NYP LIT

Python Games Workshop Day 2

By Planning Committee

27th Dec 2022

All sides made by Javerine and Amber

Pre Workshop Preparations

Make sure you have these prepared before the workshop:

1. Python and PyCharm/Vscode
2. Flask installed in Python environment
    a. Install flask using ‘pip install flask’ in your terminal

Note! You may refer to the PDF if you are unsure how to install these or reach out to our LIT members for help via Discord

Agenda

Tic-Tac-Toe

Hunt the Wumpus

Tic Tac Toe

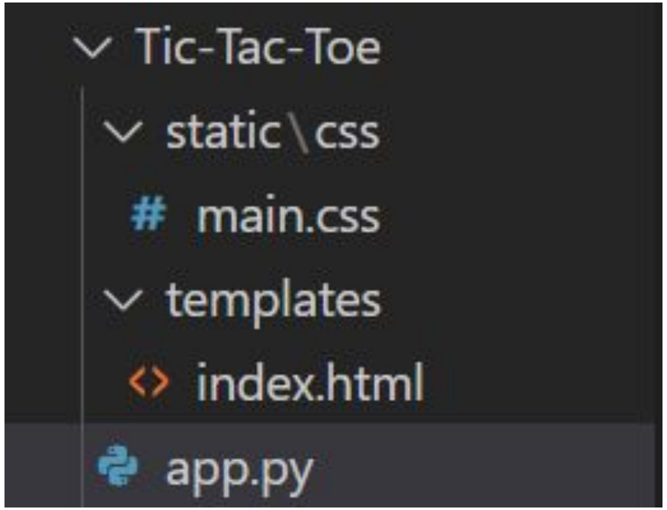
Back to Agenda Page

Tic Tac Toe

- Players take turns to mark their spaces with noughts (o) or crosses (x) on a 3 by 3 grid
- The player who place 3 of his mark in a horizontal,vertical or diagonal manner will win the game.

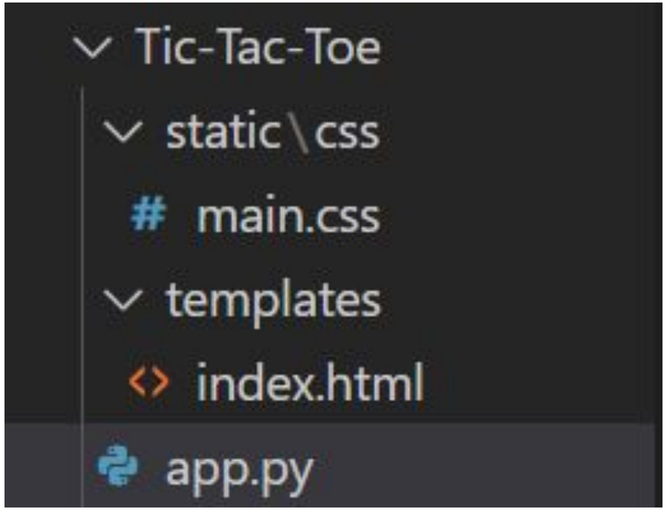
Files and Folders required

- static
  - Contains css and javascript files etc
- templates
  - Contains HTML files
  - To structure the page and take in Flask variables
- app.py
  - To run the page and ping the port



Files and Folders required

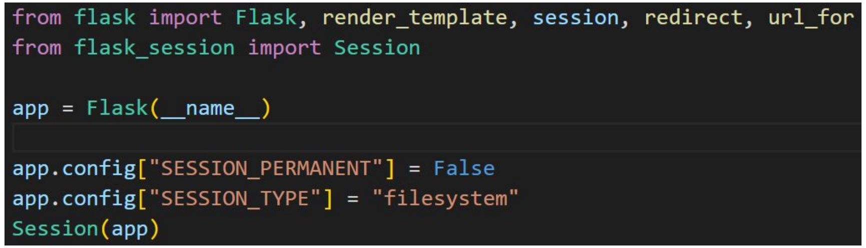
- Create a new project folder and add the following files and folders
- static
  - css
    - main.css
- templates
  - index.html
- app.py



```
Tic-Tac-Toe
  static\css
    main.css
  templates
    index.html
  app.py
```

Setting the flask application

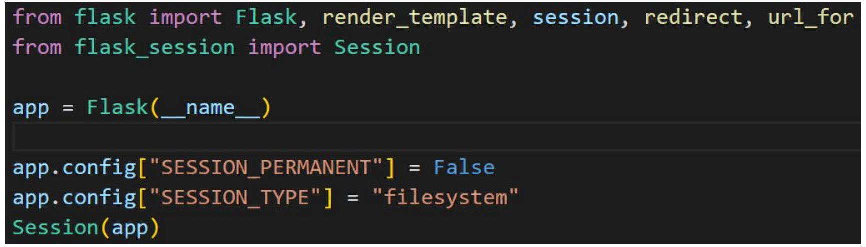
- **app.py**
- Import flask and the following modules shown
  - Flask → framework to develop the application
  - render_template → to render a webpage
  - redirect → to redirect user to the same/different page
  - url_for → generate url to an endpoint using a route
  - Session → store user information across different requests as they interact with the application
  - Flask_session → Session - to support server-side session in an application
- Create an app with flask (__name__)



```python
from flask import Flask, render_template, session, redirect, url_for
from flask_session import Session

app = Flask(__name__)

app.config["SESSION_PERMANENT"] = False
app.config["SESSION_TYPE"] = "filesystem"
Session(app)
```

Setting the flask application

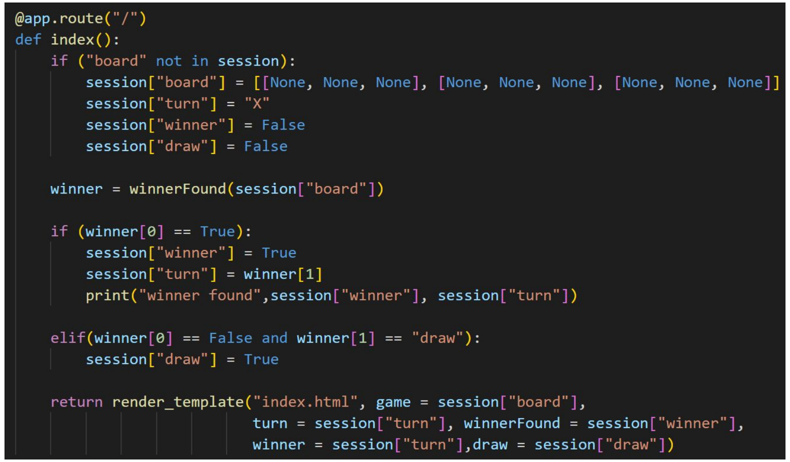
- **For app.config**
- Set session permanent to false to start a non-permanent session when running the app
- Set session type to filesystem to store session data in the flash-session directory when running the app



```
from flask import Flask, render_template, session, redirect, url_for
from flask_session import Session

app = Flask(__name__)

app.config["SESSION_PERMANENT"] = False
app.config["SESSION_TYPE"] = "filesystem"
Session(app)
```

