Massively Multiplayer Online Role Playing Dungeons and
Dragons (MMORPDND)
Manual & Programming Documentation

Developer: Antonius Torode

Version – 0.001
Latest update: August 25, 2023

This document is continuously under development.
Most Current Revision Date: August 25, 2023

Contents

This page intentionally left blank.

Introduction

Imagine having a toolkit that makes crafting an entire Dungeons and Dragons world an effortless adventure. This collection of tools and scripts has been meticulously designed to simplify the process, aiding in the creation of visually appealing HTML files and automating the linkage of files in a user-friendly wiki-style layout.

At its core, this project serves as a collaborative haven for D&D enthusiasts. Within this dynamic universe, players are invited to claim their own territories, assuming control and adding their creative essence to the ever-expanding lore. Moreover, the existing components are at the disposal of all participants, allowing for a seamless integration of established content into their unique narratives and imaginative designs.

In essence, this endeavor mirrors the essence of a massively multiplayer online role-playing experience, with a D&D twist. It's an invitation to embark on an interconnected journey, where each player becomes a co-author of a sprawling and vibrant MMORPDND universe.

Installation And Setup

In this chapter, we delve into the essential technical aspects of getting your MMORPDND project up and running. From tool installation to environment configuration, this section provides the necessary groundwork for a seamless initiation into the world of D&D universe creation.

2.1 Windows

TODO - Write this section.

2.2 Linux

2.2.1 Semi-Automated Installation with Setup Script

To expedite the installation and setup process, a convenient setup script has been provided. The provided setup script currently supports Linux environments. This script automates the installation of required dependencies and prepares your environment for the MMORPDND project. Follow these steps to get started quickly:

1. Open a terminal window.

2. Install Git if it's not already present on your system. Run the appropriate command for your operating system:

   ```
   sudo apt-get install git
   ```

3. Clone the MMORPDND repository by running the following command:

   ```
   git clone https://github.com/torodean/DnD.git
   ```

4. Navigate to the cloned repository directory:

   ```
   cd DnD/
   ```

5. Make the setup script executable by running:

   ```
   chmod +x mmorpdnd-setup.sh
   ```

6. Execute the script with administrative privileges to initiate the installation process:

   ```
   sudo ./mmorpdnd-setup.sh
   ```

The script will automatically install the necessary dependencies.

7. Once the script completes, your system will be set up with all required components for the MMORPDND project. If there are any dependencies that did not install via the script, you may need to follow the manual installation guidelines in the below section.

By utilizing Git and the setup script from the cloned repository, you can significantly streamline the installation and setup process, ensuring a swift start to your MMORPDND adventure.

2.2.2 Manual Installation

Before using the MMORPDND project, it's crucial to ensure that your system meets the necessary requirements. The following prerequisites must be satisfied to facilitate a smooth installation and setup process:

- **Git:** Install Git if it's not already present on your system. Run the appropriate command for your operating system:

  ```
  sudo apt-get install git
  ```

- **:** Clone the MMORPDND repository by running the following command:

  ```
  git clone https://github.com/torodean/DnD.git
  ```

- **Python 3:** Ensure that Python 3 is installed on your system. If it's not already installed, you can use the following commands to install it:

  ```
  sudo apt-get update
  sudo apt-get install python3
  ```

- **Python Package Installer (pip):** Install pip, the package installer for Python, using the following command:

  ```
  sudo apt-get install python3-pip
  ```

- **Python Tkinter:** This package is used for creating graphical user interfaces in Python. Install it with:

  ```
  sudo apt-get install python3-tk
  ```

- **Python Libraries:** The MMORPDND project relies on the following Python libraries:

  - `bs4`, `BeautifulSoup`: These libraries are used for web scraping and HTML parsing. Install them with:

    ```
    pip3 install bs4
    pip3 install BeautifulSoup
    ```

  - `regex`: This library provides advanced regular expression capabilities. Install it with:

    ```
    pip3 install regex
    ```

- cssbeautifier: This library is used for formatting CSS files. Install it with:

  `pip3 install cssbeautifier`

- tqdm: This library provides a fast, extensible progress bar. Install it with:

  `pip3 install tqdm`

To ensure a streamlined installation and setup experience, make sure these prerequisites are met before proceeding with the MMORPDND project.

Tools And Scripts

3.1   mmorpdnd.py

3.1.1   Purpose

This script serves as the backbone for an array of automatic linking and logistical features within MMORPDND. These functionalities typically operate on all files, encompassing crucial elements such as the generation of directories, HTML index files at the directory summits, establishment of uniform headers and navigation for all HTML documents, seamless interlinking between files, refinement of both HTML and CSS code, and an array of additional capabilities. When ran, this script will perform the following actions in the following order:

1. Create Directory Structures if not already existing.

2. Create index files for each directory.

3. Update each index file to link to all files and images within the directory.

4. Update the headers of all html files to the template html header.

5. Update the navigation of all html files to the template navigation html.

6. Search and hyperlink all words found in all html files to the appropriate html file whose name matches the words found.

7. Beautify the code.

3.1.2   Use

The mmorpdnd.py script can be ran as a gui window or using the command line options. The two main features are 'test' and 'update'. The 'test' feature will create fake files and then perform the update command on them. This is useful for testing if the features and newly added features are working properly. The 'update' feature will perform a combination of tasks which updates all appropriate files to do the steps mentioned in the list above.

GUI

The GUI window is somewhat self explanatory. The two main features are represented by red buttons - the 'test' and 'update' features. There are also other buttons to simple perform and of the individual tasks using the appropriately associated button.

Figure 3.1: The MMORPDND Main GUI

Command Line

The MMORPDND application supports some command line features.

```
usage: mmorpdnd.py [-h] [-t] [-u]

MMORPDND Tools and apps.

optional arguments:
    -h, --help      show this help message and exit
    -t, --test      Runs the test-all feature then exit.
    -u, --update    Runs the update-all feature then exit.
```

3.1.3 Programming Methods and Variables

```
class MMORPDND_VARS

1  """
2  This is a class for storing variables used by MMORPDND* classes.
3
4  Methods:
5      __init__(self)
6
```

```
7   Variables:
8       # Define the directory structure
9       self.directory_structure = { ... }
10
11      # Define the number of HTML files to create in each subdirectory
12      self.num_dummy_files_per_subdir = 3  # Stores all files created so
            far.
13      self.all_index_files = []
14
15      # Define the root directory
16      self.root_dir = os.getcwd()
17      self.script_dir = os.path.dirname(os.path.abspath(__file__))
18
19      # Directories to exclude when processing.
20      self.directories_to_exclude = ["templates", "css", ".git", ".idea
            ", ".github", "scripts", "docs"]
21
22      # Define the regular expression to match the header section
23      self.header_regex = re.compile(r"<head>.*?</head>", re.DOTALL)
24
25      # Define the regular expression to match the header section
26      self.title_regex = re.compile(r"<title>.*?</title>", re.DOTALL)
27
28      # Define the template file paths
29      self.header_template_file = "templates/headerTemplate.html"
30      self.nav_template_file = "templates/navTemplate.html"
31      self.css_path = "css/mmorpdnd.css"
32  """
```

global_vars = MMORPDND_VARS()

```
1   """
2   Define a global variable containing the declared vars. Use this so
        they are all only defined once and can be updated/stored
        throughout the applications lifetime.
3   """
```

is_image_file(file_name)

```
1   """
2       Checks if a file name is an image file based on its extension.
3
4   Args:
5       file_name (str): The name of the file to check.
6
7   Returns:
8       bool: True if the file name has an image extension, False
            otherwise.
9
10  Example:
11      >>> is_image_file('myphoto.jpg')
12      True
13      >>> is_image_file('document.pdf')
14      False
15  """
```

