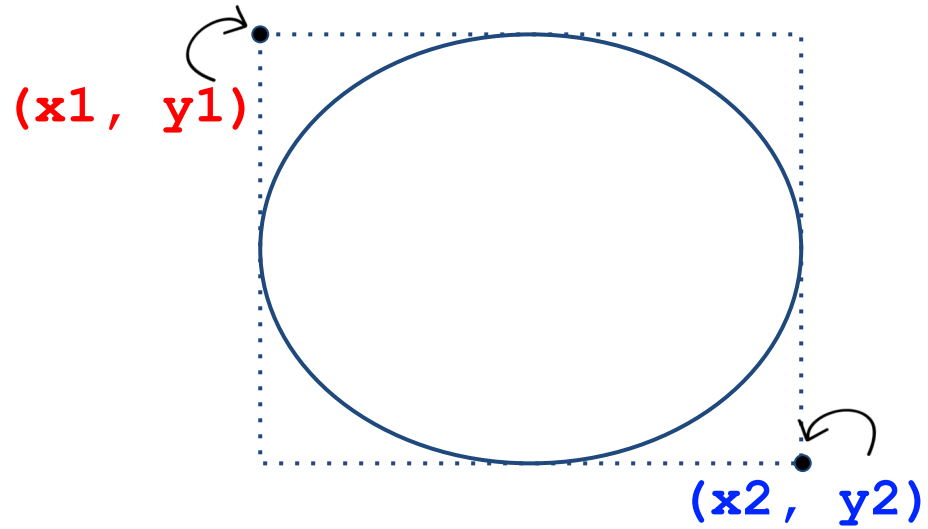
- `canvas.create_line(x1, y1, x2, y2)`
- `canvas.create_oval()`
- `canvas.create_text()`

The second point of the line is `(x2, y2)`

- `canvas.create_line(x1, y1, x2, y2)`
- `canvas.create_oval()`
- `canvas.create_text()`

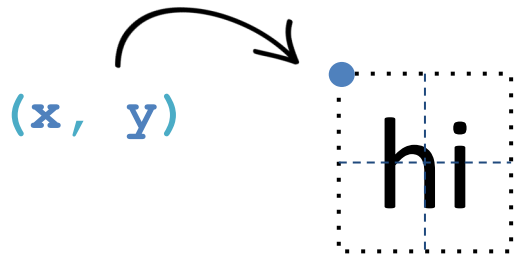
- `canvas.create_line()`
- **`canvas.create_oval(x1, y1, x2, y2)`**
- `canvas.create_text()`

- `canvas.create_line()`
- **`canvas.create_oval(x1, y1, x2, y2)`**
- `canvas.create_text()`

