**CDT 1D**

# Description

This game is a simplified and shortened version of the Nintendo Pokemon game (Yellow/Red/Blue Gameboy Version). There are three levels of trainers, each with two Pokemons, that the player has to defeat to win the game. The player will get three Pokemons and their stats will reset at the start of every level.

The objective of the game is to kill as many Pokemons and clear as many levels as possible. Each round, the player has two options to choose from - to Fight or to Change Pokemons. Every Pokemon has four moves they can choose from which consists of a combination of two or three types of the following moves: attack, which deals damage to the opponent; buff, which brings down the opponent’s defense; and debuff, which increases its own attack.

For this game, most of the information about trainers, Pokemons, and moves, is stored in individual dictionaries. We chose to store this information in dictionaries so that we can clearly reference the various properties of each object, which are stored as values, by referencing their keys, capitalising on the key-value functionality of dictionaries. Some dictionaries also called upon other dictionaries, such as a moveset dictionary being called from a Pokemon dictionary, as these two objects are clearly and fundamentally related to each other.

Sources we used include Pokedex which we used to reference the Pokemons and their movesets and a Pokemon stat calculator to set the base stats of the pokemons we choose.

Pokemons: <https://www.pokemon.com/us/pokedex/>

Stats levels: <https://pycosites.com/pkmn/stat.php>

Moves: <https://bulbapedia.bulbagarden.net/wiki/Main_Page>

# Documentation

It is required for this game to use the time and random library.

**start\_of\_battle\_text(enemy\_trainer, enemy\_pokemon, Saiyan, Saiyan\_pokemon):**

This function takes in four parameters: **enemy\_trainer**, **enemy\_pokemon**, **Saiyan** and **Saiyan\_pokemon** and prints out the text during initiation of each battle using placeholders to insert the parameters into the text.

**missed\_text(pokemon):**

This function takes in one parameter: the **pokemon** dictionary and prints out the text that the Pokemon has missed its move.

**update\_status(pokemon1, pokemon2):**

This function takes in two parameters: **pokemon1** and **pokemon2** and calculates the health difference between each Pokemon’s current health and their maximum health, and prints out a health bar for both **pokemon1** and **pokemon2** that reflects the Pokemons’ names, levels, types and a visual representation of their health. As long as the Pokemon’s health is not equal to 0, the health bar will always have at least one bar.

**accuracy\_check(move):**

This function takes in one parameter: a **move** dictionary, and compares the accuracy attribute of **move** to a random number generated and returns **True** if the accuracy attribute is equal to or greater than the random number, or **False** if otherwise.

**attack(pokemon1, pokemon2, attack\_move):**

This function takes in three parameters: the **pokemon1, pokemon2** and **attack\_move** dictionaries and contains the attack and defense multipliers based on the types of attacking move and defending Pokemon. It conducts an accuracy check of the move and if the move is successfully executed, calculates the damage dealt to **pokemon2**, based on the difference between their attack and defence stats as well as their type difference, and returns the dictionary **pokemon2** with the updated health.

**attack\_text(pokemon1, pokemon2, attack\_move):**

This function takes in three parameters: the **pokemon1, pokemon2,** and **attack\_move** dictionaries and prints out the statement that states the name of the attacking Pokemon, name of the defending Pokemon, and the move used.

**buff(pokemon, move):**

This function takes in two parameters: the **pokemon** and **move** dictionaries, and updates the attack or defence multiplier key of **pokemon** based on the number of stages the **move** changes.

**buff\_text(pokemon, move):**

This function takes in two parameters: the **pokemon** and **move** dictionaries, and prints out the name of the **pokemon** and the name of the **move** that it is using, together with the effectiveness of the buff used, based on the stage of the **move**.

**buff\_too\_much(pokemon, move):**

This function takes in two parameters: the **pokemon** and **move** dictionaries, and prints out a statement that indicates that the **move** had no effect on the **pokemon**’s stats.

**debuff(pokemon, move):**

This function takes in two parameters:the **pokemon** and **move** dictionaries, and updates the attack or defence multiplier key of **pokemon** based on the number of stages the **move** changes.

**debuff\_text(pokemon1, pokemon2, move):**

This function takes in two parameters: the **pokemon** and **move** dictionaries, and prints out the name of the **pokemon** and the name of the **move** that it is using, together with the effectiveness of the debuff used based on the stage of the **move**.