Add Game Codes

- **<u>app.py</u>**
- Edit the index route and function
- Make sure board is empty and there is no winner and draw when board is not in session.
- If there is a winner, print the statement in terminal to show the winner.
- Render the index page and pass in session variables

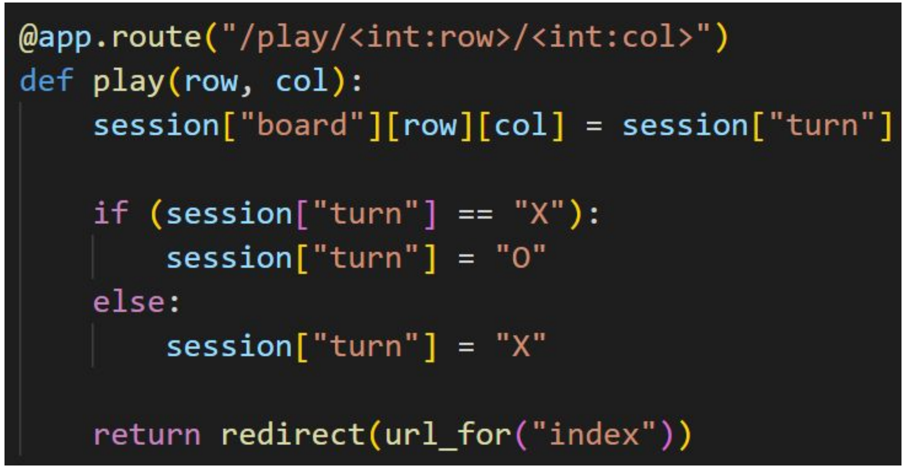


```python
@app.route("/")
def index():
    if ("board" not in session):
        session["board"] = [[None, None, None], [None, None, None], [None, None, None]]
        session["turn"] = "X"
        session["winner"] = False
        session["draw"] = False

    winner = winnerFound(session["board"])

    if (winner[0] == True):
        session["winner"] = True
        session["turn"] = winner[1]
        print("winner found",session["winner"], session["turn"])

    elif(winner[0] == False and winner[1] == "draw"):
        session["draw"] = True

    return render_template("index.html", game = session["board"],
                           turn = session["turn"], winnerFound = session["winner"],
                           winner = session["turn"],draw = session["draw"])
```

```
      |      |
[0][0]|[0][1]|[0][2]
------|------|------
      |      |
[1][0]|[1][1]|[1][2]
------|------|------
[2][0]|[2][1]|[2][2]
      |      |
```

Add Game Codes

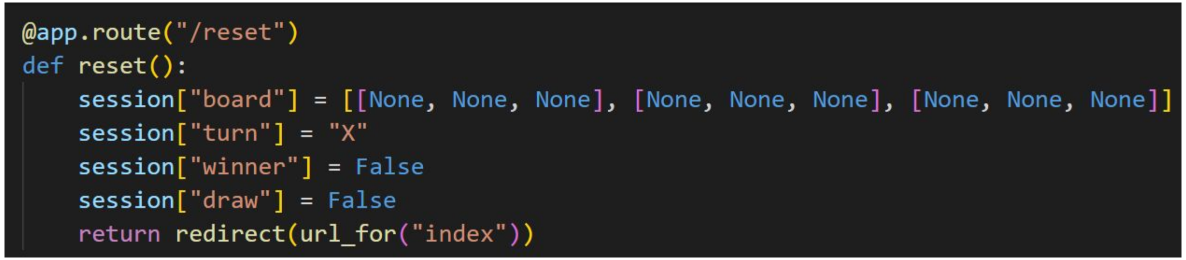
- **app.py**
- Edit the play route and function
- Depending on the player's turn and the position of the mark, the mark and the player turn will change accordingly
- Return the redirect of the index page



```
@app.route("/play/<int:row>/<int:col>")
def play(row, col):
    session["board"][row][col] = session["turn"]

    if (session["turn"] == "X"):
        session["turn"] = "O"
    else:
        session["turn"] = "X"

    return redirect(url_for("index"))
```

Add Game Codes

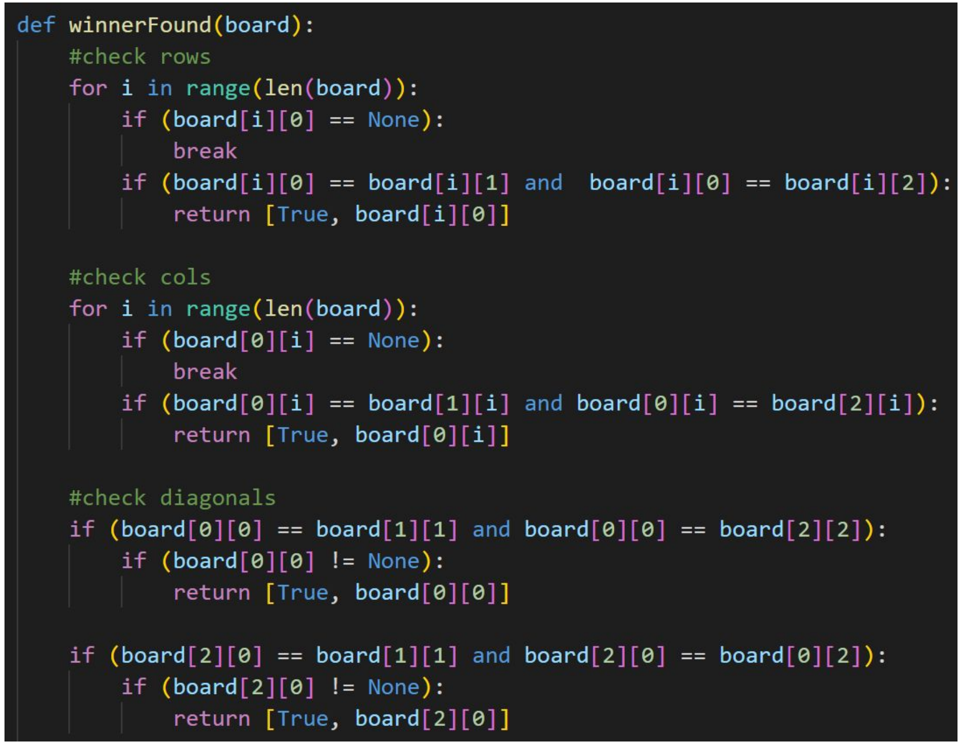
- **<u>app.py</u>**
- Add the reset route and function
- Return an empty board
- Set the mark to cross by default
- There is no winner and draw
- Return redirect of index page



```python
@app.route("/reset")
def reset():
    session["board"] = [[None, None, None], [None, None, None], [None, None, None]]
    session["turn"] = "X"
    session["winner"] = False
    session["draw"] = False
    return redirect(url_for("index"))
```

