Pygame To Web Game

asyncio:

is a Python library designed to handle asynchronous programming, which enables you to write code that can perform multiple tasks concurrently.

Using asyncio in a Python game is crucial when converting it into a web game because web browsers operate in an asynchronous environment (e.g., rendering, user input, and animations).

In other words, asynchronous programming ensures:

- The browser doesn't "freeze" while running your game.
- Input events (like keypresses or mouse clicks) and rendering updates are processed without delays.

Without an asynchronous loop, the browser environment won't execute the game properly because it expects tasks to yield control periodically to process events like rendering or user input.

pygbag:

converts Python games into web-compatible formats (using WebAssembly) that can run in browsers.

To install pygbag use the command (depending on your Python setup):

`pip install pygbag` or `pip3 install pygbag`

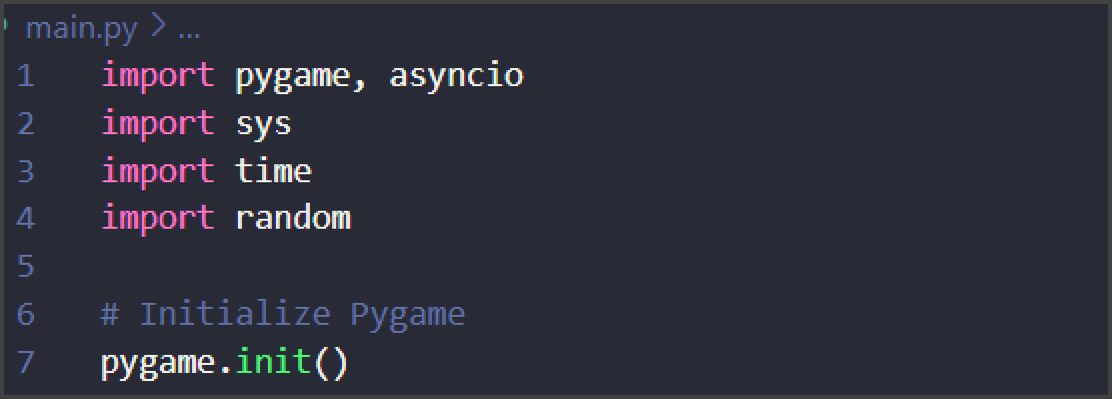
Verify with:

```
pygbag --version
```

After installing pygbag, you might need to edit your system's PATH environment variable if the pygbag command isn't recognized.

Steps

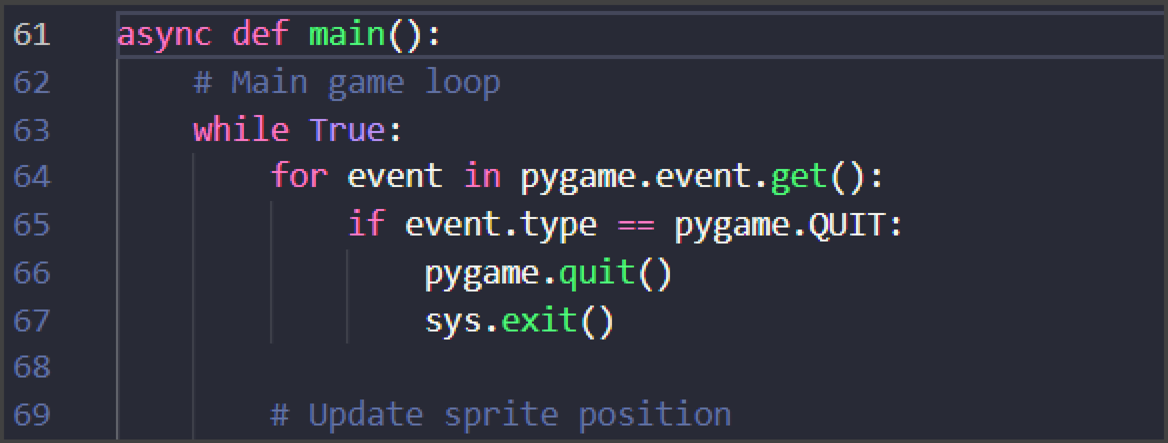
1. Import the asyncio library to your Python game:



```python
import pygame, asyncio
import sys
import time
import random

# Initialize Pygame
pygame.init()
```