8

get_relative_path(from_file, to_file)

```
1  """
2  Returns the relative path from one file to another.
3
4  Args:
5      from_file (str): The path of the source file.
6      to_file (str): The path of the target file.
7
8  Returns:
9      str: The relative path from the source file to the target file.
10
11 Raises:
12     None.
13
14 This method takes two file paths, 'from_file' and 'to_file', and
       calculates the relative path from 'from_file'
15 to 'to_file'. The relative path represents the path that, when
       followed from 'from_file', leads to 'to_file'.
16
17 Example:
18     Assuming from_file = '/path/to/source/file.html' and to_file = '/
           path/to/target/image.jpg',
19     the method will return '../../target/image.jpg' as the relative
           path.
20
21 Note:
22     The method uses the 'os.path.relpath()' function to calculate the
           relative path.
23
24 Example usage:
25     get_relative_path('/path/to/source/file.html', '/path/to/target/
           image.jpg')
26     print(relative_path)  # Output: '../../target/image.jpg'
27 """
```

create_dummy_html_files(directory=global_vars.root_dir)

```
1  """
2  Creates dummy HTML files in all directories and subdirectories for
       testing purposes.
3
4  Args:
5      directory (str): The directory path to start creating dummy HTML
           files from. Defaults to global_vars.root_dir.
6
7  Returns:
8      None.
9
10 Raises:
11     None.
12
13 This method recursively walks through the directory structure,
       creates an index.html file in each directory,
14 and creates additional HTML files with random links in each
       subdirectory.
15
```

```
16  The method performs the following steps:
17      1. Creates an index.html file in the specified directory with a
           basic HTML structure.
18      2. Recursively walks through the directory structure using 'os.
           walk'.
19      3. For each subdirectory, excluding any directories listed in '
           global_vars.directories_to_exclude':
20          - Creates an index.html file in the subdirectory with a basic
             HTML structure.
21          - Generates a specified number of dummy HTML files in the
             subdirectory, each containing a random link to another dummy
              file.
22          - Prints a message indicating the successful creation of each
             HTML file.
23      4. Creates additional dummy HTML files in the script directory (
           specified by the 'directory' argument), each containing a
           random link to another dummy file.
24      5. Prints a message indicating the successful creation of all HTML
            files.
25
26  Note:
27      The content of the generated HTML files consists of a basic HTML
           structure with a header and body.
28  Each dummy file includes a link to two randomly chosen dummy files,
       facilitating testing scenarios.
29
30  Example usage:
31      create_dummy_html_files()
32  """
```

alphabetize_links(list_of_links)

```
1   """
2   Alphabetizes the items in a list of links.
3
4   Args:
5       list_of_links (str): A multiline string representing a list of
           links in the format
6       "<li><a href="url">link_text</a></li>". Each link should be on a
           separate line.
7
8   Returns:
9       str: A multiline string representing the alphabetized list of
           links.
10
11  Example:
12      links = '''<li><a href="valen_shadowborn.html">valen_shadowborn</a
           ></li>
13      <li><a href="kaelar_stormcaller.html">kaelar_stormcaller</a></li>
14      <li><a href="thorne_ironfist.html">thorne_ironfist</a></li>
15      <li><a href="foobar.html">foobar</a></li>
16      <li><a href="aria_thistlewood.html">aria_thistlewood</a></li>
17      <li><a href="stoneshaper_golem.html">stoneshaper_golem</a></li>
18      <li><a href="elara_nightshade.html">elara_nightshade</a></li>'''
19
20      sorted_list = alphabetize_links(links)
21
```

```
22      print(sorted_list)
23  """
```

class MMORPDND

```
1   """
2   A class for all the main MMORPDND features.
3
4   Methods:
5       __init__(self)
6       create_directories(self, path: str, structure: dict) -> None
7       create_index_files(self, directory=global_vars.root_dir)
8       move_dir_items_to_end(self, string)
9       move_img_items_to_end(self, string)
10      update_index_files(self)
11      update_headers(self, directory=global_vars.root_dir)
12      update_navigation(self, directory=global_vars.root_dir)
13      beautify_files(self, directory=global_vars.root_dir)
14      find_all_html_files(self, directory=global_vars.root_dir)
15      update_html_links(self, directory=global_vars.root_dir)
16  """
```

MMORPDND.__init__(self)

```
1   """
2   Initialization method.
3   """
```

MMORPDND.create_directories(self, path: str, structure: dict) -¿ None

```
1   """
2   Recursively creates directories in the given path according to the
        structure specified in the dictionary.
3
4   Args:
5       path (str): The root path where directories will be created.
6       structure (dict): A dictionary representing the structure of the
            directories to be created.
7
8   Returns:
9       None
10  """
```

MMORPDND.create_index_files(self, directory=global_vars.root_dir)

```
1   """
2   Creates index files for all subdirectories within the specified
        directory.
3
4   Args:
5       directory (str): The directory path to start creating index files
            from. Defaults to global_vars.root_dir.
6
7   Returns:
8       None.
```

```
 9
10  Raises:
11      None.
12
13  This method traverses through all subdirectories and files starting
        from the specified directory and creates an
14  index file named "index.html" in each directory that doesn't already
        have one.
15
16  The method performs the following steps:
17      1. Loop through all directories and files using 'os.walk' starting
            from the specified directory.
18      2. Check if an index file named "index.html" already exists in the
            current directory. If so, skip that directory.
19      3. Check if the current directory matches any of the excluded
            directories defined in 'global_vars.directories_to_exclude'.
20      If so, skip that directory.
21      4. Create an index.html file in the current directory.
22      5. Write the HTML content to the index.html file, including the
            directory name in the title and header.
23      6. Print a message indicating the creation of the index file.
24
25  Note:
26      The index.html file created contains a basic HTML structure with
            the title and header set to "Index of [directory_name]".
27
28  Example usage:
29      create_index_files()
30  """
```

MMORPDND.move_dir_items_to_end(self, string)

```
 1  """
 2  Moves directory items (lines containing "/index.html") to the end of
        the input string.
 3
 4  This method takes a multi-line string as input and separates lines
        that contain "/index.html"
 5  (directory items) from other lines (non-directory items). It then
        rearranges the lines by moving
 6  the directory items to the end while maintaining the order of non-
        directory items.
 7
 8  Args:
 9      string (str): The input multi-line string to be processed.
10
11  Returns:
12      str: A modified string with directory items moved to the end.
13
14  Example:
15      input_string = "Line 1\n/index.html\nLine 2\nLine 3\n/index.html"
16      result = move_dir_items_to_end(input_string)
17      # result will be "Line 1\nLine 2\nLine 3\n/index.html\n/index.html
            "
18  """
```

