PokeBattlES

AI 2001 – Final Project

Rule Based Expert System

Poke BattlES

Team

- Hibba Imtiaz
- Michael Molnar
- Tejas Vyas

What is PokeBattlES?

- Rule-based expert system
- It’s a simple trading card game which allows a human player to play Pokemon Trading Cards against an AI and uses Pokemon Type Rules and respective Attack and Defense values to enrich the gameplay.
- The objective is to get your opponent to 0 HP.
- Written using PyKnow.

Background and Rationale

Background

- Pokémon is world’s 3rd biggest video game franchise of all time.
- Created by Satoshi Tajiri, in 1996 as a role-playing video game for the Game Boy handheld game console.
- We take inspiration from this game and create and expert system using the trading card game.

Rationale

- We went with this project because:
  - It’s a perfect simulation set for gaming industry expert systems, covering general idea behind all card games
  - It’s fun and nostalgic! **We made the cards a meme version to go Plus Ultra!**

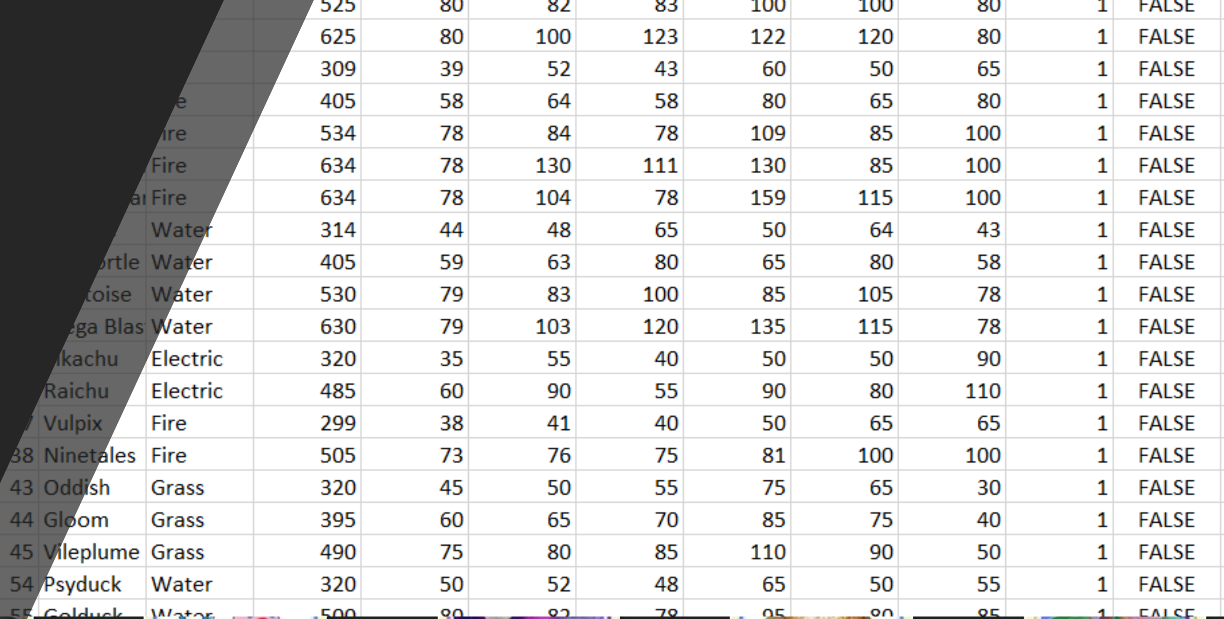
Architecture

- The project contains 3 components:
  - Excel and Assets containing card details
  - PyKnow module containing game logic and rules
  - Flask webapp hosting the game

