Johto War Card Generator Guide

*Expand the tasks below to learn more about the tasks.*

## **Understanding the Card Generator**

### **Understanding the Files and Folder Structure**

The folder structure must be kept the way it is so that the python scripts know where to pull assets from and where to output files to. See below for the folder structure:

1. 📁card\_generator: This folder contains all Python scripts that will be called on. The main files to note here include:
   1. 📃config: This script will need to be updated in the following scenarios:
      1. New Bosses are added
      2. New Items are added that need to show on Pokémon cards
      3. You wish to change the file structure
      4. New fonts are added/changed
   2. 📃utils: This script will need to be updated in the following scenarios:
      1. The cube name is changed
      2. New fonts are to be used
   3. 📃main\_[X]: These scripts will be the scripts that are run to create the cards. They call on the 📃generate\_[X] scripts in the order necessary
   4. 📁output: This folder contains the individual output folders for each card type. Each folder may have the following structure (except for 📁Deck Objects):
      1. 📁card\_backs: This holds the images for the back of the cards.
      2. 📁card\_fronts: This holds the images for the front of the cards.
      3. 📁decks: This holds the images for the 7x10 array of cards.
      4. 📁moves: This holds the images for the moves and ability texts.
      5. 📁Deck Objects: This folder contains the completed JSON files to be imported into Tabletop Simulator.
2. 📁generator\_assets: This folder contains all images and assets to be used by the card generator.
   1. 📁archetypes: This folder contains the images for each of the archetypes. The text is added to the card via the cube, so these images do not have text.
   2. 📁biomes: This folder contains background images for each biome.
   3. 📁card\_backs: This folder contains the base image of each card back.
   4. 📁card\_bases: This folder contains the base image of each card front.
   5. 📁card\_images: This folder contains the image to be added to the top section of each non-Pokémon card.
   6. 📁climates: This folder contains background images for each climate.
   7. 📁emblems: This folder contains the icon signature for the creator.
   8. 📁encounter\_icons: This folder contains the icons for each encounter tier as well as the shiny icon.
   9. 📁evolution\_icons: This folder contains the images to be used to differentiate between Pokémon that can evolve and Pokémon that cannot.
   10. 📁fonts: This folder contains the font files used.   
       ⚠Make sure to update the 📃config and 📃utils if the fonts are changed or when adding a new font.
   11. 📁held\_item\_bases: This folder contains the images for required held items.   
       ⚠ Make sure to update the 📃config if adding new images.
   12. 📁location\_bases: This folder contains the background image for the location icon.
   13. 📁location\_icons: This folder contains images of the location icons.
   14. 📁object\_templates: This folder contains the JSON files for making cards and decks.
   15. 📁pokemon: This folder contains the images of all Pokémon to be created using the card generator. The file name is based on their national dex number followed by any different form. These files names must match the pokedex\_number in the Johto cube. These files must also have square dimensions.
   16. 📁stats\_bases: This folder contains the background image for HP and Initiative values.
   17. 📁stats\_icons: This folder contains the icons for HP and Initiative values.
   18. 📁tactic\_bases: This folder contains the card bases for the Tactics cards.
   19. 📁trainer\_card\_assets: This folder contains subfolders containing assets specifically for the trainer card generation.
       1. 📁\_backs: This folder contains the base image of each card back.
       2. 📁bronze: This folder contains the base images for the front of the Bronze trainer cards.
       3. 📁emblems: This folder contains the icon signature for the creator.
       4. 📁gold: This folder contains the base images for the front of the Gold trainer cards.
       5. 📁object\_templates: This folder contains the JSON files for making cards and decks.
       6. 📁silver: This folder contains the base images for the front of the Silver trainer cards.
       7. 📁trainers: This folder contains images of all Trainer Cards to be created using the card generator. The file name must match the trainer\_class in the Johto cube. These files must also have 2x1 dimension ratio.
       8. 📷locked\_blue: this image is the lock icon for the Moderate ability.
       9. 📷locked\_red: this image is the lock icon for the Strong ability.
   20. 📁trainer\_icons: This folder contains the icons that display for Trainer Pokémon.
   21. 📁trainers: This folder contains the images of each Trainer to be added to Trainer Pokémon. These files must also have square dimensions.
   22. 📁types: This folder contains the icons for each Pokémon type. These files must also have square dimensions.
   23. 📷frame\_base: this image is the base for the frame of the Pokémon card front.
   24. 📷move\_base: this image is the base for the move text.

