Benjamin Matich & Nathan Rao

Colleen van Lent

Data-Oriented Programming

22 April 2025

SI 206 Final Project Report

https://github.com/nwrao/Big-Brass/tree/main

INITIAL GOALS

Our Main goal for this project was to research two different video game APIs and to notice the trends in damage types across weapons types and damage types.

To achieve this goal, we selected the Elden Ring and Monster Hunter Wilds APIs. Both of these games contain an extensive amount of weapons with differing damage types. Through this goal we had subgoals within each video game. For the Elden Ring Api, there were the goals of finding out how many weapons belonged to a certain type of damage and what the average damage for each different damage type. For the Monster Hunter Wilds API, there were the goals of finding out how many weapons of each type there were, and the average damage for each weapon type. We wanted to compare and contrast these averages and distributions between games.

ACHIEVED GOALS

We were able to find out the main goal of the trends in weapons damages amongst the different weapon types and damage types and found that Weapons that were longer had the stronger damage and the weapons that were physical had the greater damage in Monster Hunter Wilds and Elden Ring respectively. The Sword and Shield was found to be the most common weapon type in Monster Hunter Wilds, while the Physical weapon was the most common in Elden Ring.

PROBLEMS FACED

The main problems we faced was learning how to limit the data to 25 per submission. We both figured out ways to go around this by making sure that the count of data was limited to 25 by seeing the number of data in the database and making sure only 25 was added at a time.

Another problem we faced was understanding how to join the databases in the calculations, but we both realized ways that could be achieved in both projects. The Elden Ring api connected all three tables to get the average damage and the damage name while the Monster Hunter Wilds connected mhw_weapons and mhw_weapon_types, to achieve this goal.

CALCULATIONS

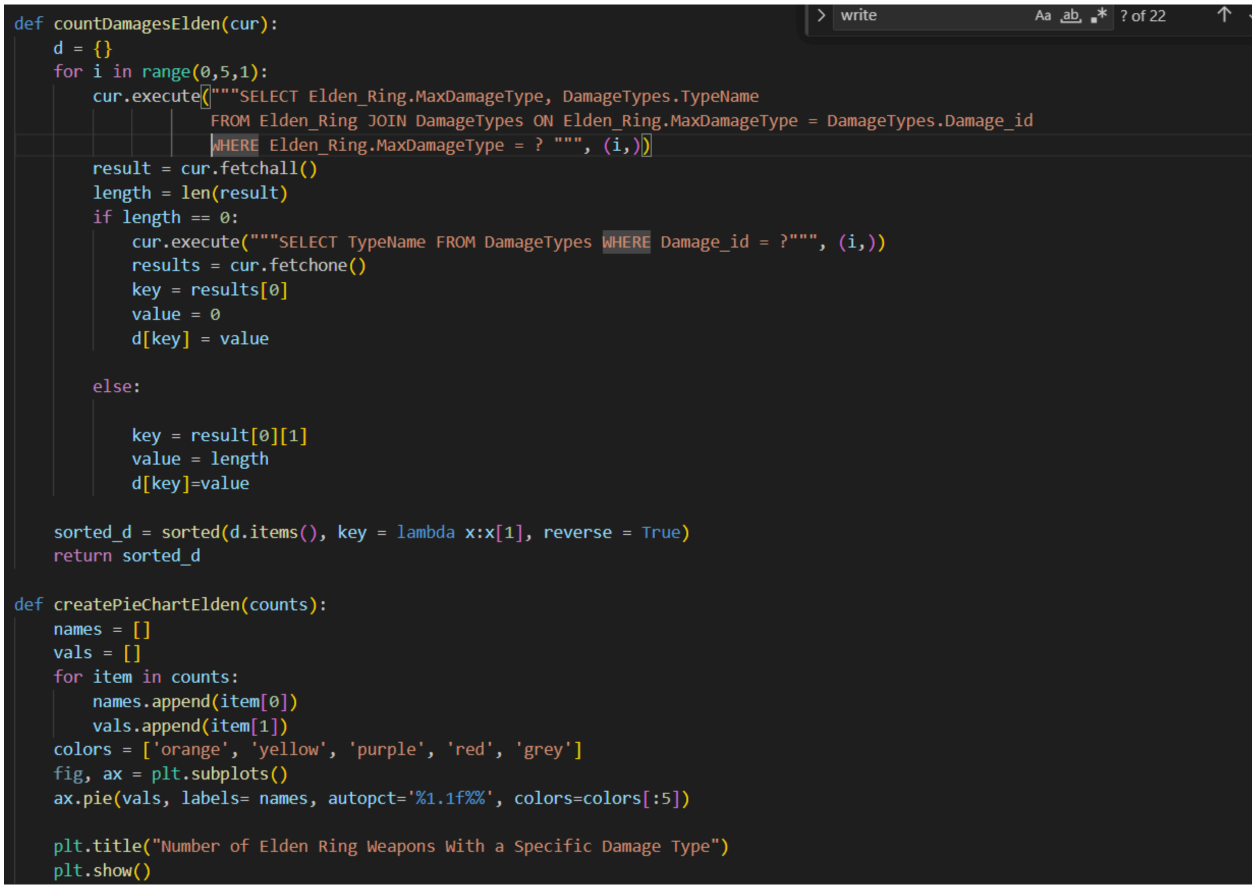
The calculations we created are based upon the Goals and subgoals of the APIs