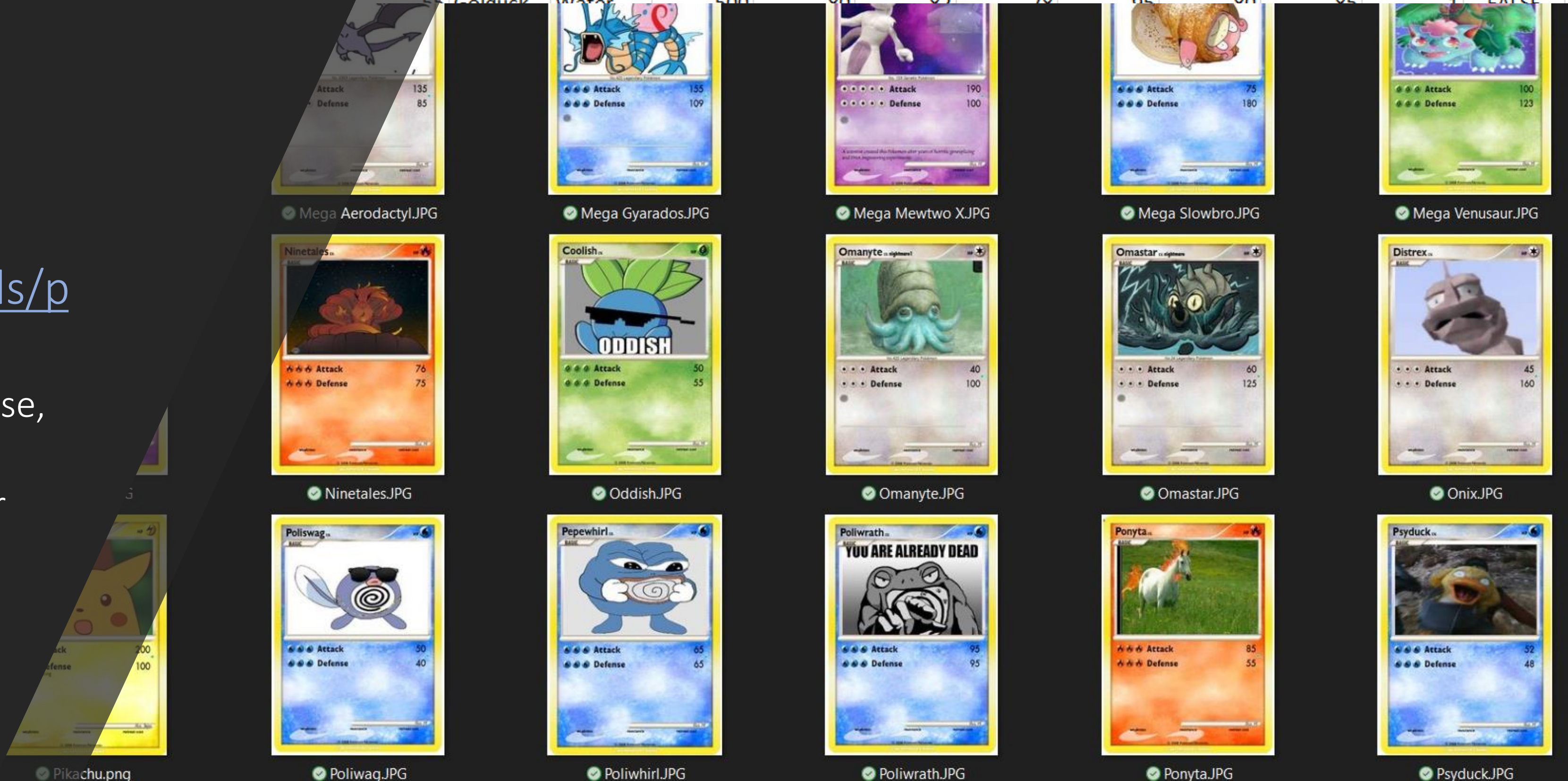
Available at: https://github.com/tejas1794/PokeBattlES

