EMF2026 – Battlesnake workshop

Presented by: Bastiaan Slee

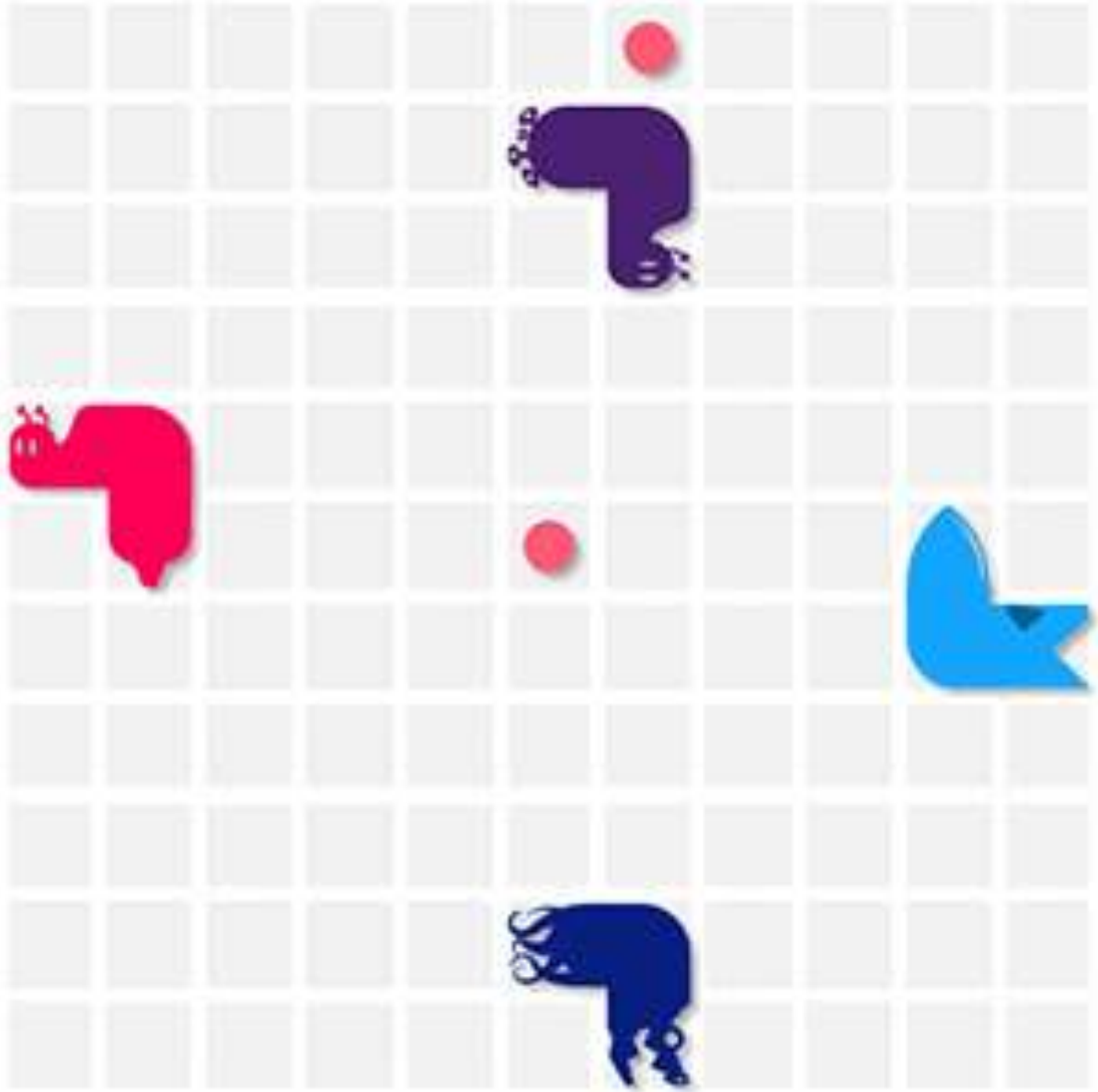
Coded by: YOU!!!!

What is Battlesnake?

Battlesnake is a competitive programming game where your code is the controller.

In the game, each Battlesnake is controlled in real-time by a live web server, responding to the Battlesnake API. It navigates the game board based on **your** algorithm, trying to find food, avoid other Battlesnakes, and survive as long as possible.


NOKIA
0000
1 ao
2 abc
def 3
4 ghi
5 jkl
mno 6
7 pqrs
8 tuv
wxyz 9
* +
0
#



How does a Battlesnake game work?

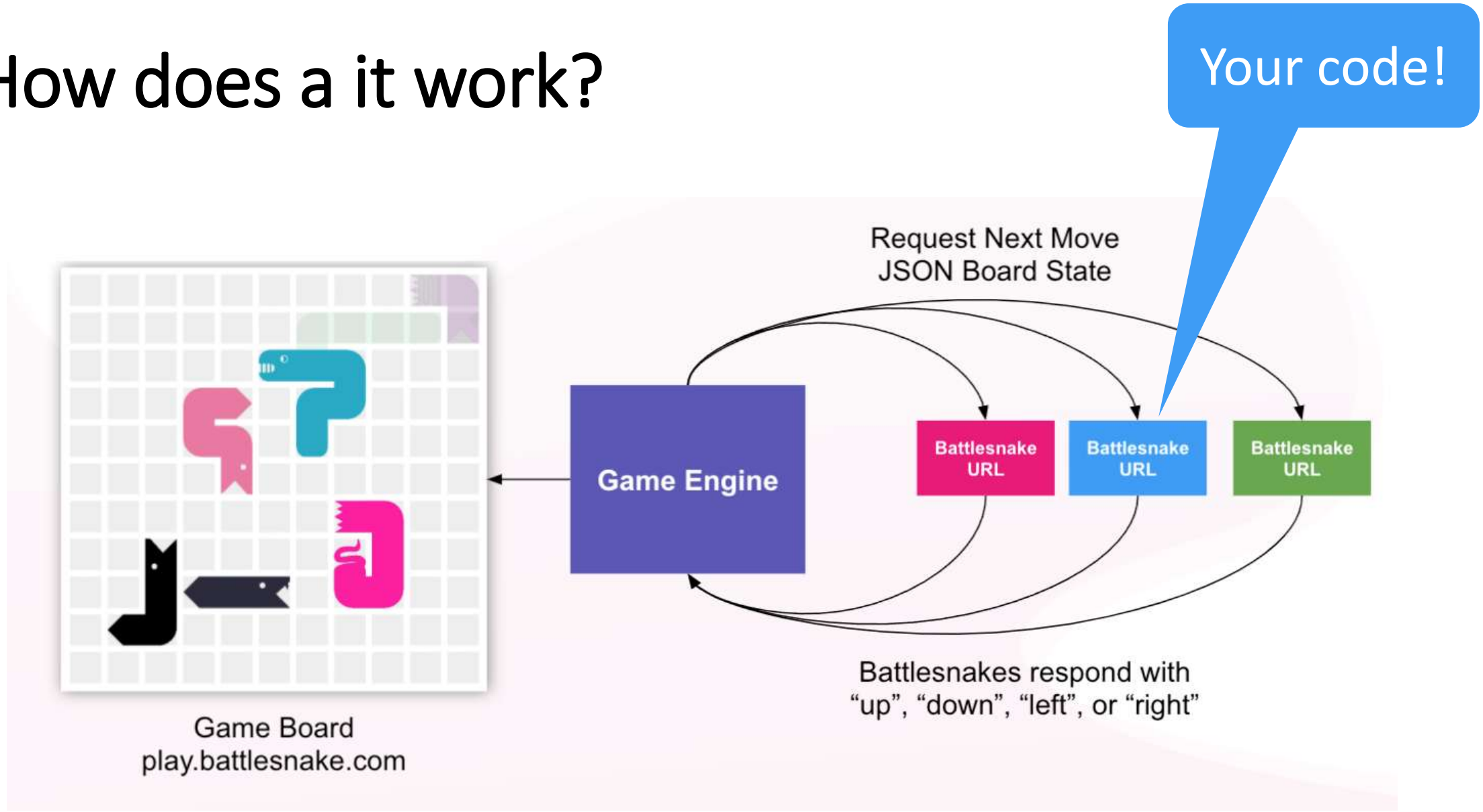
To play Battlesnake you'll need to build and deploy a **web server** that **implements the Battlesnake API**.

During each game, the **game engine** will send requests to **your web server**. The **code you write** to respond to these requests will determine how your Battlesnake behaves and ultimately whether you win or lose.

Games can be played against multiple users, who all deployed their own server and code. **Tournament will be done later this weekend*!**

* have to check availability with the Arcade, to be announced later

How does a it work?



Battlesnake Workshop

This workshop will help you deploy your first Battlesnake using a **Python Starter Project**, by following the steps from the battlesnake.com website.

This project implements just enough code for a simple Battlesnake that moves randomly (and not much else).

As a Battlesnake developer, you can use any tech stack you would like

If you're not sure which language to choose, try starting with Python or JavaScript as they're both widely supported and popular in the Battlesnake community. In the workshop we will follow the Python starter pack, but if you are more comfortable with JavaScript (or any of the other languages), please use what suits you best.

Deploy our own server and game!

For hosting your Battlesnake there are endless options. Choose what fits you best. I've documented 3 ways to host:

1) **HOSTED** on REPLIT (easiest)
2) **LOCAL** hosted on your laptop (hardest)
3) Hosted on your EMF Tildagon **BADGE** (has bonus features)

HOSTED and BADGE are done today,

For LOCAL; follow the guide from the website, and then return to this presentation

Deploy your game HOSTED on Replit

Deploy our own server and game!

For the Battlesnake we build in this workshop, we use the **Python Starter Project**. This project implements just enough code for a simple Battlesnake that moves randomly (and not much else).

The starter project can be easily be deployed to Replit

Deploy our own server and game!

Create an account at Replit: https://replit.com

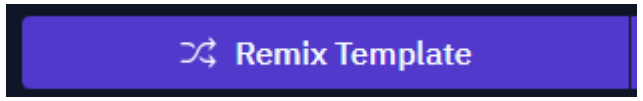
Start with making sure you are logged in on the Replit server

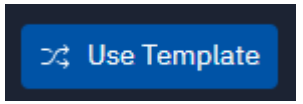
Deploy our own server and game!

https://github.com/BattlesnakeOfficial/starter-snake-python

Use the 

button

And then 

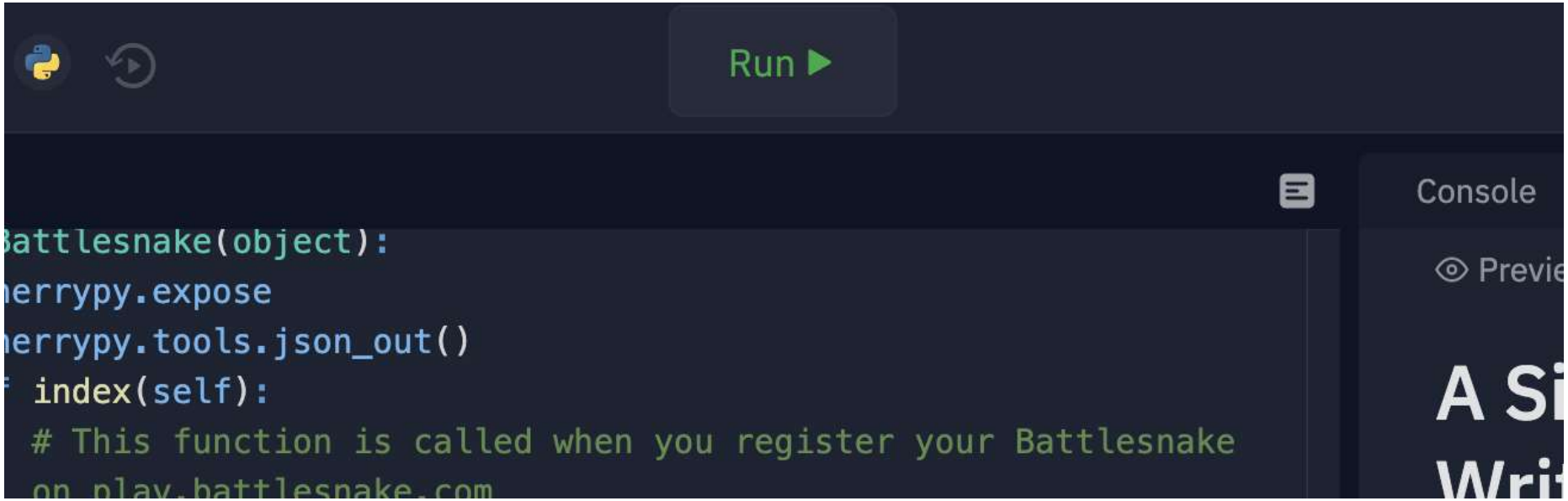
Keep the standard name, and options, click 

Blink with your eyes, is there a green RUN button on the top? Then you are done! (otherwise, blink with your eyes some more times)

You can close the Assistant tab for some more space on your screen.