Add Game Codes

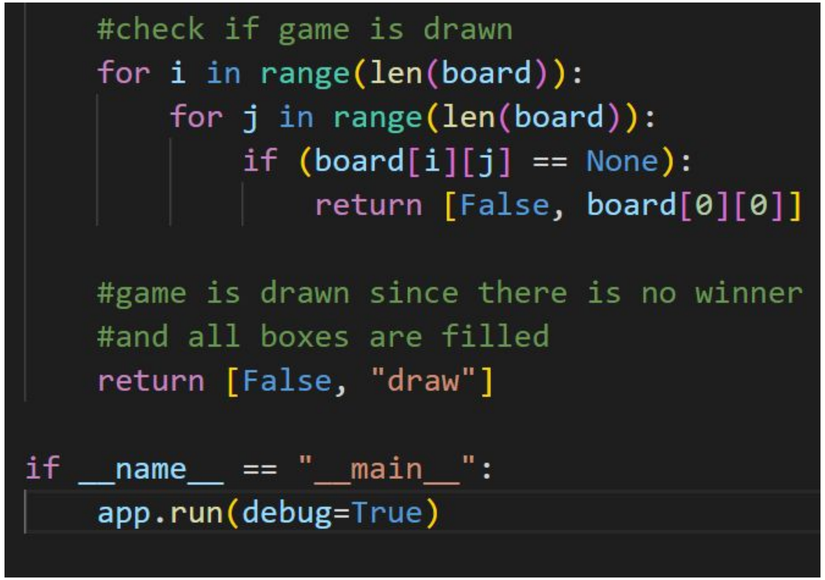
- **app.py**
- Edit the winner found function
- To check for winners and see if the winner wins based on horizontal,vertical or diagonal win
- This is done by comparing certain positions of the board for each type of win



```python
def winnerFound(board):
    #check rows
    for i in range(len(board)):
        if (board[i][0] == None):
            break
        if (board[i][0] == board[i][1] and  board[i][0] == board[i][2]):
            return [True, board[i][0]]

    #check cols
    for i in range(len(board)):
        if (board[0][i] == None):
            break
        if (board[0][i] == board[1][i] and board[0][i] == board[2][i]):
            return [True, board[0][i]]

    #check diagonals
    if (board[0][0] == board[1][1] and board[0][0] == board[2][2]):
        if (board[0][0] != None):
            return [True, board[0][0]]

    if (board[2][0] == board[1][1] and board[2][0] == board[0][2]):
        if (board[2][0] != None):
            return [True, board[2][0]]
```

Add Game Codes

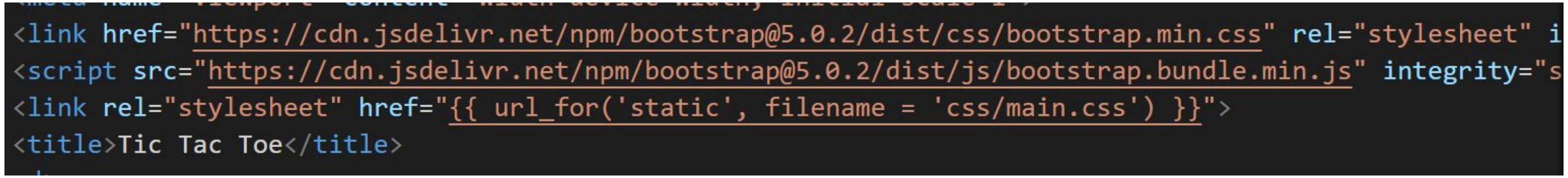
- **app.py**
- To check if there is a draw in the game
- Add the if statement and app.run(debug=True) in order for the app to run



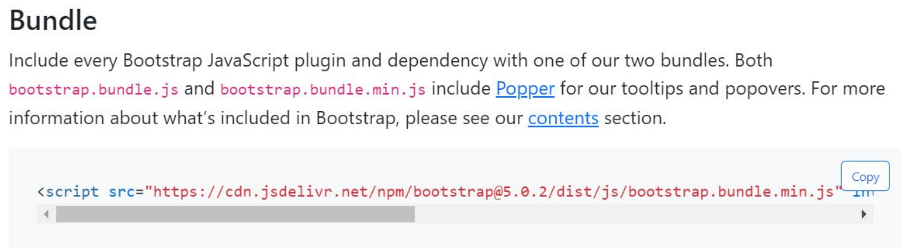
```
    #check if game is drawn
    for i in range(len(board)):
        for j in range(len(board)):
            if (board[i][j] == None):
                return [False, board[0][0]]

    #game is drawn since there is no winner
    #and all boxes are filled
    return [False, "draw"]

if __name__ == "__main__":
    app.run(debug=True)
```

Create the Webpage

- **index.html**
- Go to bootstrap 5 and copy and paste the cdn codes for css and js in the head section
- https://getbootstrap.com/docs/5.0/getting-started/introduction/
- Add link rel to link the main.css to the index.html using url_for



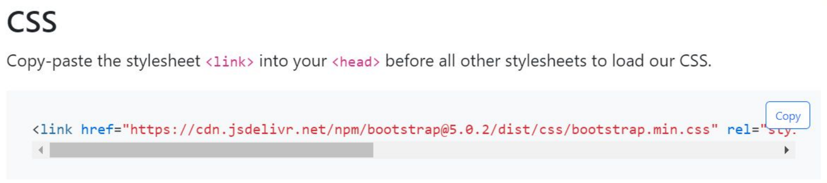
```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" i
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" integrity="s
<link rel="stylesheet" href="{{ url_for('static', filename = 'css/main.css') }}">
<title>Tic Tac Toe</title>
```



Bundle

Include every Bootstrap JavaScript plugin and dependency with one of our two bundles. Both `bootstrap.bundle.js` and `bootstrap.bundle.min.js` include Popper for our tooltips and popovers. For more information about what's included in Bootstrap, please see our contents section.