MMORPDND.move_img_items_to_end(self, string)

```
"""
Moves items containing "img/" to the end of the string while
    preserving their original order.

Args:
    string (str): A string containing items in the format '<li><a href
        ="url">link_text </a></li>'.

Returns:
    str: A new string with items containing "img/" moved to the end
        while preserving their original order.

Example:
    >>> string = '''<li><a href="valen_shadowborn.html">
        valen_shadowborn </a></li>
    ...             <li><a href="kaelar_stormcaller.html">
        kaelar_stormcaller </a></li>
    ...             <li><a href="thorne_ironfist.html">thorne_ironfist
        </a></li>
    ...             <li><a href="foobar.html">img/foobar </a></li>
    ...             <li><a href="aria_thistlewood.html">img/
        aria_thistlewood </a></li>
    ...             <li><a href="stoneshaper_golem.html">
        stoneshaper_golem </a></li>
    ...             <li><a href="elara_nightshade.html">
        elara_nightshade </a></li>'''
    >>> new_string = move_img_items_to_end(string)
    >>> print(new_string)
    <li><a href="valen_shadowborn.html">valen_shadowborn </a></li>
    <li><a href="kaelar_stormcaller.html">kaelar_stormcaller </a></li>
    <li><a href="thorne_ironfist.html">thorne_ironfist </a></li>
    <li><a href="aria_thistlewood.html">img/aria_thistlewood </a></li>
    <li><a href="stoneshaper_golem.html">stoneshaper_golem </a></li>
    <li><a href="elara_nightshade.html">elara_nightshade </a></li>
    <li><a href="foobar.html">img/foobar </a></li>
"""
```

MMORPDND.update_index_files(self)

```
"""
Updates all index files in the directory and subdirectories to
    include links to other files in the same directory.

Returns:
    None.

Raises:
    None.

This method performs the following steps:
    1. Prints a message indicating that index files are being updated.
    2. Retrieves a list of all HTML index files in the current
        directory and subdirectories , excluding any directories listed
        in 'global_vars.directories_to_exclude '.
    3. For each index file found:
```

```
14      - Reads the file data.
15      - Identifies the HTML files present in the same directory as
           the current index file.
16      - If the index links div section does not exist in the file, it
           adds the div section just before the closing </body> tag.
17      - Updates the index links by generating HTML code for each HTML
           file in the directory (excluding the index.html file) and
           appending it to the index links div.
18      - Replaces the old index links section in the file with the
           updated index links div.
19      - Writes the updated file data back to the file.
20      - Prints a message indicating that the index file has been
           updated.
21    4. Prints a message indicating that all index.html files have been
         updated.
22
23  Note: The method relies on regular expressions for searching and
      updating the index links section in each index file.
24
25  Example usage:
26    update_index_files()
27  """
```

MMORPDND.update_headers(self, directory=global_vars.root_dir)

```
1   """
2   Updates the headers of HTML files in the specified directory and its
       subdirectories to match a predefined template.
3
4   Args:
5       directory (str): The directory path to update the HTML files in.
           Defaults to global_vars.root_dir.
6
7   Returns:
8       None.
9
10  Raises:
11      None.
12
13  This method performs the following steps:
14    1. Loop through all HTML files in the specified directory and its
         subdirectories.
15    2. For each HTML file found that does not contain "Template" in
         its filename:
16      - Read the contents of the HTML file.
17      - Check if the file is in the list of directories to exclude.
18      - Read the contents of the header template file.
19      - Replace the header section in the HTML file with the contents
           of the template.
20      - Generate a title based on the filename and replace the title
           section in the HTML file.
21      - Determine the relative path to the CSS file.
22      - Calculate the number of subdirectories between the HTML file
           and the CSS file.
23      - Create the correct link path for the CSS file.
24      - Replace the placeholder in the HTML file with the link to the
           CSS file.
```

14

```
25      - Overwrite the HTML file with the updated contents.
26      - Print a progress update indicating the file that has been
            updated.
27   3. Print a message indicating that the headers and CSS have been
        updated in all relevant HTML files.
28
29 Note:
30   The method relies on regular expressions for pattern matching and
        modification.
31
32 Example usage:
33   update_headers()
34 """
```

MMORPDND.update_navigation(self, directory=global_vars.root_dir)

```
1  """
2  Updates the navigation block of HTML files in the specified directory
       and its subdirectories to match a predefined template.
3
4  Args:
5     directory (str): The directory path to update the HTML files in.
          Defaults to global_vars.root_dir.
6
7  Returns:
8     None.
9
10 Raises:
11    None.
12
13 This method performs the following steps:
14    1. Loop through all HTML files in the specified directory and its
          subdirectories.
15    2. For each HTML file found that does not contain "Template" in
          its filename:
16       - Read the contents of the HTML file.
17       - Check if the file is in the list of directories to exclude.
18       - Print a progress message indicating the file being processed.
19       - Read the contents of the navigation template file.
20       - Find the navigation block in the original HTML file using a
             regular expression.
21       - If a navigation block is found:
22       - Replace the navigation block in the HTML file with the
             contents of the template.
23       - Write the modified HTML back to the file.
24       - Print a message indicating that the navigation block has been
             replaced.
25       - If no navigation block is found:
26       - Print a message indicating that the navigation block was not
             found.
27       - Insert the navigation contents at the start of the body tag
             in the HTML file.
28       - Overwrite the file with the updated contents.
29       - Print a message indicating that the navigation contents have
             been inserted.
30
31 Note:
```

```
32      The method relies on regular expressions for pattern matching and
            modification.
33
34  Example usage:
35      update_navigation()
36  """
```

MMORPDND.beautify_files(self, directory=global_vars.root_dir)

```
1   """
2   Beautifies HTML and CSS files in the specified directory and its
        subdirectories.
3
4   Args:
5       directory (str): The directory path to beautify the files in.
            Defaults to global_vars.root_dir.
6
7   Returns:
8       None.
9
10  Raises:
11      None.
12
13  This method performs the following steps:
14      1. Modifies the directories_to_exclude list to include template
            and CSS files.
15      2. Loops through all files and subdirectories in the specified
            directory.
16      3. For each HTML or CSS file found (excluding those in the
            modified directories_to_exclude list):
17          - Constructs the file path.
18          - Checks if the file is an HTML or CSS file, and skips it if
                not.
19          - Reads the contents of the file.
20          - If the file is an HTML file:
21          - Uses BeautifulSoup to parse the HTML and prettify it.
22          - If the file is a CSS file:
23          - Uses cssbeautifier to prettify the CSS code.
24          - Writes the prettified code back to the file.
25          - Prints a message indicating that the file has been prettified
                .
26
27  Note:
28      The method relies on BeautifulSoup for HTML parsing and
            prettifying, and cssbeautifier for CSS prettifying.
29
30  Example usage:
31      navigator = Navigator()
32      navigator.beautify_files()
33  """
```