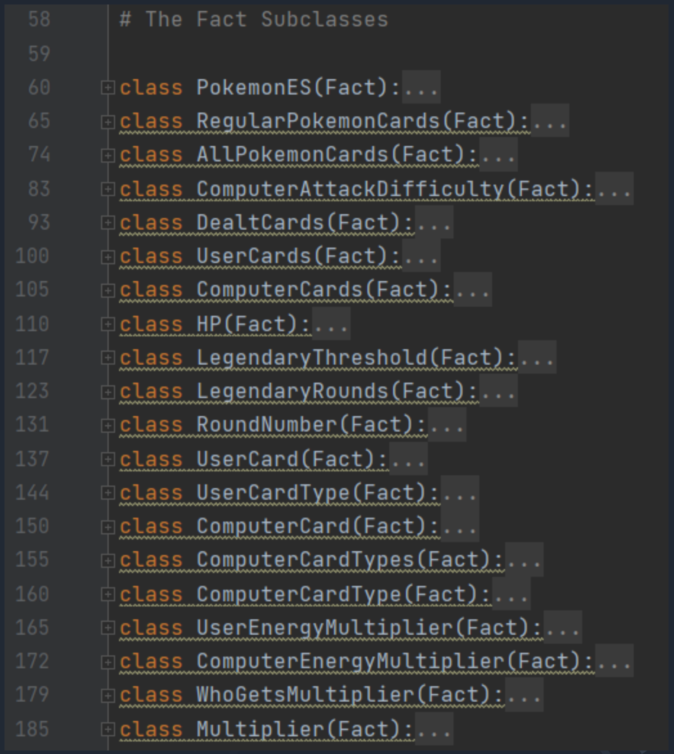
Assets and Data

- The app contains CSV which is used as a dataset to be used for the Knowledge Engine setup
- Also contains images of cards mapped to each record on CSV
- The data was imported from Kaggle: https://www.kaggle.com/abcsds/pokemon
  - Updated to contain Attack, Defense, Type for core gameplay
  - Added Legendary/Energy card for advanced gameplay