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js" in
```



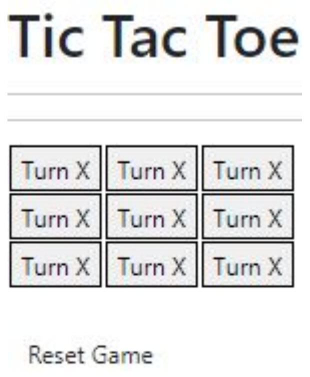
CSS

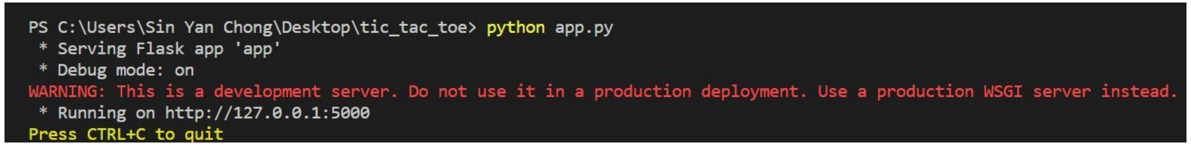
Copy-paste the stylesheet `<link>` into your `<head>` before all other stylesheets to load our CSS.

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="sty
```

Create the Webpage

- **index.html**
- Add the html codes to create the basic webpage of the tic-tac-toc game
- In your terminal, type python app.py and click on the local host link to open and view webpage.

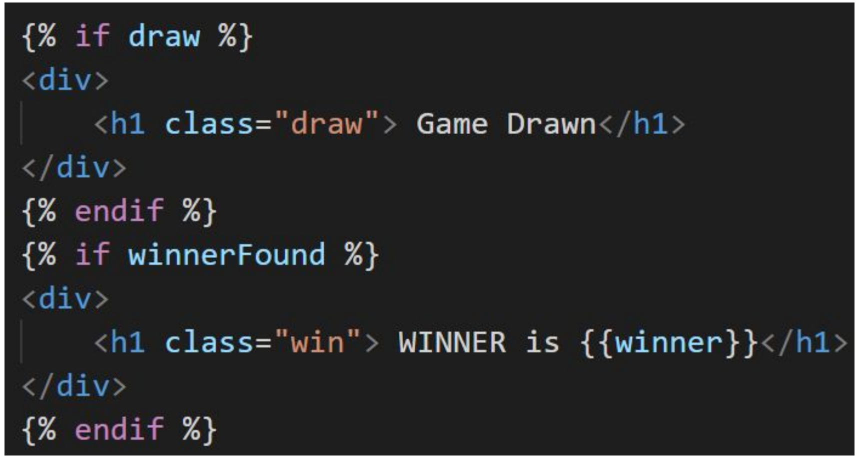




```
PS C:\Users\Sin Yan Chong\Desktop\tic_tac_toe> python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

Create the Webpage

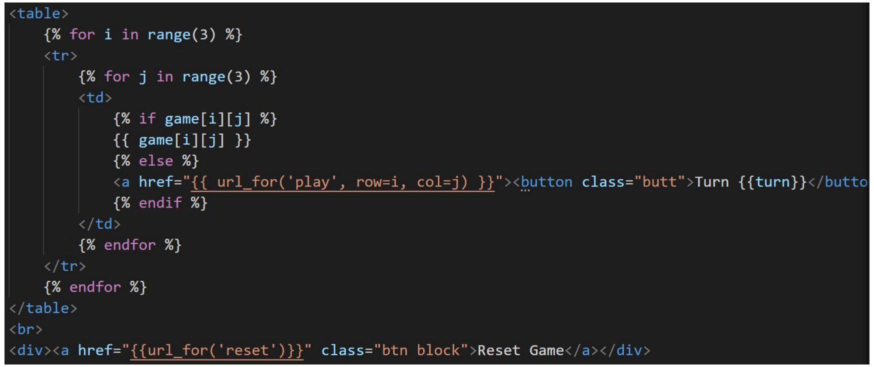
- **index.html**
- Add the the jinja tags and codes to display the respective heading based on whether the game has a winner or a draw
- If there is a winner, the webpage will display the winner name which was pass in earlier on in the webpage.



```
{% if draw %}
<div>
    <h1 class="draw"> Game Drawn</h1>
</div>
{% endif %}
{% if winnerFound %}
<div>
    <h1 class="win"> WINNER is {{winner}}</h1>
</div>
{% endif %}
```

Create the Webpage

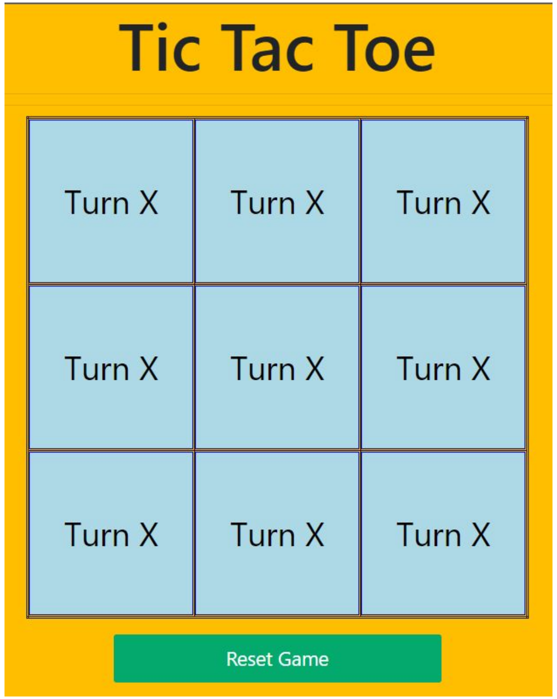
- **<u>index.html</u>**
- Add the the jinja tags and codes to display the table with the mark depending on the player
- Add a reset button to reset the game using url_for to link to reset route from app.py