Start your Battlesnake server

Once your project is ready to go in the Replit editor, launch your Battlesnake server by clicking **Run** at the top of the screen.



Start your Battlesnake server

The first time you start your Battlesnake server you may see packages and dependencies being installed in the Replit console window. This may take a few moments, but will only happen on the first run.

Once installation is complete your Battlesnake server will start. You should see live output from your server:

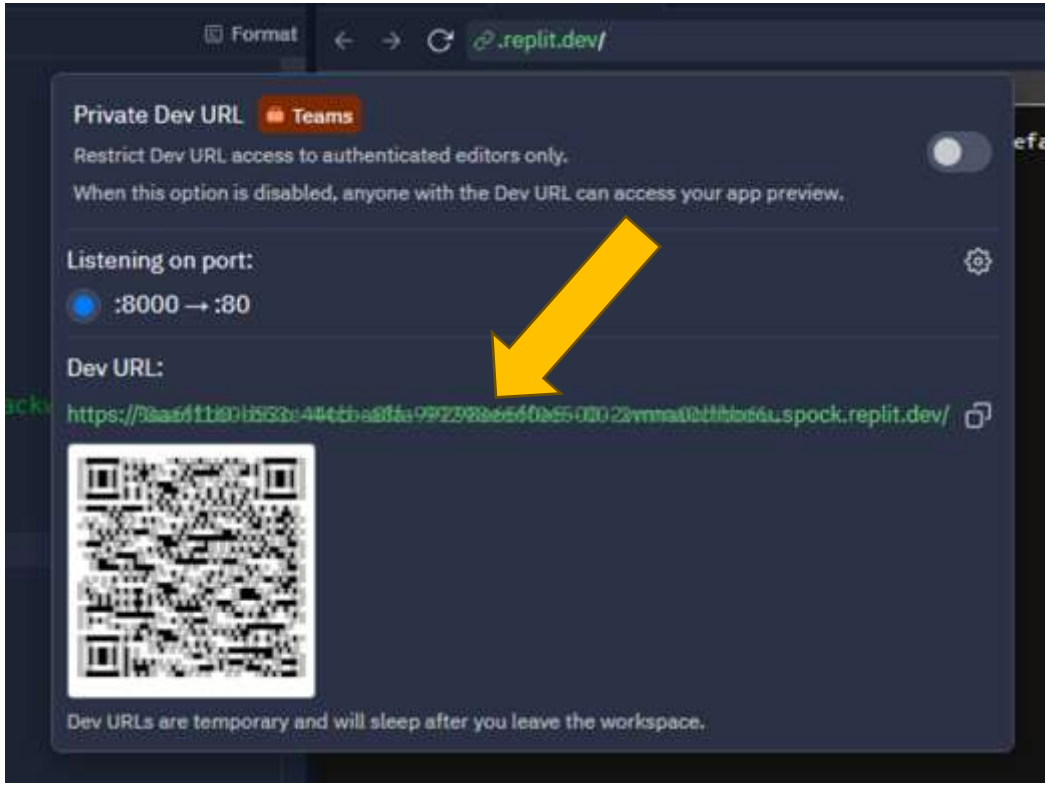
{"apiversion": "1", "author": "", "color": "#888888", "head": "default", "tail": "default"}

This means that your Battlesnake server is running correctly and is ready to play games!

Start your Battlesnake server

The Replit URL to link to the Battlesnake server is hidden, but can be revealed when clicking on .replit.dev/. You will need the Dev URL to create your Battlesnake in the next step.

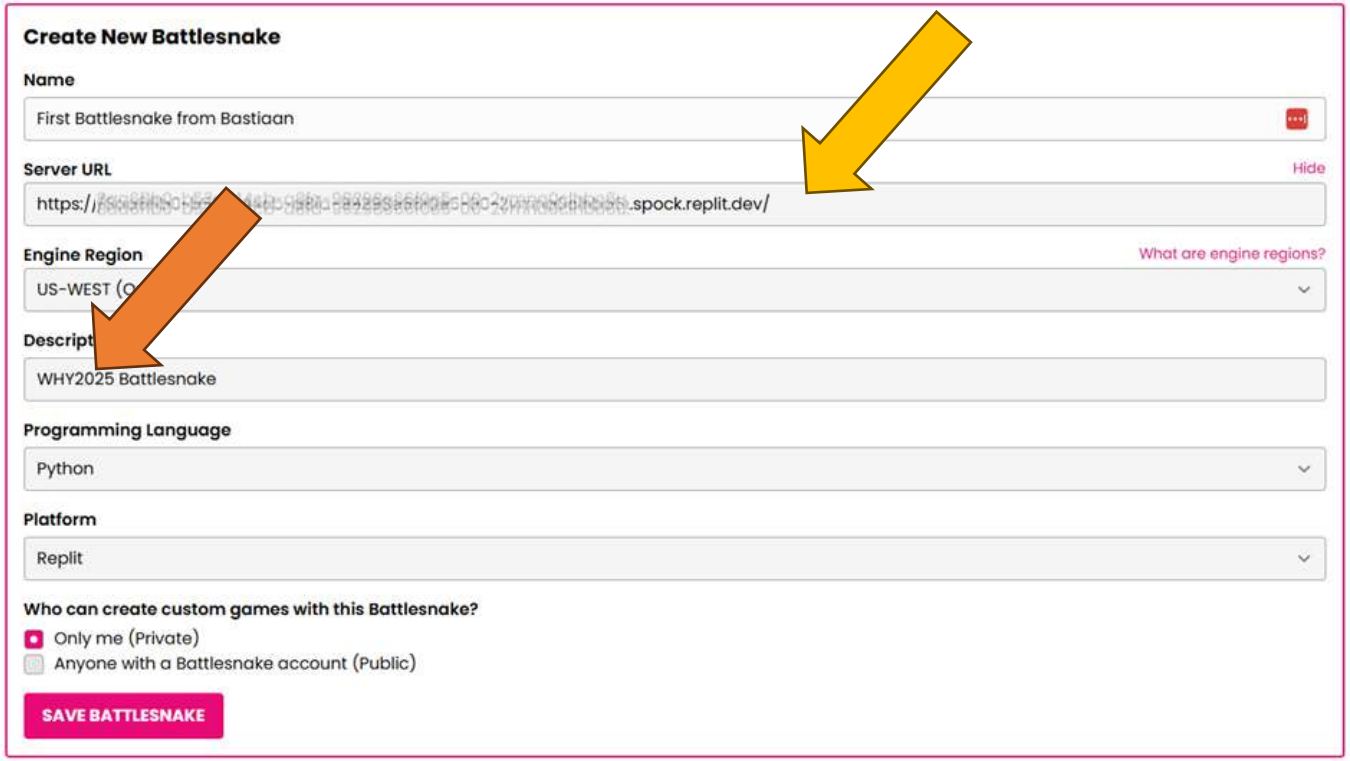
Once you have a working URL, you can register it on the Battlesnake website and start playing games.
You only need to register a Battlesnake's URL once - after that you can edit your code and restart your server as much as you'd like in between games.



Create your Battlesnake game

Create a new Battlesnake by going to https://play.battlesnake.com/account/battlesnakes.

Copy your server URL from Replit (see previous step) into the 'URL' field for your new Battlesnake. Next, give it a sn(e)aky name, and click save.



Create your Battlesnake game

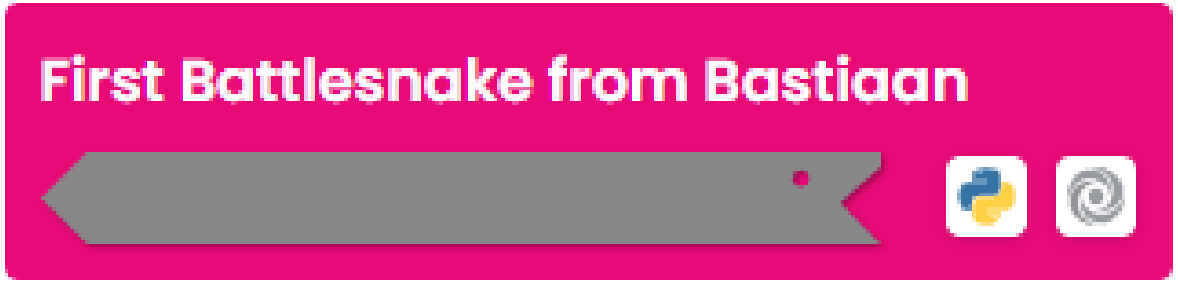
If everything is set up correctly your Battlesnake will show as operational. Some additional information will also be displayed, such as the latency between the game engine and your Battlesnake server.

If there is ever a problem or error with your Battlesnake, you can make edits to your code or configuration and then click **Ping** to test it again.

Customize your Battlesnake

To make sure everything is working correctly you can edit your Battlesnake's appearance (head, tail, and color).

The Starter Pack creates a little bit of a boring standard snake:



Lets first change that, make it your own fun snake!

Customize your Battlesnake

Go to your Replit screen, in the main.py file, you should see code that defines a head, tail, and color for your Battlesnake on line 26 – 28

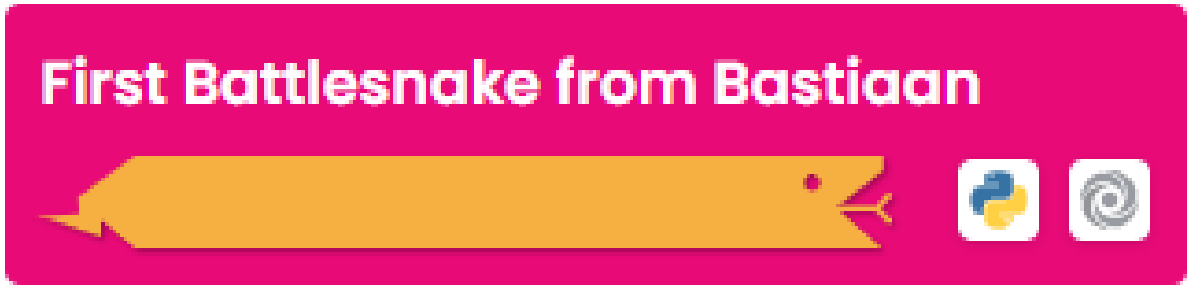
You can change these values to be whatever you'd like, selecting from the https://play.battlesnake.com/customizations.

```
23      return {
24          "apiversion": "1",
25          "author": "bastiaanslee",  # TODO: Your Battlesnake Username
26          "color": "#f5b041",  # TODO: Choose color
27          "head": "tongue",  # TODO: Choose head
28          "tail": "bolt",  # TODO: Choose tail
29      }
```

Customize your Battlesnake

Restart your Replit server (click **Stop** and **Start**) and click the **Ping** button in your Battlesnake dashboard.

Your Battlesnake should update to reflect your new appearance!



