OUTLINE

Bare minimum:

- Pokemon objects that contain important stats, available moves they CAN learn, and moves that there are GOING to use in battle
- Move objects that contain important stats and any special statuses they inflict
- Status objects that can inflict those statuses by using like a use_status() function or smth
- Tournament object that holds all of these and decides how turns play out
- Dashboard to visualize all this
- Some way to create custom Pokemon object w/ any of current moves
- Some way to measure power of Pokemon
- AI that player can battle against
- Contains all Gen 1 Pokemon and SIMPLE Gen 1 moves
- Gen 1 Damage formula

Good things we should probably have:

- ALL Gen 1 moves (including annoying ass moves like Substitute)
- Ways to create custom moves (ideally do damage and inflict simple status)
- Pokemon images (can probably be scraped)
- Extra button that lets player use generic healing item or general status cure-all (after all there are some statuses the will literally stick around forever without special items) with limited use
- Allow users to change level of pokemon to use / fight against

Nice things we don't really need:

- Ways to limit which pokemon can use the custom moves we implement (e.g. what if we want magikarp to use an overpowered custom move?)
- Ways to create custom statuses
- Ways to create custom moves using complex rules / statuses
- Extra button that lets players choose to use a support item (like one that boosts defense, specific burn cureness, PP up, etc) with limited use
- AI vs AI
- Different AI strategies to show effectiveness

God no (...unless?):

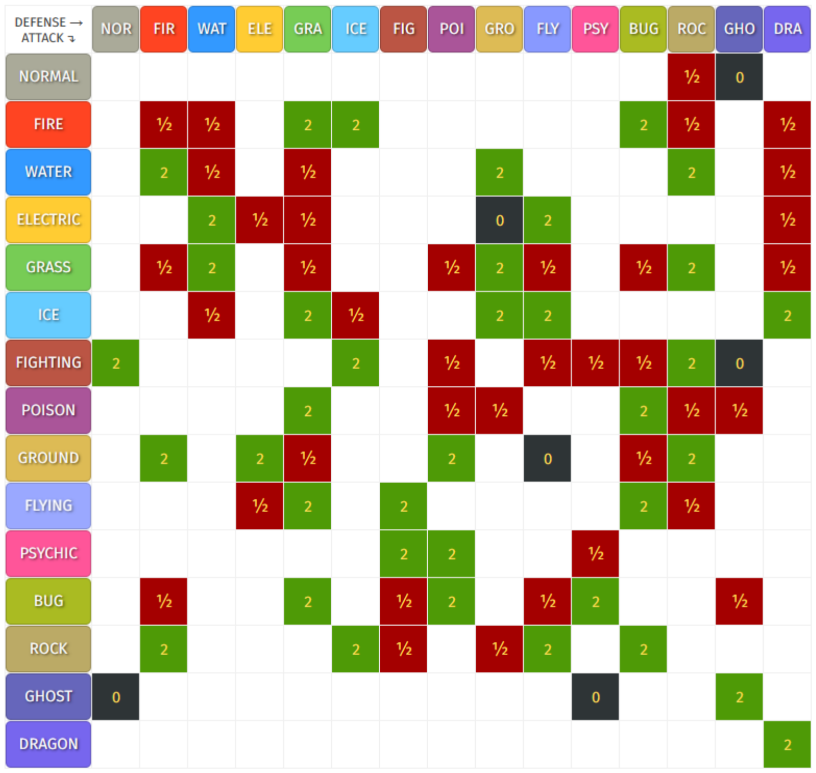
- Support for multi-gen Pokemon, moves, damage formulas
- Differentiate between wild battles and trainer battles (e.g. some moves have different effect on wild pokemon, can catch pokemon)
- Multi-pokemon battle

DAMAGE

Damage formula:

$$Damage = \left(\frac{\left(\frac{2 \times Level \times Critical}{5} + 2\right) \times Power \times A/D}{50} + 2 \right) \times STAB \times Type1 \times Type2 \times random$$

