 GAMING

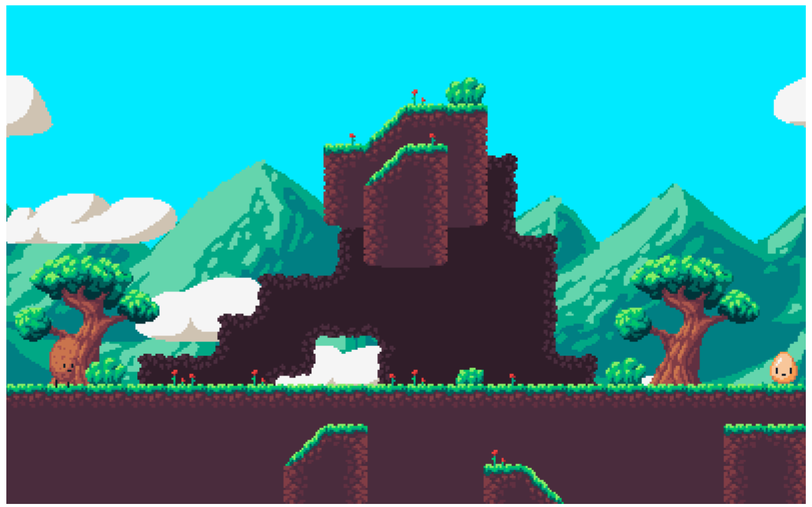
May 24th, 2022

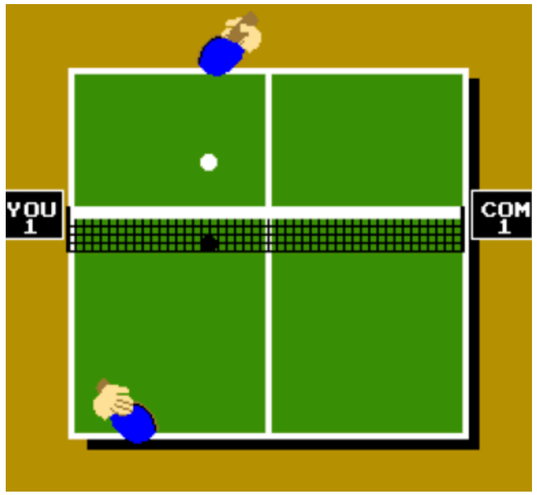
Schedule

- Next Week Overview
- Unittesting
- Pygame

Making Games in Python

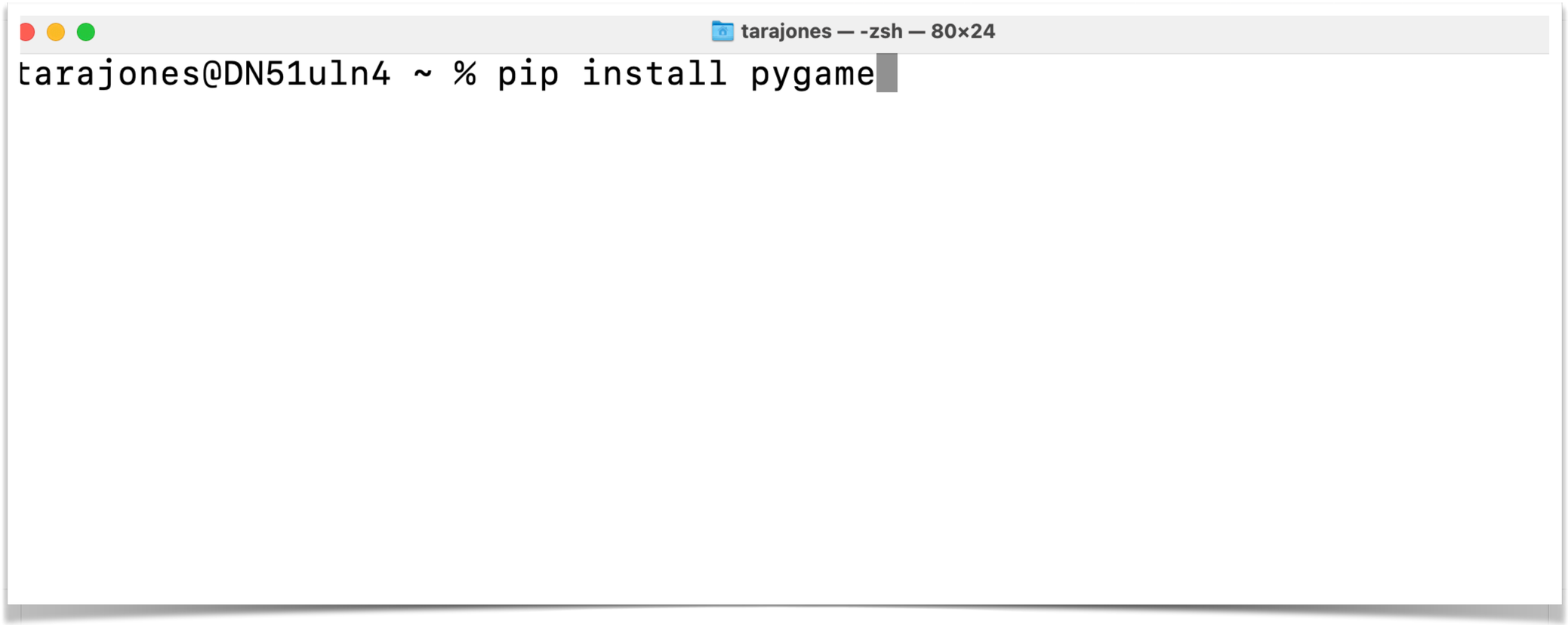
Example Games





Installation

- Non-SL package like we learned about last week



GameLoop

Gameloop

-Center of game play control

-Updates state of the game

-Every cycle is a frame

Events

- User actions that our game loops are centered on
- Access them with an event handler
- Every event has a type

QUIT
ACTIVEEVENT
KEYDOWN
KEYUP
MOUSEMOTION
MOUSEBUTTONUP
MOUSEBUTTONDOWN
JOYAXISMOTION
JOYBALLMOTION
JOYHATMOTION
JOYBUTTONUP
JOYBUTTONDOWN
VIDEORESIZE
VIDEOEXPOSE
USEREVENT

Events

- These events are stored in a queue
- We can access them with the following:
  - `for event in pygame.event.get():`
- <Event(768-KeyDown {'unicode': '', 'key': 1073741906, 'mod': 0, 'scancode': 82, 'window': None})>
- <Event(769-KeyUp {'unicode': '', 'key': 1073741906, 'mod': 0, 'scancode': 82, 'window': None})>

Lets build our first graphics window

```
import pygame
```

```
import pygame

pygame.init()
screen = pygame.display.set_mode((500, 500))
```

```
import pygame

pygame.init()
screen = pygame.display.set_mode((500, 500))
```

How we will initialize the window

```
import pygame

pygame.init()
screen = pygame.display.set_mode((500, 500))
white = (255, 255, 255)
blue = (0, 0, 255)
```

```
import pygame

pygame.init()
screen = pygame.display.set_mode((500, 500))
white = (255, 255, 255)
blue = (0, 0, 255)
```

Tuples for RGB values

```
import pygame

pygame.init()
screen = pygame.display.set_mode((500, 500))
white = (255, 255, 255)
blue = (0, 0, 255)

running = True

while running:
```

How we will initialize the game loop

```
import pygame

pygame.init()
screen = pygame.display.set_mode((500, 500))
white = (255, 255, 255)
blue = (0, 0, 255)

running = True

while running:

    for event in pygame.event.get():
```

```
import pygame

pygame.init()
screen = pygame.display.set_mode((500, 500))
white = (255, 255, 255)
blue = (0, 0, 255)

running = True

while running:

    for event in pygame.event.get():
```

This line gets all of the actions in the event queue

```
import pygame

pygame.init()
screen = pygame.display.set_mode((500, 500))
white = (255, 255, 255)
blue = (0, 0, 255)

