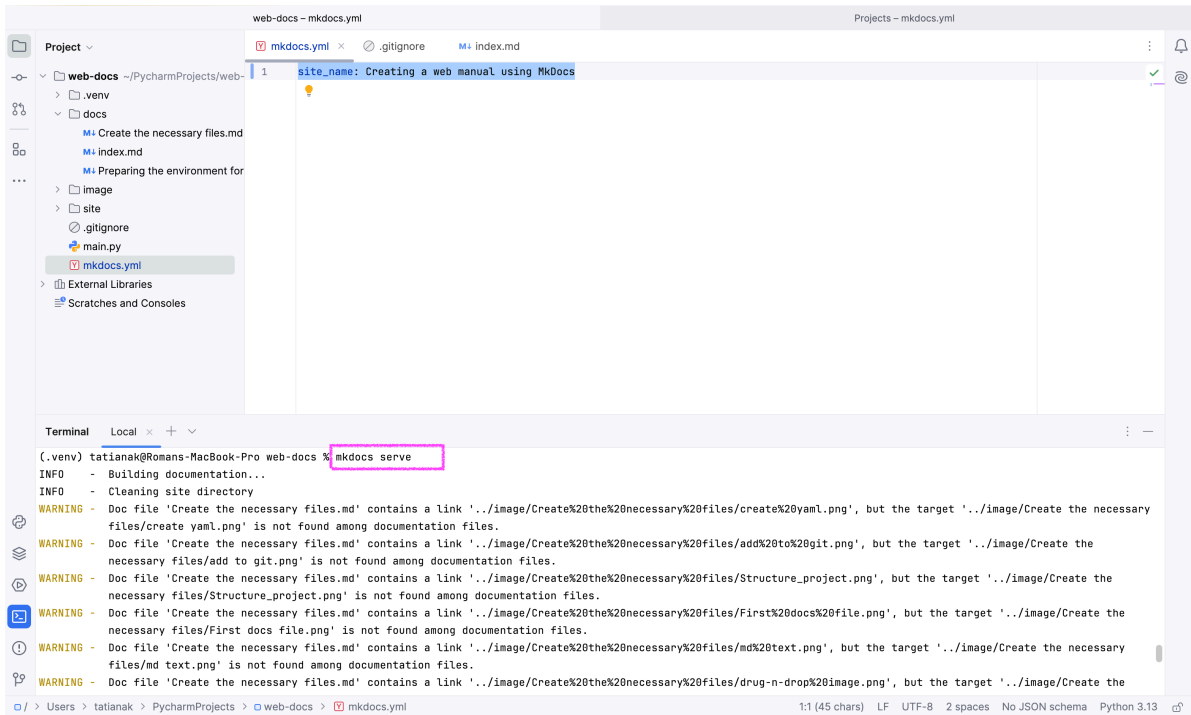


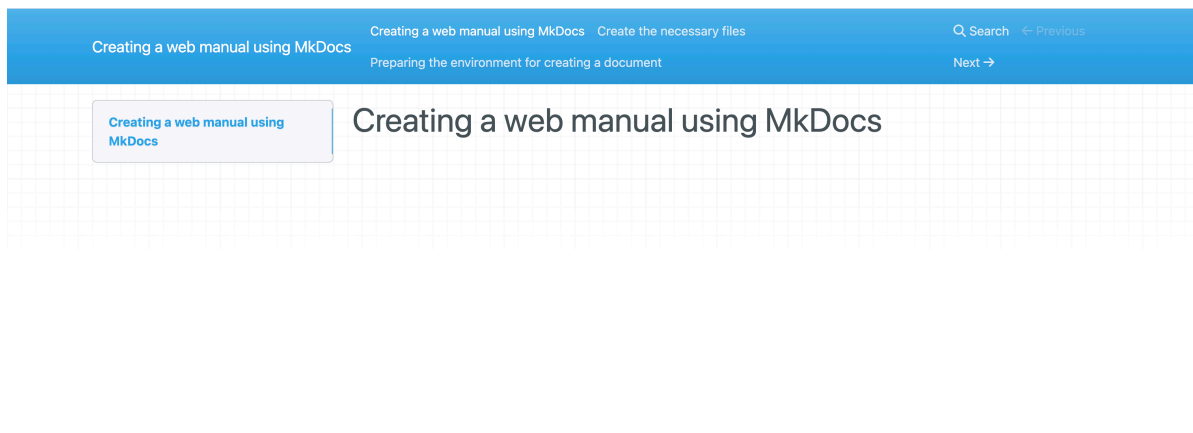
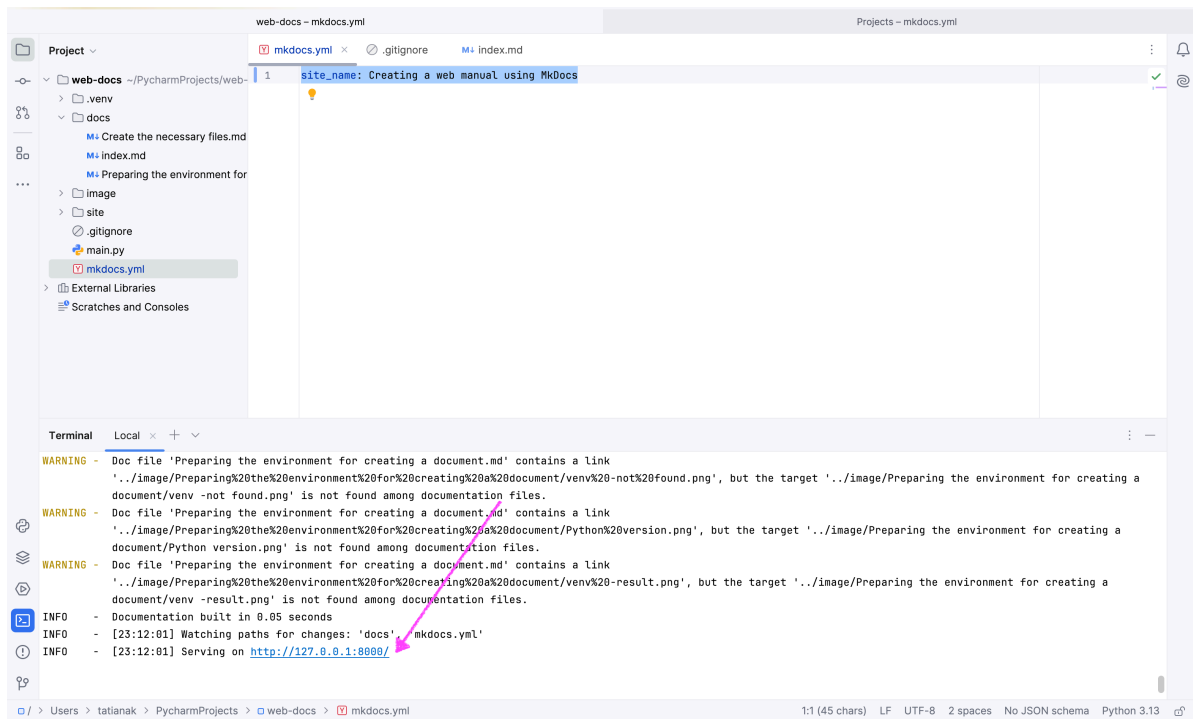
## Step 8: Deploy Documentation to GitHub Pages

**Build and preview the documentation using the MkDocs built-in web server:**

- Run the following command in the **terminal**: `mkdocs serve`



The site will be available locally for testing.