<u>Level</u>: Pokemon's Level (should be the same for all Pokemon)
<u>Critical</u>: Generate a random number from 0 to 255. If the move has a "High Crit" tag, divide this number by 8, rounded down. If this number is strictly less than half the Pokemon's speed (rounded down), then Critical = 2. Otherwise, Critical = 1.
<u>Power</u>: Power stat of move, included in object.
<u>A</u>: Attack stat of Pokemon using the move if the move is in the 'Physical' category, otherwise use the Special Attack stat.
<u>D</u>: Defense stat of Pokemon being targeted if the move is in the 'Physical' category, otherwise use the Special Defense stat.
<u>STAB</u>: Equal to 1.5 if the user Pokemon has a type that is the same type as the move (e.g. Grass type Pokemon using a Grass type move). Otherwise, equal to 1.
<u>Type1</u>: Type effectiveness, refer to the entry below. Equal to 1 for blank entries.



DEFENSE → ATTACK ↴	NOR	FIR	WAT	ELE	GRA	ICE	FIG	POI	GRO	FLY	PSY	BUG	ROC	GHO	DRA
NORMAL													½	0	
FIRE		½	½		2	2						2	½		½
WATER		2	½		½				2				2		½
ELECTRIC			2	½	½				0	2					½
GRASS		½	2		½			½	2	½		½	2		½
ICE			½		2	½			2	2					2
FIGHTING	2					2		½		½	½	½	2	0	
POISON					2			½	½			2	½	½	
GROUND		2		2	½			2		0		½	2		
FLYING				½	2		2					2	½		
PSYCHIC							2	2			½				
BUG		½			2		½	2		½	2			½	
ROCK		2				2	½		½	2		2			
GHOST	0										0			2	
DRAGON															2

<u>Type 2</u>: Type effectiveness, same table but only use for target Pokemon for multiple types (e.g. Venusaur is both Grass and Poison). Equal to 1 if the target has only one type.
<u>Random</u>: Generate a random integer from 217 to 255, inclusive. Divide this by 255.

MOVE TAGS (Mostly Complete)

Useless: Does literally nothing in trainer battles

Burn: Applies burn status effect. Burned Pokemon will lose 1/16 of their max HP after they attack, or at the end of the turn if they didn't attack; they also get their Attack stat halved. Fire type Pokemon cannot be burned.

Freeze: Applies freeze status effect. Target Pokemon will be unable to act unless hit with a move that can inflict burn. Essentially a de-facto win unless we implement a way to use items to alleviate this effect.

Paralyze: Allies paralysis status effect. Target Pokemon will have its speed divided by 4 (rounded down), and any move it uses can fail a quarter of the time.

Flinch: Applies flinch status effect. Pokemon will not be able to use a move for a single turn. This effect is cured after this happens.

Poison: Applies poison status effect. Poisoned Pokemon will lose 1/16 of their max HP after they attack, or at the end of the turn if they didn't attack.

Confuse: Applies confusion effect. Lasts for 1-4 turns (turns where the Pokemon is asleep, frozen, or flinched don't count). The target has a 50% chance of hitting itself instead of executing its chosen move. (Self damage follows the above formula for a move that has no type, cannot score critical hits, and has power = 40).

High Crit: Move will modify the critical chance formula described above.

Instakill: Will insta-kill target if it lands. The move must have a base accuracy of 30%, but will always fail if the user has a lower Speed compared to the target.

Recoil: User will lose HP equal to 25% of the damage dealt using this move. If the move is Struggle, the user will receive 50% of the damage dealt instead.

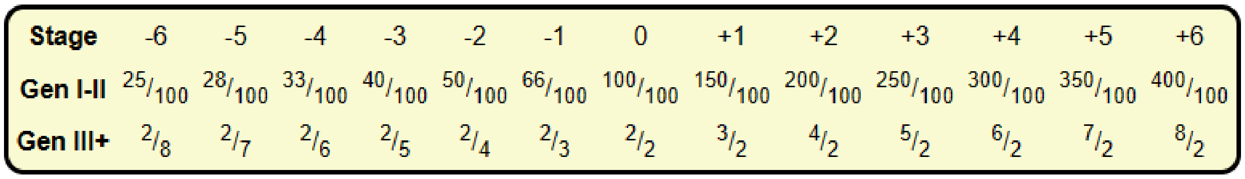
Crash: If the attack misses, the user loses 1 HP.