MMORPDND.find_all_html_files(self, directory=global_vars.root_dir)

```
1   """
2   Finds all HTML files (not index.html files) in a directory and its
        subdirectories.
```

```
3
4  Args:
5      directory: The directory to search. Defaults to the current
           directory.
6
7  Returns:
8      A list of dictionaries containing the name (without extension),
           name (with extension), and full path
9  of each HTML file found.
10 """
```

MMORPDND.update_html_links(self, directory=global_vars.root_dir)

```
1  """
2  Update the links in the various HTML files to link to the appropriate
       file.
3
4  Args:
5      directory (str): The directory to search for HTML files. Defaults
           to global_vars.root_dir.
6
7  Returns:
8      None
9  """
```

class MMORPDND_GUI

```
1  """
2  Class to store GUI functions and operations. The majority of the
       methods in this class simply connect the UI elements to the
       associated MMORPDND class methods.
3
4  Methods:
5      __init__(self)
6      run(self)
7      test_all(self)
8      update_all(self)
9      create_directories(self)
10     create_dummy_html_files(self)
11     create_index_files(self)
12     update_index_links(self)
13     update_headers(self)
14     update_navigation(self)
15     beautify_files(self)
16     update_html_links(self)
17 """
```

main()

```
1  """
2  Main method to start the gui and take optional arguments/parameters
       for running via command line.
3  """
```

3.2 templates/creator.py

3.2.1 Purpose

The creator tool is a practical solution for converting basic template text files into functional HTML pages. It takes care of the technical aspects by automatically creating HTML files and filling in missing details, like character stats or other content. The tool understands the structure of your template files, recognizes placeholders, and replaces them with accurate data. Whether you're building character profiles, story summaries, or any content with consistent formatting, this tool ensures your HTML documents are correctly formatted and ready for use. It simplifies the process, letting you focus on content creation while it handles the conversion from templates to HTML.

3.2.2 Use

The creator tool is a powerful tool that contains many subtle features. It is important to understand these subtleties before using the tool and therefore I encourage the reader to read this enture section before use.

GUI
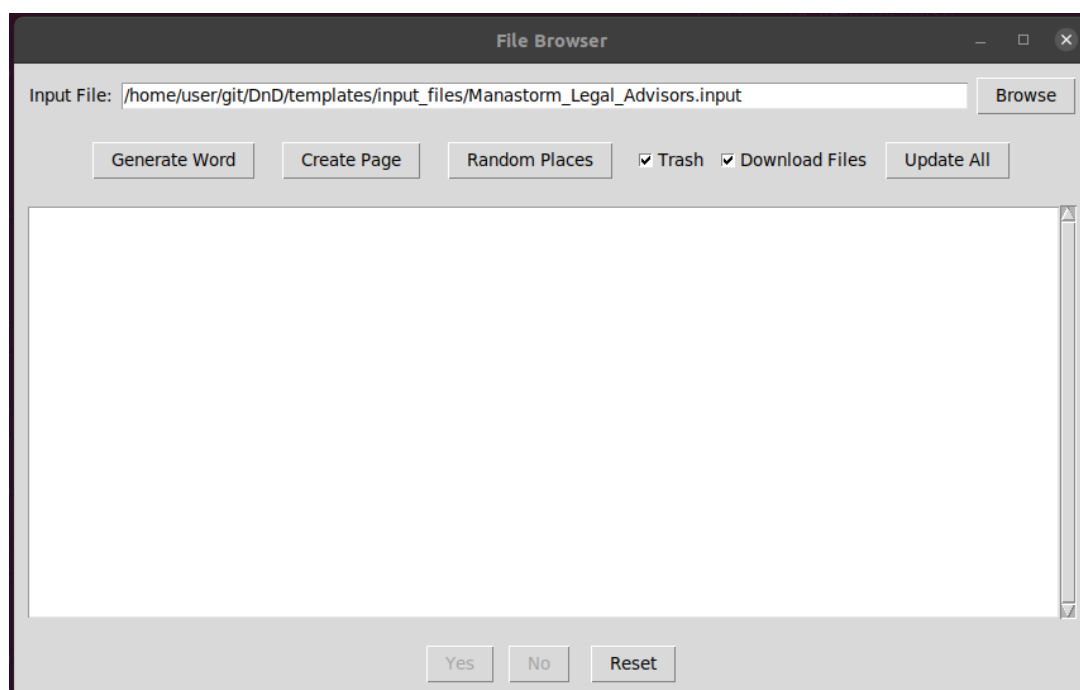
Figure 3.2: The MMORPDND Main GUI

SECTION IN PROGRESS!

Input Files (.input)

The input files are essentially just text files with the file extension '.input'. The main feature of the creator.py tool is to read in these input files, parse them, and generate the appropriate html files from them. These input files must follow a strict formatting but offer a few various helpful features. Each line of the input file represents a 'section' of the html file that will be generated. Each line follows the following format:

```
Heading Text[type]=Information text
```

The "Heading Text" is what will appear as the header for that html section. The "type" value is what determines the properties of this section and how it is interpreted by the creator tool. The "Information text" is the actual information for that section. The only exception to this rule is the first line which defines the "folder" that the html file generated from the input file will be placed into. This "folder" line is special in that it has no type and the creator tool will recognize the folder keyword and store this information separately. This folder line must be formatted as follows:

```
folder=path/to/where/I/want/my/generated/file
```

The various types determine the formatting and decoding of the input file, and each type has small subtleties. The various types are

1. dnd-image: This type is used to display an image or multiple images.

2. dnd-list: This is used to display a list of items.

3. dnd-info: This is used for general information and sections. It also supports listing information between paragraphs.

4. dnd-music: This is a section for storing various music for the regions.

dnd-image type

SECTION IN PROGRESS!

dnd-list type

SECTION IN PROGRESS!

dnd-info type

SECTION IN PROGRESS!

dnd-music type

SECTION IN PROGRESS!

Example Input File

SECTION IN PROGRESS!

Character Files (.char)

SECTION IN PROGRESS!

The character files are text files with the ".char" extension. These are a special type of input file that generates an html page specific for DnD character information. This was actually designed before the more general input file formatting outlined in the sections above and therefore lacks some of the versatility and extra features that can be included in the input files. Nevertheless, the character format creates an html page that should more than suffice for any npc characters and has many of its own unique features and capabilities. The .char file contains lines that define the various elements of a character. These lines to not currently support the types that are supported with the input files. Each line has a specific output format in the generated html file based on a template and the information itself. Some lines in the file are required while others are optional. The required lines must be present for the character

file generated to be completed, while the optional lines will automatically be randomized and generated in the event that they are missing. Eahc line of the character file must be formatting in the following way:

```
Keyword=Value
```

The "Keyword" is used by the creator tool to determine which type of information is being input, and the "Value" is the value of the information. For example "level = 3" would let the creator know that the character level is 3, and it will use that information appropriately.

Required Fields

SECTION IN PROGRESS!

Optional Fields

SECTION IN PROGRESS!