**debuff\_too\_much(pokemon1, pokemon2, move):**

This function takes in three parameters: the **pokemon1, pokemon2** and **move** dictionaries, and prints out a statement that indicates that the **move** used, and that ithad no effect on the **pokemon**’s stats.

**win\_text():**

This function takes in one parameter: the **trainer** dictionary, and prints out a statement that says that the player has defeated an enemy trainer.

**lose\_text():**

This function does not take in any parameters and prints two statements which inform the player that they no longer have any more Pokemons alive and as a result have lost the game.

**ai\_choose\_move(pokemon):**

This function takes in one parameter: the **pokemon** dictionary, and returns a random move out of the four moves the Pokemon has using the randint() function.

**user\_move(pokemon):**

This function takes in one parameter: the **pokemon** dictionary, and prompts the player to choose the move they want their Pokemon to use using the input() function, and returns the move selected through key-value pairing selection from the dictionary.

**check\_remaining\_pokemon(trainer):**

This function takes in one parameter: the **trainer** dictionary of the trainer that the player is fighting against that round, and checks the health of all the trainer’s Pokemons and returns **remaining\_pokemon**, which is an integer of the number of alive Pokemons the trainer has.

**change\_pokemon(trainer, pokemon):**

This function takes in two parameters: the **trainer** dictionary of the trainer who is making the move, and the **pokemon** dictionary of one of the trainer’s pokemon. It sets the Pokemon chosen by the trainer as the global variable, and prints a line which will inform the player of the Pokemon that is chosen.

**fainting(trainer):**

This function takes in one parameter: the **trainer** dictionary, and checks the number of the trainer’s Pokemons that are alive using the function **check\_remaining\_pokemon(trainer)**.

If the enemy trainer is input and there is only one Pokemon alive, it changes the Pokemon the trainer is using to that alive Pokemon. If there are no Pokemons alive, it sets the global variable **battle\_over** to be ‘True’ and the round ends.

If the player is input and there is only one Pokemon alive, it changes the Pokemon the player is using to that alive Pokemon. If there are no Pokemons alive, it sets the global variable for **battle\_over** to be ‘True’ and the game ends.

If there is more than one alive Pokemon, it checks which Pokemons’ health is not equal to zero using a for-loop and prints out the alive Pokemons. It will then prompt the player to choose their next Pokemon using the input function, which receives an integer that becomes the parameter for **change\_pokemon(trainer, pokemon)**.

**reset(trainer):**

This function takes in one parameter: the **trainer** dictionary, resets all the trainer’s Pokemons back to their original values and updates the dictionary before returning the **trainer** dictionary.

**reset\_pokemon\_multiplier(pokemon):**

This function takes in one parameter: the **pokemon** dictionary, resets the Pokemon’s attack and defence multipliers to zero and returns the updated **pokemon** dictionary.

**reset\_multiplier(trainer):**

This function takes in one parameter: the **trainer** dictionary, uses the function **reset\_pokemon\_multiplier(pokemon)** to reset the attack and defence multiplier of all the trainer’s Pokemons to zero and returns the updated **trainer** dictionary.

**battle(trainer, Saiyan):**

This function takes in two parameters: the **trainer** dictionary of the trainer that the player is fighting against that round and the **Saiyan** dictionary of the player. The pseudocode is as follows:

