

# Adventures in Coding 2

# Python

---

Erik Fredericks, 2018

Homepage: <http://efredericks.net>

GitHub: <https://github.com/ou-sbselab/SECS-SummerCamp>

# If you are interested...

All class materials are posted to my lab's GitHub page

<https://github.com/ou-sbselab/SECS-SummerCamp>

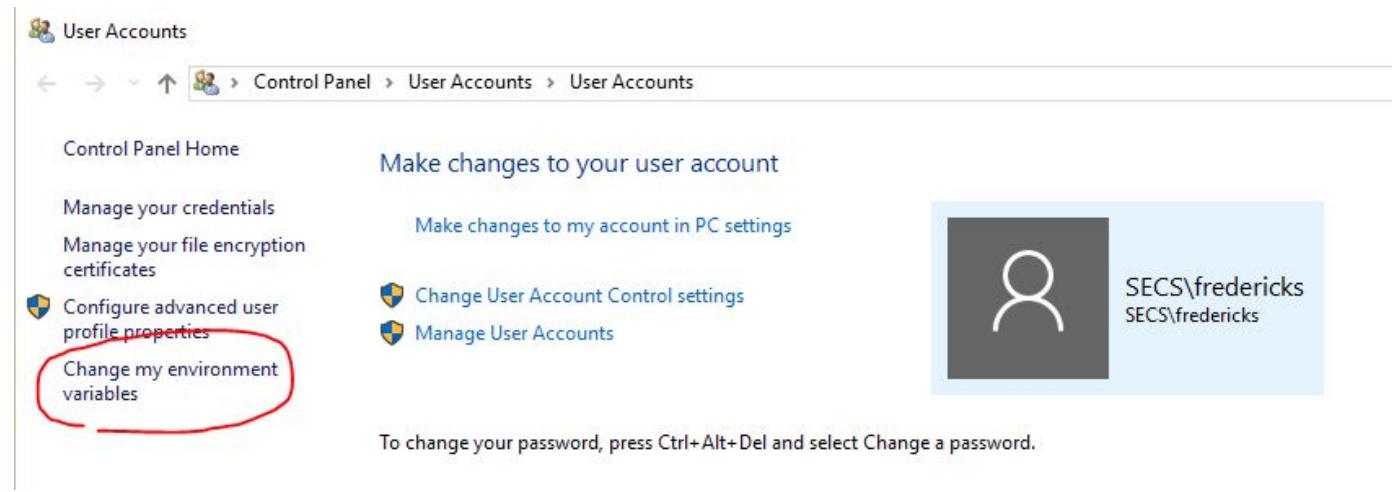
*(Your stuff is in adventures-in-coding-2)*

Here you can find all the presentations, code, guides, etc.

# Python Setup

Python is broken on these machines! Let's fix that.

Search for **User Accounts**, open it, and then click **Change my Environment Variables**