Example Character File

SECTION IN PROGRESS!

```
folder = characters/non−player
name = Lyra Nightshade
level = 3
ac = 15
hp = 75
size = Medium
type = Humanoid
alignment = Lawful Neutral
speed = 30 feet
resistances = None
immunities = None
senses = Passive Perception 14, Darkvision 60 ft.
languages = Common, Elvish
image = lyra_nightshade.jpg
race = Half−Elf
class = Ranger
background = Outlander
strength = 14
dexterity = 18
constitution = 13
intelligence = 12
wisdom = 16
charisma = 10
abilities = Favored Enemy, Natural Explorer
equipment = Longbow, short sword, leather armor, explorer's pack, and a druidic
    focus − a carved wooden token from the heart of the Whispering Woods
proficiencies = Survival, Perception, Nature
information = Lyra Nightshade is a level 3 Half−Elf ranger.
notes = Lyra's friends call her Ly ("Lee").
```

List Files (.list)

Command Line

The creator application does NOT currently support command line features.

3.2.3 Programming Methods and Variables

get_youtube_video_name(url)

```python
"""
Retrieves the title of a YouTube video based on the provided URL.

Args:
    url (str): The URL of the YouTube video.

Returns:
    str or None: The title of the YouTube video if it can be
        retrieved successfully,
                None if there was an error.

Raises:
    None

Example:
    >>> url = "https://www.youtube.com/watch?v=dQw4w9WgXcQ"
    >>> title = get_youtube_video_name(url)
    >>> print(title)
    "Rick Astley - Never Gonna Give You Up (Official Music Video)"
"""
```

find_longest_and_shortest(words)

```python
"""
Find the lengths of the longest and shortest words in a given list.

Args:
    words (list): A list of words.

Returns:
    tuple: A tuple containing the lengths of the longest and shortest
        words.

Raises:
    ValueError: If the input list is empty.

Examples:
    >>> word_list = ["one", "two", "three", "four", "five"]
    >>> longest_length, shortest_length = find_longest_and_shortest(
        word_list)
    >>> print("Longest word length:", longest_length)
    >>> print("Shortest word length:", shortest_length)
    Longest word length: 5
    Shortest word length: 3
"""
```

remove_numbers_at_start(string)

```python
"""
Remove numbers at the start of a string.

Args:
```

```
5       string (str): The input string.
6
7   Returns:
8       str: The string with numbers removed from the start.
9   """
```

append_to_file(file_path, string_to_append)

```
1   """
2   Append a string to a file.
3
4   Parameters:
5       file_path: The path to the file to append to.
6       string_to_append: The string to append to the file.
7   """
```

read_lines_from_file(file_name)

```
1   """
2   Reads lines from a file and returns them as a list.
3
4   Parameters:
5       file_name (str): The name of the file to read.
6
7   Returns:
8       A list of strings, where each string represents a line from the
            file.
9       Any leading and trailing whitespace is stripped from each line.
10
11  Raises:
12      FileNotFoundError: If the specified file does not exist.
13      PermissionError: If the specified file cannot be opened due to
            insufficient permissions.
14  """
```

calculate_hp(class_type: str, level: int, constitution: int) -> int

```
1   """
2   Calculate the hit points (hp) of a Dungeons & Dragons (DnD) 5th
        edition character
3   based on their class, level, and constitution modifier.
4
5   Args:
6       class_type (str): The character's class (e.g. 'fighter', 'wizard
            ', 'rogue').
7       level (int): The character's level, between 1 and 20.
8       constitution (int): The character's constitution score, between 1
            and 30.
9
10  Returns:
11      int: The character's hit points, based on their class and level,
            modified by their
12          constitution modifier.
13
14  Raises:
15      ValueError: If the given class_type is not recognized.
```

22

```
16    """
```

calculate_proficiency_bonus(level)

```
1    """
2    Calculate the proficiency bonus based on character level.
3
4    Parameters:
5        level (int): The character's level.
6
7    Returns:
8        int: The character's proficiency bonus.
9    """
```

roll_4d6_drop_lowest()

```
1    """
2    Rolls 4d6 and returns the sum of the highest 3 dice.
3    Returns:
4        int: The sum of the highest 3 dice.
5    """
```

get_stat_priority(character_class)

```
1    """
2    Returns a list of attributes ordered by the stat priority for a given
        class.
3    Args:
4        character_class (str): The class to get the stat priority for.
5    Returns:
6        list: The list of attributes ordered by the stat priority.
7    """
```

adjust_stats_for_level(assigned_stats, level)

```
1    """
2    Adjusts the stats for a character based on their level.
3
4    Args:
5        assigned_stats (dict): A dictionary representing the character's
            current stats, where keys are
6          the stat names and values are the corresponding stat values.
7        level (int): The level of the character.
8
9    Returns:
10        dict: A dictionary representing the adjusted stats based on the
            character's level.
11
12    Raises:
13        None.
14
15    This method takes the current stats of a character, represented by
        the 'assigned_stats' dictionary,
16    and adjusts the stats based on the character's level. The adjusted
        stats are returned as a new dictionary.
```

```
17
18  The adjustment of stats is determined by the character's level:
19   - For levels below 4, no adjustments are made, and the original
         stats are returned.
20   - For levels 4 to 7, 2 bonus attribute points are awarded.
21   - For levels 8 to 11, 4 bonus attribute points are awarded.
22   - For levels 12 to 15, 6 bonus attribute points are awarded.
23   - For levels 16 to 18, 8 bonus attribute points are awarded.
24   - For levels 19 and above, 10 bonus attribute points are awarded.
25
26  The method prints the awarded bonus points and the original and
       updated attributes for informational purposes.
27
28  Note: The stats dictionary is assumed to have numeric values for each
         stat.
29
30  Example usage:
31      assigned_stats = {'strength': 10, 'dexterity': 12, 'intelligence
           ': 14}
32      adjusted_stats = adjust_stats_for_level(assigned_stats, 8)
33  """
```

generate_character_stats(character_class, level=1)

```
1   """
2   Generates a list of six stats for a character based on their class
        and level.
3   Args:
4       character_class (str): The character's class.
5   Returns:
6       list: The list of six stats.
7   """
```

calculate_modifier(attribute_value)

```
1   """
2   Calculate the DnD attribute modifier based on the value of the
        attribute.
3
4   Args:
5       attribute_value (int): The value of the attribute.
6
7   Returns:
8       int: The modifier value for the attribute.
9
10  Example:
11      >>> calculate_modifier(15)
12      2
13  """
```

copy_file_to_directory(file_path, directory_path)

```
1   """
2   Copy a file to a directory.
3
4   Args:
```

```
5       file_path (str): The path to the file to copy.
6       directory_path (str): The path to the directory to copy the file
            to.
7
8   Raises:
9       ValueError: If the file or directory doesn't exist.
10
11  Returns:
12      None
13  """
```

move_file_to_directory(file_path, directory_path)

```
1   """
2   Move a file to a directory.
3
4   Args:
5       file_path (str): The path to the file to move.
6       directory_path (str): The path to the directory to move the file
            to.
7
8   Raises:
9       ValueError: If the file or directory doesn't exist.
10
11  Returns:
12      None
13  """
```