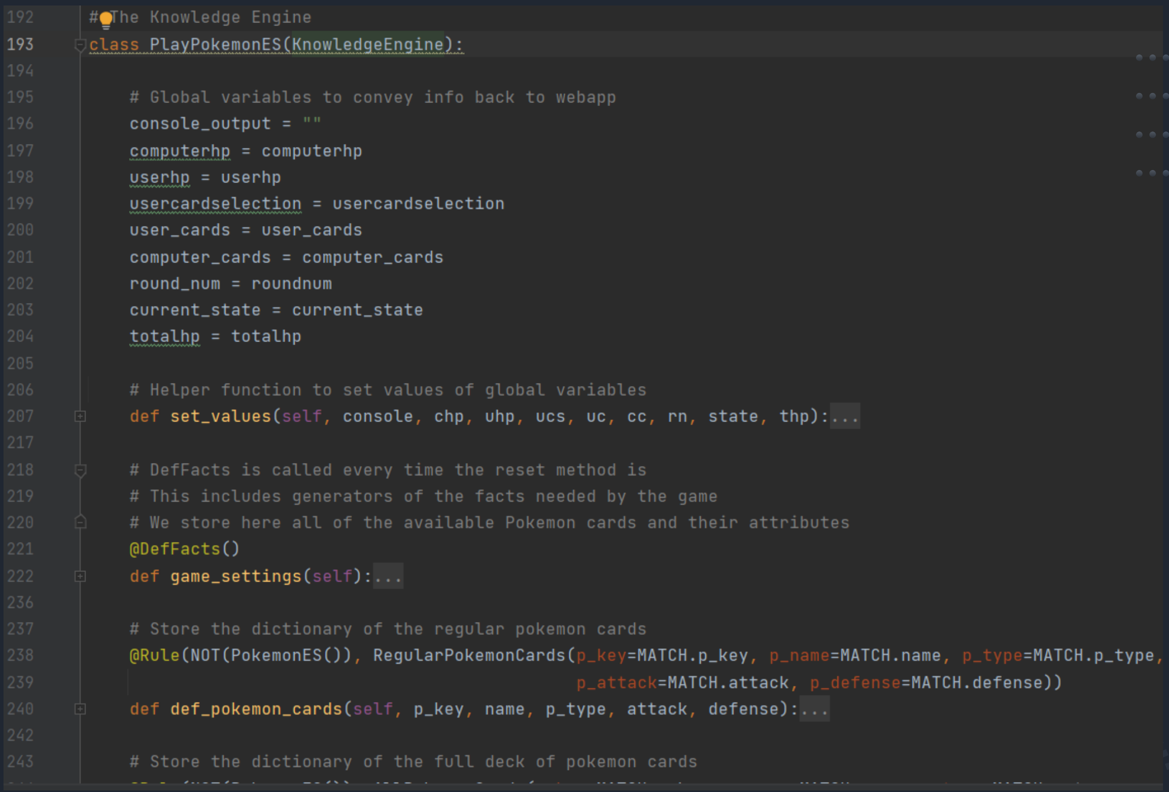


			525	80	82	83	100	100	80	1	FALSE
			625	80	100	123	122	120	80	1	FALSE
			309	39	52	43	60	50	65	1	FALSE
		e	405	58	64	58	80	65	80	1	FALSE
		ire	534	78	84	78	109	85	100	1	FALSE
		Fire	634	78	130	111	130	85	100	1	FALSE
	a	Fire	634	78	104	78	159	115	100	1	FALSE
		Water	314	44	48	65	50	64	43	1	FALSE
	rtle	Water	405	59	63	80	65	80	58	1	FALSE
	oise	Water	530	79	83	100	85	105	78	1	FALSE
	ga Blas	Water	630	79	103	120	135	115	78	1	FALSE
	kachu	Electric	320	35	55	40	50	50	90	1	FALSE
	Raichu	Electric	485	60	90	55	90	80	110	1	FALSE
7	Vulpix	Fire	299	38	41	40	50	65	65	1	FALSE
38	Ninetales	Fire	505	73	76	75	81	100	100	1	FALSE
43	Oddish	Grass	320	45	50	55	75	65	30	1	FALSE
44	Gloom	Grass	395	60	65	70	85	75	40	1	FALSE
45	Vileplume	Grass	490	75	80	85	110	90	50	1	FALSE
54	Psyduck	Water	320	50	52	48	65	50	55	1	FALSE
55	Golduck	Water	500	80	82	78	95	80	85	1	FALSE





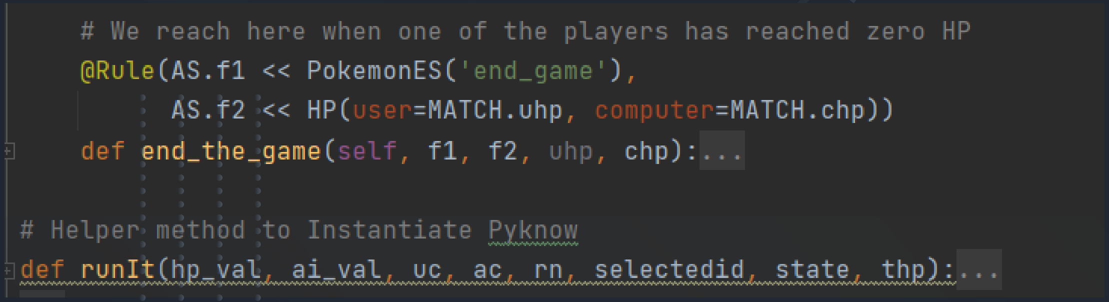
```
# The Fact Subclasses

class PokemonES(Fact):...
class RegularPokemonCards(Fact):...
class AllPokemonCards(Fact):...
class ComputerAttackDifficulty(Fact):...
class DealtCards(Fact):...
class UserCards(Fact):...
class ComputerCards(Fact):...
class HP(Fact):...
class LegendaryThreshold(Fact):...
class LegendaryRounds(Fact):...
class RoundNumber(Fact):...
class UserCard(Fact):...
class UserCardType(Fact):...
class ComputerCard(Fact):...
class ComputerCardTypes(Fact):...
class ComputerCardType(Fact):...
class UserEnergyMultiplier(Fact):...
class ComputerEnergyMultiplier(Fact):...
class WhoGetsMultiplier(Fact):...
class Multiplier(Fact):...
```