1. Initialise the global variable **Saiyan\_pokemon\_choice**
2. Initialise empty dictionaries: **Saiyan\_pokemon\_dict** and **Saiyan\_pokemon\_types**
3. Add the names of the Pokemons the player selects at the start of the game into **Saiyan\_pokemon\_dict** and their types into **Saiyan\_pokemon\_types** using a **for**-loop to access the Pokemon list in the **Saiyan** dictionary
4. Print a numbered list of the names and types of the Pokemons the player selected for them to choose their starting Pokemon
5. Prompt the player to select their starting Pokemon by keying in an integer that corresponds to the printed list using the **input** function.
6. Signal the start of a round using the function **start\_of\_battle\_text(enemy\_trainer, enemy\_pokemon, Saiyan, Saiyan\_pokemon)**, where the ‘Name’ key of the trainer’s dictionary is the input for **enemy\_trainer**, the first Pokemon in the **pokemon** list in the enemy trainer’s dictionary is the input for **enemy\_pokemon**, the ‘Name’ key of Saiyan’s dictionary as the input for **Saiyan** and the player’s Pokemon choice as the input for **Saiyan\_pokemon**. All these information are then printed out.
7. Initialise global variable **battle\_over**
8. Set **battle\_over** to be False
9. Initialise **own\_turn** as True
10. Initialise global variable **Enemy\_pokemon\_choice**
11. Set **Enemy\_pokemon\_choice** to be zero
12. while **battle\_over** is False:
    1. while trainer’s Pokemon and player’s Pokemon are alive (health more than zero):
       1. if **own\_turn** (player’s turn):
          1. Print the stats bar of both Saiyan and the enemy trainer’s Pokemons using the function **update\_status(pokemon1, pokemon2)**, where the player’s Pokemon choice is the input for **pokemon1** and the enemy trainer’s Pokemon is **pokemon2**
          2. Set **turn\_choice** as player’s input of choice of move (either fight or change pokemon)
          3. If **turn\_choice** is 1 (Player chooses to fight) :
             1. Set **move** with the function **user\_move(pokemon)**, where the player’s Pokemon choice is the input for **pokemon**
             2. If move type of the move selected is ‘Attack’:

Print information of the move using the function **attack\_text(pokemon1, pokemon2, attack\_move)**, where the player’s Pokemon choice is the input for **pokemon1**, the enemy trainer’s Pokemon is **pokemon2** and the move the player selected is **attack\_move**

The move is executed by the player’s Pokemon.

Print updated stats of Pokemons using the function **update\_status(pokemon1, pokemon2)**, where the player’s Pokemon is **pokemon1** and the enemy trainer’s Pokemon is **pokemon2**

* + - * 1. Else if move type of move selected is ‘Buff’:

If multiplier of opposing Pokemon’s attack is 3:

Print a statement that says that their Pokemon’s stat cannot be increased using the function **buff\_too\_much(pokemon, move)**, where the player’s Pokemon is **pokemon** and their chosen move is **move**

Else:

Adjust the attack multiplier of the player’s Pokemon using the function **buff(pokemon, move)**

* + - * 1. Else if move type of player’s selected move is ‘Debuff’:

If opposing Pokemon’s defence multiplier is equal to -3:

Print a statement that says that their Pokemon’s stat cannot be decreased **debuff\_too\_much(pokemon1, pokemon2, move)**, where the the player’s Pokemon is **pokemon1**, the enemy trainer’s Pokemon is **pokemon2** and the selected move is **move**

Else:

Adjust the trainer’s defence multiplier using the function **debuff(pokemon, move)**, where the trainer’s Pokemon is **pokemon** and the player’s selected move is **move**

Print information using the function **debuff\_text(pokemon1, pokemon2, move)**, where the player’s Pokemon is **pokemon1**, the trainer’s Pokemon is **pokemon2** and the player’s selected move is **move**

* + - * 1. Else:

assert(“Move Type error!”) to help with debugging of the game.

* + - 1. If **turn\_choice** is 2 (Player chooses to change Pokemon):
         1. Reset Saiyan’s current Pokemon’s multiplier stats using the function **reset\_pokemon\_multiplier(pokemon)**.
         2. An empty dictionary called **other\_pokemon** is created to contain all the other Pokemons that Saiyan has available left in his squad.
         3. Another empty dictionary called **other\_pokemon\_types** is created to contain the types of all the other Pokemons that Saiyan has available left in his squad.
         4. A **for**-loop is then used to check if each Pokemon in his squad has 0 health.

If his Pokemon has 0 health:

Continue the loop

else(Pokemon has more than 0 health):

Add the Pokemon into the **other\_pokemon** dictionary with the key as the index of the **for**-loop.

Add the Pokemon’s type into the **other\_pokemon\_types** dictionary with the key as the index of the **for**-loop.

* + - * 1. Check the length of the **other\_pokemon** dictionary. If the length of the dictionary is 0:

Print a statement that informs the player that Saiyan has no more Pokemons left and prompts the user to choose again and assign the input to **turn\_choice**.

* + - * 1. else:(There are pokemon left)

A **for** loop is used with **zip(other\_pokemon, other\_pokemon types)** to print the remaining Pokemon’s name and type and the index representing them.