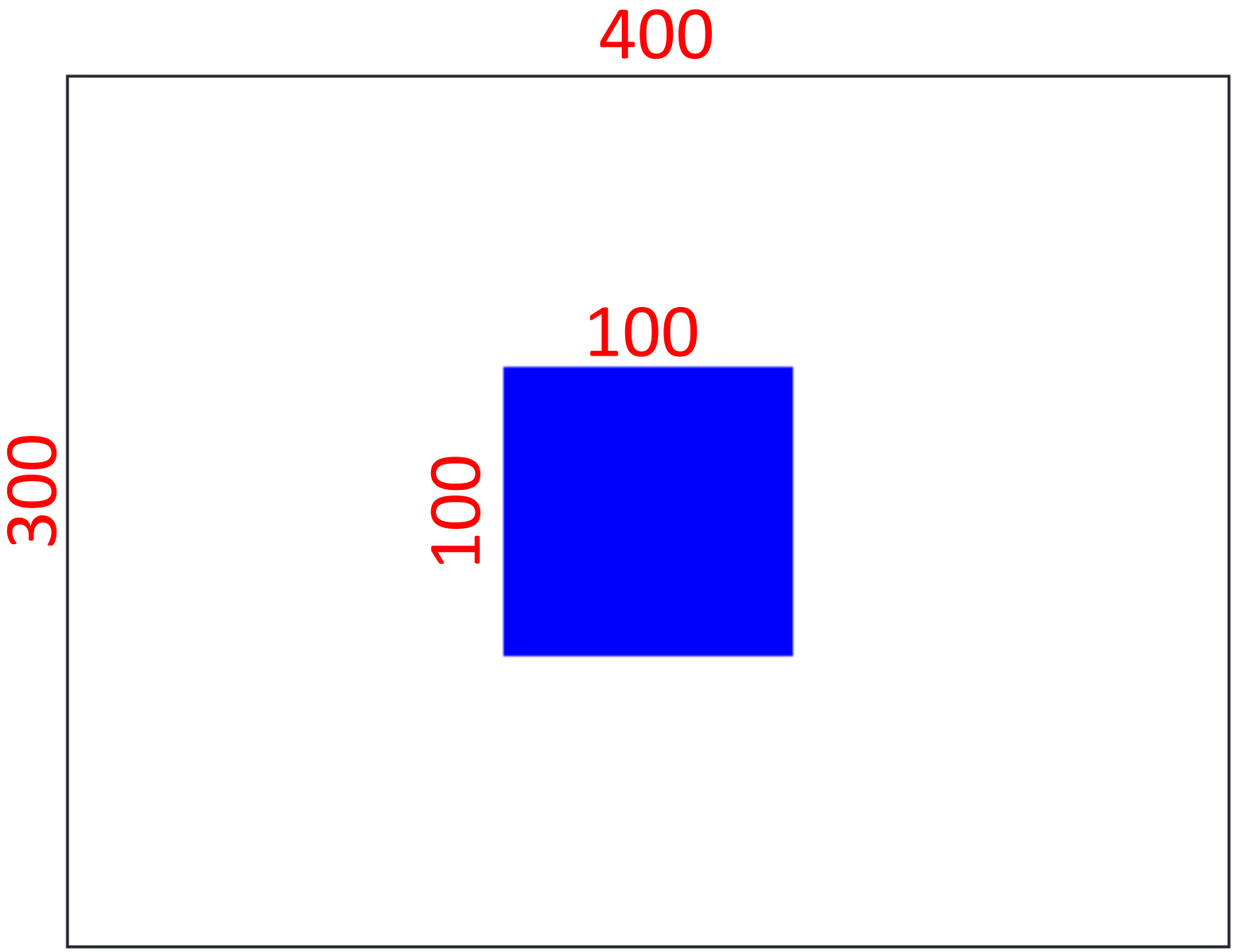


- `canvas.create_line()`
- `canvas.create_oval()`
- **`canvas.create_text(x, y, text='hi')`**