# Fixing Python

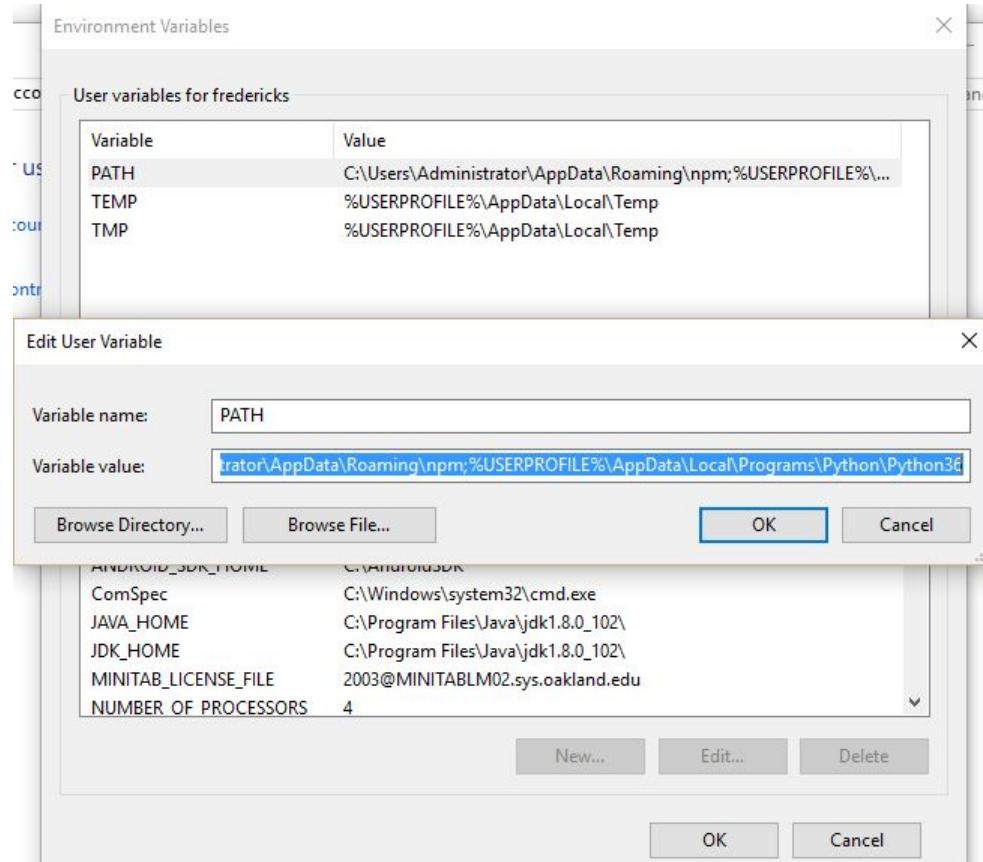
Click **PATH** at the top and then click **Edit..**

After Variable value:

Add  
**;%USERPROFILE%\AppData\Local\Programs\Python\Python36**

Click OK to save.

THE SEMI-COLON IS IMPORTANT!



# Python works!

Run a command prompt. Start → cmd. Type **python** and you should see this.

```
C:\Windows\System32\cmd.exe - python
C:\Windows\system32>python
Python 3.6.2 (v3.6.2:5fd33b5, Jul  8 2017, 04:57:36) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

`exit()` will quit Python

# Install pygame (we'll need it later)

In the command prompt, navigate (cd) to

**%USERPROFILE%\AppData\Local\Programs\Python\Python36\Scripts**

Run:

**python -m pip install pygame**

```
C:\Users\fredericks\AppData\Local\Programs\Python\Python36\Scripts>python -m pip install pygame
Collecting pygame
  Downloading https://files.pythonhosted.org/packages/e4/1f/23e1620de98d3eff46cbd470dc9465c5b860c06d37787781d756135b512b
  /pygame-1.9.3-cp36-cp36m-win_amd64.whl (4.2MB)
    100% |██████████| 4.2MB 319kB/s
Installing collected packages: pygame
Successfully installed pygame-1.9.3
You are using pip version 9.0.1, however version 10.0.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Users\fredericks\AppData\Local\Programs\Python\Python36\Scripts>
```

# Check to make sure it works!

Run python:

```
C:\> python
```

```
>>> import pygame
```

# Adventures in Coding 2 Topics

Quick Python review

Python classes

Continuation of game design concepts



# Wake up!

Launch the Python interpreter (command line → python)

- 1) Create a variable that holds **your name**
- 2) Create another variable that holds **your school**
- 3) Print out “My name is **your name** and I attend **your school**.”

So if I did this, it would say:

My name is Professor Erik, and I attend Oakland University.

Python reminder:

```
variable = "string"  
print("I am printing out the %s" % (variable))
```

# The story of the command line and the text file

There is **no** Python GUI on these machines (that works)

*Normally we'd use IDLE or Thonny*

Open up Notepad (or your favorite text editor) → NOT WORD

Save the file on your Desktop as <your last name>.py

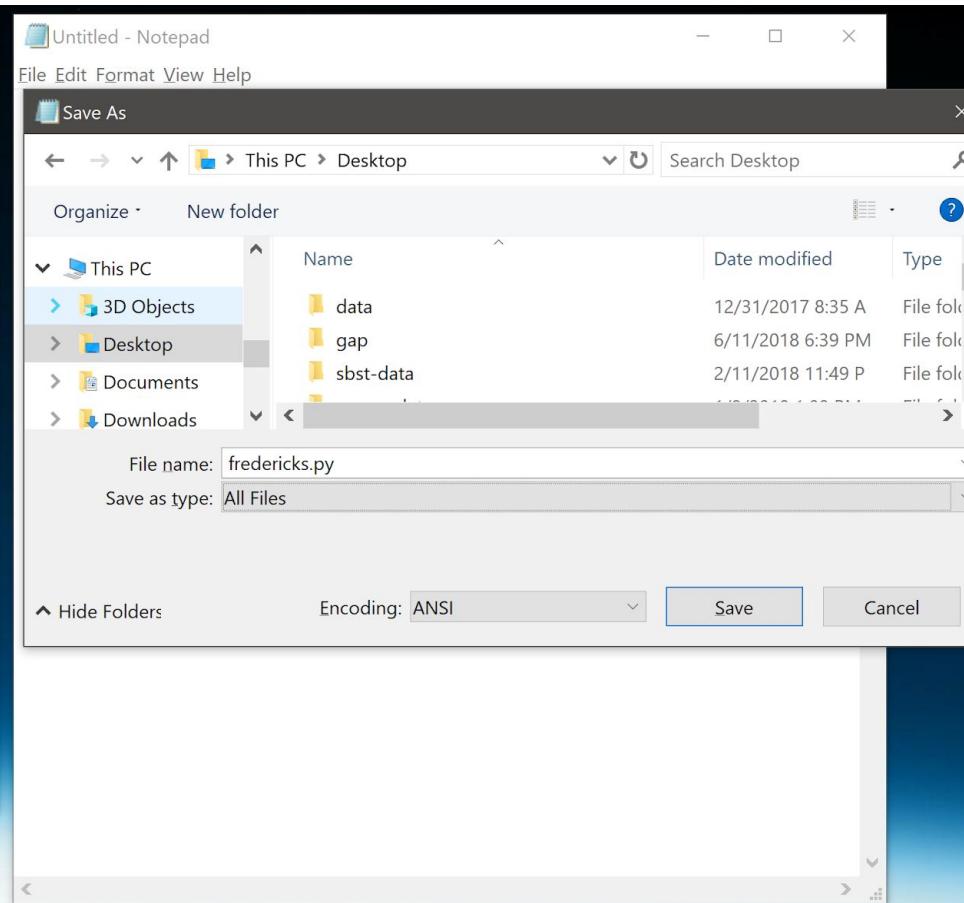
Make sure it is of type All Files, not Text Documents!

Open up a command prompt and navigate to your Desktop:

```
cd %USERPROFILE\Desktop  
python <your last name>.py
```

Command Prompt

```
C:\Users\erik>cd %USERPROFILE%\Desktop  
C:\Users\erik\Desktop>python fredericks.py
```



# Decisions

```
x = True
y = 5.2
if x:
    print("x was True!")
else:
    print("Apparently x was False!")

#####
if (y <= 3.14159626):
    print("y is smaller than pi")
elif (y > 3.14159636) and (y < (2 * 3.14159626)):
    print("y is between pi and 2pi")
else:
    print("y is larger than 2pi")
```

# Loops

```
numbers = range(10)
for i in len(numbers):
    print(numbers)

for number in numbers:
    print(number)

for i in range(10):
    print(number[i])
```

```
done = True
i = 10
while not done:
    print(i)
    i -= 1
    if (i <= 0):
        done = True
print("Done")
```

# Lists