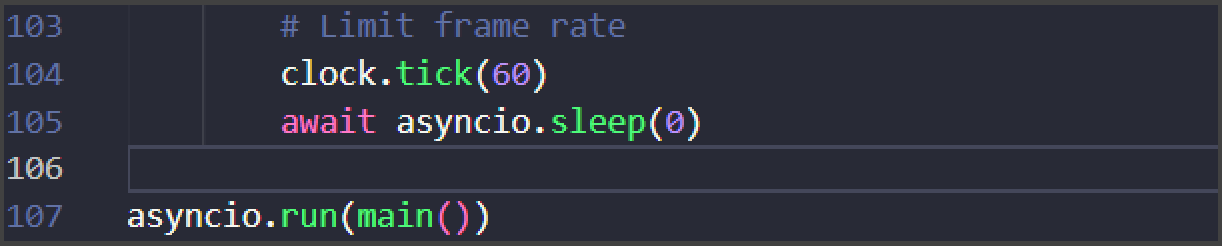
2. Wrap your game loop (while loop) with async def main():



```python
async def main():
    # Main game loop
    while True:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                sys.exit()

        # Update sprite position
```

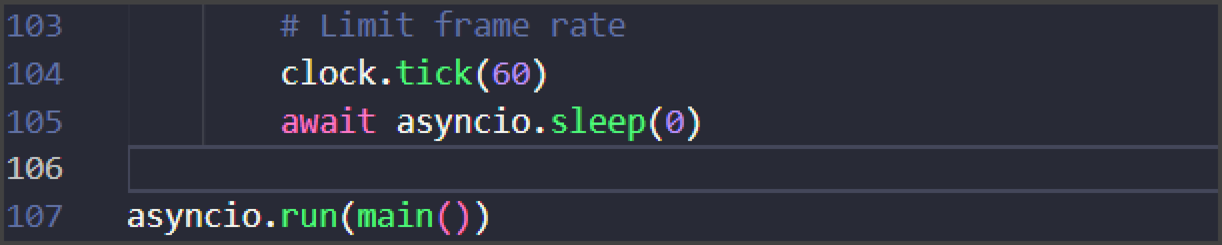
3. The last line of your game loop includes:



```python
        # Limit frame rate
        clock.tick(60)
        await asyncio.sleep(0)

asyncio.run(main())
```

4. Outside of the game loop call the game:

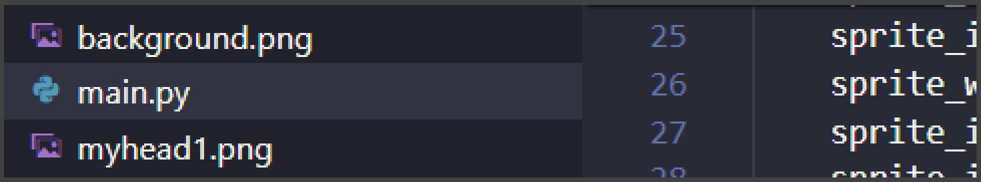


```
        # Limit frame rate
        clock.tick(60)
        await asyncio.sleep(0)

asyncio.run(main())
```

5. Fix any variable errors.

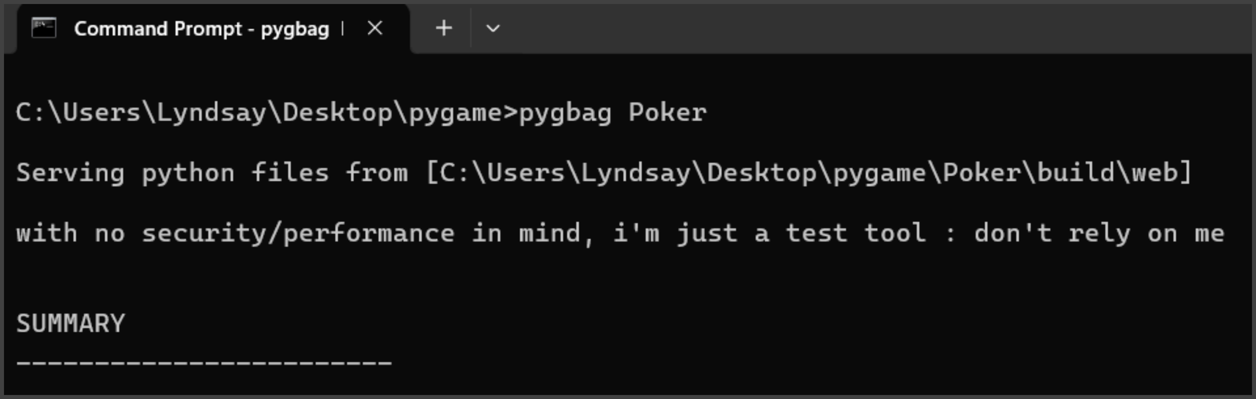
6. Rename your game file to main.py.



```
background.png      25      sprite_i
main.py             26      sprite_w
myhead1.png         27      sprite_i
```

