Purpose of each file

game.py:

This is the main script that runs the game. It starts by generating and displaying multiple card decks for the player to choose from. Once the deck is selected, it sets up the player and computer (cat) as Player objects and initiates the game loop. During each turn, the user can choose to play a card, view their deck, or inspect the current stats. Each card triggers an effect that either causes damage or modifies stats. The cat (computer) also takes its turn by selecting a card based on strategic logic. This continues until either the player or the cat is defeated. This script integrates logic from all other modules and provides the game's user interface via the terminal.

deck_selection.py:

This file provides the functionality for deck selection at the beginning of the game. It displays the player's available decks (pre-generated) and their card details, prompts the player to choose one, and randomly assigns a deck to the computer opponent. It ensures the player makes a valid selection through input validation. This is the entry point into customizing gameplay and affects the strategy and outcome of the match.

make_deck.py:

This module is responsible for generating balanced, randomized card decks from a data file. It contains the Card class, which represents a card and includes logic for displaying card details. The make_deck() function reads a structured text file, parses the card information, and selects a set of cards based on type and power level constraints. Each deck includes at least one attack card and one buff/debuff to ensure gameplay variety. The goal is to create fair but unique decks for each game session.

game_menu.py:

This file implements the user interaction menu during the player's turn. It allows the user to choose whether to play a card, view their deck, or check the current player and cat stats. The menu ensures the game feels interactive and turn-based by guiding the player through their available choices and looping until a card is selected. It also uses validation to ensure correct input formats.

input_validation.py:

The input_validation.py file provides a reusable function, validate_input, that helps ensure robust and user-friendly input handling across the game. It prompts the user for input, safely converts it to the correct type, and checks whether the value is within an allowed set, if specified. This utility is used in both the deck selection screen and the in-game menu, allowing those components to remain clean and focused on gameplay logic while delegating input validation to a centralized, well-structured helper function. Its use of conditional expressions and error handling enhances both functionality and readability.

cat_cards.txt:

Text file that contains all the cards the cat could use. Formatting goes by: name of card, description of card, type of card, magnitude of card (multiplier or tuple for damage range), power level of card (how strong it's considered), and accuracy.

player_cards.txt:

Text file that contains all the cards the player could use. Formatting goes by: name of card, description of card, type of card, magnitude of card (multiplier or tuple for damage range), power level of card (how strong it's considered), and accuracy.

Running the program

In order to run our program, the following python files are required: deck_selection.py, game.py, game_menu.py, input_validation.py, and make_deck.py. An additional two text files are required as well: cat_cards.txt, player_cards.txt. To run this program from the command line, use the following commands: python game.py. Both of these commands are required. -d/--difficulty and -l/--length are optional commands that specify the difficulty and length of the game respectively. -d/--difficulty takes the strings 'easy' or 'hard' and -l/--length takes the strings 'short' or 'long.' The default values for these commands are 'easy' and 'short' respectively. Both the cat and player text files are not required as command line arguments, since the program has the files selected already.

Using the program

The program begins by displaying 3 decks and details about the cards in each. Type the number of the deck that you want to select. The program then displays the menu and its options. You can start the battle by selecting a card (option 1), or you can show your current deck (option 2) or your player's stats (option 3). If you choose option 1, the program prompts you to enter a number for one of the cards in your deck. After selecting a card, the effect of the card is applied, and the computer chooses a card and its effect is applied. Once again the menu is displayed. This continues until either the player or computer is defeated.

Attribution

Method/function	**Primary author**	**Techniques demonstrated**
computer_card_draw	Connor	List comprehensions, key function with max()
deck_selection	Connor	
apply_card_effect	Hagan	Sequence unpacking
resolve_attack	Hagan	Optional parameters
make_deck	Ben	Regular expressions
class Card's __str__	Ben	magic method
validate_input	Jadon	Conditional expressions

game_menu	Jadon	f-strings
parse_args	Dhawal	ArgumentParser
turn_history	Dhawal	With statement

Bibliography

ASCII Art Cats - asciiart.eu. (n.d.). ASCII Art Cats - Asciiart.eu.
https://www.asciiart.eu/animals/cats
We used this website to find ASCII art that is printed in the program