```
<table>
    {% for i in range(3) %}
    <tr>
        {% for j in range(3) %}
        <td>
            {% if game[i][j] %}
            {{ game[i][j] }}
            {% else %}
            <a href="{{ url_for('play', row=i, col=j) }}"><button class="butt">Turn {{turn}}</butto
            {% endif %}
        </td>
        {% endfor %}
    </tr>
    {% endfor %}
</table>
<br>
<div><a href="{{url_for('reset')}}" class="btn block">Reset Game</a></div>
```

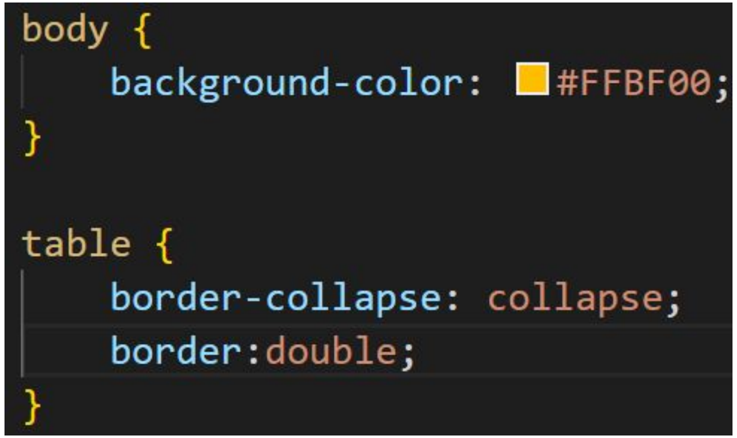
Style the Webpage

- **main.css**
- Add the css codes to style the basic webpage of the tic-tac-toc game



Style the Webpage

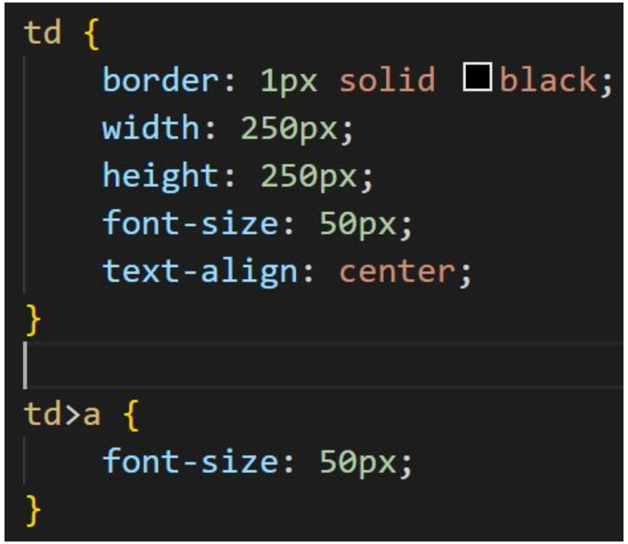
- **main.css**
- In body tag, add background-colour to change to a yellow background (#FFBF00)
- In table tag, add border-collapse to collapse the table into a single border,
- and add border:double to create 2 borders



```css
body {
    background-color: #FFBF00;
}

table {
    border-collapse: collapse;
    border:double;
}
```

Style the Webpage

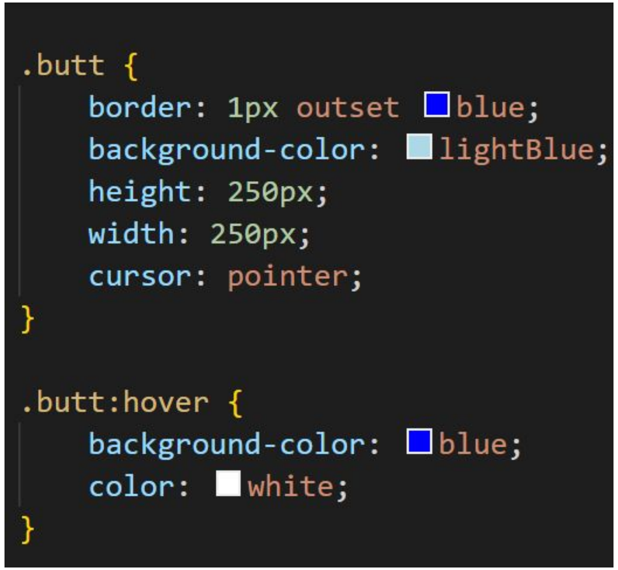
- **main.css**
- In td tag, add border to create a solid black border of 1px,
- add width and height of 250px,
- add font-size to increase table cell text to 50px
- and add text-align to center table cell text
- In td>a tag, add font-size to increase table cell a tag text to 50px



```
td {
    border: 1px solid ■black;
    width: 250px;
    height: 250px;
    font-size: 50px;
    text-align: center;
}

td>a {
    font-size: 50px;
}
```

Style the Webpage

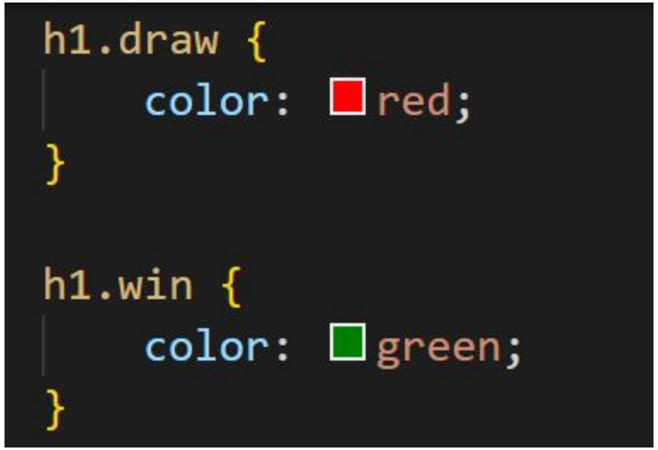
- **<u>main.css</u>**
- In .butt tag,
- add border to create a outset blue border of 1px,
- add background-color to change to lightBlue,
- add width and height of 250px,
- add cursor to change to pointer when user hover the table cell
- In .butt:hover tag, add background-color and color to change the background colour to blue and text colour to white when user hover the table cell



```
.butt {
    border: 1px outset blue;
    background-color: lightBlue;
    height: 250px;
    width: 250px;
    cursor: pointer;
}

.butt:hover {
    background-color: blue;
    color: white;
}
```

Style the Webpage

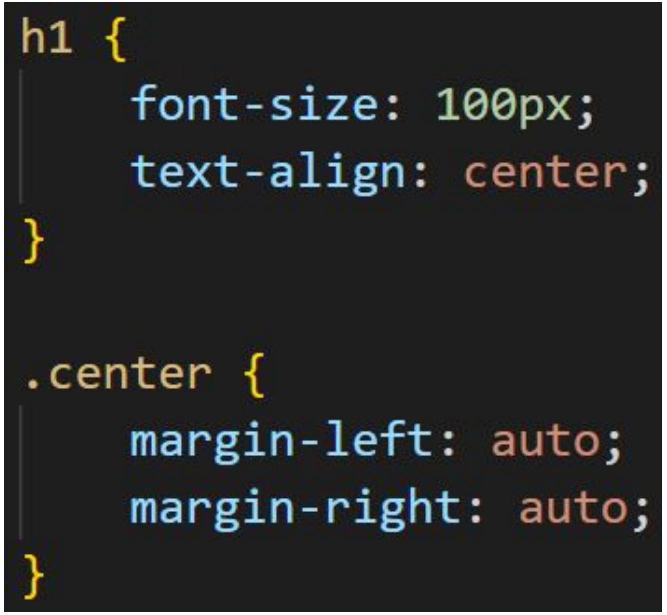
- **main.css**
- In h1.draw tag, add colour to change the draw heading text to red
- In h1.win tag, add colour to change the winner heading to green