### **Understanding the Johto Cube**

Most of the card edits are done by editing cells in the Johto Cube excel. Below is a breakdown of which columns are safe to edit, and which have specific requirements to be used by the generator. The following breakdown is divided by each sheet in the excel.

1. 📑pokemon: This sheet contains all information for making the Pokémon cards.
   1. Pokedex\_number: this must match the file names in 📁pokemon.
   2. Pokedex\_name: this is the text that will appear at the top of the card.
   3. Type\_1: this must match the file names in 📁types.
   4. Type\_2: this must match the file names in 📁types. Leave blank if the Pokémon only has 1 type.
   5. Hp: this matches the video game stats of the Pokémon. This was used in the initial calculations for HP.
   6. Atk: this matches the video game stats of the Pokémon. This was used in the initial calculations for Initiative.
   7. Def: this matches the video game stats of the Pokémon. This was used in the initial calculations for HP.
   8. Sp\_atk: this matches the video game stats of the Pokémon. This was used in the initial calculations for Initiative.
   9. Sp\_def: this matches the video game stats of the Pokémon. This was used in the initial calculations for HP.
   10. Spd: this matches the video game stats of the Pokémon. This was used in the initial calculations for Initiative.
   11. Classification: this is the category of the Pokémon and is used in the card’s tooltip.
   12. Biome: this must match the file names in 📁biomes.
   13. Climate: this must match the file names in 📁climates.
       1. [Climate]\_[Biome] should match the file names in 📁location\_icons. If a match is not found, it will use unknown.png for the location\_icon.
   14. \_initiative: this contains the Initiative calculated by the function:
   15. \_health: this contains the HP calculated by the function:
   16. I\_mod: this contains modifications to the calculated Initiative.
   17. H\_mod: this contains modifications to the calculated HP.
   18. Initiative: this contains the final initiative of that Pokémon.
   19. Health: this contains the final HP of that Pokémon.
   20. Total: this contains the stat total of that Pokémon.
   21. Move\_1: this contains the first learnable type of the Pokémon. It will match the first type of the Pokémon.
   22. Move\_2: this contains the second learnable type of the Pokémon. It will match the second type of the Pokémon, if any.
   23. Move\_3: this contains the third learnable type of the Pokémon.
   24. Move\_4: this contains the fourth learnable type of the Pokémon.
   25. Evolve\_into: this contains the name of the Pokémon’s evolution, if any. This must match the text found in the internal\_name column (AA). For split evolutions, the split evolutions must be split by “/”.
   26. Evolve\_cost: this contains the value each Pokémon can evolve for. For trainers, this value is hidden by placing the word “Hidden” in the field.
2. Internal\_name: this contains the internal name for the Pokémon. This must match the Evolve\_into column (Y) if a Pokémon evolves into it. It should also match the name of the models in-game. If a model is not showing for a Pokémon card, most likely, there is a mismatch between the model name and the internal\_name.
3. Description: this is the text that will appear beneath the pokedex\_name.
4. Move\_name: this must match a move name found on the 📑moves.
5. Move\_type: this pulls in the type of the move from 📑moves.
6. Move\_attack\_strength: this pulls in the attack strength of the move from 📑moves.
7. Move\_effect: this pulls in the move effect of the move from 📑moves.
8. Is\_starter: this field denotes whether a Pokémon should have the Starter Pokemon tag and starter encounter icon. A value of 1 indicates a starter. All others should be 0.
9. Is\_legendary: this field denotes whether a Pokémon should have the Legendary Encounter tag. A value of 1 indicates a legendary. All others should be 0.
10. Is\_shiny: this field denotes whether a Pokémon should have the Shiny Pokemon tag and shiny icon. A value of 1 indicates a shiny Pokémon. All others should be 0.
11. Encounter\_tier: this field denotes which encounter icon should be used.
12. Trainer: this must match the file names in 📁trainers. Leave blank if it is not a Trainer Pokémon.
    * 1. ⚠When making a new Trainer, you must also update the deck\_configurations in 📃generate\_pokemon\_decks
13. Evo\_only: this field denotes whether a Pokémon should only exist in the Evolution deck and not in an Encounter Deck. A value of 1 indicates it should only be in the Evolution deck. All others should be 0.
14. Number\_in\_deck: this field denotes whether a copy of this Pokémon should be in the evolution deck. A value of 1 indicates it will only exist in the encounter decks. A value of 2 indicates that it will exist in both the encounter decks and the Evolution Deck. The value in Evo\_only will override this field.