```
a = range(10)
b = ["This", "is", "Adventures", "in", "Coding", 2]
c = [[1, 2, 3, 4],
      [5, 6, 7, 8],
      [9, 10, 11, 12]]

print(a)
print(b)
print(c)

for word in b:
    print(word)
```

# Functions

```
def this_function_is_neat(loop):  
    for i in range(0,loop):  
        print("%d is the best number" % i)
```

- 1) What does **this\_function\_is\_neat** actually do?
- 2) How would you call it?

# main in Python

```
# Functions functions functions...
...
...
...
if __name__ == "__main__":
    print('this is the main function')
```

# Task:

Make a Python program that:

- 1) Asks you for your first name
  - Input → `myvar = input("What do you want to know? ")`
- 2) Runs a loop 10 times that prints your first name
- 3) Uses the Python main style function
  - `if __name__ == "__main__":`

# Classes



# Classes

## C#

```
public class MyClass
{
    public MyClass()
    {
        // Initialize class
        int x      = 1;
        String y = "name";
    }
}

var my_class = MyClass();
```

## Python

```
class MyClass(object):
    def __init__(self):
        self.x = 1
        self.y = "name"

my_class = MyClass()
```

# HelloWorld Class

```
class HelloWorld(object):
    def __init__(self, in_name):
        print "Hello there! I am printed whenever you instantiate me"
        self.student_name = in_name

hello = HelloWorld("Prof. Erik")
```

# Student class

```
class Student(object):
    # Initialize student information
    def __init__(self, name, sid, age, hobby):
        self.name = sname
        self.sid = sid
        self.age = age
        self.hobby = hobby

    # Print out the student's age
    def printAge(self):
        print("%s is %d years old." % (self.name, self.age))
```

# Inheritance!

One common OOP concept is **inheritance**

One class **inherits** from another

```
# Define a class that specifies a Person
class Person(object):
    def __init__(self, name, age):
        self.name = name
        self.age  = age

    def Print(self):
        print("Name [%s], Age [%d]" % (self.name, self.age))
```

# Inherit from Person

```
# Create a student → derives from Person
class Student(Person):
    def __init__(self, name, age, student_id):
        Person.__init__(name, age)
        self.student_id = student_id

    def Print(self):
        print("Name [%s], Age [%d], Student ID [%d]" % \
              (self.name, self.age, self.student_id))

p = Person("Your Name", 40)
s = Student("Your Name", 40, 123456789)
```

# Other things to remember → DECISIONS and LOOPS

Decisions:

```
something = "do it"
if (something == "do it"):
    print("I did something")
else:
    print("I did something else!")
```

# LOOPS (and random and lists)

```
import random

students = []
for i in range(10):
    student_name = "Student %d" % i
    s = Student(student_name, random.randint(10,20), \
                random.randint(1,10000))

    students.append(s)

for student in students:
    student.Print()
```

# Questions!

- Why is it Student 0 instead of Student 1?
- How would you **extend the Student class** to include the classes a student is taking?

# Extend Student class

```
# Classes is a list!
class Student(object):
    def __init__(self, name, age, student_id, classes):
        ...
        self.classes = classes

    # Print a list of classes per student
    def PrintClasses(self):
        for c in classes:
            print(c)

s = Student("Name", 12, 1234456, ["Math", "Social Studies"])
s.PrintClasses()
```

# Remember pygame? Let's do that again!



```
import pygame
# Hello world of graphics
pygame.init()
screen = pygame.display.set_mode((640, 480))