print_prob_matrix(prob_matrix)

```
1   """
2   Prints a probability matrix to the console in JSON format.
3
4   Parameters:
5       prob_matrix (dict): A dictionary representing the probability
            matrix,
6       where each key is an input character and the corresponding value
            is a dictionary
7       of output characters and their probabilities.
8   """
```

generate_prob_matrix(words)

```
1   """
2   Generates a probability matrix based on a list of words.
3
4   Parameters:
5       words (list of str): A list of words to use in generating the
            probability matrix.
6
7   Returns:
8       A dictionary representing the probability matrix, where each key
            is an input character
9       and the corresponding value is a dictionary of output characters
            and their probabilities.
10
```

```
11   Example:
12       >>> words = ["cat", "dog", "cut", "cog", "cot", "caught"]
13       >>> prob_matrix = generate_prob_matrix(words)
14       >>> prob_matrix
15   {
16       'c': {'a': 0.4, 'u': 0.2, 'o': 0.4},
17       'a': {'t': 0.5, 'u': 0.5},
18       't': {},
19       'd': {'o': 1.0},
20       'o': {'g': 0.6666666666666666, 't': 0.3333333333333333},
21       'g': {'h': 1.0},
22       'u': {'t': 0.5, 'g': 0.5},
23       'h': {'t': 1.0}
24   }
25
26   Note that the probabilities for each output character are normalized
         so that they sum to 1.0.
27   """
```

generate_word(prob_matrix, min_length=4, max_length=10)

```
1    """
2    Generate a random word using a probability matrix.
3
4    Parameters:
5        prob_matrix (dict): A dictionary representing the probability
             matrix for generating words.
6        min_length (int): The minimum length of the generated word.
             Default value is 4.
7        max_length (int): The maximum length of the generated word.
             Default value is 10.
8
9    Returns:
10       A string representing the generated word.
11
12   Algorithm:
13       1. Choose a random length between min_length and max_length.
14       2. Initialize the word with a random input character.
15       3. Generate the next characters based on the probabilities in the
             matrix.
16           a. Get the output probabilities for the current input
                character.
17           b. Create a list of possible next characters based on their
                probabilities.
18           c. Choose a random next character from the list.
19           d. Update the word and the input character for the next
                iteration.
20       4. Return the generated word.
21   """
```

move_file(source_file_path, destination_folder_path)

```
1    """
2    Move a file from the source path to the destination folder.
3
4    Args:
```

```
5       source_file_path (str): The path to the file to be moved.
6       destination_folder_path (str): The path to the destination folder
            .
7
8   Returns:
9       None
10  """
```

class Variables

```
1   """
2   A class to store app wide variables.
3
4   Variables:
5       self.current_prob_matrix = None
6       self.current_file = ""
7       self.current_list = []
8       self.output_file_folder = ""
9       self.character_template_file = "characterTemplate.html"
10
11      # Define directories to exclude
12      self.directories_to_exclude = ["templates", "css", ".git", ".idea
            ", ".github", "scripts", "docs"]
13
14      # Define the root directory
15      self.root_dir = os.getcwd()
16      self.trash_dir = self.root_dir + "/trash"
17
18  Methods:
19      __init__(self)
20      trash_file(self, file)
21      reset(self)
22  """
```

Variables.__init__(self)

```
1   """
2   Initializes the Variables class by setting initial variable status'.
3   """
```

Variables.trash_file(self, file)

```
1   """
2   Move a file to the trash folder.
3
4   Parameters:
5       file: The file to move.
6
7   Returns:
8       None
9   """
```

Variables.reset(self)

```
1   """
2   Reset the state of some objects to their initial values.
3
4   This method resets the state of the object by clearing the values of
        the current_file, current_list,
5   and current_prob_matrix attributes. After calling this method, the
        object is restored to its initial
6   state, ready for new data to be processed and stored.
7
8   Example:
9       my_object = Variables()
10      my_object.current_file = "data.txt"
11      my_object.current_list = [1, 2, 3]
12      my_object.current_prob_matrix = {"A": 0.2, "B": 0.3}
13      my_object.reset()
14      # After resetting, my_object's attributes are cleared and ready
            for new data.
15  """
```

global_vars = Variables()

```
1   """
2   Define a global variable containing the declared vars. Use this so
        they are all only defined once and can be updated/stored
        throughout the applications lifetime.
3   """
```

get_character_fields(file)

```
1   """
2   Read a file containing character fields and their values, and return
        a dictionary of the fields.
3
4   Args:
5       file (str): The path to the file containing character fields.
6
7   Returns:
8       dict: A dictionary mapping character fields to their
            corresponding values.
9
10  Raises:
11      None.
12
13  The method opens the specified file and reads its contents. Each line
        in the file is expected to
14  represent a character field and its value, separated by an equals
        sign (=). The method parses each
15  line, extracts the field name and value, converts them to lowercase,
        and stores them in a dictionary.
16  The resulting dictionary is returned.
17
18  If the 'class' field is not found in the character fields, an error
        message is printed, and an
19  empty dictionary is returned.
```

```
20
21  Example usage:
22      character_fields = get_character_fields('character_data.txt')
23  """
```

split_list(lst, n)

```
1   """
2   Split a list into n sublists of approximately equal size.
3
4   Args:
5       lst (list): The input list to be split.
6       n (int): The number of sublists to create.
7
8   Returns:
9       list: A list containing n sublists.
10
11  Example:
12      >>> my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
13      >>> sublists = split_list(my_list, 3)
14      >>> print(sublists)
15      [[1, 2, 3, 4], [5, 6, 7], [8, 9, 10]]
16  """
```

create_html_list(values)

```
1   """
2   Create an HTML list from a string of semi-colon-separated values.
3
4   Args:
5       values (str): The string of semi-colon-separated values.
6
7   Returns:
8       str: The HTML list generated from the values.
9
10  Example:
11      >>> create_html_list("Item 1; Item 2; Item 3")
12      '<ul>\n<li>Item 1</li>\n<li>Item 2</li>\n<li>Item 3</li>\n</ul>'
13  """
```

separate_header_and_info(string)

```
1   """
2   Separates the header value and information from a string in the
        format "*header* Information here".
3
4   Args:
5       string (str): The input string in the specified format.
6
7   Returns:
8       tuple: A tuple containing the title value and the information.
9
10  Example:
11      >>> separate_title_and_info("*header* Information here")
12      ('header', 'Information here')
13  """
```

create_html_info(values)

```
"""
Create an HTML info block from a string of semi-colon-separated
    values.

Args:
    values (str): The string of semi-colon-separated values.

Returns:
    str: The HTML info block generated from the values.

Example:
    >>> create_html_info("Item 1; - Item 2; - Item 3; Item 4")
    '<p class="first-paragraph">Item 1</p>\n<p><ul><li>Item 2</li>\n<
        li>Item 3</li>\n</ul></p>\n<p>Item 4</p>\n'
"""
```

create_html_table(input_line)

```
"""
Convert a single line input to an HTML table format.

Args:
    input_line (str): Single line input containing semi-colon-
        separated values.

Returns:
    str: HTML table structure representing the input values.

Example:
    For a table of the form:

    -----------
    | a1 | a2 |
    |---------|
    | b1 | b2 |
    |---------|
    | c1 | c2 |
    -----------

    input: "2,a1,a2,b1,b2,c1,c2"
    output: '<table><tr><td>Value 1</td><td>Value 2</td></tr><tr><td>
        Value 3</td><td>Value 4</td></tr><tr><td>Value 5</td><td>Value
        6</td></tr></table>'
"""
```

download_image(url, file_path)

```
"""
Download an image from a URL and save it to a file path.

Args:
    url (str): The URL of the image to download.
    file_path (str): The file path to save the downloaded image.