Prompt user for input for the index of the next Pokemon which they would like to choose and assign it to a variable, **choice**.

A **while** loop is used to ensure that **choice** is a valid input.

Otherwise, prompt the player for input for the index of the next pokemon which they would like to choose again and assign it to **choice**.

**Choice** is then made an integer minus 1 to correspond to the index in the **Saiyan** dictionary.

Change Saiyan’s pokemon using the **change\_pokemon(trainer, pokemon)** function.

* + - * 1. else(neither 1 or 2 input):

Prompt user to re-enter input for **turn\_choice**.

* + - 1. else: (opponent’s turn)
         1. If opponent’s current pokemon is Miltank:

If Miltank’s health is more than 120:

If Miltank’s Attack Multiplier is 3:

If Miltank’s Defence Multiplier is 3:

Assign **move** to index for Miltank’s **Hyper Beam**

else(Miltank’s Defence Multiplier is less than 3)

Randomize the indexes for Miltank’s **Advanced Harden** and **Hyper Beam** and assign the index to **move**

Else if Miltank’s Defence Multiplier is 3:

Randomize the indexes for Miltank’s **Maternal Instinct** and **Hyper Beam** and assign the index to **move**

else(Both Miltank’s Defence and Attack Multiplier is not 3)

Randomize the indexes for Miltank’s **Maternal Instinct**, **Advanced Harden** and **Hyper Beam** and assign it to **move**

else:(Miltank’s health is less than or equal to 120)

Assign the index for Miltank’s **Milk Drink** to **move**

* + - * 1. else(opponent’s current pokemon is not Miltank):

Randomize move used by pokemon using the **ai\_choose\_move** function and assign the move index to **move**

* + - * 1. If **move** name is **Milk Drink**:

Print “Miltank used Milk Drink!”

Set Miltank’s health to 300

Print “Miltank is now at full health”

* + - * 1. Else if **move** is an Attack:

Print the text for attacking using **attack\_text(pokemon1, pokemon2, attack\_move)** where **pokemon1** is the opponent’s pokemon, **pokemon2** is Saiyan’s pokemon and **attack\_move** is the **move** that was chosen.

* + - * 1. Else if **move** is a Buff:

If opponent’s pokemon has a multiplier of 3 for the affected stat of **move**:

Print the buff was ineffective using **buff\_too\_much(pokemon, move)** function where **pokemon** is the opponent’s pokemon and **move** is the **move** chosen.

else:(multiplier of affected stat is not 3)

Increase the multiplier of the affected stat of the pokemon by using the **buff(pokemon, move)** function where **pokemon** is the opponent’s pokemon and **move** is the **move** chosen.

Update the user that the buff has taken place using **buff\_text(pokemon, move)** function where **pokemon** is the opponent’s pokemon and **move** is the **move** chosen.

* + - * 1. Else if **move** is a Debuff:

If Saiyan’s pokemon has a multiplier of -3 for the affected stat of **move**:

Print the debuff was ineffective using **debuff\_too\_much(pokemon1, pokemon2, move)** function where **pokemon1** is the opponent’s pokemon, **pokemon2** is Saiyan’s pokemon and **move** is the **move** chosen.

else:(multiplier of affected stat is not 3)

Decrease the multiplier of the affected stat of the pokemon by using the **debuff(pokemon, move)** function where **pokemon** is Saiyan’s pokemon and **move** is the **move** chosen.

Update the user that the debuff has taken place using **debuff\_text(pokemon1, pokemon2, move)** function where **pokemon1** is the opponent’s pokemon, **pokemon2** is Saiyan’s pokemon and **move** is the **move** chosen.

* + 1. Set **own\_turn** as **not own\_turn**
    2. If trainer’s pokemon fainted:
       1. Check what to do next using the function **fainting(trainer)**, inputting the trainer’s dictionary into **trainer**
       2. Switch to the player’s turn by setting boolean **own\_turn** to be True
       3. If **battle\_over** is True:
          1. Print text telling the player they won using the function **win\_text(trainer)**
          2. return True (player won the round)
    3. Else if the player’s pokemon faints:
       1. Print updated stats of the pokemons using the function **update\_status(pokemon1, pokemon2)**, where the player’s pokemon is **pokemon1** and the trainer’s pokemon is **pokemon2**
       2. Check what to do next using the function **fainting(trainer)**, inputting Saiyan’s dictionary into **trainer**
       3. Switch to the player’s turn by setting boolean **own\_turn** to be True
       4. If **battle\_over** is True:
          1. Print text telling the player they lost
          2. return False (player lost the game)
    4. Else:
       1. continue

**main():**

This is the function that runs the game. The pseudocode is as follows.