15. Uncapturable: this field denotes that the Pokémon cannot be caught. A value of 1 indicates it cannot be caught. All others should be 0.
16. State: this field denotes what state the Pokémon should be in. The first Pokémon in a state should have a value of 1 and increases by 1 for each other state needed. These must be in order. If the excel is out of order, these will break. Pokémon without states should have a value of 0.
17. 📑moves: This sheet contains the move info for making the Pokémon and TM cards.
    1. Move\_name: this field is the name of the move and must match the Move\_name found on the 📑pokemon.
    2. Move\_type: this must match the file names in 📁types.
       1. ⚠If adding new types/translating, you must also edit 📃generate\_pokemon\_moves and 📃generate\_tm\_moves. It is not advised to translate “blank”.
    3. Move\_tier: this field denotes which TM deck this move should show up in. Moves that should not be TMs should be labeled as “legendary” as there is no Legendary TM deck.
       1. ⚠If translating the Plasma Moves/Abilities or Shadow Moves, you must also edit 📃generate\_tm\_front.
    4. Move\_attack\_strength: this field is the number of dice rolled for the attack.
    5. Move\_effect: this field is the move’s effect. Do not use Alt+Enter to make extra lines as this will cause errors.
    6. Archetype\_1: this contains the 1st archetype of the move. This must be filled first before the other archetype columns. Leave blank if none.
    7. Archetype\_2: this contains the 2nd archetype of the move. Leave blank if none.
    8. Archetype\_3: this contains the 3rd archetype of the move. Leave blank if none.
    9. Archetype\_count: this calculates the number of archetypes a move has. This is important to calculate which archetype base to use.
    10. Number\_in\_deck: this field denotes how many copies of this move should be found in the TM decks.
    11. States: this field denotes what state the card should be in. The first card in a state should have a value of 1 and increases by 1 for each other state needed. These must be in order. If the excel is out of order, these will break. Cards without states should have a value of 0.
18. 📑abilities: This sheet contains the info for making the ability cards.
    1. Ability\_name: this field is the name of the ability.
    2. Classification: this field denotes what card type the ability is.
    3. Move\_type: this is the left type of the ability. This must match the file names in 📁types.
    4. Move\_type2: this is the right type of the ability. This must match the file names in 📁types.
    5. Move\_effect: this field is the ability’s effect. Do not use Alt+Enter to make extra lines as this will cause errors.
    6. Number\_in\_deck: this field denotes how many copies of this move should be found in the Ability deck.
    7. States: this field denotes what state the card should be in. The first card in a state should have a value of 1 and increases by 1 for each other state needed. These must be in order. If the excel is out of order, these will break. Cards without states should have a value of 0.
19. 📑tactics: This sheet contains the info for making the Battle Tactics.
    1. Tactic\_name: this field is the name of the Tactic. It will be used as part of the output name of the tactic.
    2. trainer: this must match the file names in 📁tactic\_bases.
    3. Move\_name: this is the 1st move/ability on the tactic and takes up the 2nd move slot. This must match a move name found on the 📑tactic\_moves.
    4. ability\_name: this is the 2nd move/ability on the tactic and takes up the 3rd move slot. This must match a move name found on the 📑tactic\_moves.
    5. Number\_in\_deck: this field denotes how many copies of this move should be found in the Tactics deck.
    6. States: this field denotes what state the card should be in. The first card in a state should have a value of 1 and increases by 1 for each other state needed. These must be in order. If the excel is out of order, these will break. Cards without states should have a value of 0.
20. 📑tactic\_moves: This sheet contains the move info for making the Tactic moves.
    1. Move\_name: this field is the name of the move and must match the Move\_name found on the 📑tactics.
    2. Move\_type: this must match the file names in 📁types.
    3. Move\_attack\_strength: this field is the number of dice rolled for the attack. To make a tactic ability, type “blank”.
    4. Move\_effect: this field is the move’s effect. Do not use Alt+Enter to make extra lines as this will cause errors.
    5. Archetype\_1: this contains the 1st archetype of the move. This must be filled first before the other archetype columns. Leave blank if none.
    6. Archetype\_2: this contains the 2nd archetype of the move. Leave blank if none.
    7. Archetype\_3: this contains the 3rd archetype of the move. Leave blank if none.