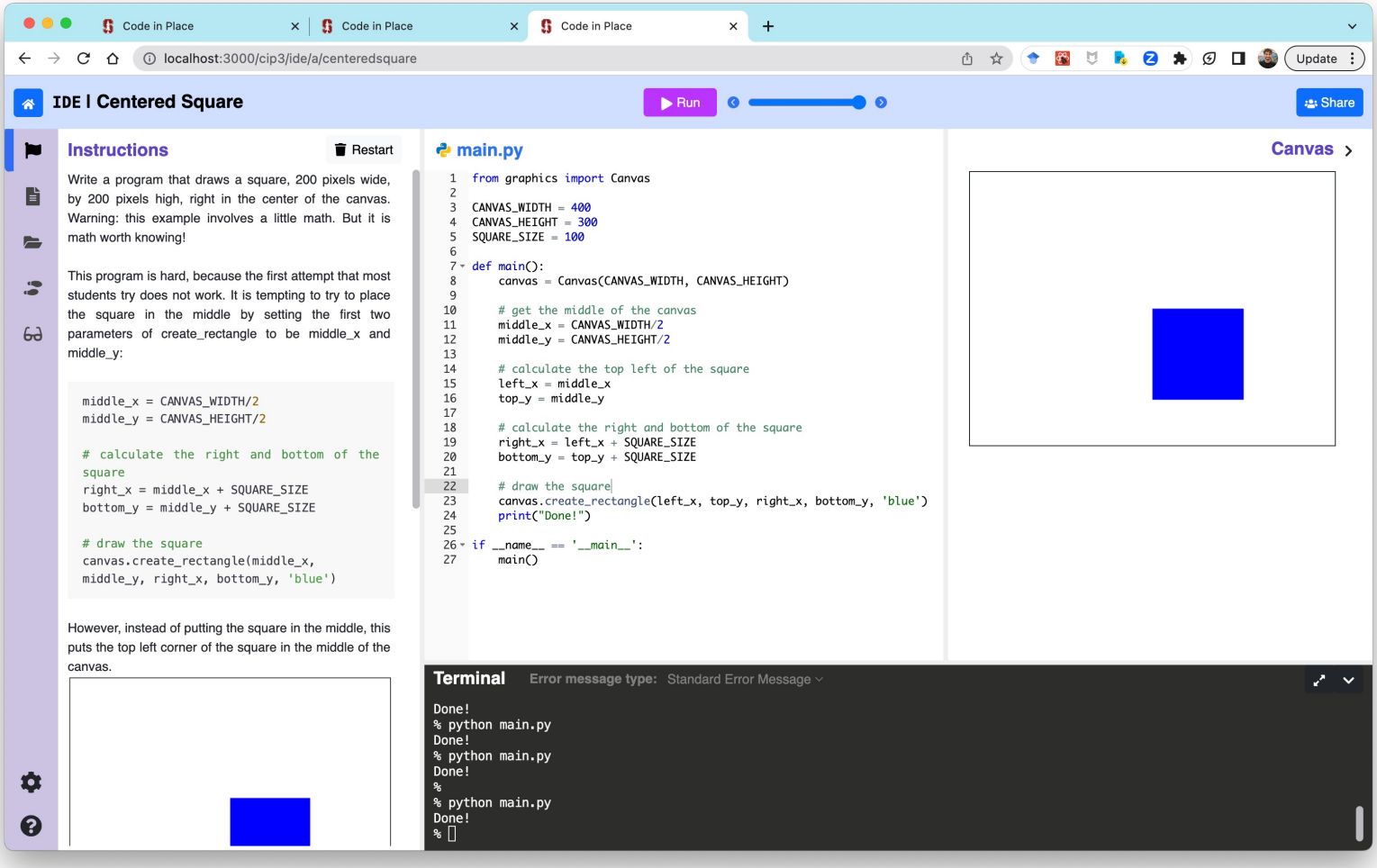
- `canvas.create_line()`
- `canvas.create_oval()`
- **`canvas.create_text(x, y, text='hi')`**