```
h1.draw {
    color: 🟥red;
}

h1.win {
    color: 🟩green;
}
```

Style the Webpage

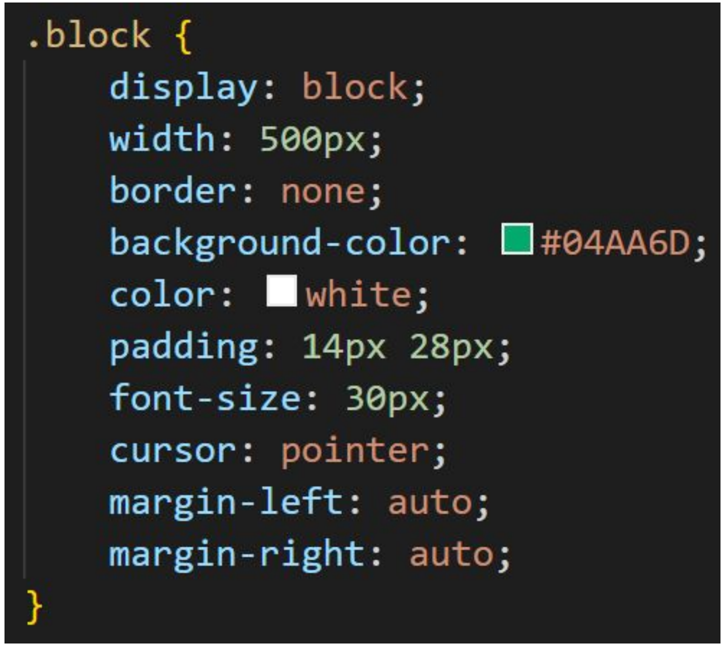
- **<u>main.css</u>**
- In h1 tag, add font-size to increase all h1 headings to 100px,
- add text-align to align all h1 headings to center
- In .center tag, add margin-left and margin-right auto to center the table



```css
h1 {
    font-size: 100px;
    text-align: center;
}

.center {
    margin-left: auto;
    margin-right: auto;
}
```

Style the Webpage

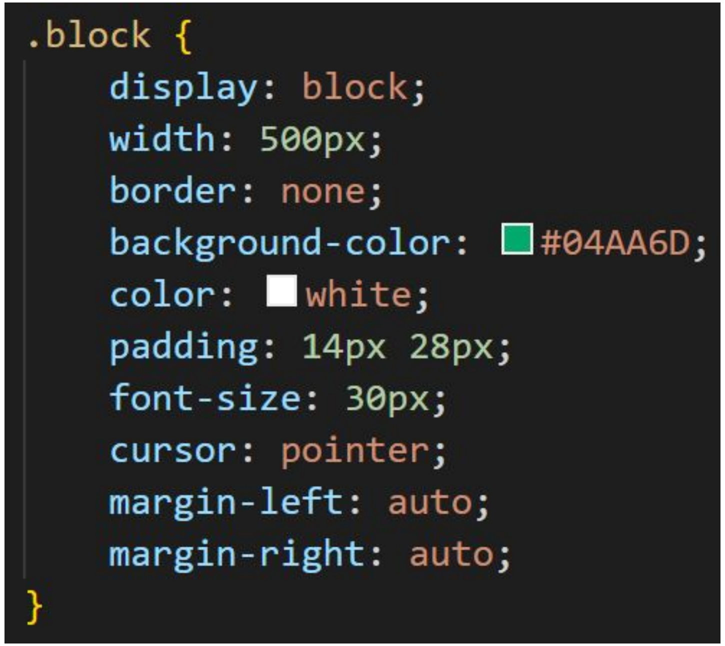
- **<u>main.css</u>**
- In .block tag, add display to display as a block element,
- add width of 500px,
- add border:none to create no border,
- add background-color to change to green (#04AA6D),
- add color to change text to white,
- add padding to add top and bottom padding of 14px and left and right padding of 28px.



```
.block {
    display: block;
    width: 500px;
    border: none;
    background-color: 🟩#04AA6D;
    color: ⬜white;
    padding: 14px 28px;
    font-size: 30px;
    cursor: pointer;
    margin-left: auto;
    margin-right: auto;
}
```

Style the Webpage

- **main.css**
- In .block tag, add font-size to increase text to 30px,
- add cursor to pointer when user hover over the button,
- add margin-left and margin right auto to center the button



```css
.block {
    display: block;
    width: 500px;
    border: none;
    background-color: #04AA6D;
    color: white;
    padding: 14px 28px;
    font-size: 30px;
    cursor: pointer;
    margin-left: auto;
    margin-right: auto;
}
```

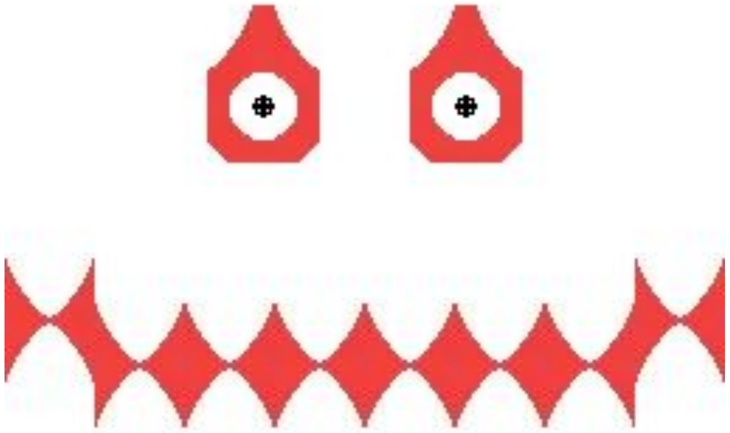
Break

Back to Agenda Page

Hunt the Wumpus

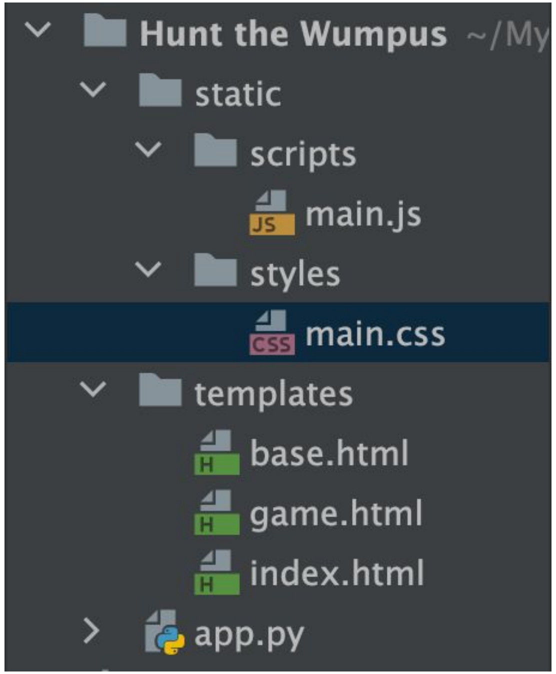
Hunt the Wumpus

- In Hunt the Wumpus, the player is an adventurer entering a dungeon seeking treasure.
- To win, you need to find the treasure
- You lose if you fall into a pit or are eaten by Wumpus, a man eating monster



Files and Folders required

- Static
  - Contains css and javascript files
  - For styling the webpage
  - In here, images can also be stored
- Templates
  - Contains HTML files
  - To structure the page and take in Flask variables
- app.py
  - To run the page and ping the port