    8. Archetype\_count: this calculates the number of archetypes a move has. This is important to calculate which archetype base to use.
21. 📑legend: This sheet contains the info for making Legendary Quest cards.
    1. Image\_type: this must match the file names in 📁card\_bases.
    2. Card\_name: this field denotes the name of the Card in the tooltip.
    3. Pokedex\_name: this field is the name of the Legendary Quest and is written at the top of the Legendary Quest card.
    4. Image\_name: this must match the file names in 📁card\_images.
    5. Classification: this field denotes the tag the card should have.
    6. Description: this field contains the text in the card’s tooltip.
    7. Quest\_info: this field contains the text in the middle of the card.
    8. Objectives: this field contains any objectives at the lower part of the card.
    9. Next\_state: this field contains the text at the bottom fo the card.
    10. Boon: this is just a notes field to track what each boon consists of.
    11. Number\_in\_deck: this field denotes how many copies of this move should be found in the Legendary Quest deck. Though there are multiple copies of Entei, Suicune, and Raikou in game, there is more work needed to complete these cards, so I only put “1” in this column.
    12. States: this field denotes what state the card should be in. The first card in a state should have a value of 1 and increases by 1 for each other state needed. These must be in order. If the excel is out of order, these will break. Cards without states should have a value of 0.
22. 📑shrine: This sheet contains the info for making Legendary Quest cards.
    1. Image\_type: this must match the file names in 📁card\_bases.
       1. ⚠The “Warp” image type is used in the scripting. If changing this, 📃generate\_shrine must be edited.
    2. utility\_name: this field denotes the output name of the card.
    3. Internal\_name: this field is the name of the card and is written on the card itself and is the name in the tooltip.
    4. Card\_effect: this field is the card’s effect.
    5. Number\_in\_deck: this field denotes how many copies of this move should be found in the Shrine deck.
    6. States: this field denotes what state the card should be in. The first card in a state should have a value of 1 and increases by 1 for each other state needed. These must be in order. If the excel is out of order, these will break. Cards without states should have a value of 0.
23. 📑others: This sheet contains the info for making treasure cards, utility cards, fortune cards, disaster cards, and quest cards.
    1. Image\_type: this must match the file names in 📁card\_bases.
       1. ⚠The “Warp” image type is used in the scripting. If changing this, 📃generate\_utility, 📃generate\_utility\_decks, and 📃generate\_utility\_deck\_object must be edited.
       2. ⚠All image types are used in the scripting. If changing this, 📃generate\_utility\_decks and 📃generate\_utility\_deck\_object must be edited.
    2. utility\_name: this field denotes the output name of the card.
    3. card\_name: this field is the name of the card and is written on the card itself.
    4. internal\_name: this field is the name in the tooltip.
    5. Utility\_type: this must match the file names in 📁card\_bases.
       1. ⚠The “QUEST” utility type is used in the scripting. If changing this, 📃generate\_utility must be edited.
       2. ⚠The “BATTLE” and “COMPANION” utility types are used in the scripting. If changing this, 📃generate\_utility\_deck\_object must be edited.
    6. Rocket\_ignore: this field denotes whether an Influence card should have the rocket logo in the corner. A value of 1 indicates it should have the logo. All others should be 0. This is important for the Team Rocket Trainer Card effect.
    7. Classification: this field denotes the description of the card in the tooltip.
    8. Card\_effect: this field is the card’s effect.
    9. Number\_in\_deck: this field denotes how many copies of this move should be found in the decks.
    10. States: this field denotes what state the card should be in. The first card in a state should have a value of 1 and increases by 1 for each other state needed. These must be in order. If the excel is out of order, these will break. Cards without states should have a value of 0.
24. 📑trainer\_cards: This sheet contains all information for making the Trainer cards. This sheet can only hold up to 70 Trainers. If more is needed, the next 70 should go on 📑trainer\_cards\_2. If more is needed, more sheets will need to be added and more scripts created.
    1. Trainer\_class: this must match the file names in 📁trainer\_card\_assets > 📁trainers.
    2. Ability\_1\_name: this field denotes the Moderate ability name.
    3. Ability\_1\_description: this field denotes the Moderate ability text.
    4. Ability\_2\_name: this field denotes the Strong ability name.
    5. Ability\_2\_description: this field denotes the Strong ability text.