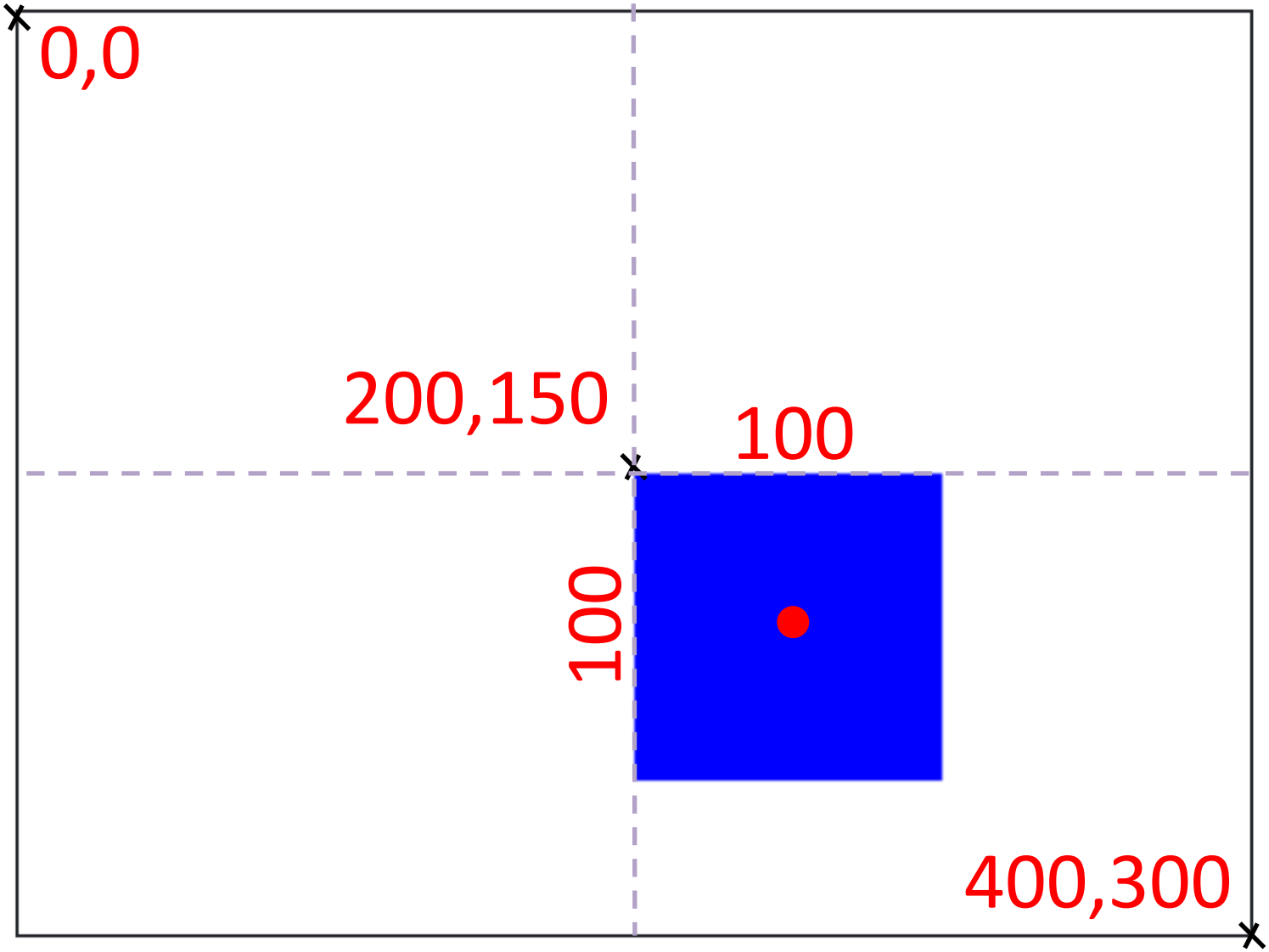
Centered Square



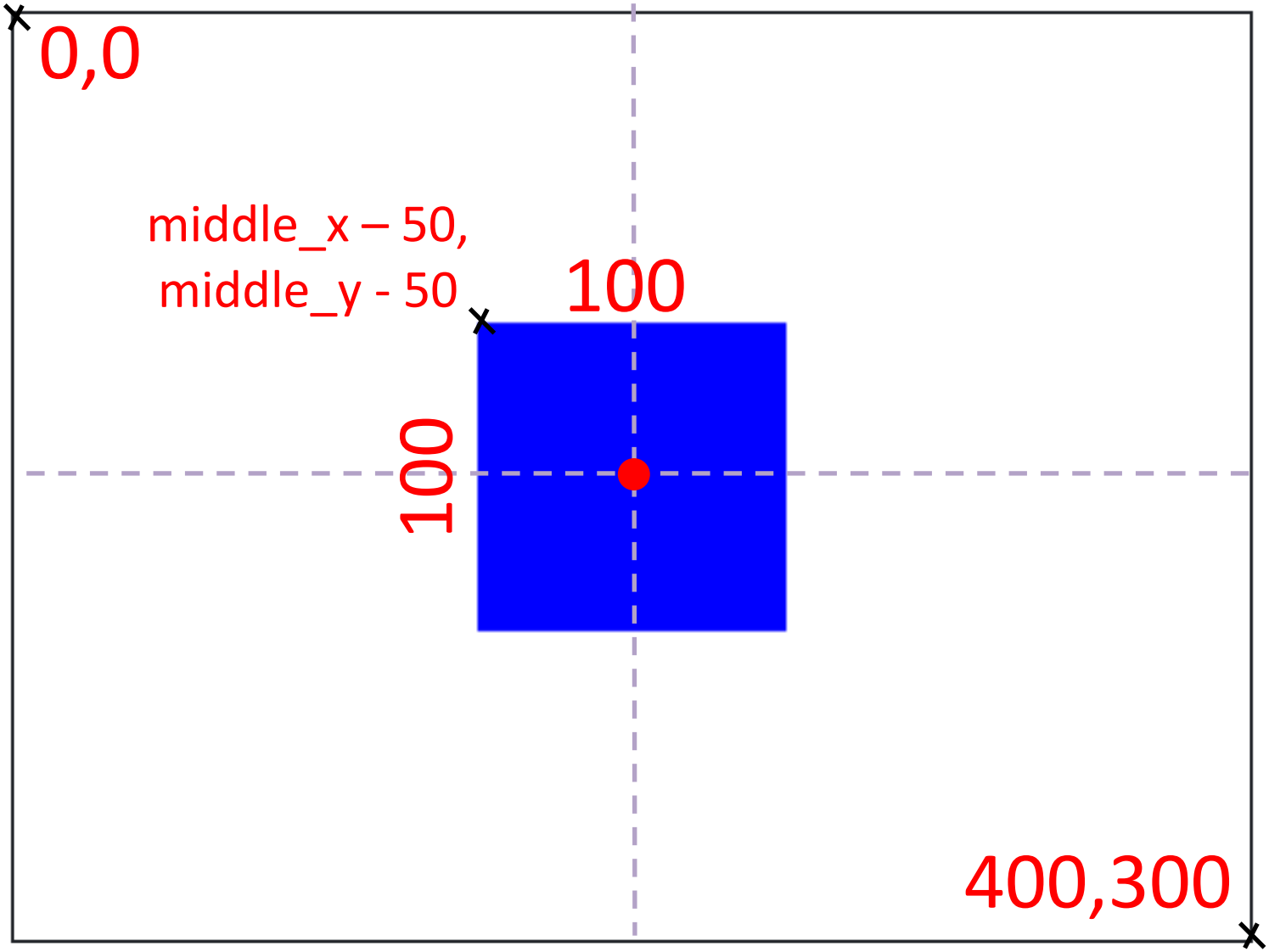
Centered Square



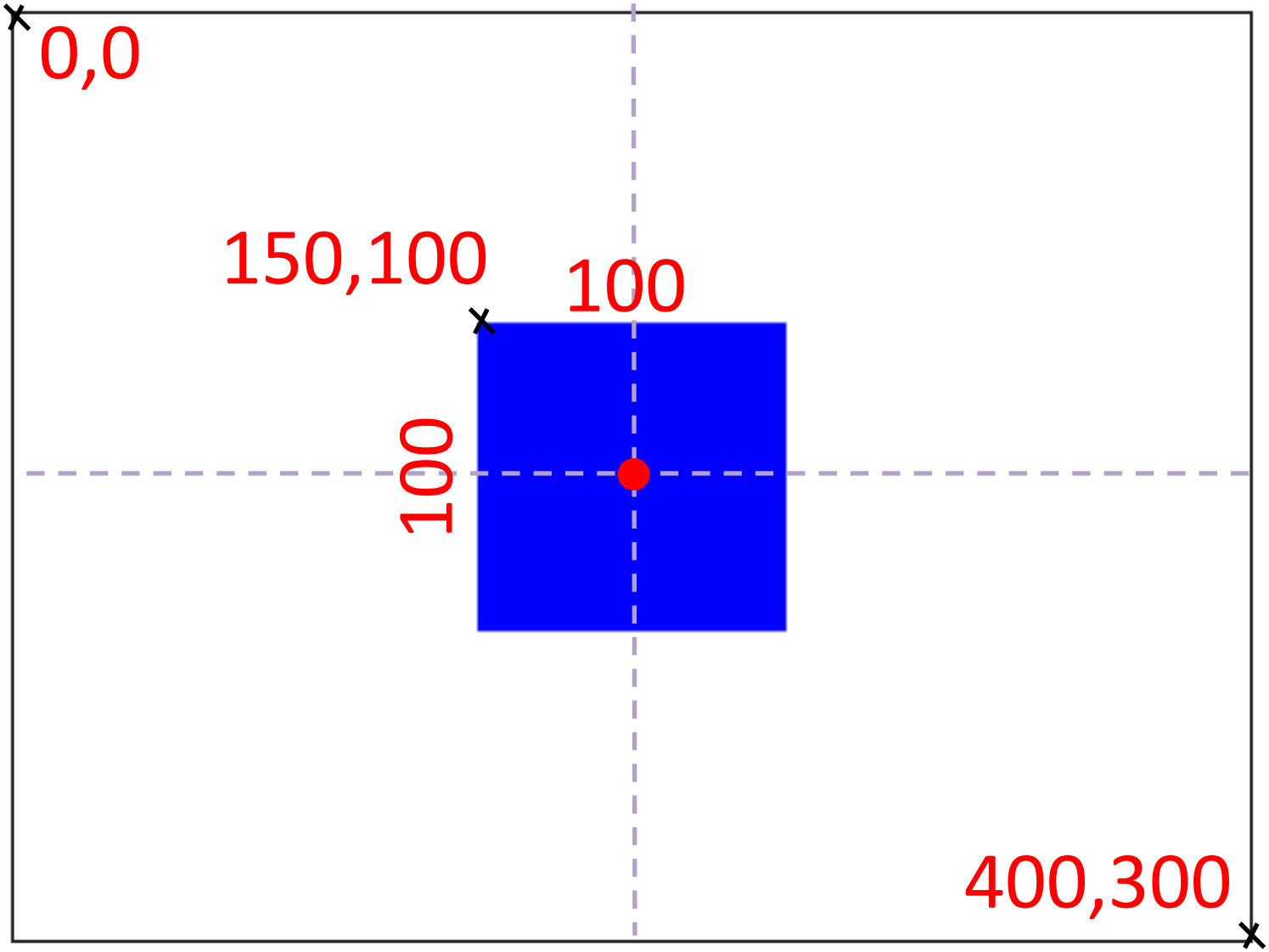
Centered Square



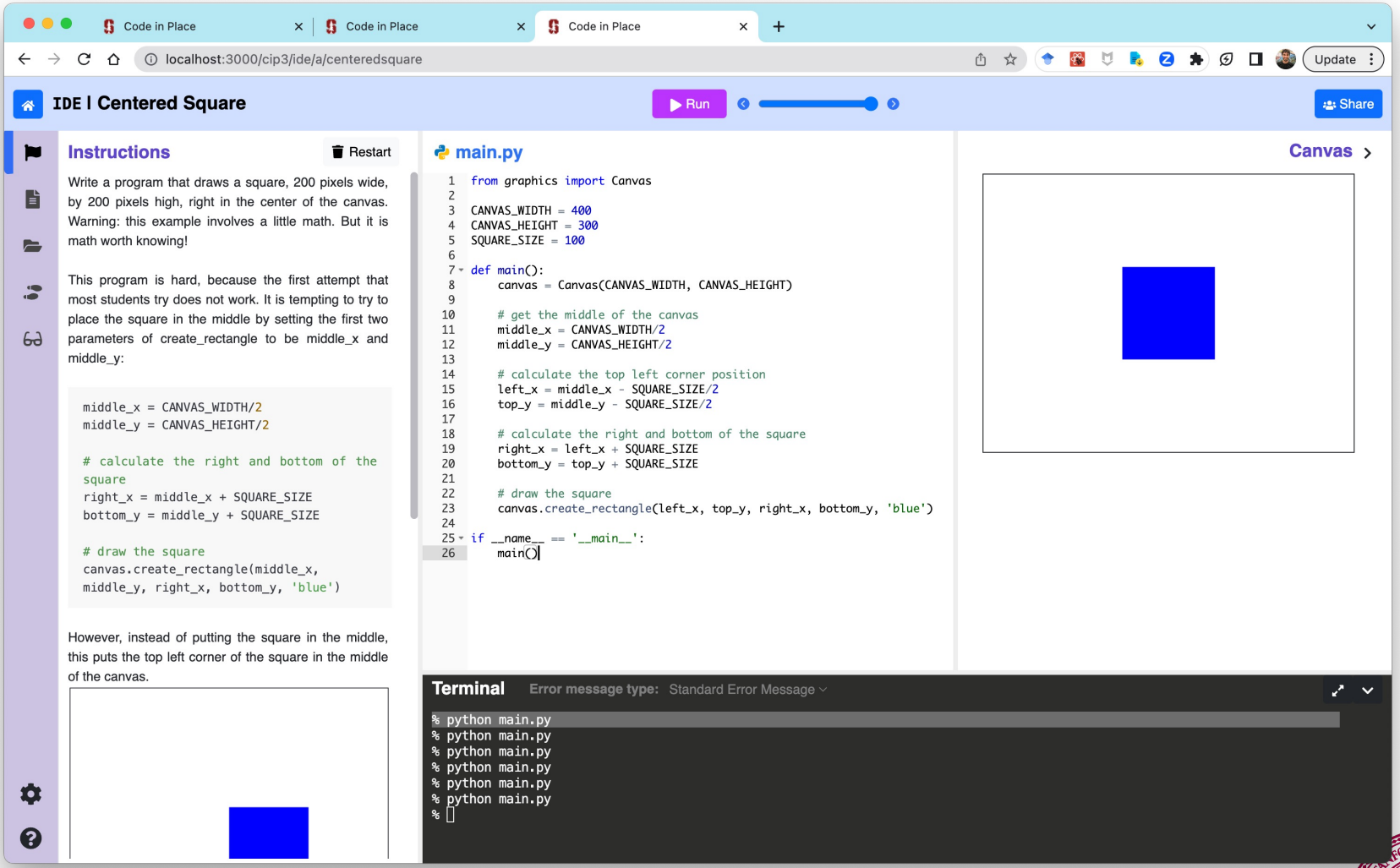
Centered Square



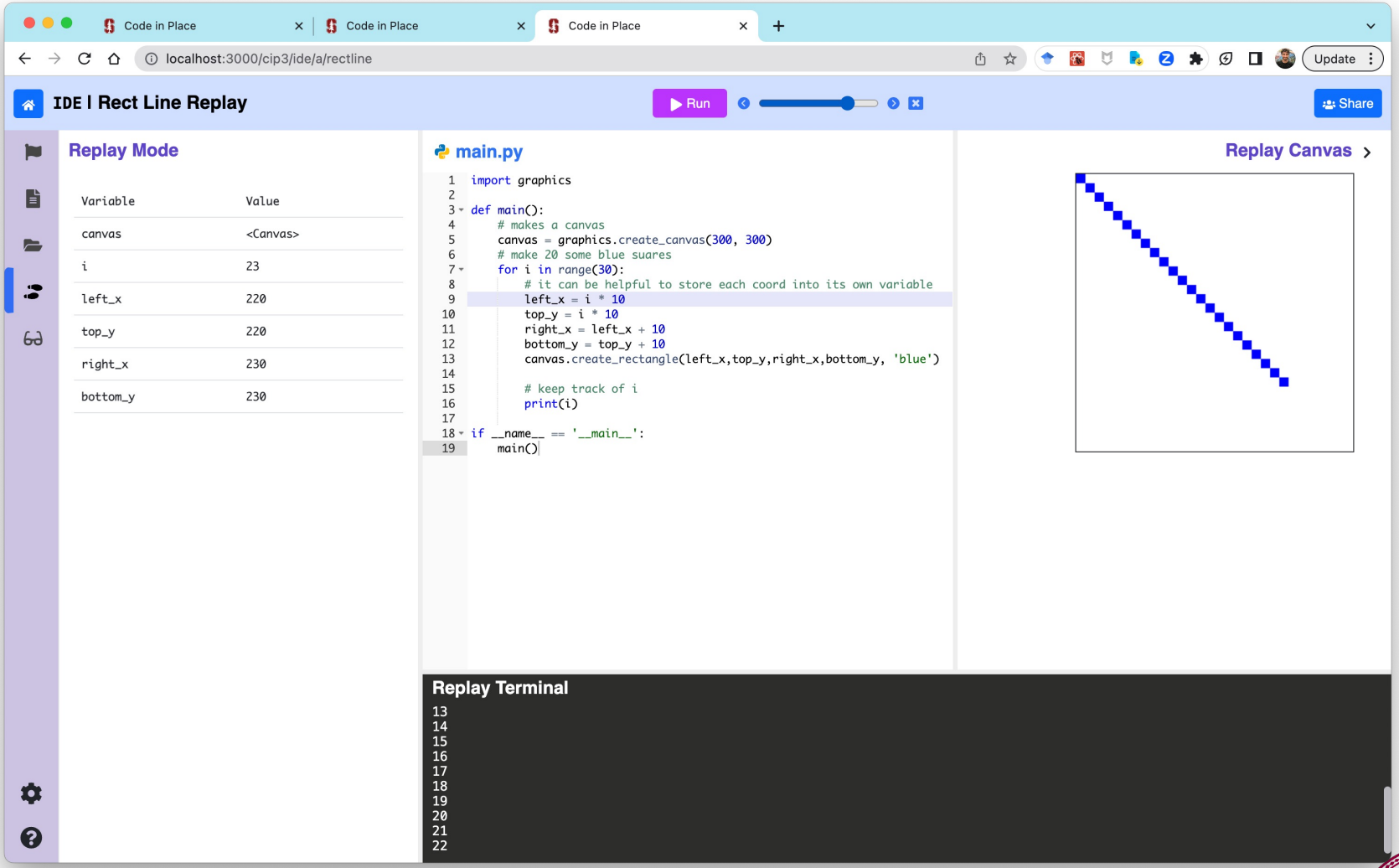
Centered Square



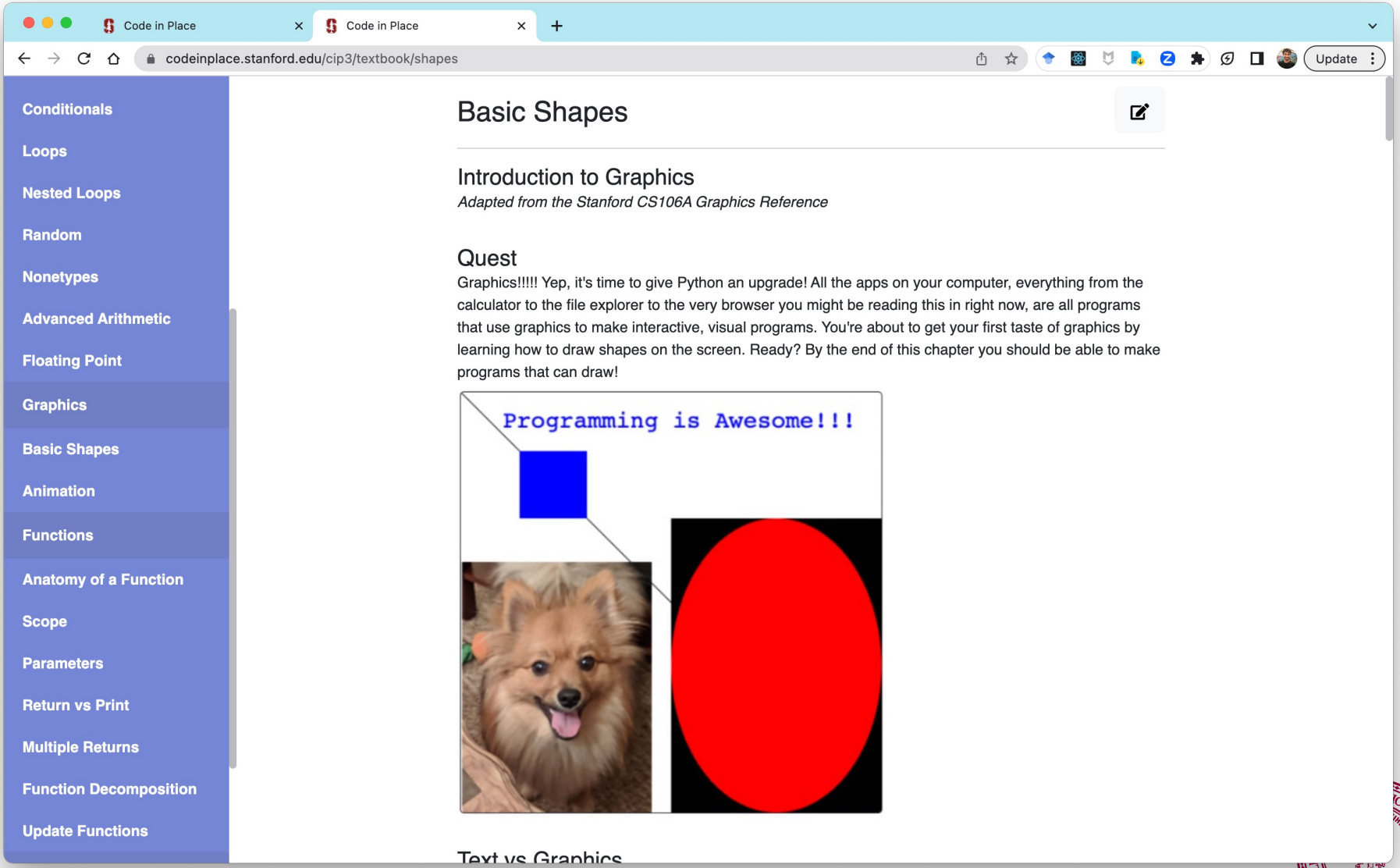
Centered Square



Step by Step


Code in Place
Code in Place