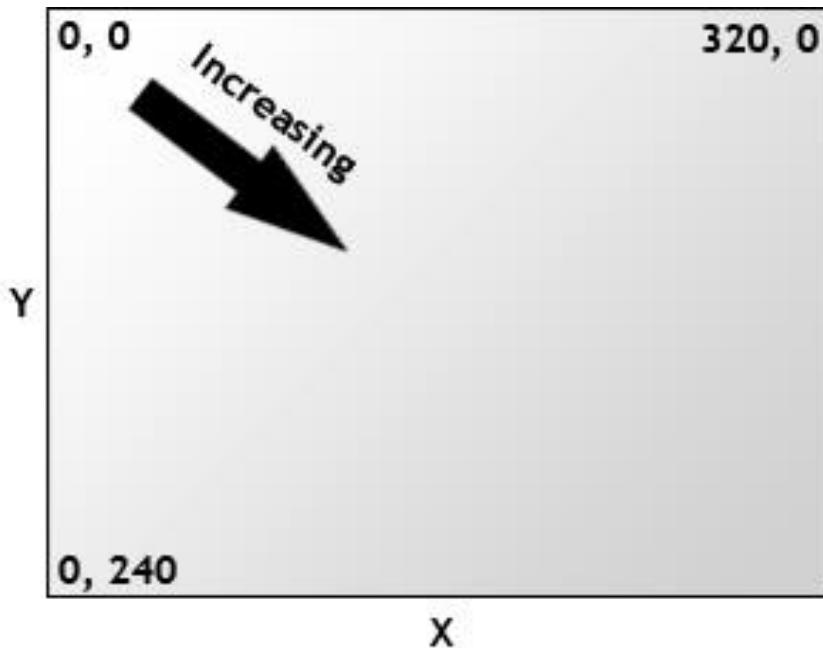
done = False
while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True

    pygame.draw.rect(screen, (0,128,255), pygame.Rect(30,30,60,60))
    pygame.display.flip()
```

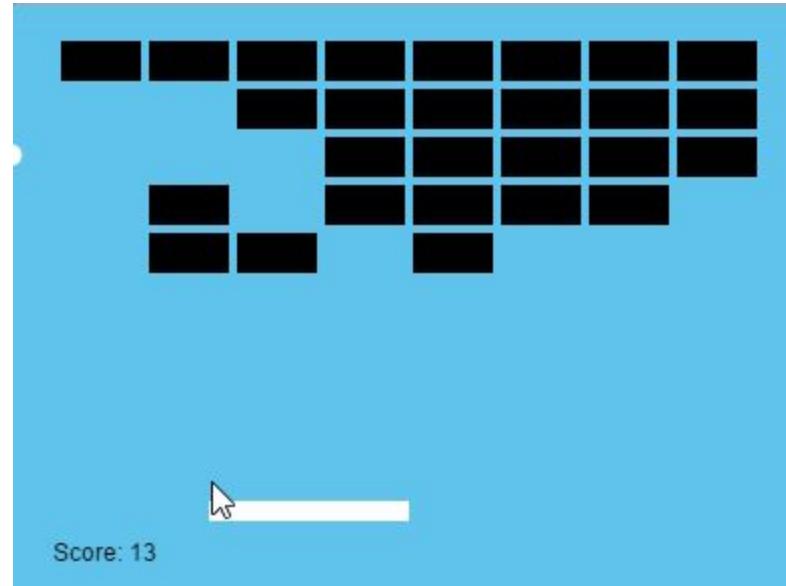
# Drawing to the screen

Back to the rectangle...

The monitor →



# Game for the rest of the day: Breakout



# What do we need to make a game?

# Imports we'll need

```
import pygame  
import sys  
import random
```

# Constants

SCREEN\_W = 640

SCREEN\_H = 480

BRICK\_W = 60

BRICK\_H = 15

PADDLE\_W = 60

PADDLE\_H = 12

BALL\_DIAMETER = 16

BALL\_RADIUS = int(BALL\_DIAMETER / 2)

# More Constants

MAX\_PADDLE\_X = SCREEN\_W - PADDLE\_W

MAX\_BALL\_X = SCREEN\_W - BALL\_DIAMETER

MAX\_BALL\_Y = SCREEN\_H - BALL\_DIAMETER

PADDLE\_Y = SCREEN\_H - PADDLE\_H - 10

BLACK = (0, 0, 0)

GOLD = (135, 113, 72)

WHITE = (255, 255, 255)

BLUE = (0, 0, 255)

# A few more!

```
# State machine  
S_BALL_IN_PADDLE = 0  
S_PLAYING          = 1  
S WON              = 2  
S GAME_OVER         = 3
```



Add your  
constants!

# Game class

Let's setup a skeleton that we'll fill out throughout the rest of the day.