## **Installing the Card Generator**

### **Download the Johto Card Generator Folder**

The folder has been uploaded to [link]. Download and place in an accessible location.

### **Install Anaconda**

These scripts use Anaconda Prompt to run the python scripts. Download Anaconda [here](https://www.anaconda.com/docs/getting-started/anaconda/install).

### **Install Python**

These scripts use Python to run the python scripts. Download Python [here](https://www.python.org/).

### **Create Anaconda Virtual Environment**

The scripts run in an Anaconda virtual environment. We will need to create the environment that we will un the scripts in.

1. Open Anaconda Prompt on your computer.
2. Type the following into the prompt window and then press Enter.
3. That will run and install all necessary items. Once complete, you are ready to run the scripts.

## **Running the Card Generator**

### **Activate the Anaconda Virtual Environment**

To activate the space, you have two main options:

1. Search for an activated virtual environment:
   1. In your computer search bar, look for “py37”.
   2. Select the option that says “Anaconda Prompt (py37)”
2. Activate the virtual environment:
   1. Open Anaconda Prompt on your computer.
   2. Type the following into the prompt window and then press Enter.

### **Head to the Card Generator Folder**

Once in your Anaconda Virtual Environment, you need to let the program know where to look for the scripts.

1. Type “cd “ and then paste the filepath to the **📁**card\_generator. For example:
2. Press Enter.

### **Run the Scripts**

Once in your Anaconda Virtual Environment, you need to let the program know which script you’d like to run.

1. Type “python “ followed by the file name of the script you wish to run. It is recommended to run the main\_[X] scripts. For example:

### **Importing the Decks**

On the last phase of each script, it will ask you to upload these files to the Steam Cloud Manager. To do so:

1. Open Tabletop Simulator.
2. Create a room.
3. At the top of the screen, press Modding.
4. Then press Cloud Manager.
5. Create folder structures, as needed.
6. Upload the deck images found in 📁decks within each folder found in 📁output.
7. This will create an ID for each item that will look like a long link. When the python script asks for the links, copy and paste the links into the Anaconda virtual environment window and press Enter. This will repeat until all necessary links are provided.
8. After the script finishes running, it will output a JSON file to 📁Deck Objects found in 📁output.
9. Now, you will need to copy these files into your Tabletop Simulator Saved Objects. To do so, go to your 📁Documents > 📁My Games > 📁Tabletop Simulator > 📁Saves > 📁Saved Objects.
10. Once the JSON files are added in there, you can now import them into the game by going back to Tabletop Simulator.
11. At the top of the screen, press Objects.
12. Then press Saved Objects.
13. Click on the files you want to import and place them in the game.