```

```
8    Returns:
9        bool: True if the image was successfully downloaded and saved,
            False otherwise.
10   """
```

add_number_to_filename(filename, number)

```
1    """
2    Add a number to the filename before the extension.
3
4    Args:
5        filename (str): The original file name.
6        number (int): The number to add.
7
8    Returns:
9        str: The updated file name with a number added before the
            extension.
10
11   Example Usage:
12       >>> new_filename = add_number_to_filename("document.txt")
13       >>> print(new_filename, 3)
14       document (3).txt
15   """
```

is_image_file(file_name)

```
1    """
2    Checks if a file name is an image file based on its extension.
3
4    Args:
5        file_name (str): The name of the file to check.
6
7    Returns:
8        bool: True if the file name has an image extension, False
            otherwise.
9
10   Example:
11       >>> is_image_file('myphoto.jpg')
12       True
13       >>> is_image_file('document.pdf')
14       False
15   """
```

create_html_img(input_line)

```
1    """
2    Create an HTML block for an image section.
3
4    Args:
5        input_line (str): The input string of the form "image_file,
            image_source, caption".
6
7    Returns:
8        str: The HTML block representing the image section.
9
10   Notes:
```

```
11        - If the image file already exists, it will be used. Otherwise,
             the image will be downloaded from the provided image source
             URL.
12        - The image file, image source URL, and caption are extracted
             from the input line.
13
14    Example:
15        input_line = "image.jpg; https://www.example.com/image.jpg; A
             beautiful sunset"
16        html_block = create_html_img(input_line)
17    """
```

fix_image_extensions()

```
1    """
2    Updates all image extensions by running the fix_image_extensions.py
        script.
3
4    Example usage:
5          fix_image_extensions()
6    """
```

update_all()

```
1    """
2    Updates all components of the MMORPDND system.
3
4    This function updates the MMORPDND system by performing the following
        steps:
5        1. Retrieves the current working directory.
6        2. Changes the current working directory to the parent directory.
7        3. Constructs a command to update the system by running './
            mmorpdnd.py -u'.
8        4. Executes the update command using the system shell.
9        5. Changes the current working directory back to the original
            directory.
10
11    Note: This function assumes that the 'mmorpdnd.py' script is located
        in the parent directory.
12
13    Example usage:
14        update_all()
15    """
```

get_random_line(file_path)

```
1    """
2    Return a random line from a file.
3
4    Args:
5        file_path (str): The path to the file.
6
7    Returns:
8        str: A random line from the file.
9
10    Raises:
```

```
11          FileNotFoundError: If the file does not exist.
12          IOError: If there is an error reading the file.
13      """
```

extract_first_integer(string)

```
1   """
2   Extracts the first integer from a given string.
3
4   Args:
5       string (str): The input string.
6
7   Returns:
8       int or None: The first integer found in the string, or None if no
9           integer is found.
10  Example:
11      >>> string = "'6 (barbarian 3, rogue 3)'"
12      >>> first_integer = extract_first_integer(string)
13      >>> print(first_integer)
14      6
15  """
```

class Creator

```
1   """
2   This is the main class for the Crteator GUI and functionality.
3
4   Variables:
5       self.last_user_input = None
6       self.gui = tk.Tk()
7       self.gui.geometry("850x500")
8       self.gui.title("File Browser")
9       self.gui.tk.call('wm', 'iconphoto', self.gui._w, icon)
10      # Other GUI variables.
11
12  Methods:
13      __init__(self)
14      create_html_music(self, urls)
15      download_youtube_video_as_mp3(self, url, output_path="../music")
16      random_place(self, number=100)
17      create_pages(self)
18      create_page(self, file=global_vars.current_file)
19      checkbox_changed(self)
20      generate_char(self, file=global_vars.current_file)
21      output_text(self, text)
22      test(self)
23      get_user_choice(self)
24      yes(self)
25      no(self)
26      reset(self)
27      browse_files(self)
28      update_input_file(self)
29      generate_word(self)
30      run(self)
31  """
```

Creator.create_html_music(self, urls)

```python
"""
Creates an HTML list with links to the provided URLs and a folder
    icon.

Args:
    urls (str or list): The URL or list of URLs as a semicolon-
        delimited string or a list of strings.

Returns:
    str: The HTML list portion with links and folder icons.

Example:
    >>> urls = "https://www.youtube.com/watch?v=dQw4w9WgXcQ;https://
        www.youtube.com/watch?v=VIDEO2_ID"
    >>> html_list = self.create_html_music(urls)
    >>> print(html_list)
    <ul>
    <li><a href="https://www.youtube.com/watch?v=dQw4w9WgXcQ">Video
        1</a><a href="local/path/video1.mp3"><i class="fas fa-folder
        "></i></a></li>
    <li><a href="https://www.youtube.com/watch?v=VIDEO2_ID">Video 2</
        a><a href="local/path/video2.mp3"><i class="fas fa-folder"></i
        ></a></li>
    </ul>
"""
```

Creator.download_youtube_video_as_mp3(self, url, output_path="../music")

```python
"""
Downloads a YouTube video as a high-quality MP3 file.

Args:
    url (str): The URL of the YouTube video.
    output_path (str): The path to the directory where the MP3 file
        will be saved.

Returns:
    str or None: The path of the downloaded MP3 file if the download
        and conversion
                are successful, None if there was an error.

Raises:
    None

Example:
    >>> url = "https://www.youtube.com/watch?v=dQw4w9WgXcQ"
    >>> output_path = "/path/to/output/directory"
    >>> mp3_path = download_youtube_video_as_mp3(url, output_path)
    >>> print(mp3_path)
    "/path/to/output/directory/output.mp3"
"""
```

Creator.random_place(self, number=100)

```
1  """
2  Generates random place names along with their corresponding types.
3
4  Args:
5      number (int, optional): The number of random place names to
           generate. Defaults to 50.
6
7  Returns:
8      None
9  """
```

Creator.create_pages(self)

```
1   """
2   Generate page files for a file or each file within a directory.
3
4   This method checks if the current file (global_vars.current_file) is
        a directory.
5   If it is a file, it calls the generate_char() or create_page() method
         based on the file extension.
6   If it is a directory, it iterates through each file within the
        directory and calls the generate_char()
7   or create_page() method for each individual file.
8
9   Returns:
10      None
11
12  Raises:
13      None
14  """
```