1. Number of weapons in a Damage Type in Elden Ring
2. Average Damage per damage type in Elden Ring
3. Number of weapons in a Weapon Type in Monster Hunter Wilds.
4. Average Attack Power per weapon type in Monster Hunter Wilds
5. Average damages per weapon types across both APIs

Number of Weapons in a Damage Type Category in Elden Ring

Elden ring possesses a lot of different weapons in the game. Each weapon also deals different areas of damage like Physical or Fire damage. We wanted to calculate what damage type is the most common damage type from amongst the weapons.

Our Database is structured in the Elden_Ring table to have a foreign key pointing to the WeaponDifferentDamages, which contains all the damages of a weapon and the max damage of that weapon. This max damage was selected from the EldenRingData.py file by selecting which damage type the weapons dealt the most and was entered into the WeaponDifferentDamages table. There is another table connected via a foreign key to Elden_Ring, which is the Damage types table. Our process was to go through every damage type possibility and achieve the number of weapons with that damage type. This is all completed through the countsDamagesElden function and returns a list of tuples of the damage names and its number of weapons.

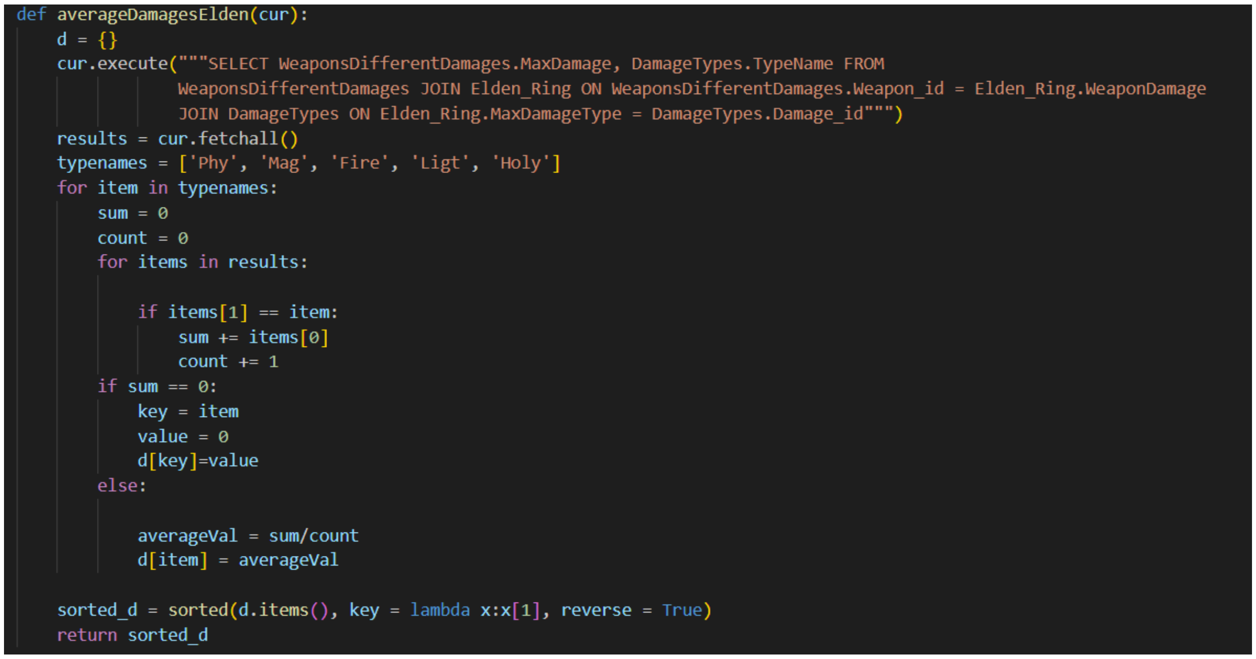


```
def countDamagesElden(cur):
    d = {}
    for i in range(0,5,1):
        cur.execute("""SELECT Elden_Ring.MaxDamageType, DamageTypes.TypeName
                    FROM Elden_Ring JOIN DamageTypes ON Elden_Ring.MaxDamageType = DamageTypes.Damage_id
                    WHERE Elden_Ring.MaxDamageType = ? """, (i,))
        result = cur.fetchall()
        length = len(result)
        if length == 0:
            cur.execute("""SELECT TypeName FROM DamageTypes WHERE Damage_id = ?""", (i,))
            results = cur.fetchone()
            key = results[0]
            value = 0
            d[key] = value

        else:

            key = result[0][1]
            value = length
            d[key]=value

    sorted_d = sorted(d.items(), key = lambda x:x[1], reverse = True)
    return sorted_d

def createPieChartElden(counts):
    names = []
    vals = []
    for item in counts:
        names.append(item[0])
        vals.append(item[1])
    colors = ['orange', 'yellow', 'purple', 'red', 'grey']
    fig, ax = plt.subplots()
    ax.pie(vals, labels= names, autopct='%1.1f%%', colors=colors[:5])

    plt.title("Number of Elden Ring Weapons With a Specific Damage Type")
    plt.show()
```

