
# INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY NAYA RAIPUR

## Pokémon Battle Simulator

Team Members:
Iinesh Jain (241010232)
Jayant Yadav (241010233)
Joy Sarkar (241010234)
Karan Jadhav (241010235)
Karishma Singha Roy (241010236)

Course Name: Object-Oriented Programming (OOP)

Instructor Name: Dr. Santosh Kumar

Date of Submission: 29th April, 2025

**GitHub**

## Abstract

This project is a simple Pokémon Battle Simulator developed using Python to demonstrate object-oriented programming concepts.
The objective of this project is to simulate a battle between two Pokémon, incorporating type effectiveness, move selection, and turn-based gameplay.
Key features include player and computer-controlled Pokémon selection, dynamic damage calculations based on type matchups, and a complete battle loop with victory conditions.

## Introduction

The Pokémon Battle Simulator is designed to model Pokémon battles in a simplified form, making use of core OOP principles.
Classes like Pokemon, Move and Battle encapsulate attributes and behaviors of real-world Pokémon battles. Inheritance isn't heavily used here, but encapsulation, abstraction, and modularity are clearly demonstrated.
The project highlights how OOP structures make it easier to extend functionality — e.g., adding more Pokémon, moves, or mechanics.

## Implementation

Platform/Tools used:

Visual Studio Code
Python 3.13.3

Libraries used:

Python standard library (random)

OOP concepts used:

- Classes: Organization of components like Move, Pokemon, LegendaryPokemon, and Battle.
- Objects: Instances of Pokémon and Moves are created.
- Inheritance: LegendaryPokemon inherits from Pokemon.
- Encapsulation: Attributes are contained within objects.
- Polymorphism: Overriding methods and treating child objects as parent objects.
- Composition: Pokemon objects composed of Move objects.
- Access Modifiers: Managed via methods (Python convention).
- Type Chart and Dynamic Behavior: Runtime behavior changes.
- Control Structures and Exception Handling: Smooth user interaction.

Screenshots:

Pokémon Data:

```
pokemons = [
    Pokemon("Charizard", "Fire", 100, 30, 20, [move("Flamethrower", 40, "Fire"), move("Fire Spin", 35, "Fire")]),
    Pokemon("Blastoise", "Water", 100, 28, 22, [move("Water Gun", 35, "Water"), move("Hydro Pump", 50, "Water")]),
    Pokemon("Venusaur", "Grass", 100, 29, 21, [move("Razor Leaf", 36, "Grass"), move("Solar Beam", 50, "Grass")]),
    Pokemon("Onix", "Rock", 110, 25, 30, [move("Rock Slide", 45, "Rock"), move("Stone Edge", 55, "Rock")]),
    Pokemon("Pikachu", "Electric", 90, 35, 18, [move("Thunderbolt", 50, "Electric"), move("Thunder Shock", 30, "Electric")]),
    Pokemon("Golem", "Rock", 105, 28, 28, [move("Rock Throw", 40, "Rock"), move("Earthquake", 50, "Rock")]),
    Pokemon("Raichu", "Electric", 98, 34, 21, [move("Spark", 37, "Electric"), move("Volt Tackle", 45, "Electric")]),
    Pokemon("Magmar", "Fire", 98, 29, 19, [move("Fire Punch", 36, "Fire"), move("Lava Plume", 40, "Fire")]),
    Pokemon("Mewtwo", "Psychic", 120, 40, 25, [move("Psychic", 45, "Psychic"), move("Shadow Ball", 50, "Ghost")]),
    Pokemon("Machamp", "Fighting", 110, 35, 30, [move("Dynamic Punch", 50, "Fighting"), move("Close Combat", 45, "Fighting")]),
    Pokemon("Gengar", "Ghost", 90, 36, 18, [move("Shadow Ball", 45, "Ghost"), move("Lick", 30, "Ghost")]),
    Pokemon("Alakazam", "Psychic", 85, 37, 22, [move("Confusion", 40, "Psychic"), move("Shadow Ball", 50, "Ghost")]),
    Pokemon("Tyranitar", "Rock", 115, 30, 35, [move("Crunch", 45, "Dark"), move("Stone Edge", 50, "Rock")]),
    Pokemon("Dragonite", "Dragon", 120, 38, 28, [move("Dragon Claw", 45, "Dragon"), move("Hyper Beam", 50, "Normal")]),
    Pokemon("Lugia", "Psychic", 130, 34, 40, [move("Aeroblast", 50, "Flying"), move("Psychic", 45, "Psychic")]),
    Pokemon("Zapdos", "Electric", 100, 35, 25, [move("Thunder", 50, "Electric"), move("Drill Peck", 45, "Flying")]),
    Pokemon("Snorlax", "Normal", 140, 33, 35, [move("Body Slam", 40, "Normal"), move("Hyper Beam", 50, "Normal")]),
    Pokemon("Exeggutor", "Grass", 110, 32, 30, [move("SolarBeam", 55, "Grass"), move("Leech Seed", 0, "Grass")]),
    Pokemon("Lanturn", "Water", 100, 28, 26, [move("Surf", 40, "Water"), move("Thunderbolt", 45, "Electric")]),
]
```

Classes & Methods:

```
class Pokemon:
    def __init__(self, name, type_, hp, attack, defense, moves):
        self.name = name
        self.type = type_
        self.max_hp = hp
        self.hp = hp
        self.attack = attack
        self.defense = defense
        self.moves = moves

    def take_damage(self, dmg):
        self.hp = max(0, self.hp - dmg)

    def is_fainted(self):
        return self.hp == 0

    def display(self):
        print(f"{self.name} ({self.type}) | HP: {self.hp}/{self.max_hp} | ATK: {self.attack}, DEF: {self.defense}")
        for m in self.moves:
            print(f"   - {m.name} (Type: {m.type}, Power: {m.power})")

    def display_moves(self):
        for i, m in enumerate(self.moves):
            print(f"{i+1}. {m.name} (Type: {m.type}, Power: {m.power})")
```

Key Parts of the code:

**Class Definitions:** Move, Pokemon, LegendaryPokemon, Battle
**Damage Calculation:** Based on type effectiveness
**Inheritance Example:** LegendaryPokemon inherits Pokemon
**Battle System**: Turn-based attacks
**Exception Handling:** For move selections

## Conclusion

This project simulates a basic Pokémon battle with turn-based gameplay and type effectiveness. Challenges included designing a scalable damage system based on Pokémon types and ensuring player and computer turns were handled cleanly.
In future versions, features like abilities, multiple battles (tournaments), more complex type interactions can be added and graphical interfaces can be further improved to enhance realism and playability.

Challenges faced:

- Managing user input validation robustly.
- Correctly implementing type effectiveness logic.
- Balancing stats and move powers.
- Handling turn-based battle flow.

## References

Pokémon Official Type Chart: https://pokemondb.net/type

Python random Module Documentation: https://docs.python.org/3/

Stack Overflow discussions on object-oriented design

TutorialsPoint - Python OOP Concepts

GeeksForGeeks - Inheritance and Polymorphism