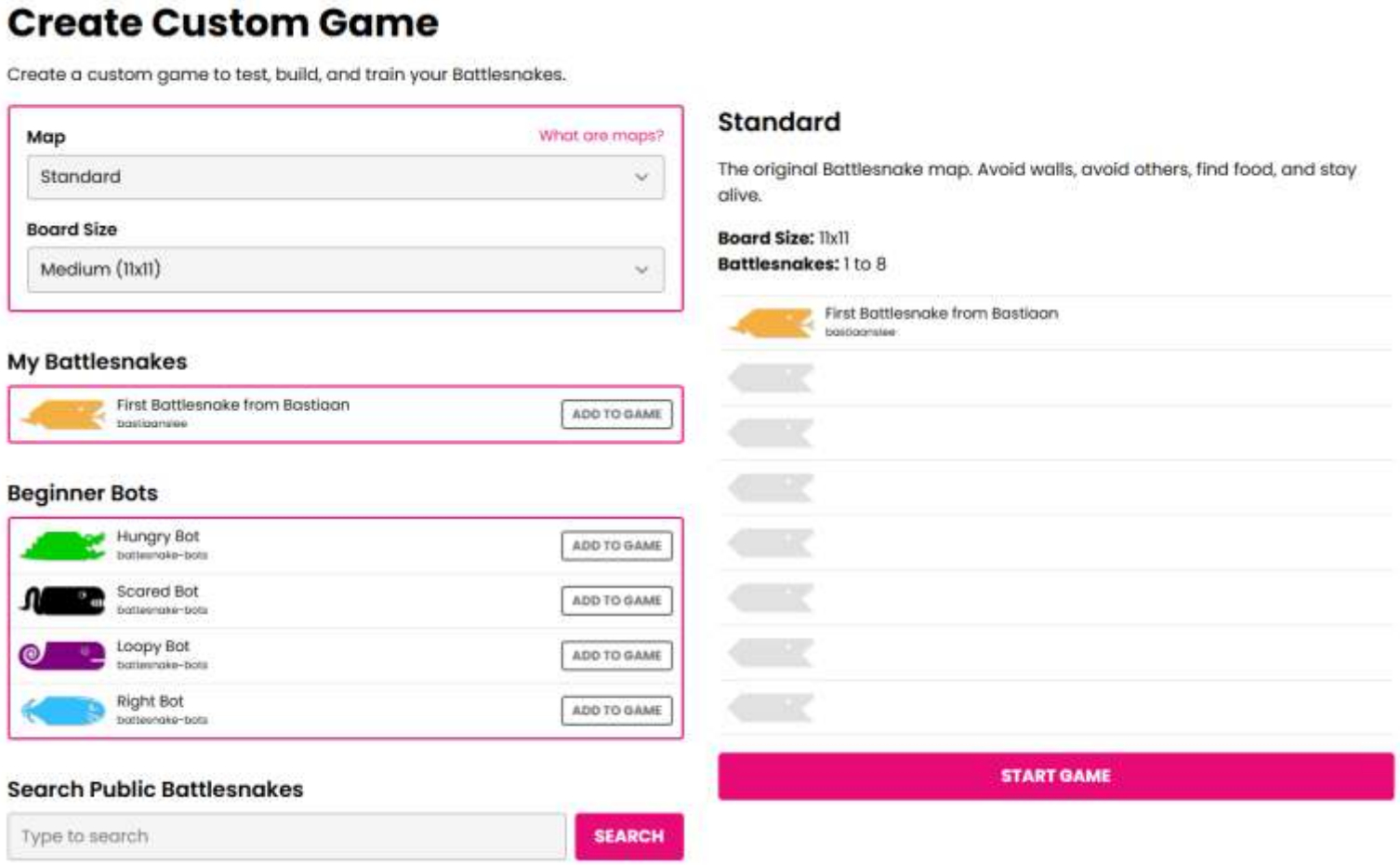
Play your first game

Your Battlesnake is now ready to play in live games! It's not very smart yet, but that's okay! Every prize winning Battlesnake has to start somewhere.

Create a custom game by going to https://play.battlesnake.com/account/games/create

Play your first game

You should see your new Battlesnake in the list. Click 'Add To Game’ to add it to the game, then click 'Start Game' to get rolling!



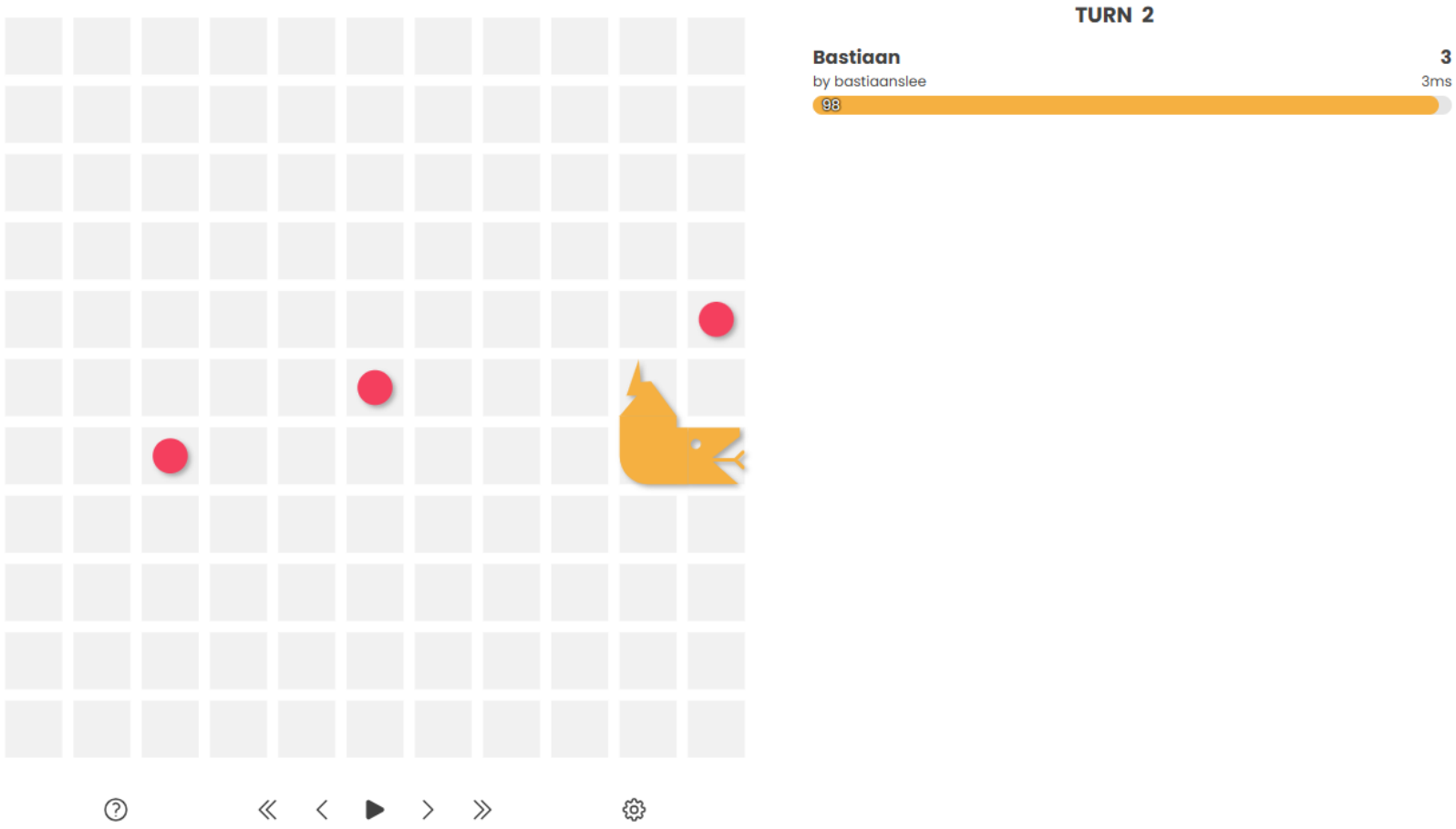
Play your first game

Now you see a game board that includes your Battlesnake.

Click the ▶ button to start playback and watch your Battlesnake in action.

You can also use the pause, forward, and back buttons to step through the game turn by turn. This can be very helpful when reviewing your server logs for each turn.

Play your first game



Play your first game

While the game runs, you see the snake moving on the board.

After the game finished, ith the control buttons, you go steps back and forward to watch your Battlesnake in action. So you can see where it did go wrong (or right!)

Play your first game

Once the game is complete, you can create a new game with the same configuration by clicking the **Create Rematch** button.

Congratulations! You've deployed your first Battlesnake!

Create YOUR Battlesnake!

Rules of the game

Winning a game of Battlesnake is simple; be the last Battlesnake remaining.

To do this your Battlesnake requires two things: **space to move** and **health to survive** - if it runs out of either, it will be eliminated!

As the game progresses, all Battlesnakes will move and grow in length. Eventually they will all run out of room and health, so finding the best path forward is critical to success.

Rules of the game

The server will send a request to each Battlesnake on the board. Each (identical) request includes detailed information about the game board and all Battlesnakes on it, including health and location.

Details are available in the Battlesnake API Reference: https://docs.battlesnake.com/api

```
{
  "game": {
    "id": "totally-unique-game-id",
    "ruleset": {
      "name": "standard",
      "version": "v1.1.15",
      "settings": {
        "foodSpawnChance": 15,
        "minimumFood": 1,
        "hazardDamagePerTurn": 14
      }
    },
    "map": "standard",
    "source": "league",
    "timeout": 500
  },
  "turn": 14,
  "board": {
    "height": 11,
    "width": 11,
    "food": [
      {"x": 5, "y": 5},
      {"x": 9, "y": 0},
      {"x": 2, "y": 6}
```

```
"food": [
  {"x": 5, "y": 5},
  {"x": 9, "y": 0},
  {"x": 2, "y": 6}
],
"hazards": [
  {"x": 3, "y": 2}
],
"snakes": [
  {
    "id": "snake-508e96ac-94ad-11ea-bb37",
    "name": "My Snake",
    "health": 54,
    "body": [
      {"x": 0, "y": 0},
      {"x": 1, "y": 0},
      {"x": 2, "y": 0}
    ],
    "latency": "111",
    "head": {"x": 0, "y": 0},
    "length": 3,
    "shout": "why are we shouting??",
    "customizations":{
      "color":"#FF0000",
```

```
    {
      "id": "snake-b67f4906-94ae-11ea-bb37",
      "name": "Another Snake",
      "health": 16,
      "body": [
        {"x": 5, "y": 4},
        {"x": 5, "y": 3},
        {"x": 6, "y": 3},
        {"x": 6, "y": 2}
      ],
      "latency": "222",
      "head": {"x": 5, "y": 4},
      "length": 4,
      "shout": "I'm not really sure...",
      "customizations":{
        "color":"#26CF04",
        "head":"silly",
        "tail":"curled"
      }
    }
  ]
},
"you": {
  "id": "snake-508e96ac-94ad-11ea-bb37",
```

```
      }
    }
  ]
},
"you": {
  "id": "snake-508e96ac-94ad-11ea-bb37",
  "name": "My Snake",
  "health": 54,
  "body": [
    {"x": 0, "y": 0},
    {"x": 1, "y": 0},
    {"x": 2, "y": 0}
  ],
  "latency": "111",
  "head": {"x": 0, "y": 0},
  "length": 3,
  "shout": "why are we shouting??",
  "customizations": {
    "color":"#FF0000",
    "head":"pixel",
    "tail":"pixel"
  }
}
}
```

Rules of the game

Battlesnakes respond with their next move. After receiving the request, each Battlesnake has limited time to respond with their next move.

If a Battlesnake's fails to provide a valid response in time, they will continue moving in the same direction as the previous.

```
{
  "move": "up",
  "shout": "I guess I'll go up then."
}
```

Rules of the game

Moves are resolved by the game engine. After all moves have been received, the game engine will update the game board accordingly.