```
Hunt the Wumpus ~/My
  static
    scripts
      main.js
    styles
      main.css
  templates
    base.html
    game.html
    index.html
  app.py
```

Setting the flask application

app.py

- Import flask and the following modules shown
- Flask(__name__) creates an application object
- render_template() renders the html file and passed to it, with the kwargs after it passed to jinja tags in the file, which are processed during the rendering
- request is used to access incoming request data such as forms
- Add app.run with debug set as True to run app.py, so any changes is immediately reflected in the app

```
from flask import Flask, render_template,
request

app = Flask(__name__)


@app.route("/")
def index():
    return render_template('index.html')


if __name__ == "__main__":
   app.run(debug=True)
```

Set up base.html template

In <head>

1. Add link rel to link the main.css to the index.html using url_for
2. Copy cdn link for JQuery (for easier DOM manipulation)
3. Copy cdn link for feather (svg icons)

```
<link rel= "stylesheet" type= "text/css" href="{{url_for('static',filename='styles/main.css')}}">
<!--JQuery-->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.1/jquery.min.js"></script>
<!--Icons with js-->
<script src="https://cdn.jsdelivr.net/npm/feather-icons/dist/feather.min.js"></script>
```

Set up base.html template

In <body>:

1. Create a header that links to the root
2. Add jinja block tags to use index.html and game.html as extensions

```
<header>
    <a href='/'><h2>Hunt the Wumpus</h2></a>
</header>
<main>
     {% block content %}
     {% endblock %}
</main>
```

Set up index.html

1. Add jinja tags to extend base.html
2. In block content, create a form so that players can chose size of dungeon map
   a. Use post method to send data back to app.py

```
{% extends 'base.html' %}
{% block content %}

<div class='setup'>
    <h1>Select the size of the dungeon</h1>
    <form method='POST'>
        <input type="number" id="height" class="size_input"
name="height" min="5" max="10" value="5"/>
        <large>X</large>
        <input type="number" id="width" class="size_input"
name="width" min="5" max="10" value="5"/>
        <button class='play' type="submit">Enter the
Dungeon</button>
    </form>
</div>

{% endblock %}
```

Handling form data in app.py

- uuid4().hex() generates random 32-character lowercase hexadecimal string
- It is then passed to app.config[‘SECRET_KEY’] to be used to secure forms
- In decorator pass [‘GET, ’‘POST’] to methods parameter to allow the page to respond to post requests
- Access form data from request.form dictionary, with <input> attribute name as keys
- Check if request.method == 'POST': for post requests from the form
- Render game.html and pass dungeon width and height to it as keyword arguments

```python
from uuid import uuid4

app = Flask(__name__)
app.config['SECRET_KEY'] = uuid4().hex()

@app.route("/", methods=['GET', POST'])
def index():
    if request.method == 'POST':
        return
render_template('game.html',
width=int(request.form['width']),
height=int(request.form['height']))
    return render_template('index.html')
```

Set up game.html

1. Add jinja tags to extend base.html
2. In <script> assign variables with the width and height data passed with jinja
3. Link main.js using url_for()

```
{% extends 'base.html' %}
{% block content %}
<script>
    var w = {{width}}, h = {{height}};
</script>
<script src="{{ url_for('static',
filename='scripts/main.js') }}"></script>
```

Create Dungeon

- Use jinja tags to create a nested for loop to create <div>s that represent dungeon rooms based on width and height
    - id will be later used to select the <div> s in main.js
- .current room represent the room the player is in

```
<div class="game">
{% for i in range(height) %}
      <div class="line">
      {% for j in range(width) %}
        {% if i == 0 and j == 0 %}
            <div class="room
curr_room" id="{{j}}-{{i}}"></div>
        {% else %}
            <div class="room"
x="{{j}}-{{i}}"></div>
        {% endif %}
    {% endfor %}
    </div>
{% endfor %}
</div>
```

Dungeon CSS

- Include display: inline-block; so that rooms will appear in a row with width and height properties respected

```
.room {
    width: 30px;
    height: 30px;
    border: 1px dotted;
    margin: 3px;
    display: inline-block;
}

.curr_room {
    background: yellow;
    border: 2px solid;
    width: 28px;
    height: 28px;
 }
```

Arrow Button Controls

- Use <i> for icons
- feather.replace() replaces tags with data-feather attribute with the corresponding svg icon
- text-align; center; centers the up arrow button

```
.controls {
    display: inline-block;
    text-align: center;
}
```

```
<div class="controls">
    <button id="up"><i
data-feather="arrow-up"></i></button>
    <div>
        <button id="left"><i
data-feather="arrow-left"></i></button>
        <button id="down"><i
data-feather="arrow-down"></i></button>
        <button id="right"><i
data-feather="arrow-right"></i></button>
    </div>
</div>

<script>
    feather.replace()
</script>
```

Win Lose Messages

- display: none; hides the messages
- Create play again buttons that link back to the root (index.html)

```css
/* main.css */
.win, .lose {
    display: none;
}
```

```html
<!--game.html-->
<div class="win">
    <h4>Congratulations! You won!</h4>
    <a href='/'><button class='play'>Play
 again</button></a>
</div>