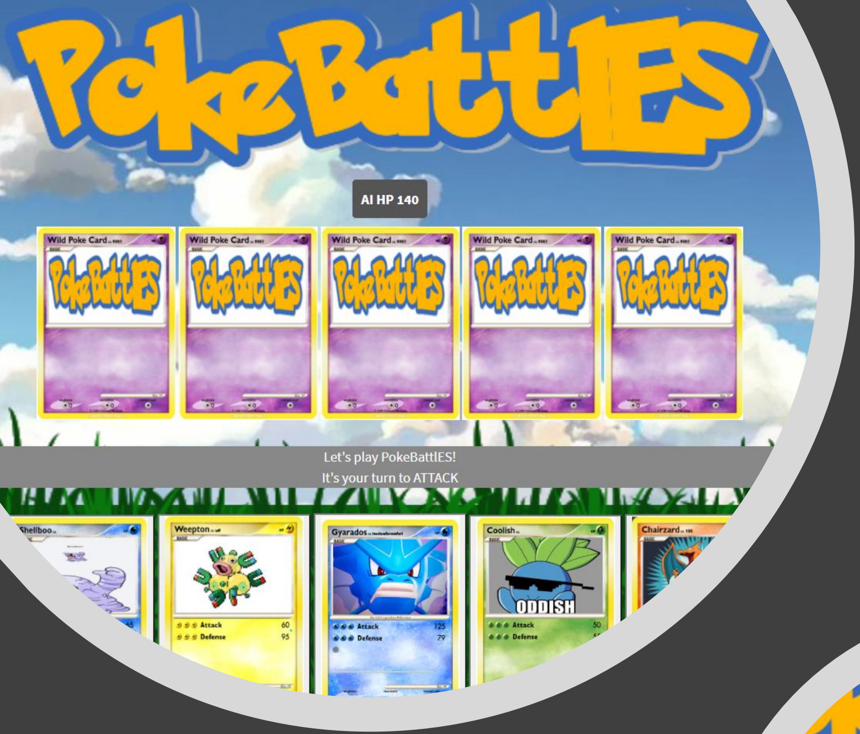
```
#The Knowledge Engine
class PlayPokemonES(KnowledgeEngine):

    # Global variables to convey info back to webapp
    console_output = ""
    computerhp = computerhp
    userhp = userhp
    usercardselection = usercardselection
    user_cards = user_cards
    computer_cards = computer_cards
    round_num = roundnum
    current_state = current_state
    totalhp = totalhp

    # Helper function to set values of global variables
    def set_values(self, console, chp, uhp, ucs, uc, cc, rn, state, thp):...

    # DefFacts is called every time the reset method is
    # This includes generators of the facts needed by the game
    # We store here all of the available Pokemon cards and their attributes
    @DefFacts()
    def game_settings(self):...

    # Store the dictionary of the regular pokemon cards
    @Rule(NOT(PokemonES()), RegularPokemonCards(p_key=MATCH.p_key, p_name=MATCH.name, p_type=MATCH.p_type,
                                                p_attack=MATCH.attack, p_defense=MATCH.defense))
    def def_pokemon_cards(self, p_key, name, p_type, attack, defense):...

    # Store the dictionary of the full deck of pokemon cards
```

PyKnow Module

- Contains core game logic
- Contains rules for:
  - Type specific interactions
  - Game state specific interactions
  - Legendary Card Interactions
  - Energy Card Interactions



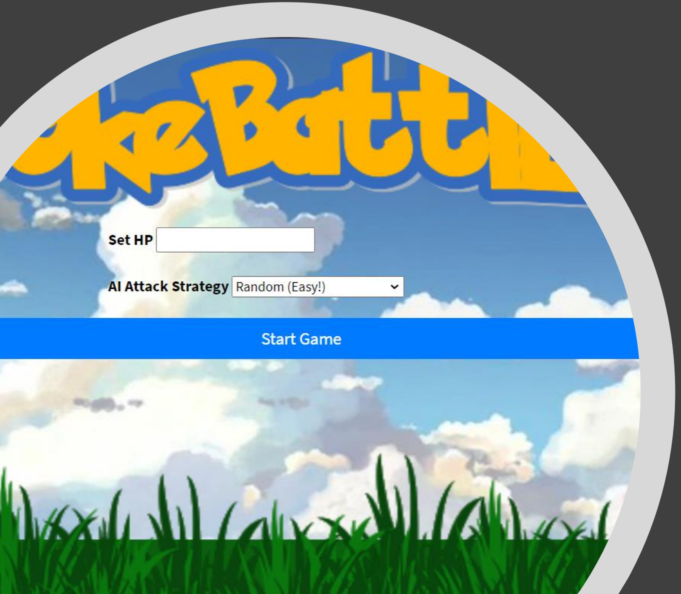
```
    # We reach here when one of the players has reached zero HP
    @Rule(AS.f1 << PokemonES('end_game'),
          AS.f2 << HP(user=MATCH.uhp, computer=MATCH.chp))
    def end_the_game(self, f1, f2, uhp, chp):...

# Helper method to Instantiate Pyknow
def runIt(hp_val, ai_val, uc, ac, rn, selectedid, state, thp):...
```



Let's play PokeBattlES!
It's your turn to ATTACK

card expert system is a game made using Rule expert systems.

You Win! You're a poke master! (^_^)



Set HP

AI Attack Strategy: Random (Easy!)

Flask Webapp

- Flask Webapp acts as the point of interaction for Users
- Uses PyKnow Module, maintains and displays game state
- Available on: https://pokebattles.azurewebsites.net/