This happens in a very specific order:

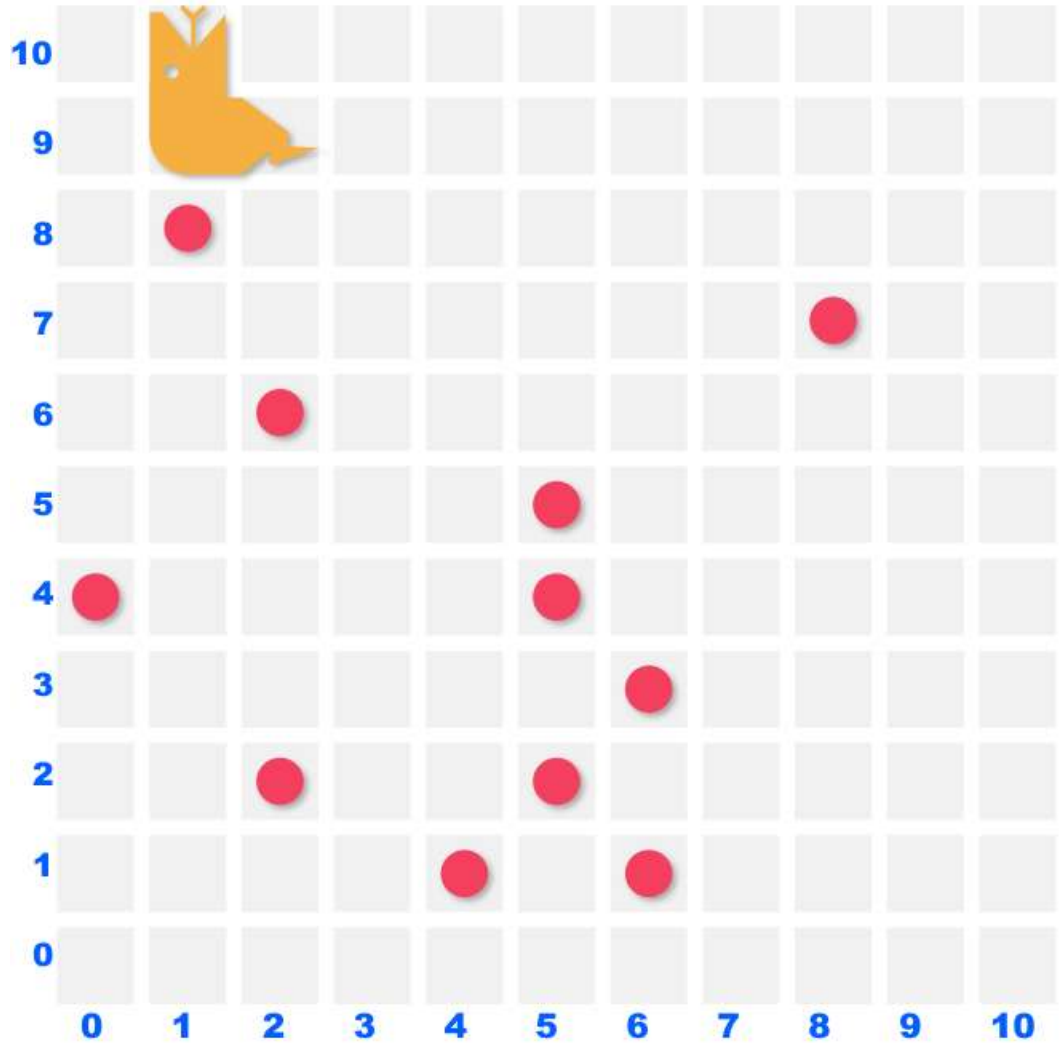
- Each Battlesnake will have its chosen move applied.
- Any Battlesnake that has found food will consume it.
- Any Battlesnake that has been eliminated is removed from the game board:
  - Health less than or equal to 0
  - Moved out of bounds
  - Collided with themselves
  - Collided with another Battlesnake
  - Collided head-to-head and lost

X and Y coordinates

Battlesnake is played on a **rectangular grid** with several Battlesnakes competing at the same time, each of variable length.

The default Battlesnake board, is 11 blocks wide and 11 blocks high.

We start counting at 0 in the left bottom corner, see this grid:



Making your Battlesnake smarter...

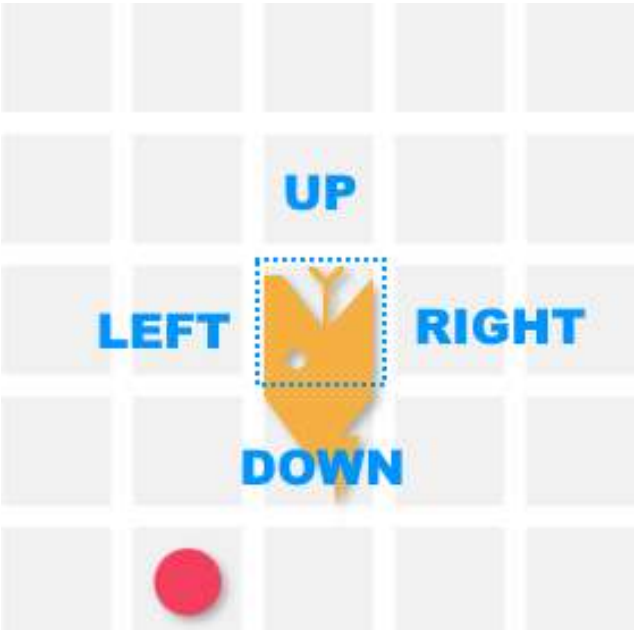
Now you're ready for the fun part -- making your Battlesnake a winner.

Typical Battlesnake development looks like this:

- Decide how you want your Battlesnake to behave differently
- Program this behaviour into your server's move response
- Restart your server with the new changes
- Create a new game and test the new behaviour
- Repeat until unstoppable

Making your Battlesnake smarter...

All we are telling back to the server is where to go:



Your code decides what direction, after you teached it some tricks

```
{
  "move": "up",
  "shout": "I guess I'll go up then."
}
```

Making your Battlesnake smarter...

In the code, we first just verify which directions are safe to go

```
is_move_safe = {
  "up": True,
  "down": True,
  "left": True,
  "right": True
}
```

Your safeguarding logic will be between these two parts

In a later step, we do return in which direction we go:

```
# Are there any safe moves left?
safe_moves = []
for move, isSafe in is_move_safe.items():
    if isSafe:
        safe_moves.append(move)

if len(safe_moves) == 0:
    print(f"MOVE {game_state['turn']}: No safe moves detected! Moving down")
    return {"move": "down"}

# Choose a random move from the safe ones
next_move = random.choice(safe_moves)

print(f"MOVE {game_state['turn']}: {next_move}")
return {"move": next_move}
```

Starter pack: example trick

The starter pack has one example trick included:

we are not going backwards, running into ourselves.

```
"you": {
  "id": "snake-508e96ac-
  "name": "My Snake",
  "health": 54,
  "body": [
    {"x": 0, "y": 0},
    {"x": 1, "y": 0},
    {"x": 2, "y": 0}
  ],
```

```
54    # We've included code to prevent your Battlesnake from moving backwards
55    my_head = game_state["you"]["body"][0]  # Coordinates of your head
56    my_neck = game_state["you"]["body"][1]  # Coordinates of your "neck"
57
58    if my_neck["x"] < my_head["x"]:  # Neck is left of head, don't move
59        is_move_safe["left"] = False
60
61    elif my_neck["x"] > my_head["x"]:  # Neck is right of head, don't mov
62        is_move_safe["right"] = False
63
64    elif my_neck["y"] < my_head["y"]:  # Neck is below head, don't move
65        is_move_safe["down"] = False
66
67    elif my_neck["y"] > my_head["y"]:  # Neck is above head, don't move
68        is_move_safe["up"] = False
```

TODO1: Board Boundaries

```
"board": {
  "height": 11,
  "width": 11,
  "food": [
```

You'll want to be **aware of the boundaries of the game board** at all times. Moving outside these boundaries does result in immediate elimination.

```
# TODO: Step 1 - Prevent your Battlesnake from moving out of bounds
# board_width = game_state['board']['width']
# board_height = game_state['board']['height']
```

TODO1: Board Boundaries

Possible solution:

All single IF statements, as you can hit 2 board edges at the same time in the corners!

```
# TODO: Step 1 - Prevent your Battlesnake from moving out of bounds
board_width = game_state['board']['width']
board_height = game_state['board']['height']

if my_head["x"] == 0:  # Head is on left board edge, don't move left
    is_move_safe["left"] = False

if my_head["x"] + 1 == board_width:  # Head is on right board edge, don't move right
    is_move_safe["right"] = False

if my_head["y"] == 0:  # Head is on bottom board edge, don't move down
    is_move_safe["down"] = False

if my_head["y"] + 1 == board_height:  # Head is on top board edge, don't move up
    is_move_safe["up"] = False
```

TODO2: Self Collisions

If your Battlesnake collides with its own body, it will be eliminated immediately. This is the easiest type of collision to avoid. The logic you will create now, will be reused later.

```
  "you": {
    "id": "snake-508e96ac-
    "name": "My Snake",
    "health": 54,
    "body": [
      {"x": 0, "y": 0},
      {"x": 1, "y": 0},
      {"x": 2, "y": 0}
    ],
```

```
# TODO: Step 2 - Prevent your Battlesnake from colliding with itself
# my_body = game_state['you']['body']
```

TODO2: Self Collisions

Possible solution:

All single IF statements, as your snake can have a U form!

This duplicates with not going backwards, so you can remove that bit.

```
# TODO: Step 2 - Prevent your Battlesnake from colliding with itself
my_body = game_state['you']['body']

for body_part in my_body:
    if my_head["x"] - 1 == body_part["x"] and my_head["y"] == body_part["y"]:
        is_move_safe["left"] = False

    if my_head["x"] + 1 == body_part["x"] and my_head["y"] == body_part["y"]:
        is_move_safe["right"] = False

    if my_head["x"] == body_part["x"] and my_head["y"] - 1 == body_part["y"]:
        is_move_safe["down"] = False

    if my_head["x"] == body_part["x"] and my_head["y"] + 1 == body_part["y"]:
        is_move_safe["up"] = False
```

TODO3: Other snake Collisions

Battlesnake is a game of not just wits, but also deep honor.

If your Battlesnake attempts to consume another Battlesnake by moving into it, it will be eliminated from the game.

So now avoid other snakes!

First think of what we have done so far, and the dataset the server provides us.

```
"snakes": [
  {
    "id": "snake-508e96ac-94ad-11ea-bb37",
    "name": "My Snake",
    "health": 54,
    "body": [
      {"x": 0, "y": 0},
      {"x": 1, "y": 0},
      {"x": 2, "y": 0}
    ],
    "latency": "111",
    "head": {"x": 0, "y": 0},
    "length": 3,
    "shout": "why are we shouting??",
    "customizations":{
      "color":"#FF0000",
      "head":"pixel",
      "tail":"pixel"
    }
  },
  {
    "id": "snake-b67f4906-94ae-11ea-bb37",
```

TODO3: Other snake Collisions

Possible solution:

```
# TODO: Step 3 - Prevent your Battlesnake from colliding with other Battlesnakes
opponents = game_state['board']['snakes']

for opponent in opponents:
    opponent_body = opponent['body']

    for body_part in opponent_body:
        if my_head["x"] - 1 == body_part["x"] and my_head["y"] == body_part["y"]:
            is_move_safe["left"] = False

        if my_head["x"] + 1 == body_part["x"] and my_head["y"] == body_part["y"]:
            is_move_safe["right"] = False

        if my_head["x"] == body_part["x"] and my_head["y"] - 1 == body_part["y"]:
            is_move_safe["down"] = False

        if my_head["x"] == body_part["x"] and my_head["y"] + 1 == body_part["y"]:
            is_move_safe["up"] = False
```

Keeping an eye on your health!

Health

On most maps each Battlesnake will begin the game with full health, and on every turn your Battlesnake will lose one point of health.

Battlesnakes that reach zero health will be eliminated, meaning that to stay alive as long as possible, you must find ways to replenish your health.