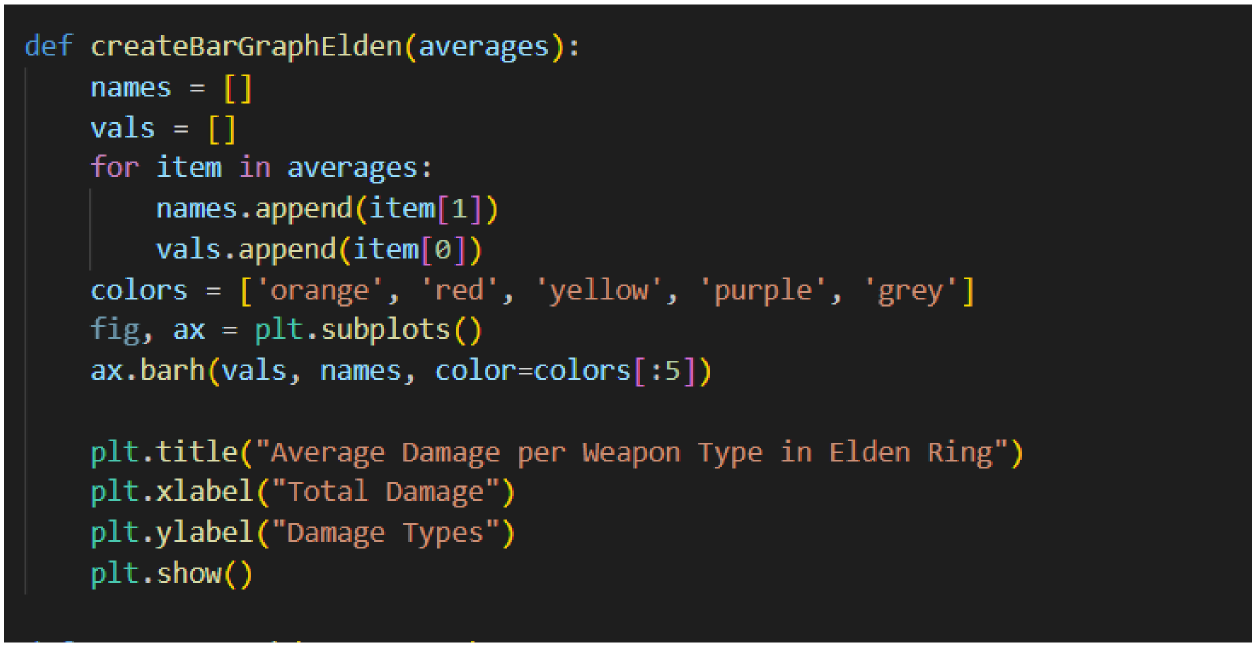
Average Damage per weapon type in Elden Ring

In addition to understanding which weapon type is the most common, it is important to see which weapon type produces the largest damage overall. This can be useful as a player of the game to know which weapon type is best to use in combat.

We accomplished this via accessing the join between the Elden_Ring table and the WeaponDifferentDamages and selecting the value of the damages. We also joined the DamageTypes table to receive the name label of the damages. This allowed us to calculate the average damage across all the weapon types.