```
### CONSTANTS  
... (from prior slides) → don't type this in.
```

```
### GAME CLASS  
class OUBreakout(object):  
    def __init__(self):  
        print 'Init'
```

```
### MAIN  
if __name__ == '__main__':  
    game = OUBreakout()
```

(in OUBreakout class)

```
# Initialize the our main class and pygame
def __init__(self):
    pygame.init()

    self.screen = pygame.display.set_mode((SCREEN_W, SCREEN_H))
    pygame.display.set_caption('OU Breakout')
    self.clock = pygame.time.Clock()

    if pygame.font:
        self.font = pygame.font.Font(None, 30)
    else:
        self.font = None

    self.initialize_game()
```

```
def initialize_game(self):
    self.lives = 3
    self.score = 0
    self.state = S_BALL_IN_PADDLE

    self.paddle = pygame.Rect(300, PADDLE_Y, PADDLE_W, PADDLE_H)
    self.ball    = pygame.Rect(300, PADDLE_Y - BALL_DIAMETER \
                                BALL_DIAMETER, BALL_DIAMETER)

    self.ball_velocity = [5, -5]

    self.create_bricks()
```

```
def create_bricks(self):
    y_offset      = 35
    self.bricks = []

    for i in range(7):      # rows
        x_offset = 35
        for j in range(8): # columns

            self.bricks.append( \
                pygame.Rect(x_offset, y_offset, BRICK_W, BRICK_H))
            x_offset += BRICK_W + 10

        y_offset += BRICK_H + 5
```

```
def draw_bricks(self):  
    for brick in self.bricks:  
        pygame.draw.rect(self.screen, GOLD, brick)
```

```
def run(self):
    while True: # Main game loop
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                sys.exit()

        self.clock.tick(60)          # lock to 60 FPS
        self.screen.fill(BLACK)    # Redraw screen

        self.draw_bricks()

    pygame.display.flip()
```

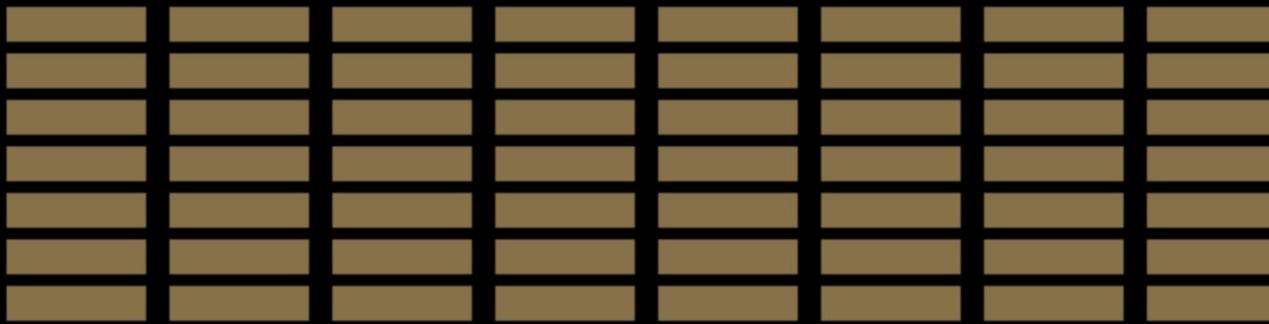


Add  
game.run() to  
the end!



OU Breakout

- □ ×



Time to  
handle user  
input



*New OUBreakout class function*

```
def check_input(self):
    keys = pygame.key.get_pressed()

    if keys[pygame.K_ESCAPE]:
        sys.exit()
```

*Inside def run(self):*

...