Code in Place
localhost:3000/cip3/ide/a/rectline
Update
IDE | Rect Line Replay
Run
Share
Replay Mode
Variable
Value
canvas
<Canvas>
i
23
left_x
220
top_y
220
right_x
230
bottom_y
230
main.py
1 import graphics
2
3 def main():
4 # makes a canvas
5 canvas = graphics.create_canvas(300, 300)
6 # make 20 some blue suares
7 for i in range(30):
8 # it can be helpful to store each coord into its own variable
9 left_x = i * 10
10 top_y = i * 10
11 right_x = left_x + 10
12 bottom_y = top_y + 10
13 canvas.create_rectangle(left_x,top_y,right_x,bottom_y, 'blue')
14
15 # keep track of i
16 print(i)
17
18 if __name__ == '__main__':
19 main()
Replay Canvas
Replay Terminal
13
14
15
16
17
18
19
20
21
22

Pedagogy


Code in Place
Code in Place
codeinplace.stanford.edu/cip3/textbook/shapes
Update
Conditionals
Loops
Nested Loops
Random
Nonetypes
Advanced Arithmetic
Floating Point
Graphics
Basic Shapes
Animation
Functions
Anatomy of a Function
Scope
Parameters
Return vs Print
Multiple Returns
Function Decomposition
Update Functions
Basic Shapes
Introduction to Graphics
Adapted from the Stanford CS106A Graphics Reference
Quest
Graphics!!!!! Yep, it's time to give Python an upgrade! All the apps on your computer, everything from the calculator to the file explorer to the very browser you might be reading this in right now, are all programs that use graphics to make interactive, visual programs. You're about to get your first taste of graphics by learning how to draw shapes on the screen. Ready? By the end of this chapter you should be able to make programs that can draw!
Programming is Awesome!!!
Text vs Graphics