Creator.create_page(self, file=global_vars.current_file)

```
1   """
2   Create an HTML page based on the input file.
3
4   Reads the input file specified by global_vars.current_file and
        extracts the content to generate an HTML page.
5   The input file should have a '.input' extension.
6   The output HTML file is created in the specified destination folder
        or the current directory if not specified.
7
8   Returns:
9       None
10  """
```

Creator.checkbox_changed(self)

```
1   """
2   This method is called when checkboxes are checked or unchecked.
3   It retrieves the values of the checkboxes and prints a message
        indicating whether they are enabled or disabled.
4   """
```

Creator.generate_char(self, file=global_vars.current_file)

```
1   """
2   Generate a character file based on the provided input file.
3
4   This method updates the input file, checks if it has the correct file
        type (.char),
5   generates character statistics and fields if they are not defined,
       replaces the fields
6   in the template file with the character information, and writes the
       new character file.
7
8   The process involves the following steps:
9       1. Verifies if the input file has the correct file type (.char).
           If not, it displays an error message and exits.
10      2. Retrieves the character fields from the input file.
11      3. Determines the character class and level.
12      4. If any fields are missing, generates default values for certain
            attributes and displays them.
13      5. Calculates the character's hit points (hp) if not already
           defined.
14      6. Generates the character file path and filename based on the
           character's name.
15      7. Reads the character template file.
16      8. Processes and replaces the fields in the template with the
           character information.
17          - Replaces general character fields.
18          - Calculates and inserts modifier values for certain attributes
             .
19          - Handles proficiencies and adds proficiency bonus to
              corresponding skills.
20          - Populates information, notes, and image blocks.
21          - Updates abilities and equipment lists.
22      9. Writes the new character file at the specified filepath.
23      10. If the option is selected, moves the input file to the trash.
24
25  Returns:
26      None
27  """
```

Creator.output_text(self, text)

```
1   """
2   Output the given text to the GUI window and the large_text widget.
3
4   This method displays the provided text in the GUI window and appends
       it to the large_text widget.
5   It also ensures that the text is visible by scrolling to the bottom
       of the widget and updates
6   the GUI window to reflect the changes.
7
8   Args:
9       text (str): The text to be displayed and appended to the
           large_text widget.
10
11  Example:
12      gui_instance = MyGUI()
```

```
13        gui_instance.output_text("Processing completed successfully.")
14        # The text "Processing completed successfully." is displayed in
              the GUI window.
15    """
```

Creator.test(self)

```
1    """
2    Method for testing.
3
4    Returns:
5        None
6    """
```

Creator.get_user_choice(self)

```
1    """
2    Displays a graphical user interface with yes/no buttons and returns
         the user's choice.
3
4    Args:
5        None.
6
7    Returns:
8        bool: The user's choice. True represents "yes" and False
             represents "no".
9
10   Raises:
11       None.
12
13   This method displays a graphical user interface (GUI) with "yes" and
         "no" buttons and waits for the user to make a choice.
14   The GUI is implemented using a main event loop.
15
16   The method performs the following steps:
17       1. Enables the "yes" and "no" buttons.
18       2. Creates a BooleanVar to store the user's choice.
19       3. Defines the callback functions that will be called when the
              buttons are clicked. These functions call the appropriate
20          methods (e.g., 'self.yes()', 'self.no()', 'self.reset()') and
                 set the user_choice variable accordingly.
21       4. Configures the buttons to call the respective callback
              functions.
22       5. Starts the main event loop using 'self.gui.mainloop()'.
23       6. Disables the "yes" and "no" buttons after the user has made a
              choice.
24       7. Returns the user's choice as a boolean value.
25
26   Note: The specific details of the GUI implementation, such as the
         actual buttons and their configuration, may depend on
27   the underlying GUI framework used.
28
29   Example usage:
30       get_user_choice()
31   """
```

Creator.yes(self)

```
"""
Set last_user_input to "yes".

This method updates the last_user_input attribute to "yes" and
    displays a message indicating the change.

Example:
    gui_instance = MyGUI()
    gui_instance.yes()
    # The last_user_input attribute is updated to "yes".
"""
```

Creator.no(self)

```
"""
Set last_user_input to "no".

This method updates the last_user_input attribute to "no" and
    displays a message indicating the change.

Example:
    gui_instance = MyGUI()
    gui_instance.no()
    # The last_user_input attribute is updated to "no".
"""
```

Creator.reset(self)

```
"""
Reset last_user_input and provide status.

This method resets the last_user_input attribute to "reset", displays
    a reset message using the output_text
method, and confirms the change with a print statement.

Example:
    gui_instance = MyGUI()
    gui_instance.reset()
    # The last_user_input attribute is reset to "reset", and the GUI
        provides a reset status.
"""
```

Creator.browse_files(self)

```
"""
Open a file dialog to select a file path.

This method opens a file dialog to allow the user to select a file
    path. The selected file path is then
displayed in the editable box on the GUI.

Example:
    gui_instance = MyGUI()
    gui_instance.browse_files()
```

```
10        # The user selects a file path using the file dialog , and the
              selected path is displayed in the GUI.
11    """
```

Creator.update_input_file(self)

```
1     """
2     Update the current input file and associated data.
3
4     This method updates the current input file based on the path entered
          in the GUI. If no file path
5     is provided , an appropriate message is displayed using the
          output_text method. If the provided
6     file path is different from the current file path , the global_vars
          object is reset , and the new
7     file path is set as the current file. Additionally , if the file's
          extension matches certain
8     predefined extensions (such as '.char', '.names', or '.list'), the
          lines from the file are read
9     and stored in the current_list attribute.
10
11    Example :
12        gui_instance = MyGUI ()
13        gui_instance.path_text.set("data.txt")
14        gui_instance.update_input_file ()
15        # The current input file is updated to 'data.txt', and associated
              data is adjusted.
16    """
```

Creator.generate_word(self)

```
1     """
2     Generates a word based on a probability matrix and offers the option
          to append it to a file.
3
4     Args :
5         None.
6
7     Returns :
8         None.
9
10    Raises :
11        None.
12
13    This method generates a word using a probability matrix and prompts
          the user whether to append the generated word
14    to a file.
15
16    The method performs the following steps :
17        1. Prints a message indicating that a word is being generated.
18        2. Updates the input file.
19        3. Generates a probability matrix based on the current list.
20        4. Prints the generated probability matrix.
21        5. Enters a loop that continues until the last user input is "
              reset".
22            a. Generates a word using the current probability matrix.
```

```
23        b. Outputs the generated word.
24        c. Asks the user if they want to append the word to the file.
25        d. Retrieves the user's choice.
26        e. If the user chooses "yes", the word is appended to the file
             .
27        f. If the user chooses "no", a message is outputted indicating
             that the word was not appended to the file.
28        g. If the user's choice is neither "yes" nor "no", the loop
             continues.
29
30 Note: The specific details of how the probability matrix is generated
       and how the word is outputted may depend on
31 the implementation of the methods used within this method.
32
33 Example usage:
34     generate_word()
35 """
```

Creator.run(self)

```
1 """
2 Runs the main GUI.
3 """
```

References

[1]