```
self.screen.fill(BLACK)
self.check_input()
```

...

*After calling draw\_bricks() inside of def run:*

```
self.draw_bricks()  
  
# Draw the paddle and the ball  
pygame.draw.rect(self.screen, BLUE, self.paddle)  
pygame.draw.circle(self.screen, WHITE, \  
    (self.ball.left + BALL_RADIUS, \  
     self.ball.top + BALL_RADIUS), \  
    BALL_RADIUS)  
  
...
```

*Inside of check\_input():*

```
# Check pygame.K_ESCAPE
```

```
...
```

```
if keys[pygame.K_LEFT]:  
    self.paddle.left -= 5  
    if self.paddle.left < 0:  
        self.paddle.left = 0
```

```
if keys[pygame.K_RIGHT]:  
    self.paddle.left += 5  
    if self.paddle.left > MAX_PADDLE_X:  
        self.paddle.left = MAX_PADDLE_X
```

```
...
```

*Inside def run above draw\_bricks*

```
if self.state == S_BALL_IN_PADDLE:  
    self.ball.left = self.paddle.left + self.paddle.width/2  
    self.ball.top  = self.paddle.top  - self.ball.height  
  
self.draw_bricks()  
...
```



And what  
else?



*New class function:*

```
def move_ball(self):
    self.ball.left += self.ball_velocity[0]
    self.ball.top  += self.ball_velocity[1]

    if self.ball.left <= 0:
        self.ball.left = 0
        self.ball_velocity[0] = -self.ball_velocity[0]

    if self.ball.left >= MAX_BALL_X:
        self.ball.left = MAX_BALL_X
        self.ball_velocity[0] = -self.ball_velocity[0]

    if self.ball.top < 0:
        self.ball.top = 0
        self.ball_velocity[1] = -self.ball_velocity[1]
```

*In def run():*

```
...
elif self.state == S_PLAYING:
    self.move_ball()
```

*In check\_input():*

```
...
if keys[K_SPACE] and self.state == S BALL IN PADDLE:
    self.ball_velocity = [5, -5]
    self.state = S_PLAYING
```



*New class function:*

```
def handle_collisions(self):
    for brick in self.bricks:
        if self.ball.colliderect(brick):
            self.score += 3
            self.ball_velocity[1] = -self.ball_velocity[1]
            self.bricks.remove(brick)
            break

    if len(self.bricks) == 0:
        self.state = S_WON
```

*In def run():*

```
if self.state == S_PLAYING:  
    self.move_ball()  
    self.handle_collisions()  
  
elif self.state == S BALL IN PADDLE:  
    ...  
    self.show_message("Press SPACE to begin.")  
elif self.state == S WON:  
    self.show_message("You won! Press ENTER to play again.")  
elif self.state == S GAME OVER:  
    self.show_message("You lost! Press ENTER to play again.")
```

*New class function: show a message!*

```
def show_message(self, message):
    if self.font:
        size = self.font.size(message)
        font_surface = self.font.render(message, False, WHITE)
        x = (SCREEN_W - size[0]) / 2
        y = (SCREEN_H - size[1]) / 2
        self.screen.blit(font_surface, (x, y))
```

A close-up photograph of a green lizard's head, showing its mouth open with a red tongue and a row of sharp teeth. A green speech bubble is positioned on the right side of the lizard's head, containing the text "Last things!" in white, handwritten-style font.

Last  
things!

*Handle paddle and misses → at end of handle\_collisions*

```
def handle_collisions(self):
    ...
    if self.ball.colliderect(self.paddle):
        self.ball.top = PADDLE_Y - BALL_DIAMETER
        self.ball_velocity[1] = -self.ball_velocity[1]

    elif self.ball.top > self.paddle.top:
        self.lives -= 1
        if self.lives > 0:
            self.state = S_BALL_IN_PADDLE
        else:
            self.state = S_GAME_OVER
```

*New class function*

```
def show_stats(self):
    if self.font:
        font_surface = self.font.render("SCORE: %d LIVES: %d" \
            % (self.score, self.lives), False, WHITE)
        self.screen.blit(font_surface, (205, 5))
```

*In self.run():*

```
...
self.show_stats()
pygame.display.flip()
```



Simple  
game!

Add in location for hitting paddle!

Add in levels!