running = True

while running:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
```

```
import pygame

pygame.init()
screen = pygame.display.set_mode((500, 500))
white = (255, 255, 255)
blue = (0, 0, 255)

running = True

while running:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
```

We saw the event types from before, and we can check the event's type like so

```
import pygame

pygame.init()
screen = pygame.display.set_mode((500, 500))
white = (255, 255, 255)
blue = (0, 0, 255)

running = True

while running:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Fill the background with white
    screen.fill(white)
```

```
import pygame

pygame.init()
screen = pygame.display.set_mode((500, 500))
white = (255, 255, 255)
blue = (0, 0, 255)

running = True

while running:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Fill the background with white
    screen.fill(white)

    # Draw a solid blue circle in the center
    pygame.draw.circle(screen, blue , (250, 250), 75)
```

```
import pygame

pygame.init()
screen = pygame.display.set_mode((500, 500))
white = (255, 255, 255)
blue = (0, 0, 255)

running = True

while running:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Fill the background with white
    screen.fill(white)

    # Draw a solid blue circle in the center
    pygame.draw.circle(screen, blue , (250, 250), 75)
```

Third param is x,y param

Fourth is radius

```
import pygame

pygame.init()
screen = pygame.display.set_mode((500, 500))
white = (255, 255, 255)
blue = (0, 0, 255)

running = True

while running:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Fill the background with white
    screen.fill(white)

    # Draw a solid blue circle in the center
    pygame.draw.circle(screen, blue , (250, 250), 75)

    # Call to update the display with drawing
    pygame.display.flip()
```

```
import pygame

pygame.init()
screen = pygame.display.set_mode((500, 500))
white = (255, 255, 255)
blue = (0, 0, 255)

running = True

while running:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    # Fill the background with white
    screen.fill(white)

    # Draw a solid blue circle in the center
    pygame.draw.circle(screen, blue , (250, 250), 75)

    # Call to update the display with drawing
    pygame.display.flip()

#when game is over
pygame.quit()
```