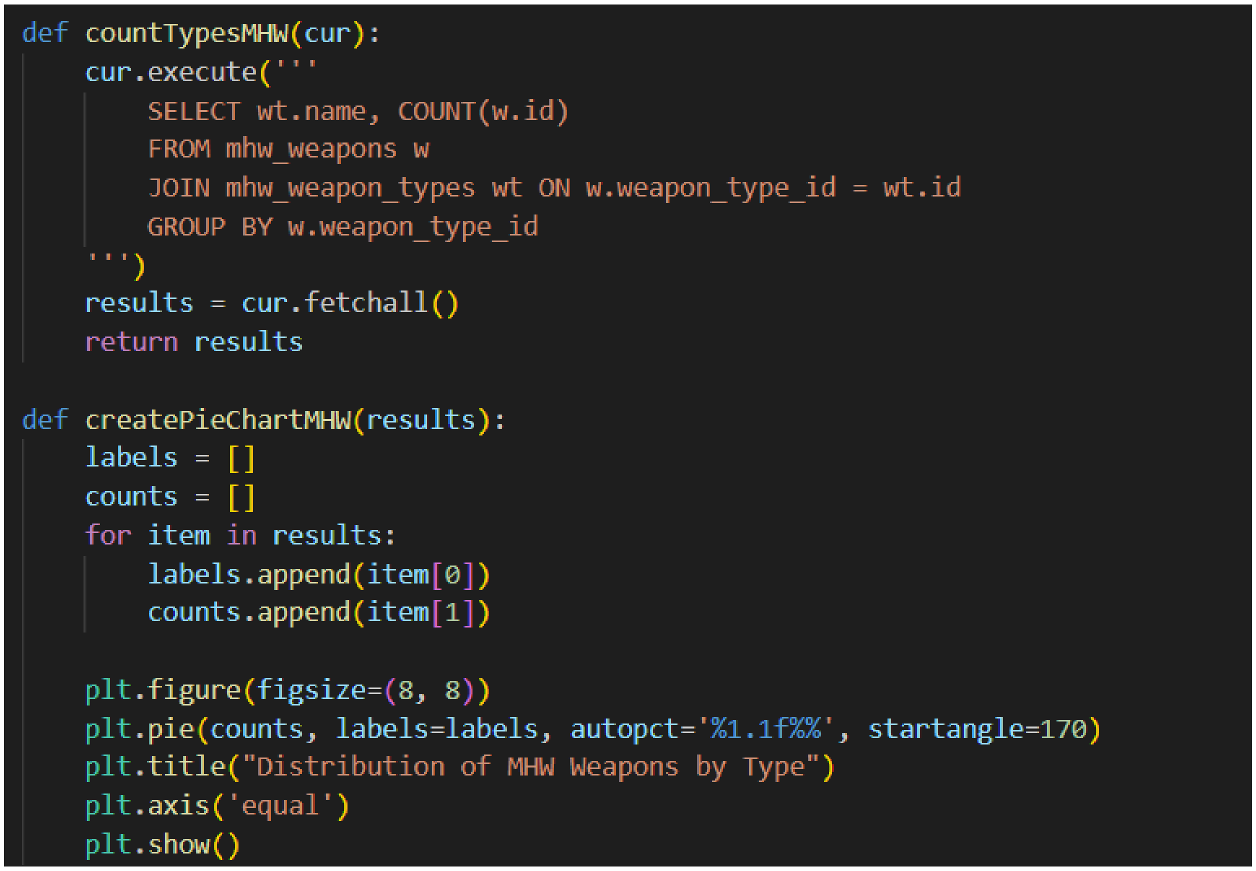
```
def averageDamagesElden(cur):
    d = {}
    cur.execute("""SELECT WeaponsDifferentDamages.MaxDamage, DamageTypes.TypeName FROM
                WeaponsDifferentDamages JOIN Elden_Ring ON WeaponsDifferentDamages.Weapon_id = Elden_Ring.WeaponDamage
                JOIN DamageTypes ON Elden_Ring.MaxDamageType = DamageTypes.Damage_id""")
    results = cur.fetchall()
    typenames = ['Phy', 'Mag', 'Fire', 'Ligt', 'Holy']
    for item in typenames:
        sum = 0
        count = 0
        for items in results:

            if items[1] == item:
                sum += items[0]
                count += 1
        if sum == 0:
            key = item
            value = 0
            d[key]=value
        else:

            averageVal = sum/count
            d[item] = averageVal

    sorted_d = sorted(d.items(), key = lambda x:x[1], reverse = True)
    return sorted_d
```



```
def createBarGraphElden(averages):
    names = []
    vals = []
    for item in averages:
        names.append(item[1])
        vals.append(item[0])
    colors = ['orange', 'red', 'yellow', 'purple', 'grey']
    fig, ax = plt.subplots()
    ax.barh(vals, names, color=colors[:5])

    plt.title("Average Damage per Weapon Type in Elden Ring")
    plt.xlabel("Total Damage")
    plt.ylabel("Damage Types")
    plt.show()
```

Percentage of weapons in a Weapon Type in Monster Hunter Wilds.