<div class="lose">
    <h4>Game Over!</h4>
    <a href='/'><button class='play'>Play
 again</button></a>
</div>
```

Game Code

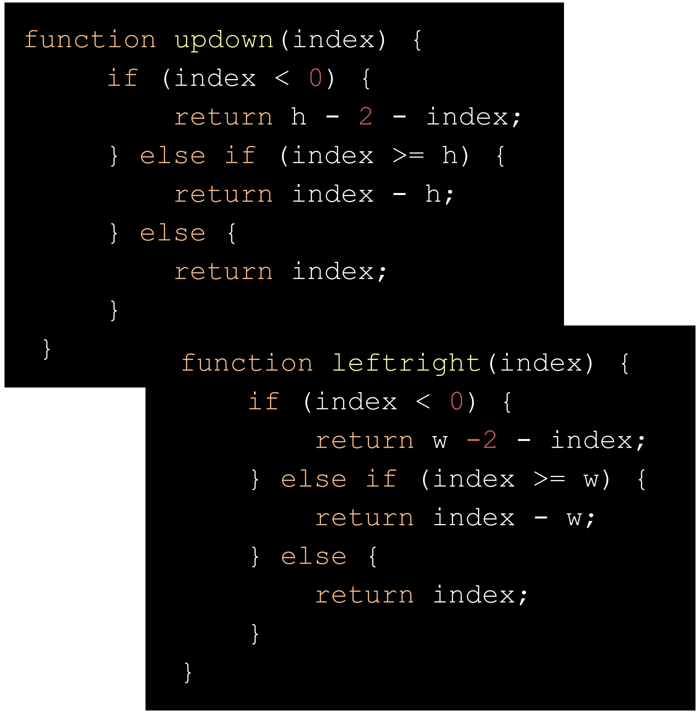
In main.js:

- Using nested for loops, create a nested array of null representing an empty dungeon (cave)
- Assign ‘Player to cave[0][0] representing the player
- Create a variable to track the coordinates of the player
- Assign a boolean representing whether a game is ongoing to a variable

```
var cave = [];
for (let i = 0, line ; i < h ; i++,
cave.push(line)) {
    line = [];
    for (let j = 0; (j < w); j++) {
        line.push(null);
    }
}
console.log(cave);
var coordinates = [0, 0];
cave[0][0] = 'Player';
playing = true;
```

Wrap the edges of the map

- Create 2 functions that processes indexes for cave
  - when the index is out of range, it starts counting elements from the front again
  - when it is negative, it counts from the back
  - Should return



```
function updown(index) {
     if (index < 0) {
         return h - 2 - index;
     } else if (index >= h) {
         return index - h;
     } else {
         return index;
     }
 }
```

```
function leftright(index) {
     if (index < 0) {
         return w -2 - index;
     } else if (index >= w) {
         return index - w;
     } else {
         return index;
     }
 }
```

Game functions

- Create a function that returns a list of the contents of nearby rooms
- Create a function that generates a random integer between 0 and the parameter
- Math.random generates a random number between 0 and 1
- Math.floor rounds down the number passed to it

```
function adjacent(x, y) {
    return [cave[updown(y-1)][x],           //above
            cave[updown(y+1)][x],           //below
            cave[y][leftright(x-1)],        //left
            cave[y][leftright(x+1)]];       //right
}
```

```
function randint(max) {         //excludes max
    return Math.floor(Math.random() * max);
}
```

Place Pits, Wumpus, and Treasure

- Create a function places items in a random empty room that is not next to the player using the previous 2 functions
- Call the function to place 2 ‘Pits’, a ‘Wumpus’, and a treasure (‘Gold’)

```
function placing(item) {
      var x, y
      do {
          x = randint(w);
          y = randint(h);
          console.log(x, y, item);
     } while (adjacent(x, y).includes("Player")
|| cave[y][x] !== null);
     cave[y][x] = item;
}
```

```
// randomise location
placing('Pit');
placing('Pit');    //2 pits
placing('Wumpus');
placing('Gold');
console.log(cave);
```

Moving

Create a function to be called when the player moves.

- It should have a parameter for direction of movement
- Coordinates should be adjusted accordingly
- $() selects DOM elements using css selectors
- .empty() removes any content in the <div> representing the room (in this case, any svg icons)

```
function move(direction) {
    if (direction === 'left') {
        coordinates[0] = leftright(coordinates[0] - 1);
    } else if (direction === 'up') {
        coordinates[1] = updown(coordinates[1] - 1);
    } else if (direction === 'down') {
        coordinates[1] = updown(coordinates[1] + 1);
    } else if (direction === 'right') {
        coordinates[0] = leftright(coordinates[0] + 1);
    }
    console.log(coordinates);
    let location = cave[coordinates[1]][coordinates[0]];

    //css
    $(".curr_room").removeClass("curr_room");
    var room = $("#"+ coordinates[0] + "-" +
coordinates[1]);
    room.empty();
    room.addClass("curr_room");
```

Moving

Continuing the earlier function:

- Check if the room the player is in has something
    - Icon is added and the end function is called to end the game
- .append() adds the string passed as content in the current room <div>
- Check if adjacent rooms have something and add icon warnings accordingly
- feather.replace() replaces tags with the data feather attribute with the corresponding svg icon

```
  var adj = adjacent(coordinates[0], coordinates[1])
  //win lose and adjacency check
  if (location === 'Pit') {
    room.append("<i data-feather='x-circle'></i>");
    end('lose');
  } else if (location === 'Wumpus') {
    room.append("<i data-feather='frown'></i>");
    end('lose');
  } else if (location === 'Gold') {
    room.append("<i data-feather='award'></i>");
    end('win');
  } else if (adj.includes('Pit')) {
    room.append("<i data-feather='wind'></i>");
  } else if (adj.includes('Wumpus')) {
    room.append("<i
data-feather='alert-triangle'></i>");
  } else if (adj.includes('Gold')) {
    room.append("<i data-feather='star'></i>");
  }
  feather.replace();
}
```

End game function

Create a function for ending the game with a parameter for the game result

- If win, use $(.win) to select the win message and display it
- .show(“fast”) removes display: none and reveals the element quickly
- Set the game state variable to false

```
function end(result) {
//result = 'win' or 'lose'
    $('.' + result).show("fast")
    playing = false
}
```

Detect clicking

- When the page is loaded, the anonymous function passed into $(document).ready() is called
- When clicked, the buttons #left will call the anonymous function passed to $('#left').click() which will call move(‘left’)
- $(document).keydown() calls the anonymous function passed to it when a keyboard key is pressed.
    - The pressed key code is passed to the anonymous function as key.keycode

```
$(document).ready(function(){
    //html arrow controller
    $('#left').click(() => {move('left');})
    $('#right').click(() => {move('right');});
    $('#up').click(() => {move('up');});
    $('#down').click(() => {move('down');});
    // detecting arrow keys
    $(document).keydown(function(key) {
        if (key.keyCode == 37) {         //left
            move('left');
        } else if (key.keyCode == 38) { //up
            move('up');
        } else if (key.keyCode == 39) { //right
            move('right');
        } else if (key.keyCode == 40) { //down
            move('down');
        }
    })
})
```

Please help us fill
In this feedback form here:
https://forms.gle/4ruYgmvrBxpcayj6A