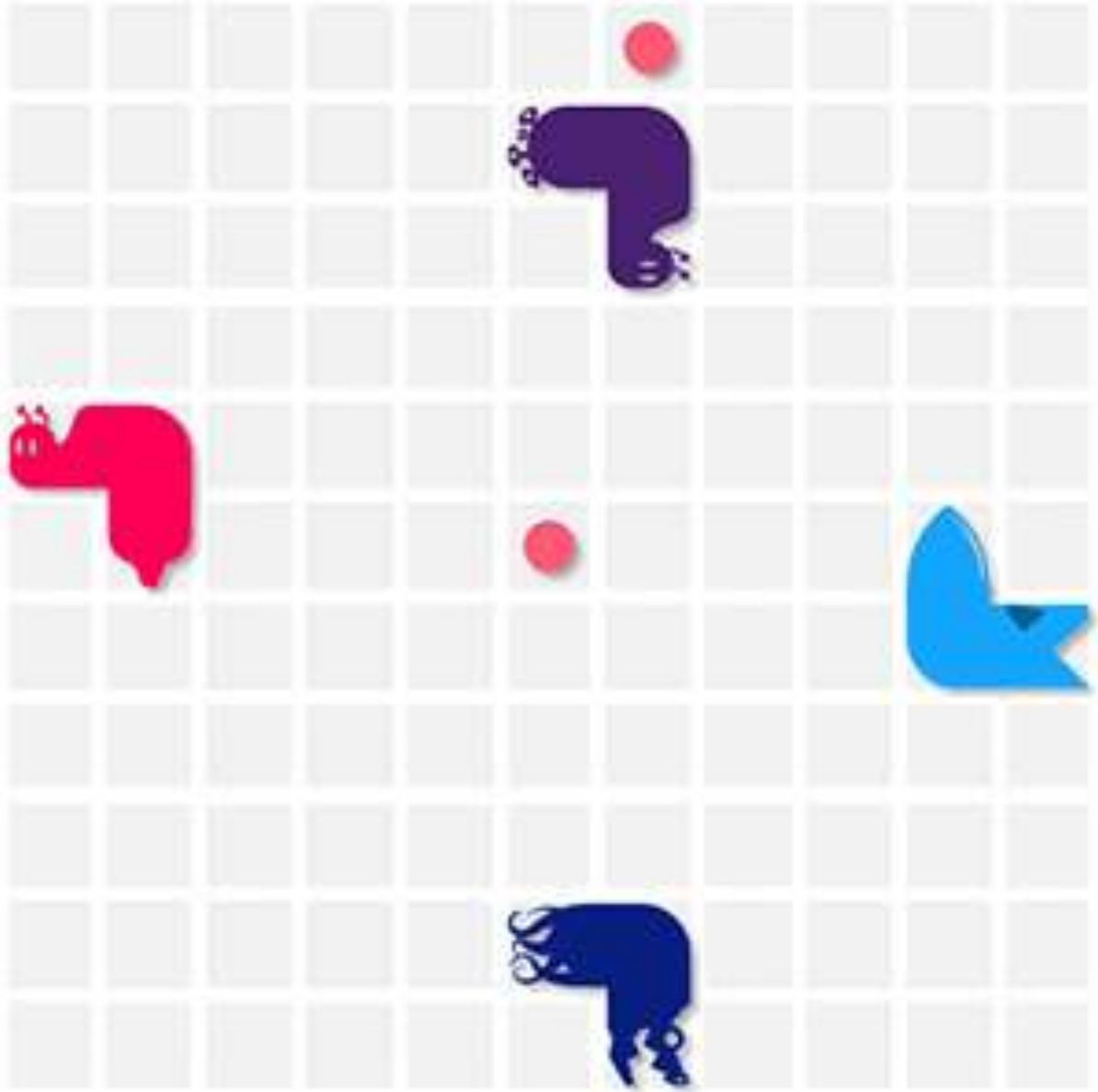
Task 4: Consuming Food

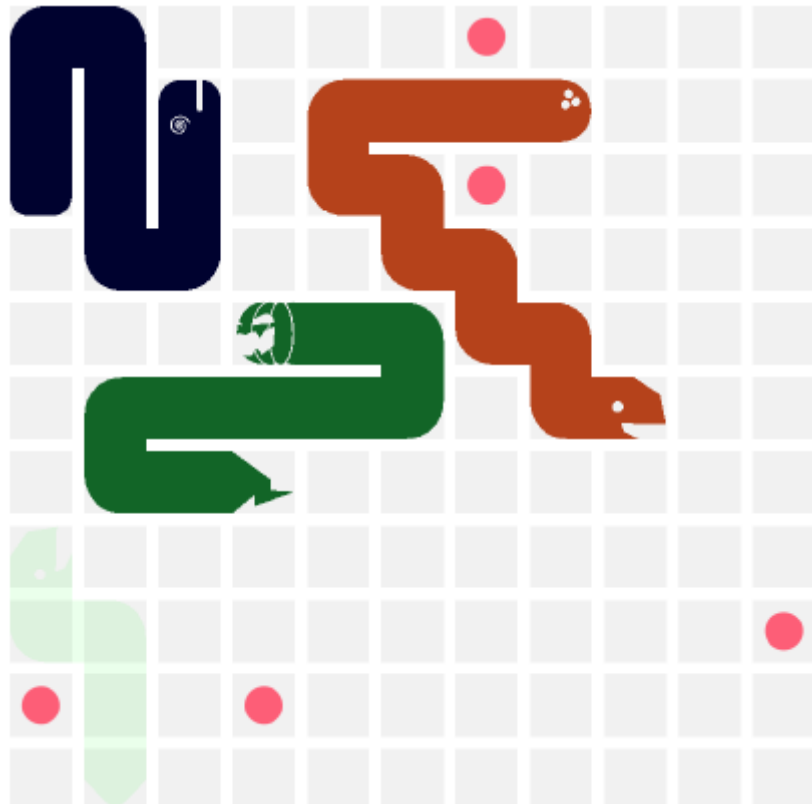
The most common way to restore health is consuming food.

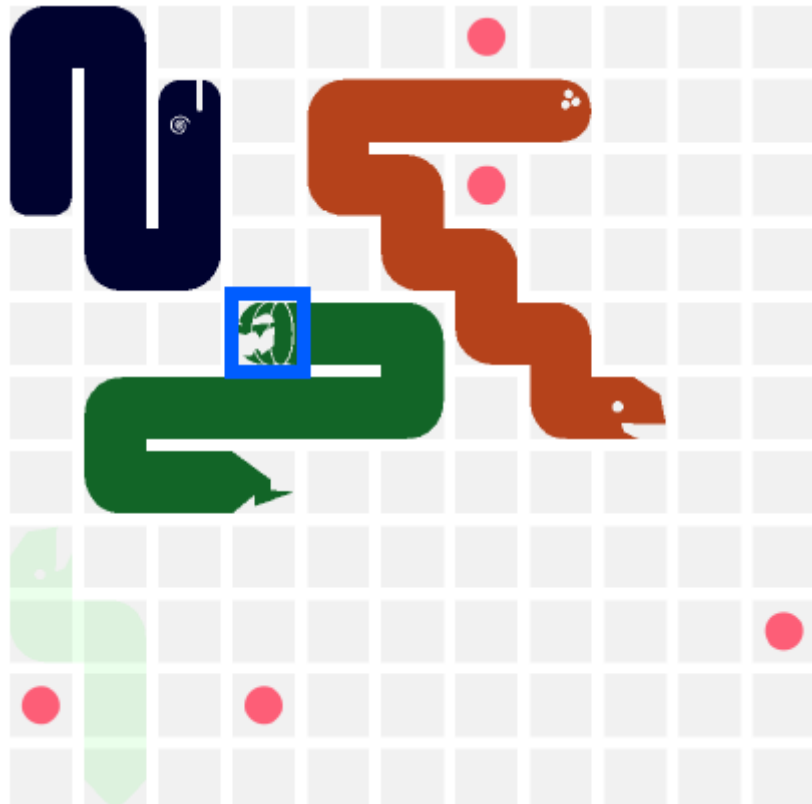
Food will appear on the game board throughout the game, and Battlesnakes that consume it, are filling their health to its maximum value (usually 100 points).

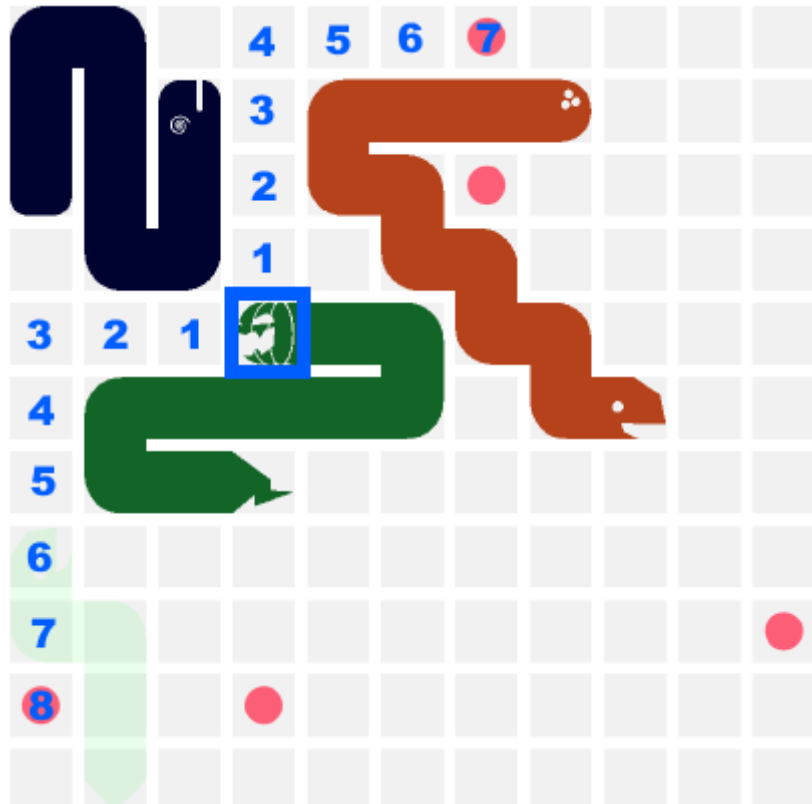
However, over-consumption comes at a cost. Each time a Battlesnake consumes food it will grow longer on the next turn, making it more challenging for itself (and others!) to survive.

On the other hand, consuming food by your snake, takes it away from other snakes. So maybe you can outlive them.








4
5
6
7
3
2
1
3
2
1
4
5
6
7
8

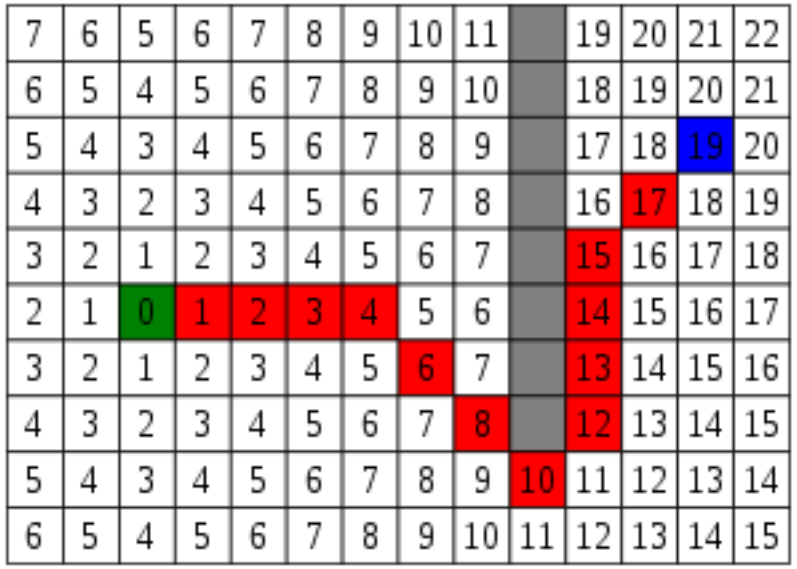
New approach

Find an algorithm

- That does what you need
- And does that fast

Examples:

- Flood Fill (see if you can escape)
- Dijkstra (finding path to target)
- A* Pathfinding (finding path to target)