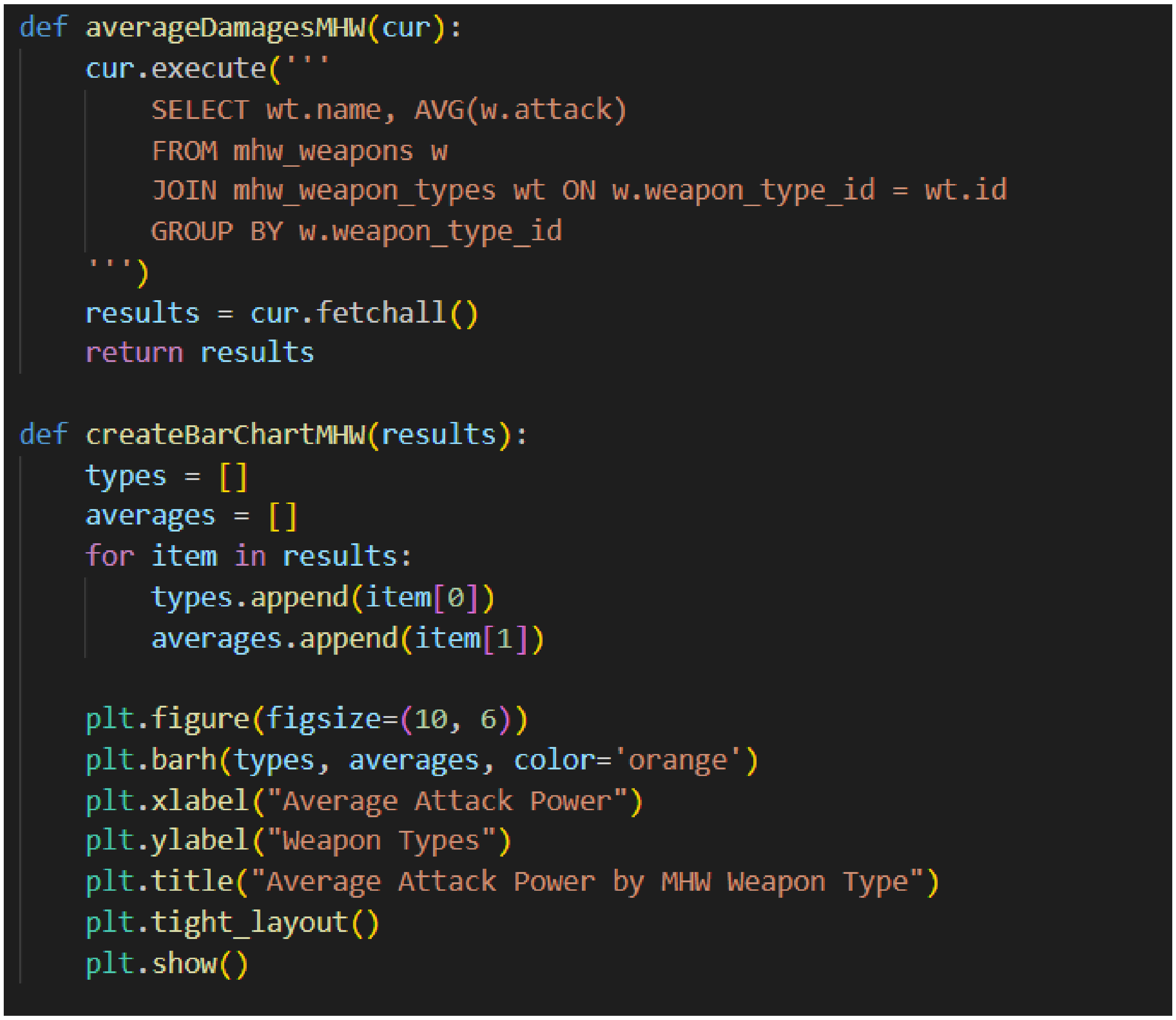
Monster Hunter Wilds possess a lot of different weapons in the game. Each weapon also deals different areas of damage like shields or longswords. We wanted to calculate what damage type is the most common damage type from amongst the weapons.



```python
def countTypesMHW(cur):
    cur.execute('''
        SELECT wt.name, COUNT(w.id)
        FROM mhw_weapons w
        JOIN mhw_weapon_types wt ON w.weapon_type_id = wt.id
        GROUP BY w.weapon_type_id
    ''')
    results = cur.fetchall()
    return results

def createPieChartMHW(results):
    labels = []
    counts = []
    for item in results:
        labels.append(item[0])
        counts.append(item[1])

    plt.figure(figsize=(8, 8))
    plt.pie(counts, labels=labels, autopct='%1.1f%%', startangle=170)
    plt.title("Distribution of MHW Weapons by Type")
    plt.axis('equal')
    plt.show()
```