Let’s run this code

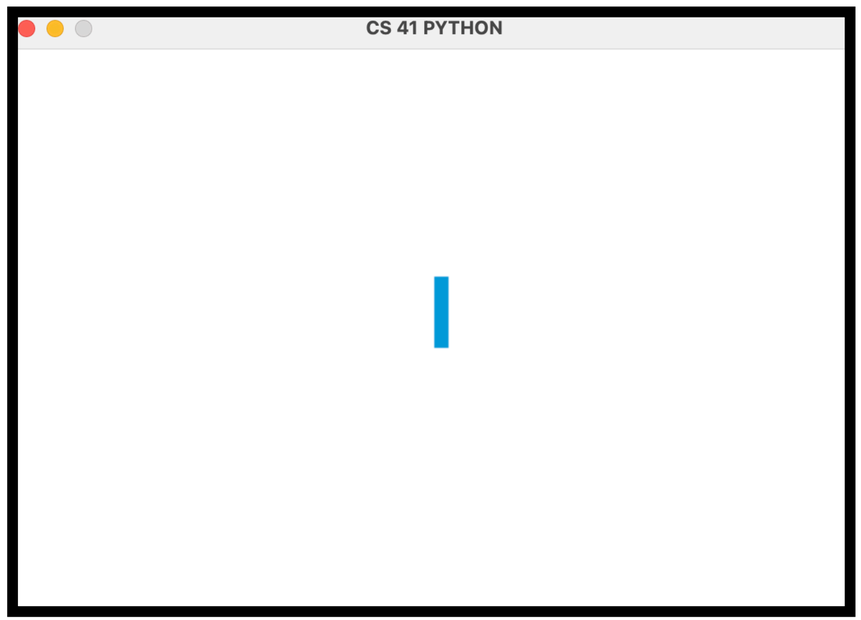
Let’s create a real game….


e, today's longest snake
length of 4491
Mancow
#1 Faux
#2 Crossfade
#3 owwie
#4 Blip
#5 Rusty
#6 Zeno's Paradox
#7 Happy Hardcore
#8 Ma$e
#9 Amek
#10 nine
ength: 364
nk: 232 of 513

Let’s start with the basics

- We know how to initialize a screen, lets add a snake

```
pygame.draw.rect(surface, color, rect)
```




```
[left, top, width, height]
```

Next, lets make it move with our keys!!

Next, lets make it move with our keys!!


```
if event.type == pygame.KEYDOWN:
```

Next, lets make it move with our keys!!


```
if event.type == pygame.KEYDOWN:
    if event.key == pygame.K_LEFT:
```

Next, lets make it move with our keys!!


```
if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_LEFT:

        if event.key == pygame.K_RIGHT:
```

Next, lets make it move with our keys!!


```
if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_LEFT:

        if event.key == pygame.K_RIGHT:

        if event.key == pygame.K_UP:
```

Next, lets make it move with our keys!!


```
if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_LEFT:

        if event.key == pygame.K_RIGHT:

        if event.key == pygame.K_UP:

        if event.key == pygame.K_DOWN:
```

How should we update our snake representation?

How should we update our snake representation?

-We currently create a list of coord tuples for each rectangle of our snake

How should we update our snake representation?

-We currently create a list of coord tuples for each rectangle of our snake

-How could we use the key events to update this list in a way to create movement?

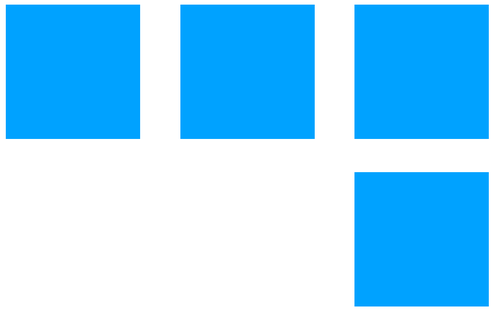
Front of the snake needs to shift down by the snake_size

Front of the snake needs to shift down by the snake_size

But now we need to traverse through this list and update all the past coords :(

ALTERNATE APPROACH

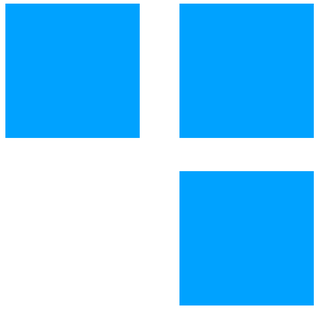
ALTERNATE APPROACH



Add a new
snake block
where the head
should be

ALTERNATE APPROACH

Delete the last one!



Important: Using Clock for Animation Speed

Important: Using Clock for Animation Speed

```
pygame.time.Clock().tick(15)
```

Ensures that the loop runs at MOST 15 frames per second if placed in the frame

Let’s write this code!

Group Activity

- Make sure you have pip installed pygame
- Download starter code from website
- Add food functionality!
- Whenever the snake runs into food, make it grow by 1!
- Food should be in a random spot on the window, should re appear once eaten
- HINT: think about what multiples your snake moves in… and how the food should only be in those coordinates
- If you want to add a different fun functionality, be my guest!!