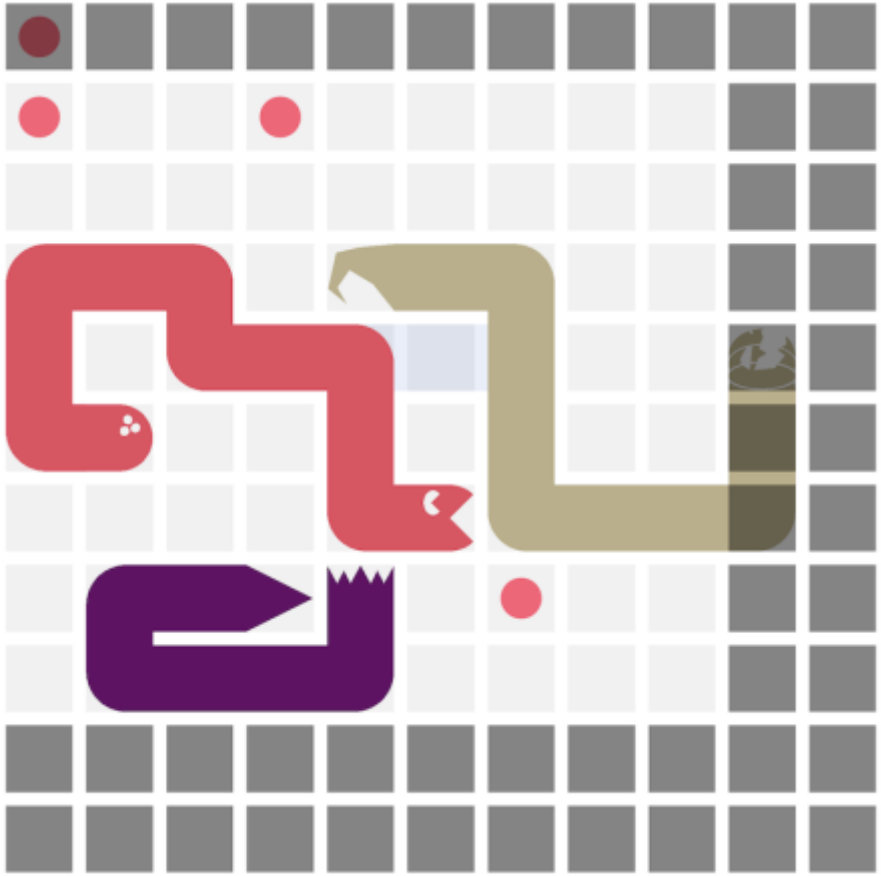
7	6	5	6	7	8	9	10	11		19	20	21	22
6	5	4	5	6	7	8	9	10		18	19	20	21
5	4	3	4	5	6	7	8	9		17	18	19	20
4	3	2	3	4	5	6	7	8		16	17	18	19
3	2	1	2	3	4	5	6	7		15	16	17	18
2	1	0	1	2	3	4	5	6		14	15	16	17
3	2	1	2	3	4	5	6	7		13	14	15	16
4	3	2	3	4	5	6	7	8		12	13	14	15
5	4	3	4	5	6	7	8	9	10	11	12	13	14
6	5	4	5	6	7	8	9	10	11	12	13	14	15

https://github.com/BattlesnakeOfficial/awesome-battlesnake

Other considerations: Hazards

With Hazards, you are allowed to go over them, but it will cost you extra energy (15 health instead of 1!)

Our **is_move_safe** “False” or “True” value, could get a new “Optional”



Other considerations: Head-to-Head Collisions

An exception to Body Collisions is when two Battlesnakes meet head-to-head on the same grid location.

In this scenario, the outcome is a little more complex:

- The longer Battlesnake will survive and the shorter will be eliminated.
- If both Battlesnakes are the same length, then both are eliminated.

Competitive Battlesnake developers use head-to-head collisions to gain advantages over their opponents and force eliminations.

Badge bonus!

Badge bonus!

There also is an EMF Tildagon Badge implementation of Battlesnake!

Keep in mind that the Badge is not a powerhouse. The standard games on the Battlesnake server are too fast (500ms response time). So we need some local setup to allow for at least 2000ms response time.

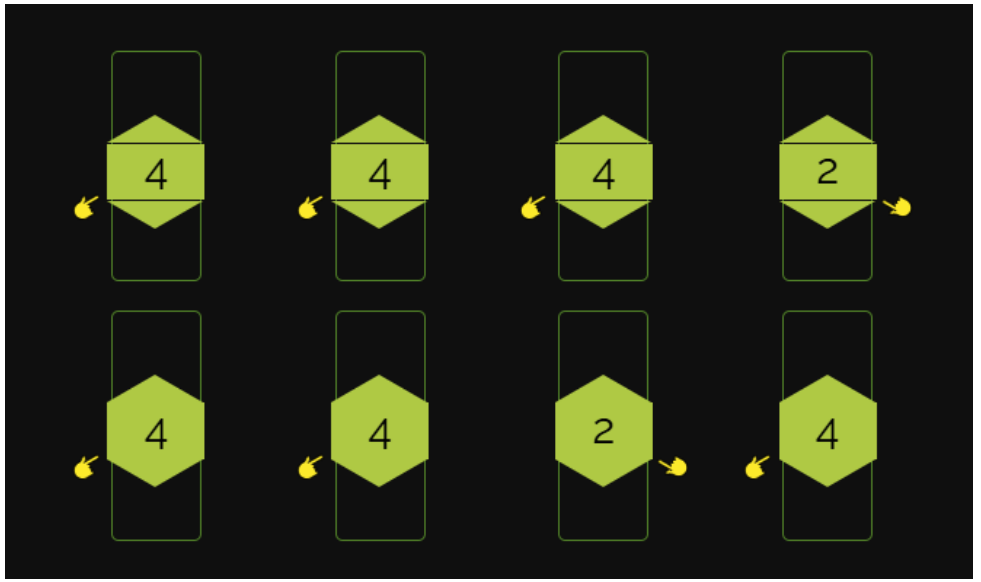
Deploy our own server and game!

For your EMF Tildagon badge (both 2024 and 2026 frontboard), there is made an app. You can deploy your code to this app for your own game logic.

Bonus functions on the badge: button + joystick, tilt movement (imu/ accelerometer) and rotation (compass) control!

Deploy our own server and game!

In the badge-app-store, download the “Battlesnake” app, or use code



Start the app for first initialization. Use “Info” to see the IP-address of your badge. For example: **http://192.168.1.234:8001**

Deploy our own server and game!

Open **http://192.168.1.234:8001** in your browser and you should see

{"apiversion":"1","author":"","color":"#888888","head":"default","tail":"default","name":"","badgemode":"logic"}

Congratulations, your Battlesnake works!

Deploy our own server and game!

Install Git on Windows: gitforwindows.org

Download the latest version and run the installer.

Once the installer has started, follow the instructions as provided in the wizard screen until the installation is complete.

Open the command prompt and type **git version** to verify Git was installed.

Deploy our own server and game!

Install Golang on Windows: go.dev

Download the latest version and run the installer.

Once the installer has started, follow the instructions as provided in the wizard screen until the installation is complete.

Make sure to select to add Go the environment path.

Open the command prompt and type **go version** to verify Go was installed.

Deploy our own server and game!

Install Battlesnake CLI

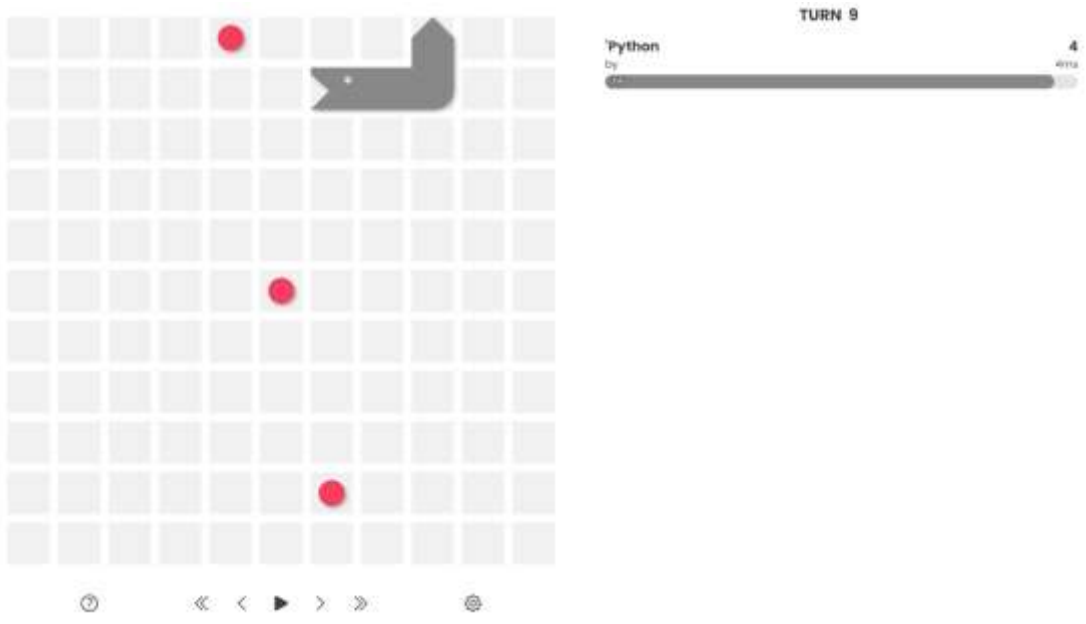
Open the command prompt and type:
go install github.com/BattlesnakeOfficial/rules/cli/battlesnake@latest

Create your Battlesnake game

Open a second command prompt and type:

battlesnake play -W 11 -H 11 -g solo --browser -t 5000 --name "Python Starter Project" --url http://192.168.1.234:8001

This will open a browser to show your game



error: Battlesnake is an unknown command

During installation, you did not select to add GO to the environment path. There are 2 options to get this fixed

Option 1:

Add to your bash profile:

export GOPATH="$HOME/go" PATH="$GOPATH/bin:$PATH"

Refresh Terminal

Option 2:

Run the game with:

~/go/bin/battlesnake play -W 11 -H 11 -g solo --browser --name "Python Starter Project" --url http://192.168.1.234:8001

Connect your laptop and badge

The badge runs Micropython, which is very similar to Python.

To edit the files, you need to make a connection between your laptop and the badge. The tool **Thonny** will take care of this.

Go to https://thonny.org/ and download the latest version

Thonny - MicroPython device :: /apps/Battlesnake/MyBattlesnake.py @ 195 : 1

File Edit View Run Tools Help

Files

This computer
C: \ Users \ basti \ EMF Badge

- Battlesnake_2
- ElectroMusicFeast

MicroPython device
/ apps / Battlesnake

- ahttpserver
- app.py
- logo.jpg
- MyBattlesnake.default
- MyBattlesnake.py
- mysnake.json
- mysnake_head_down.png
- mysnake_head_home.png
- mysnake_head_left.png
- mysnake_head_right.png
- mysnake_head_up.png
- mysnake_tail_down.png
- mysnake_tail_home.png
- mysnake_tail_left.png
- mysnake_tail_right.png
- mysnake_tail_up.png
- nosnake_head.png
- nosnake_head_down.png
- nosnake_head_home.png
- nosnake_head_left.png
- nosnake_head_right.png
- nosnake_head_up.png
- nosnake_tail.png
- nosnake_tail_down.png
- nosnake_tail_home.png
- nosnake_tail_left.png
- nosnake_tail_right.png
- nosnake_tail_up.png
- tildagon.toml

[MyBattlesnake.py]

```
1  # Welcome to
2  # __________           __    __  .__                               __
3  # \______   \_____ _/  |__/  |_|  |   ____   ______ ____ _____    |  | __ ____
4  #  |    |  _/\__  \\   __\   __\  | _/ __ \ /  ___//    \\__  \   |  |/ // __ \
5  #  |    |   \ / __ \|  |  |  | |  |_\  ___/ \___ \|   |  \/ __ \|    <\  ___/
6  #  |________/(______/__|  |__| |____/\_____>_____>___|__(______/__|__\\_____>
7  #
8  # This file can be a nice home for your Battlesnake logic and helper functions.
9  #
10 # To get you started we've included code to prevent your Battlesnake from moving backwards.
11 # For more info see docs.battlesnake.com
12
13 import random
14 import typing
15
16
17 # info is called when you create your Battlesnake on play.battlesnake.com
18 # and controls your Battlesnake's appearance
19 # TIP: If you open your Battlesnake URL in a browser you should see this data
20 def battlesnake_info() -> typing.Dict:
21     print("INFO")
22
```