Average Attack Power per weapon type in Monster Hunter Wilds

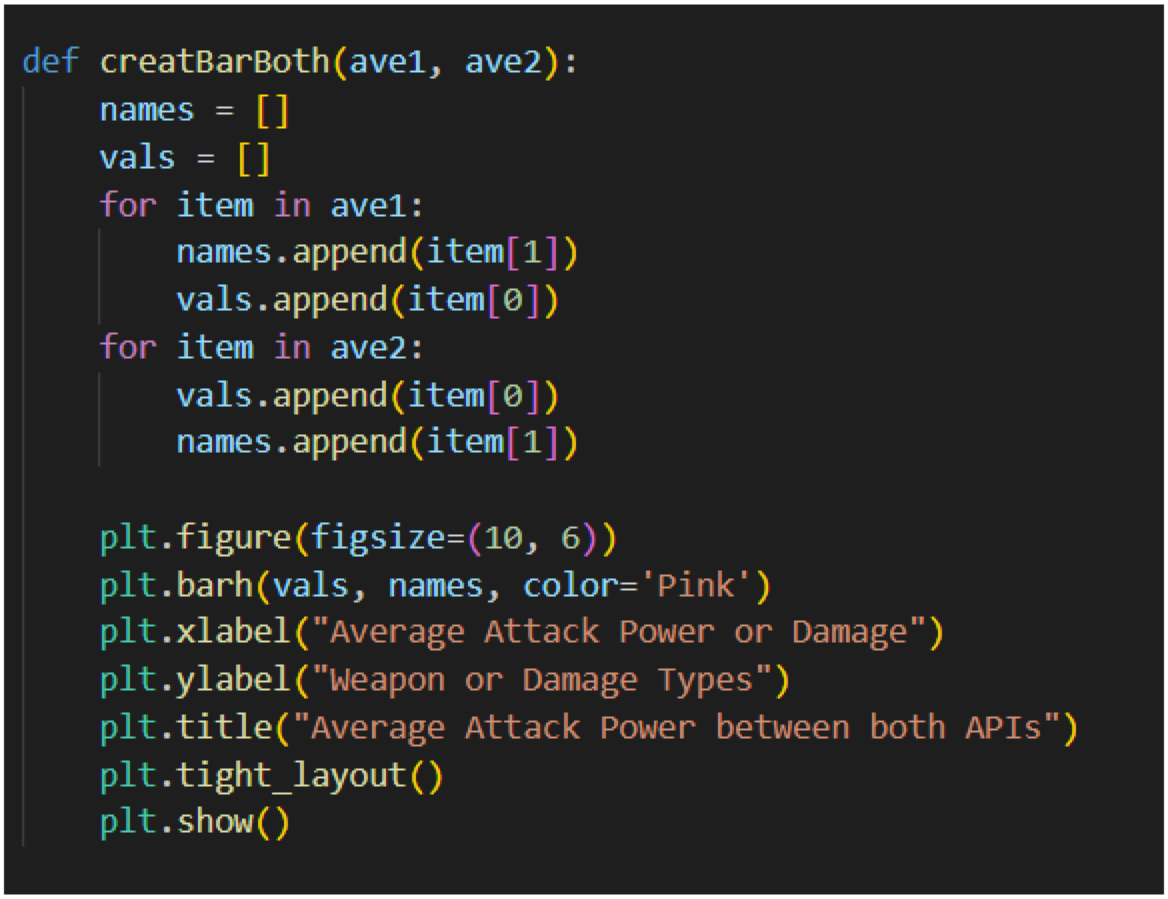
In addition to understanding which weapon type is the most common, it is important to see which weapon type produces the largest damage overall. This can be useful as a player of the game to know which weapon type is best to use in combat.



```python
def averageDamagesMHW(cur):
    cur.execute('''
        SELECT wt.name, AVG(w.attack)
        FROM mhw_weapons w
        JOIN mhw_weapon_types wt ON w.weapon_type_id = wt.id
        GROUP BY w.weapon_type_id
    ''')
    results = cur.fetchall()
    return results

def createBarChartMHW(results):
    types = []
    averages = []
    for item in results:
        types.append(item[0])
        averages.append(item[1])

    plt.figure(figsize=(10, 6))
    plt.barh(types, averages, color='orange')
    plt.xlabel("Average Attack Power")
    plt.ylabel("Weapon Types")
    plt.title("Average Attack Power by MHW Weapon Type")
    plt.tight_layout()
    plt.show()
```