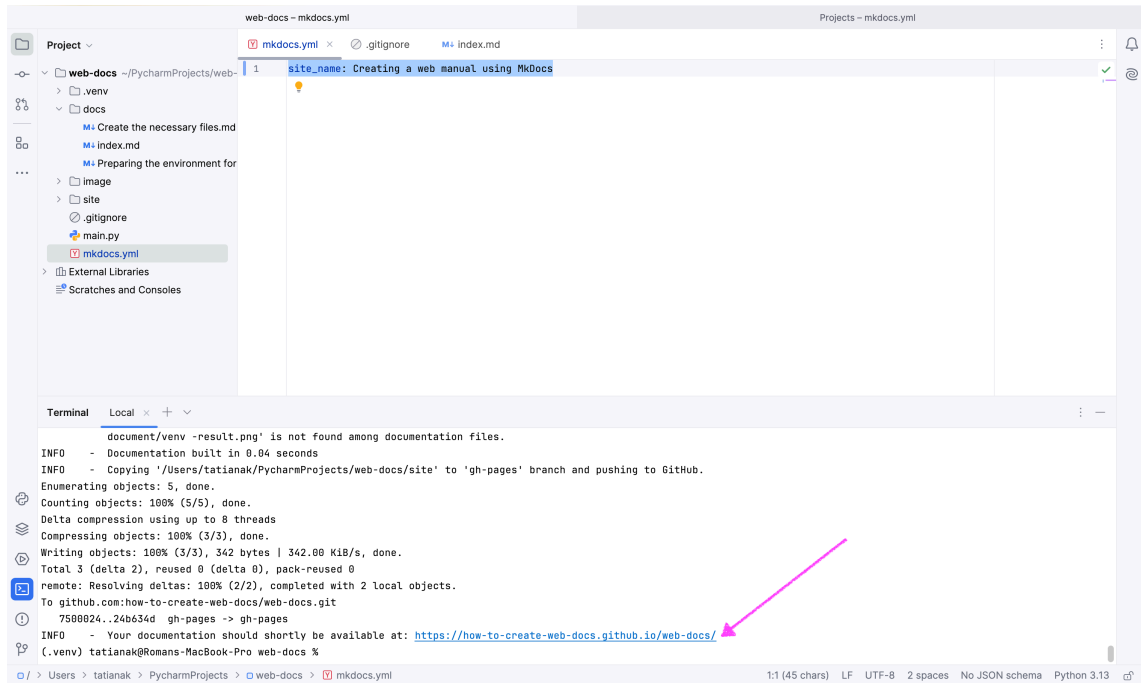


Documentation built with [MkDocs](#).

- The site will automatically reload when you make changes in **PyCharm**.
- To stop the server, press **Ctrl+C** on your keyboard.

## Deploy to GitHub Pages:

- Run the `mkdocs gh-deploy` command in the **terminal**.



The screenshot shows the PyCharm IDE interface. The top pane displays the `mkdocs.yml` file with the following content:

```
site_name: Creating a web manual using MkDocs
```

The bottom pane shows the terminal output of the `mkdocs gh-deploy` command. The output includes the following information:

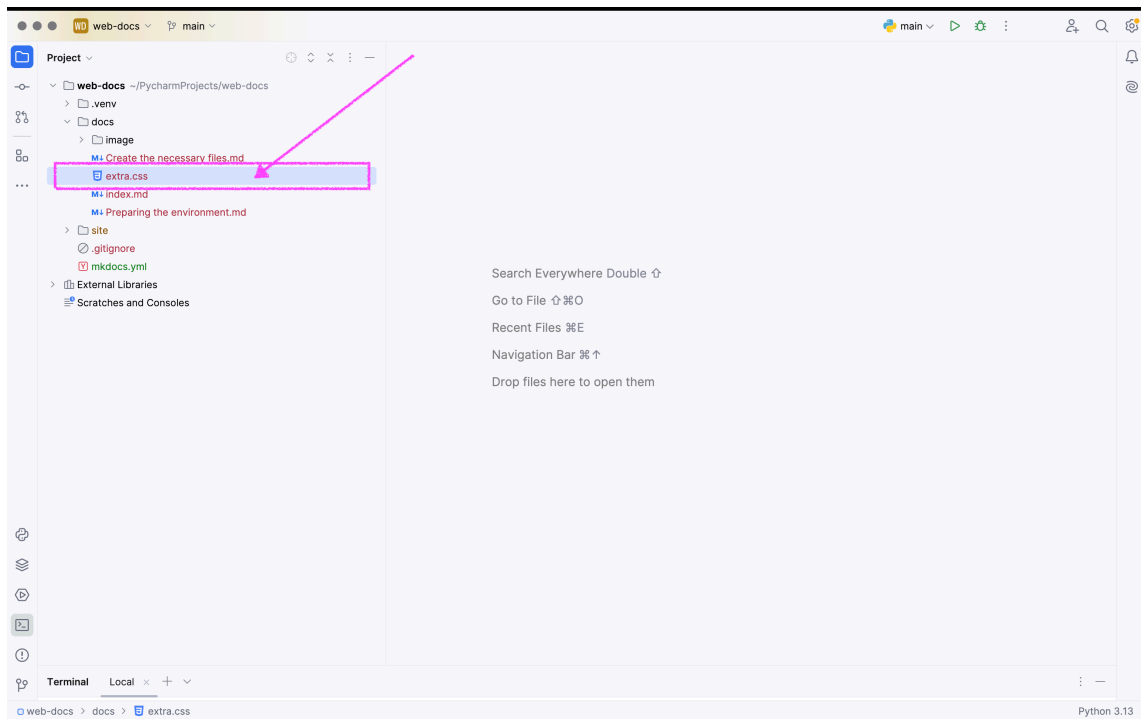
```
document/venv -result.png' is not found among documentation files.
INFO - Documentation built in 0.04 seconds
INFO - Copying '/Users/tatianak/PycharmProjects/web-docs/site' to 'gh-pages' branch and pushing to GitHub.
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 342 bytes | 342.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To github.com:how-to-create-web-docs/web-docs.git
7500024..24b634d gh-pages -> gh-pages
INFO - Your documentation should shortly be available at: https://how-to-create-web-docs.github.io/web-docs/
```