Delayed: The move does nothing on the turn it is selected. On the next turn, the move will do damage and will count as the move used for that turn as well.

Multihit (double): Hits twice. Only the first hit can be critical. The second hit will deal the same amount of damage, sans critical effect.

Multihit (multiple): Hits 2-5 times (⅜ chance for 2, ⅜ chance for 3, ⅛ chance for 4, ⅛ chance for 5). Only the first hit can be critical. The successive hits will deal the same amount of damage, sans critical effect.

StatChange: Modifies a stat for either the user or target by a "stage". Stage translation to actual stat change shown below:



Stage	-6	-5	-4	-3	-2	-1	0	+1	+2	+3	+4	+5	+6
Gen I-II	25/100	28/100	33/100	40/100	50/100	66/100	100/100	150/100	200/100	250/100	300/100	350/100	400/100
Gen III+	2/8	2/7	2/6	2/5	2/4	2/3	2/2	3/2	4/2	5/2	6/2	7/2	8/2

SelfDestruct: Causes the user to faint

Sleep: Target pokemon will not be able to act for 1 - 7 turns (chosen randomly).

Heal (lifesteal): User heals hp equal to half of the damage dealt

Heal (selfheal): User heals HP equal to ½ its max HP

AI

Up to y'all but make sure to put hella effort into it as this is the thing that makes sure this counts as a data science project

Possible strategies (feel free to suggest more):

Turn Based: AI will choose whichever move that does the most damage on a turn-by-turn basis.
PROS: Easy to implement, not computationally expensive
CONS: Hard to determine when to use status moves as some Pokemon rely on buffing themselves / status effects to do damage, may be a bit arbitrary to implement

Battle Based: AI will calculate how many turns it expects to survive (e.g. assuming the user plays optimally), and chooses the sequence of moves that will result in the most overall damage done (can be done by simply testing every possible sequence of moves)
PROS: Easier to implement more complex moves that have lasting effects (poison, buffs)
CONS: Harder to change strategy based on sudden new conditions (e.g. user Pokemon buffs a physical stat making special moves more effective)

Turn / Battle Based Hybrid: AI will calculate optimal sequence of moves like in battle-based strategy, but will recreate new sequence after each turn.
PROS: Most thorough of above strategies, as it can implement complex moves while also reacting to turn-by-turn changes
CONS: Probably the hardest to implement and most computationally expensive

OBJECT DESIGN (incomplete)

<u>Pokemon</u>: Represents Pokemon instance

- Houses stats scaled to level (can either also have variables equal to their base stat and have a method that calculates scaled stats, or simply have an outside function that calculates scaled stats and inputs that directly in __init__)
- Houses 4 Move objects that it WILL use in battle (must be drawn from housed list of moves it CAN use)
- Method that tells it to use one of its 4 moves
- Perhaps it can house the statuses it is currently affected by?
- Method that can calculate power level
- Method that can calculate optimal move based on whatever ai strategy we implement
- Maybe have a method that tells it to activate status effects (e.g. burn, poison)

<u>Move</u>: Represents an attack a Pokemon can do

- Houses basic stats that can be adjusted based on description
- Method that adds tags to move based on the description (called within __init__, essentially a helper func)
- Method that calculates damage and applies status given user Pokemona and target Pokemon (make sure to also change PP)

<u>Status</u>: Represents status affliction

- Should house values necessary for status, can be different for each status
- Should have a type that differentiates one type of status from another
- Should have a unique inflict_status() method (perhaps we can pass this in if we decide to create custom statuses)
- Thoughts?

<u>Tournament</u>: Represents actual battle

- Houses both Pokemon (assigns user and ai)
- Method to calculate turn events (Check speed vals, ask Pokemon to choose move / calculate damage / activate statuses)
- Need a way to print what happens