1. Initialise Pokemon’s moves dictionaries, Pokemon’s dictionaries and trainer dictionaries

* A Pokemon’s **move** dictionary contains the move’s Name, Type, Move-Type, Accuracy, Base-Damage for “Attack” move-types, Affected-Stat and Buff/Debuff Stage for “Buff/Debuff” move-types.
* A **pokemon** dictionary contains the Pokemon’s Name, Type, Level, Max-Health, Health, Attack, Defence, Attack-Multiplier, Defence-Multiplier and its Movesets.
* A **trainer** dictionary contains the character’s Name and a list of all his Pokemons.

1. Print story overview
2. Print from the dictionary **all\_pokemon** all the Pokemons the player can choose from using a **for**-loop to access the values of the **all\_pokemon** dictionary by calling the **keys**
3. Prompt the player to input an integer to choose the first Pokemon using the **input** function
4. Use the integer the player inputs to access the Pokemon selected from the **all\_pokemon** dictionary, remove it from the dictionary using the **pop** method and then add it to the list of Pokemons in the **Saiyan** dictionary using the **append** method
5. If the player’s input do not match any of the keys in the **all\_pokemon** dictionary, the player will be prompted repeatedly to input a number to select a Pokemon
6. while the number of selected Pokemons is not equal to three:
   1. Print remaining pokemons from the **all\_pokemon** dictionary
   2. Ask the player to input an integer to select their Pokemon
   3. Remove Pokemons selected from the **all\_pokemon** dictionary and add it to the list of Pokemons in the **Saiyan** dictionary
   4. While the player’s input is invalid, repeatedly prompt the player to input an integer to select a Pokemon
7. Continue printing the story overview
8. Set boolean **win\_Platon**, to be True if the player beats Philosopher Platon (Level 1) and False if the player loses, using the function **battle(trainer, Saiyan)**, where the **Philosopher\_Platon** dictionary is called in the **trainer** parameter, and will return either True or False
9. if **win\_Platon** is True:
   1. The stats of Saiyan’s Pokemons are reset using the function **reset(trainer)** with the **Saiyan** dictionary called in the **trainer** parameter, which returns the updated **Saiyan** dictionary
   2. Print story
   3. Set boolean **win\_Konfusion**, to be True if the player beats Grandmaster Konfusion (Level 2), and False if they lose, using the function **battle(trainer, Saiyan)**, where the **Grandmaster\_Konfusion** dictionary is called in the **trainer** parameter, which will return either True or False
   4. if **win\_Konfusion** is True:
      1. The stats of Saiyan’s Pokemons are reset using the function **reset(trainer)** with the **Saiyan** dictionary called in the **trainer** parameter, which returns the updated **Saiyan** dictionary
      2. Print story
      3. Set boolean **win\_christna**, to be True if the player beats Christna (Level 3) and False if they lose, using the function **battle(trainer, Saiyan)**, where the **Christna** dictionary is called in the **trainer** parameter, which will return either True or False
      4. while **win\_christna** is True:
         1. The stats of Christna’s Pokemons are reset using the function **reset(trainer)** with the **Christna** dictionary called in the **trainer** parameter, which returns the updated **Christna** dictionary
         2. The multipliers of Saiyan’s Pokemons are reset using the function **reset\_multiplier(trainer)** with the **Saiyan** dictionary called in the **trainer** parameter, which returns the updated **Saiyan** dictionary
         3. Print story
      5. If **win\_christna** is False:
         1. Print story
         2. Print Gatekeeper of Doom Christna: “GAME OVER!”
   5. else: (did not get past level 2)
      1. Print story
      2. Grandmaster Konfusion: “Restart to try again”
10. else: (did not get past level 1)
    1. Print story
    2. Philosopher Platon: “Restart to try again”
11. print(“The end.”)