A pink arrow points to the URL <https://how-to-create-web-docs.github.io/web-docs/> in the terminal output.

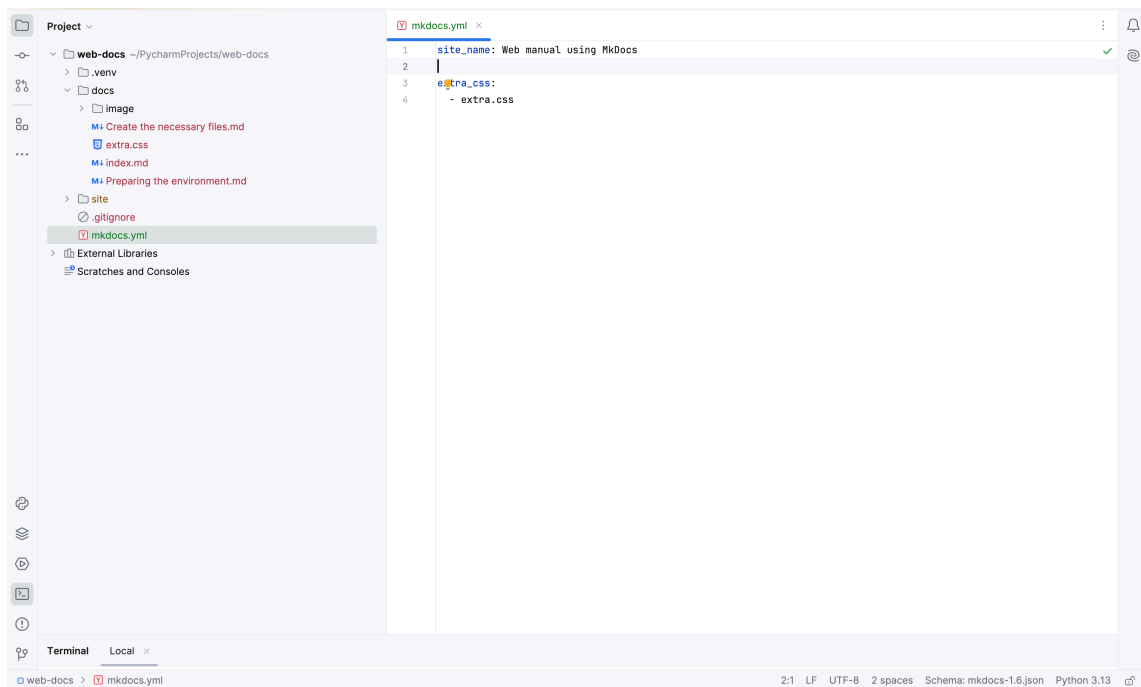
- This command creates a `gh-pages` branch, adds the compiled site to it, and pushes it to GitHub.
- The generated URL will look like this: `https://[username].github.io/[repository-name]`