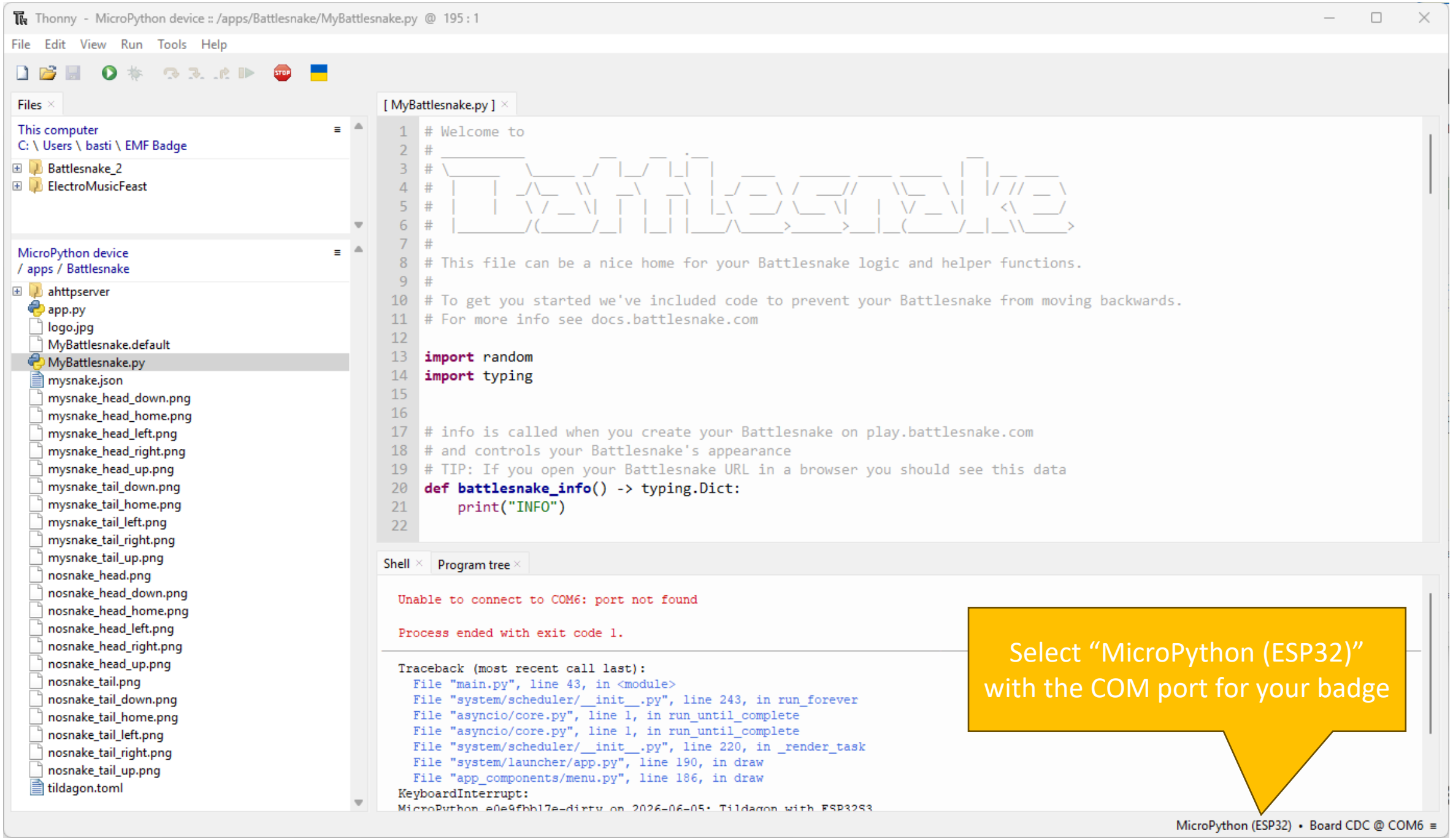
Shell | Program tree

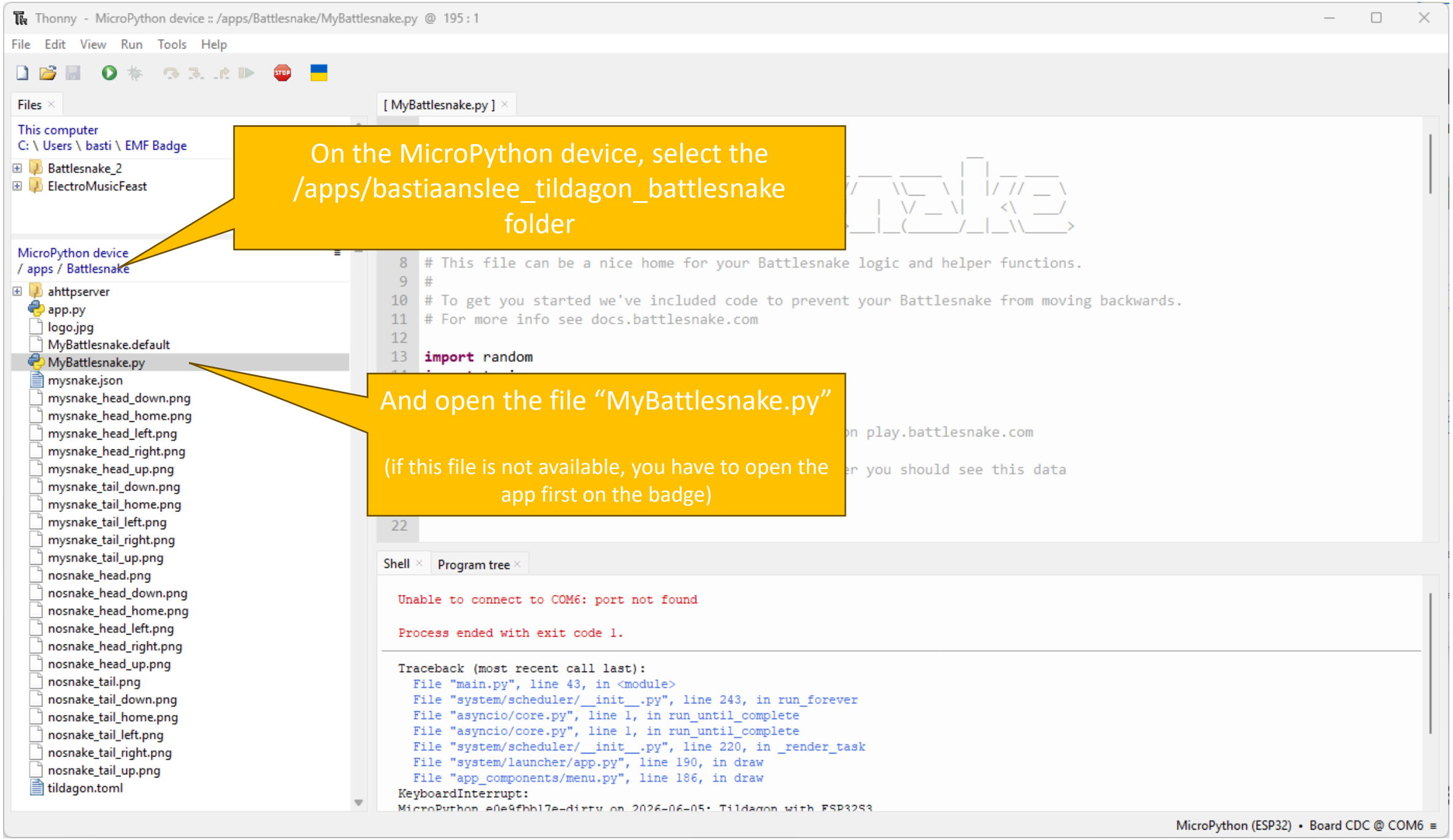
```
Unable to connect to COM6: port not found

Process ended with exit code 1.

Traceback (most recent call last):
  File "main.py", line 43, in <module>
  File "system/scheduler/__init__.py", line 243, in run_forever
  File "asyncio/core.py", line 1, in run_until_complete
  File "asyncio/core.py", line 1, in run_until_complete
  File "system/scheduler/__init__.py", line 220, in _render_task
  File "system/launcher/app.py", line 190, in draw
  File "app_components/menu.py", line 186, in draw
KeyboardInterrupt:
MicroPython e0e9fbb17e-dirty on 2026-06-05; Tildagon with ESP32S3
```