6. Once the game is running, open a CMD and navigate to the directory that contains your game folder.

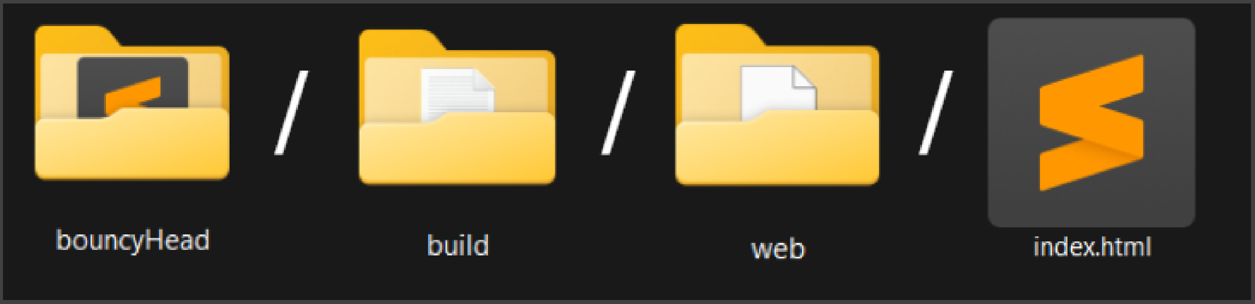
7. Run the following command to compile your game into a binary build:



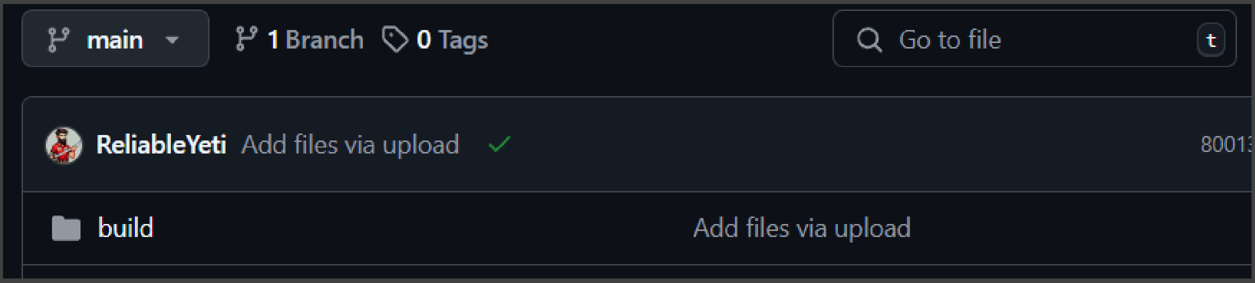
```
C:\Users\Lyndsay\Desktop\pygame>pygbag Poker
Serving python files from [C:\Users\Lyndsay\Desktop\pygame\Poker\build\web]
with no security/performance in mind, i'm just a test tool : don't rely on me

SUMMARY
------------------------
```

8. You can now type localhost in a browser search bar and play your game, or you can embed the compiled index.html file into some HTML code and add it to your web page.



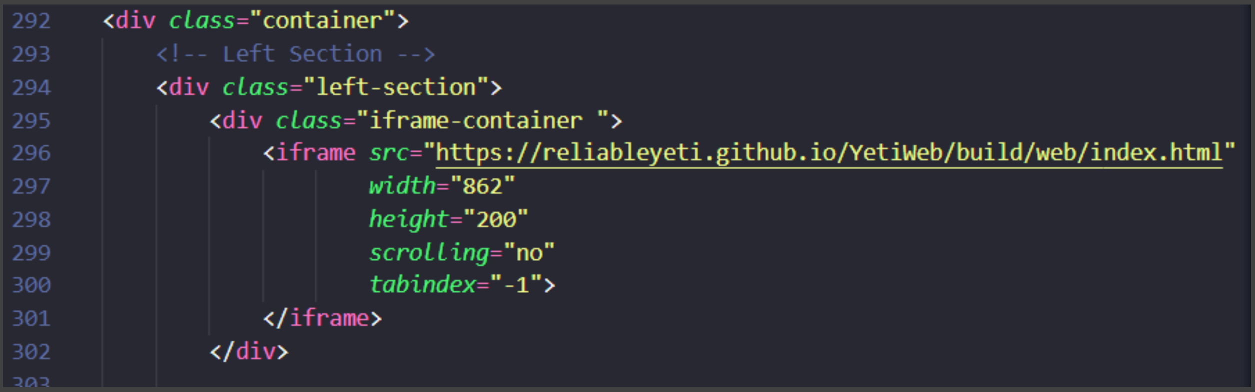
9. If you're embedding your game, you have to upload the entirety of the build folder to your host server root directory. The same directory that houses your website's index.html file.



ReliableYeti Add files via upload ✓ 80013

build Add files via upload

10. Use the full address of the games index.html file.



```
<div class="container">
    <!-- Left Section -->
    <div class="left-section">
        <div class="iframe-container ">
            <iframe src="https://reliableyeti.github.io/YetiWeb/build/web/index.html"
                    width="862"
                    height="200"
                    scrolling="no"
                    tabindex="-1">
            </iframe>
        </div>
```

Home About Topics Projects Websites Books

H