## Step 9: Edit the Appearance of Your Site

1. Create the `extra.css` file in the `docs` folder.



2. Register your custom **CSS** in the `mkdocs.yml` configuration file.



3. Add your custom styles to the `extra.css` file.

For example, in my case:

```

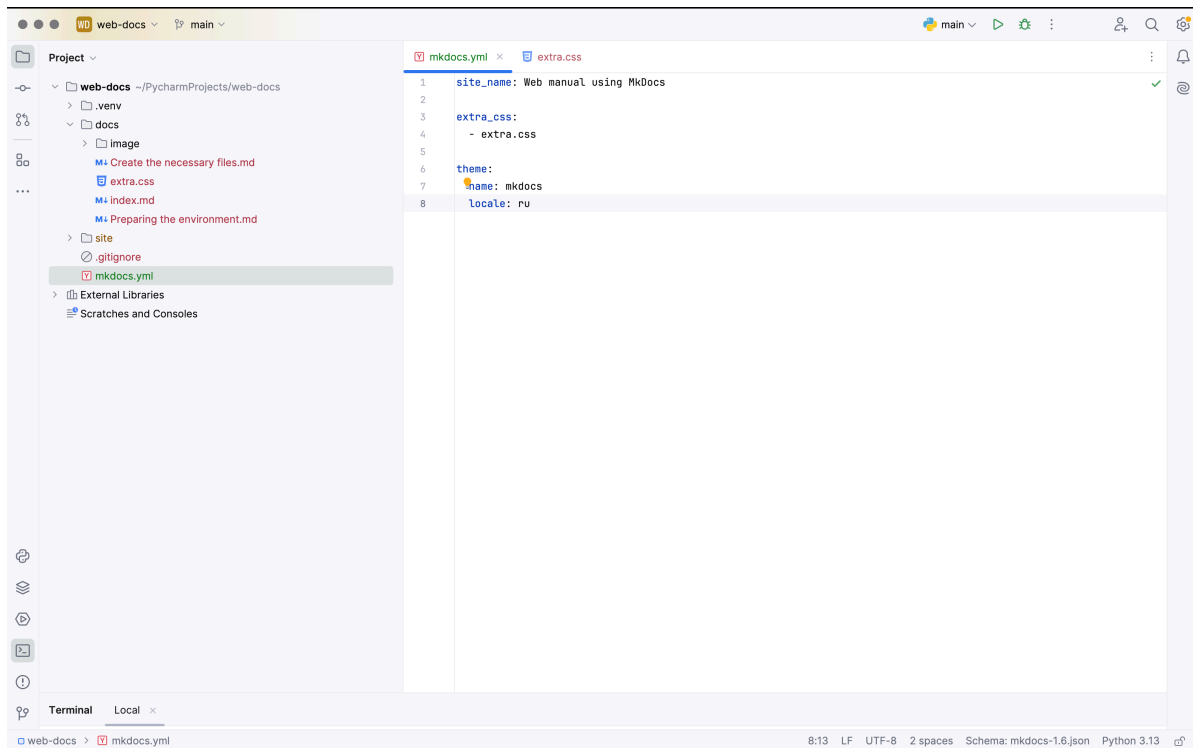
/* Header styles */
.navbar.bg-primary {
    background-image: none !important;
    background-color: #2d6da7 !important;
}

/* Removing the default footer */
footer {
    display: none !important;
}

/* Styles for the navigation bar */
.navbar .nav-link {
    color: #F8F9FA !important; /* Text color for the navigation bar */
}

.navbar .nav-link:hover {
    color: #d3d9dA !important; /* Hover text color for the navigation bar */
}

```



4. Write in our configuration (mkdocs.yml) the styling theme (in my case, it is “mkdocs”) and localization (for example, ru).

► Localization is written to translate default elements, for example, **Pagination**.

# Step 10: Committing Sources and Sending to the Git Repository

Follow these commands in sequence:

**Adding files to the staging area:** `git add .`

- This command adds all files and changes in the current directory (indicated by the `.`) to the staging area. It means you tell Git which files should be included in the next commit.
- If you need to add files from a specific folder (e.g., `docs`), use: `git add docs/` or `git add ./docs`

The choice depends on your preference, but both are valid.

**Creating a commit with a message:** `git commit -m "Description of changes"`

- This command saves the changes from the staging area to the repository with a message.
- If you don't use the `-m` flag, Git will open a text editor for you to write the description manually.

*Example:* `git commit -m "Added documentation files"`

**Pushing changes to the remote repository:** `git push origin main`

- This command sends (pushes) changes from your local repository to the remote repository (e.g., GitHub).
- `origin` is the name of the remote repository (default).
- `main` is the name of the branch where the changes are pushed. If your branch is named differently (e.g., `master`), replace `main` with the name of your branch.

Now you can view the committed changes by visiting the following link: [GitHub Repository](#)

## Adjustments Made:

1. Explained the meaning of the dot (`.`) as a reference to the current directory.
2. Added alternative commands for adding files from specific folders (`git add ./docs` and `git add docs/`).
3. Emphasized the logic behind each command for better understanding.