MicroPython (ESP32) • Board CDC @ COM6

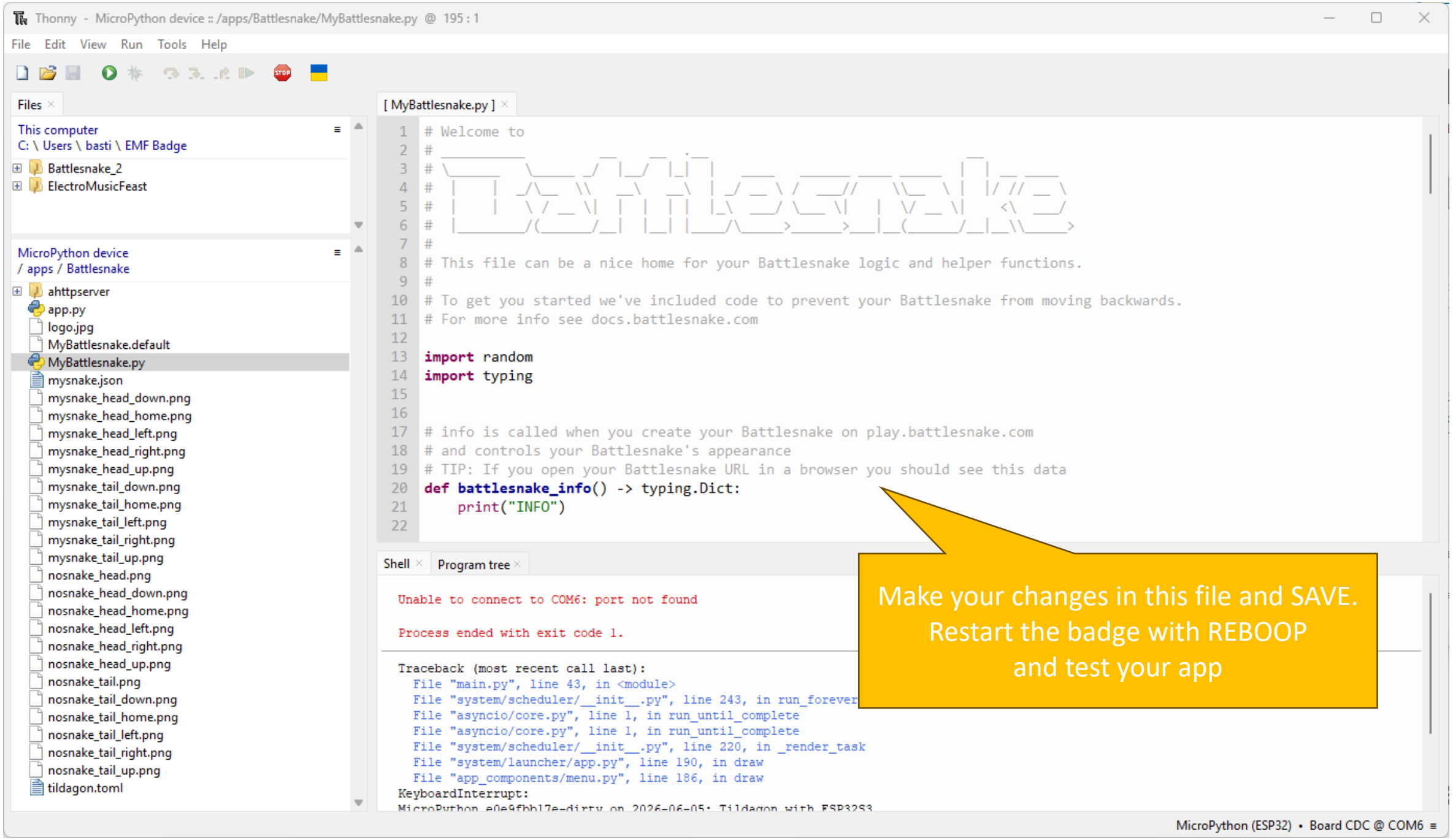

Thonny - MicroPython device :: /apps/Battlesnake/MyBattlesnake.py @ 195 : 1
File Edit View Run Tools Help
Files
This computer
C: \ Users \ basti \ EMF Badge
Battlesnake_2
ElectroMusicFeast
MicroPython device
/ apps / Battlesnake
ahttpserver
app.py
logo.jpg
MyBattlesnake.default
MyBattlesnake.py
mysnake.json
mysnake_head_down.png
mysnake_head_home.png
mysnake_head_left.png
mysnake_head_right.png
mysnake_head_up.png
mysnake_tail_down.png
mysnake_tail_home.png
mysnake_tail_left.png
mysnake_tail_right.png
mysnake_tail_up.png
nosnake_head.png
nosnake_head_down.png
nosnake_head_home.png
nosnake_head_left.png
nosnake_head_right.png
nosnake_head_up.png
nosnake_tail.png
nosnake_tail_down.png
nosnake_tail_home.png
nosnake_tail_left.png
nosnake_tail_right.png
nosnake_tail_up.png
tildagon.toml
[MyBattlesnake.py]
Welcome to
__________ __ __ .__ __
______ ____ _/ |_/ |_| | ____ ______ ____ ____ | | __ ____
| | _/__ \\ __\ __\ | _/ __ \ / ___// __ \ | |/ // __ \
| | \ / __ \| | | | | |_\ ___/ ___ \| | \/ __ \| <\ ___/
|_____/(_____/__| |__| |____/_____>_____>__|__(_____/__|_______>
#
This file can be a nice home for your Battlesnake logic and helper functions.
#
# To get you started we've included code to prevent your Battlesnake from moving backwards.
# For more info see docs.battlesnake.com
import random
import typing
# info is called when you create your Battlesnake on play.battlesnake.com
# and controls your Battlesnake's appearance
# TIP: If you open your Battlesnake URL in a browser you should see this data
def battlesnake_info() -> typing.Dict:
print("INFO")
Shell
Program tree
Unable to connect to COM6: port not found
Process ended with exit code 1.
Traceback (most recent call last):
File "main.py", line 43, in <module>
File "system/scheduler/__init__.py", line 243, in run_forever
File "asyncio/core.py", line 1, in run_until_complete
File "asyncio/core.py", line 1, in run_until_complete
File "system/scheduler/__init__.py", line 220, in _render_task
File "system/launcher/app.py", line 190, in draw
File "app_components/menu.py", line 186, in draw
KeyboardInterrupt:
Select “MicroPython (ESP32)” with the COM port for your badge
MicroPython (ESP32) • Board CDC @ COM6



On the MicroPython device, select the /apps/bastiaanslee_tildagon_battlesnake folder

And open the file “MyBattlesnake.py”

(if this file is not available, you have to open the app first on the badge)


Thonny - MicroPython device :: /apps/Battlesnake/MyBattlesnake.py @ 195 : 1
File Edit View Run Tools Help
Files
This computer
C: \ Users \ basti \ EMF Badge
Battlesnake_2
ElectroMusicFeast
MicroPython device
/ apps / Battlesnake
ahttpserver
app.py
logo.jpg
MyBattlesnake.default
MyBattlesnake.py
mysnake.json
mysnake_head_down.png
mysnake_head_home.png
mysnake_head_left.png
mysnake_head_right.png
mysnake_head_up.png
mysnake_tail_down.png
mysnake_tail_home.png
mysnake_tail_left.png
mysnake_tail_right.png
mysnake_tail_up.png
nosnake_head.png
nosnake_head_down.png
nosnake_head_home.png
nosnake_head_left.png
nosnake_head_right.png
nosnake_head_up.png
nosnake_tail.png
nosnake_tail_down.png
nosnake_tail_home.png
nosnake_tail_left.png
nosnake_tail_right.png
nosnake_tail_up.png
tildagon.toml
[MyBattlesnake.py]
Welcome to
__________ __ __ .__ __
______ ____ _/ |_/ |_| | ____ ______ ____ _____ | | __ ____
| | _/__ \\ __\ __\ | _/ __ \ / ___// __ \ | |/ // __ \
| | \ / __ \| | | | | |_\ ___/ ___ \| | \/ __ \| <\ ___/
|_____/(_____/__| |__| |____/_____>_____>__|__(_____/__|_______>
#
This file can be a nice home for your Battlesnake logic and helper functions.
#
# To get you started we've included code to prevent your Battlesnake from moving backwards.
# For more info see docs.battlesnake.com
import random
import typing
# info is called when you create your Battlesnake on play.battlesnake.com
# and controls your Battlesnake's appearance
# TIP: If you open your Battlesnake URL in a browser you should see this data
def battlesnake_info() -> typing.Dict:
print("INFO")
Shell
Program tree
Unable to connect to COM6: port not found
Process ended with exit code 1.
Traceback (most recent call last):
File "main.py", line 43, in <module>
File "system/scheduler/__init__.py", line 243, in run_forever
File "asyncio/core.py", line 1, in run_until_complete
File "asyncio/core.py", line 1, in run_until_complete
File "system/scheduler/__init__.py", line 220, in _render_task
File "system/launcher/app.py", line 190, in draw
File "app_components/menu.py", line 186, in draw
KeyboardInterrupt:
Make your changes in this file and SAVE.
Restart the badge with REBOOP
and test your app
MicroPython (ESP32) • Board CDC @ COM6

Customize your Battlesnake

To make sure everything is working correctly you can edit your Battlesnake's appearance (head, tail, and color). This works the same way as the hosted Battlesnake.

Go to your Thonny screen, in the MyBattlesnake.py file, you should see code that defines a head, tail, and color for your Battlesnake on line 23 – 31

```
19  # TIP: If you open your Battlesnake URL in a browser you should see this data
20  def battlesnake_info() -> typing.Dict:
21      print("INFO")
22
23      return {
24          "apiversion": "1",
25          "name": "Loopings",  # TODO: Your Battlesnake Username
26          "author": "Bastiaan",  # TODO: Your Battlesnake Username
27          "color": "#ffa500",  # TODO: Choose color   ffa500
28          "head": "rocket-helmet",  # TODO: Choose head
29          "tail": "rocket",  # TODO: Choose tail
30          "badgemode": "logic", # TODO: how does the batch work: logic, button, imu, compass
31      }
32
```

Customize your Battlesnake

Stop your badgeapp (**REBOOP**) and start again from the menu.

On the second command prompt, run the game again (**arrow up**)

Your Battlesnake should update to reflect your new appearance!

Copy your snake

On the badge, open the MyBattlesnake.py file in Thonny

Copy part of your code:

Line 23-31 are similar, but we have some extra lines. Setup of copy the similar lines and fill the others with your setup.

Start with “Badgemode”: “logic” (line 30)

From line 51 and further, copy your created code from TODO 1 /2 / 3

Play your badge game

battlesnake play -W 11 -H 11 -g solo --browser -t 5000 --name "Python Starter Project" --url http://192.168.1.234:8001

Or

battlesnake play -W 11 -H 11 -g default --browser -t 5000 --name "MySnake" --url http://192.168.1.234:8001 --name "Always Right" --url https://be1bb639-bdd8-495e-8d65-93530d7411be-00-3sfnu1z8jdvu3.spock.replit.dev/ --name "Loopings" --url https://8aa6f1b0-b53c-44cb-a8fa-99298e66f0e5-00-2vmna0clhbo6u.spock.replit.dev/

Badge fun!

The badge has buttons and a joystick. But also tilt movement (imu/accelerometer) and rotation (compass) sensors. Can you use these in your logic?

core_app.game_badgemode	{ logic, button, imu, compass }
core_app.game_direction	{ up, down, left, right }

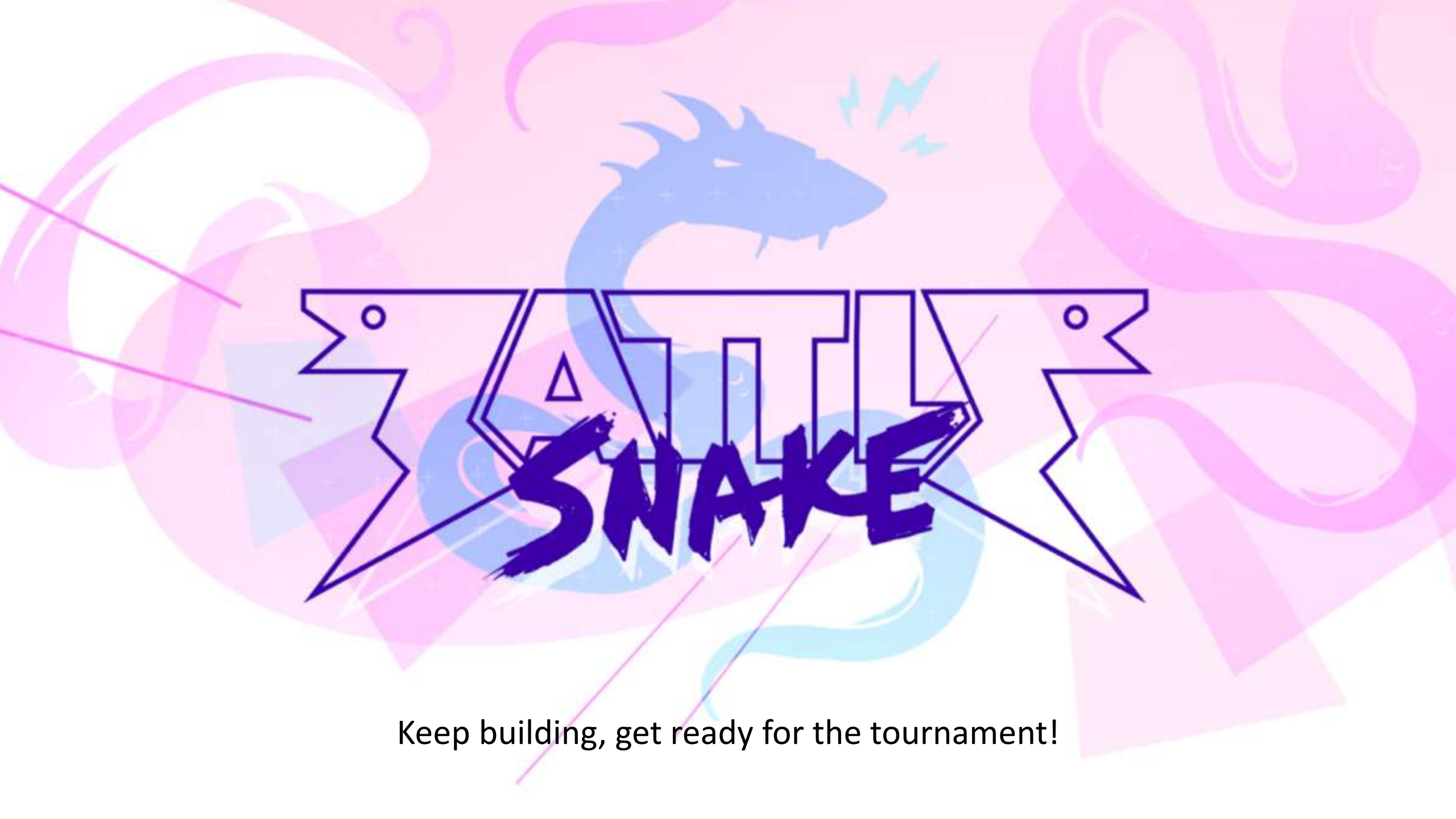
Badge fun!

Possible solution:

```
# Button mode, what did we press?
if core_app.game_badgemode == "button" and core_app.game_direction != "":
    if is_move_safe[core_app.game_direction] == True:
        print(f"MOVE {game_state['turn']}: {core_app.game_direction} is pressed!")
        return {"move": core_app.game_direction}
    else:
        print(f"Button {core_app.game_direction} is pressed, but move is not safe!")

# IMU mode, did we tilt in a good direction?
if core_app.game_badgemode == "imu" and core_app.game_direction != "":
    if is_move_safe[core_app.game_direction] == True:
        print(f"MOVE {game_state['turn']}: Tilted {core_app.game_direction}!")
        return {"move": core_app.game_direction}
    else:
        print(f"We are tilted {core_app.game_direction}, but move is not safe!")

# Compass mode, did we turn in a good direction?
if core_app.game_badgemode == "imu" and core_app.game_direction != "":
    if is_move_safe[core_app.game_direction] == True:
        print(f"MOVE {game_state['turn']}: Turned {core_app.game_direction}!")
        return {"move": core_app.game_direction}
    else:
        print(f"We are turned {core_app.game_direction}, but move is not safe!")
```



Keep building, get ready for the tournament!

EMF2026 – Battlesnake workshop