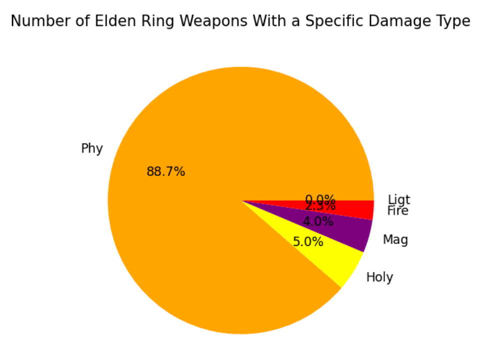
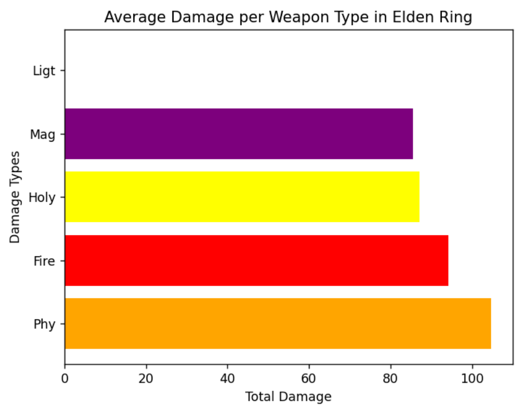
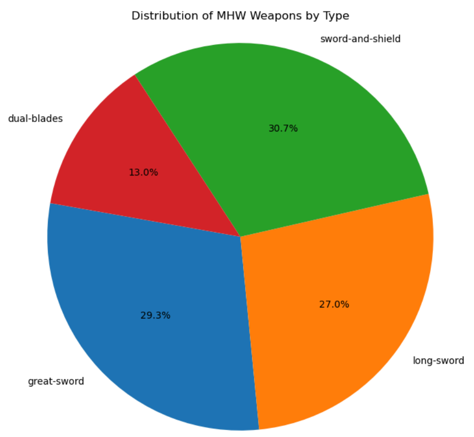
Average damages per weapon types across both APIs

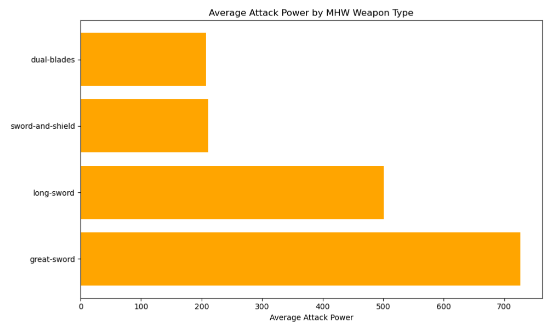
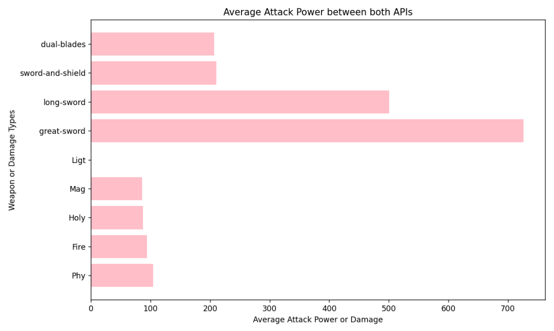
It is important to see the trend across both API’s so this calculation was more about pairing the two calculations of average attack power and average damage from both Monster Hunter Wilds and Elden Ring respectively so that we can see the trends in the data.



```python
def creatBarBoth(ave1, ave2):
    names = []
    vals = []
    for item in ave1:
        names.append(item[1])
        vals.append(item[0])
    for item in ave2:
        vals.append(item[0])
        names.append(item[1])

    plt.figure(figsize=(10, 6))
    plt.barh(vals, names, color='Pink')
    plt.xlabel("Average Attack Power or Damage")
    plt.ylabel("Weapon or Damage Types")
    plt.title("Average Attack Power between both APIs")
    plt.tight_layout()
    plt.show()
```

VISUALS

	This visualization shows the percentage of weapons that belong to a certain category of weapon damages. There is a clear majority in the weapons that demonstrate having more Physical damage than the rest. No weapons had the light categorization as a max damage.
	This visualization shows the average damages per the different weapon types. The horizontal access shows the total damage and the vertical access shows the damage types. Physical damage is seen to have the largest, while Magic has the least. No weapons had the light categorization as a max damage.
	This visualization shows the distribution of weapons in Monster Hunter World across its different weapon types. Sword and shield pairings, great swords, and long swords are all fairly equal at around 30% of the weapons in the database, but dual blades are a clear minority at just 13% in comparison.

	This visualization shows the average attack power by weapon type in Monster Hunter World. Dual blades and sword and shield pairings sit together as the weakest by a significant amount. Great swords have the greatest attack power followed by long swords.
	This visualization shows the average attack power or damage per weapon for both of the APIs. It can be seen that the Monster Hunter Wilds weapons vastly outpower the Elden Ring Weapons, which may be a result of different game mechanics. Monster Hunter Wilds Weapons are all physical, which is still the highest average damage type amongst elden ring weapons.

INSTRUCTIONS

Running our code

Our code requires that you run both of the API data files before you run the calculations file. This allows for the data being correctly added to the database and the calculations taken properly.

General Repository Guide		
File	Description	Guide
EldenRingData.py	File used to store data to the database from the Elden Ring API	<once every document is finalized>
MonsterHunterData.py	File used to store data to the database from the Monster Hunter Wilds API	

calculations.py	File used to create calculations of the data and produce the plots.	

Run EldenRingData.py (25 times)

This may seem like a lot of time, but there are 3 tables associated with the file and there are also 300 elements of data to add. The first run adds elements to the DamageTypes table. This will be over signified with text “Damage Types Complete!”. The next 12 times add data to the WeaponDifferentDamages Table. This will be over signified with the text “WeaponDifferentDamages Complete!”. The Final 12 times adds data to the Elden_Ring table. This will be signified as over with the text “Elden_Ring complete!”.

Run MonsterHunterData.py (12 Times)

This file adds 25 elements to the mhw_weapons table for a total of 300 elements. Each run will tell you the amount of elements deposited (25) as well as the current number of elements in the data. This is signified as “Finished! Total Weapons in Database {total} and Inserted {inserted_count} new weapons into Database”

Run calculations.py (1 time)

This file contains all the calculations and plots for the data. This will only need to be run once to save everything.

FUNCTION DOCUMENTATION

EldenRingData.py		
Function Name + Purpose	Inputs	Outputs
createDataset. The purpose was to make requests from the API and synthesize the data into a 300 element list. The API has multiple pages and each page holds a 100 data values, so the function synthesizes different Urls to achieve the 300 elements	baseUrl, which is the generic url of the API	FullDatad, which is a list that contains 300 elements of data from the API
damageTypes. The purpose of the function was to add the 5 different damage types to the DamageTypes table.	cur, conn, which were used to access and insert data in the data table	None.
weaponDamages. The purpose of this function was to add the different damages for each weapon and weapon type to the WeaponDifferentDamages table. It also had the column of max damage which had the largest damage of a single type.	cur, conn, FullData, which accessed the data in intervals of 25 and updated the data table.	None.
add_weapons. The purpose of this function was to add the weapon names to the Elden_Ring table and to add the two different foreign API keys as well. This added the Max damage type label foreign key to the DamagesType table and the WeaponDamage label foreign key to the WeaponDifferentDamages table.	cur, conn, FullData, which accessed the data in intervals of 25 and updated the data table.	None.

MonsterHunterData.py		
Function Name + Purpose	Inputs	Outputs
fetch_mhw_weapon_data. Fetches data from the monster Hunter World API.	None	A JSON list of weapon dictionaries
create_tables. Creates the mhw_weapon_types and mhw_weapons tables in the database (if they don’t already exist).	conn, to connect to SQLite database	None
insert_weapons_types. Inserts unique weapon type names into the mhw_weapon_types table.	conn, weapon types: list of strings	None
insert_weapons. Inserts unique weapon names into the mhw_weapons table.	conn, weapons: list of dictionaries	None
process_and_insert_data. Processes the raw JSON weapon data into a cleaned format and inserts it into the database.	weapon data: JSON list, conn	None
count_weapons_in_db. Returns the number of weapons currently in the database.	conn	Integer count of weapons

calculations.py		
Function Name + Purpose	Inputs	Outputs
countTypesMHW. This selects the number of	cur, this is used to access the database	Returns a list of tuples in the form (weapon type,

weapon under each weapon type		number)
createPieChartMHW. This creates a pie chart for MHW that contains the percentages of each weapon type and its occurence	results, which is a list of tuples	None/ saves a pie chart
averageDamageMHW. This gives the average damage per each weapon type	cur, this is used to access the database	Returns a list of tuples in the form (weapon type, average damage)
createBarChartMHW. This creates a bar chart for the MHW that contains the average attack power and weapon types.	results, which is a list of tuples	None/ saves a barchart
countDamagesElden. This selects the number of weapons under each damage type	cur, this is used to access the database	Returns a list of sorted tuple of the damage type and number
createPieChartElden. This creates a pie chart for Elden Ring that contains the percentages of each damage type and its occurence	counts, which is a list of tuples	None/ saves a pie chart
averageDamagesElden. This gives the average damage per each damage type	cur, this is used to access the database	Returns a list of sorted tuple of the damage type and average damage
createBarChartElden. This creates a bar chart for the Elden Ring that contains the average damage and damage types	averages, which is a list of tuples	None/ saves a barchart
createBarChartBoth. This creates a bar chart of both	ave1, ave2, which are both lists of tuples	None/ saves a barchart

the weapon and damage types and their respective values for both Elden Ring and Monster Hunter Wilds together		
writeToFile. This writes a summary of all the calculations to calculations.txt.	cur, this is used to access the database	None

Resources

Date	Issue Description	Location of Resource	Result (did it solve the issue?)
04/08/2025	API documentation for Elden Ring	https://docs.eldenring.fanapis.com/docs/weapons	Managed to retrieve data from API. This is a public API.
04/08/2025	API documentation for Monster Hunter Worlds	https://docs.mhw-db.com/#weapons	Managed to retrieve data from API. This is a public API.
04/22/2025	Documentation for different types of plots in MatPlotLib	https://matplotlib.org/stable/plot_types/index.html	Managed to understand and use ideas to create our plots.