✦ Member-only story

Documenting Python Projects with MkDocs

Use Markdown to quickly create a beautiful documentation page for your projects

Gustavo R Santos

Published in TDS Archive

9 min read · Nov 22, 2024

Listen Share More

Image created with DALL·E by OpenAI. https://openai.com. MkDocs in Python.

Introduction

Project documentation is necessary. **Very necessary, I would emphasize.**

At the beginning of my career, I learned the hard way that a project must be documented.

Let's go back in time — to the 2000s — when I was working as a Customer Representative for large US companies. I was part of a team and my colleagues and I had joined the company around the same month. So, for a while, there was no need to worry because nobody was going on vacation just a few weeks or months after starting a new job.

However, after some time, it inevitably would happen. And we were all assigned to back up each other. That is when **documentation** started to play a major part in my career.

The day the first person took a few days off, I panicked! I got to work and I didn't know what to do or even where to start. The tasks kept coming and piling up while I was trying to figure out how to process them.

In the end, everything turned out well. I was able to figure it out and move on. But from that day on, I knew that documentation needed to be in place for any time off or team movement, like promotions or offboardings.

In this post, we will learn how to create a simple (and effective) project documentation using `mkdocs` in Python. The final result will look similar to MkDocs documentation.

Building The Documentation

`mkdocs` is a module in Python that allows us to create simple web pages using Markdown language. The benefit is that it is highly customizable, and gives your documentation a professional look, besides easily integrating with GitHub.

Additionally, `mkdocs` leverages Markdown notation language, which is very simple to use, being just plain text with the addition of a couple of signs to point titles, subtitles, bullet points, *italic*, **bold** *etc.* To illustrate, **Medium** uses Markdown language for blogging.

> Markdown is a lightweight markup language for creating web formatted text using a plain-text editor.

Preparation

I believe that the best time to create the documentation is once we finish the project. At that point, we already know which modules were used, how it was deployed, and how the project can be started and maintained. So, it is time to document those steps for the users.

When documenting something, my experience tells me to:

- Describe it as if you were describing how to run the project to a complete layperson.
- Try to avoid highly technical terms, and acronyms used in your company.
- Describe each step using clear and simple language.
- If the concept is too dense, or the task is too complex, try breaking it down into bullets.

Before starting with the documentation, let's create a sample project real quick, using the module `uv` for virtual environment management. Here, I am using `uv` and VSCode.

1. Open a terminal and install `uv` with `pip install uv`
2. Create a new project name "p2": `uv init p2`

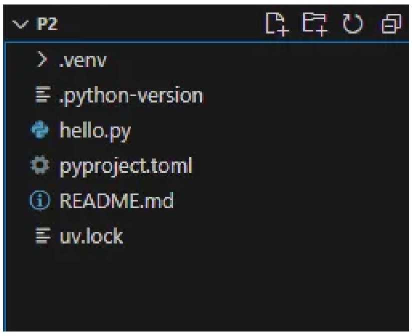
3. Change the directory to access the new folder: `cd p2`

4. Set the Python version for the project: `pyenv local 3.12.1`

5. Create a new virtual environment: `uv venv --python 3.12.1`

6. Activate the environment: `venv/Scripts/activate`

7. Add some packages: `uv add pandas, numpy, scikit-learn, streamlit`



Sample project created. Image by the author.

Getting Started

Having the project created, let's add `mkdocs` .

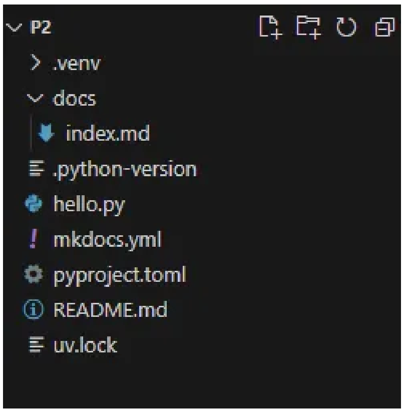
```
# Install mkdocs
uv add mkdocs
```

Next, we will create a new documentation folder.

```
# create a new documentation folder in your project
mkdocs new .
```

That command will generate a **docs** folder and the files needed for the documentation.

- File `mkdocs.yml` : It is used to configure your documentation webpage, like title, theme, and site structure, like adding new tabs.
- Folder **docs** with the file `index.md` : This file is where you will write the documentation itself.

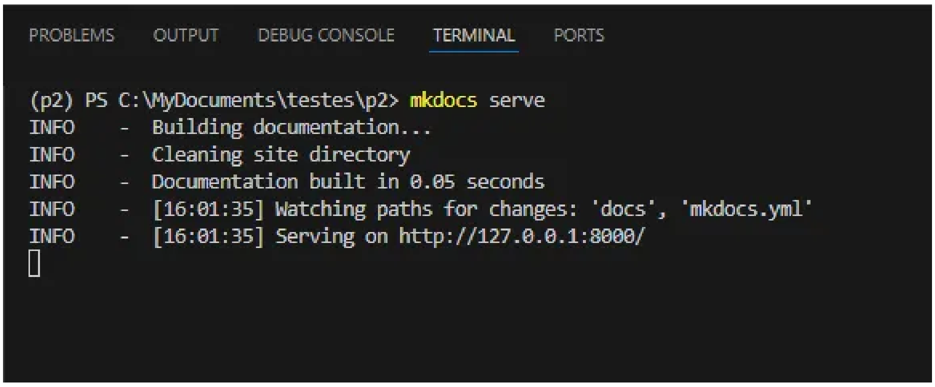


```
P2
  .venv
  docs
    index.md
  .python-version
  hello.py
  mkdocs.yml
  pyproject.toml
  README.md
  uv.lock
```

Documentation folder. Image by the author.

If we want to look at our documentation, we already can. Just use the serve command.

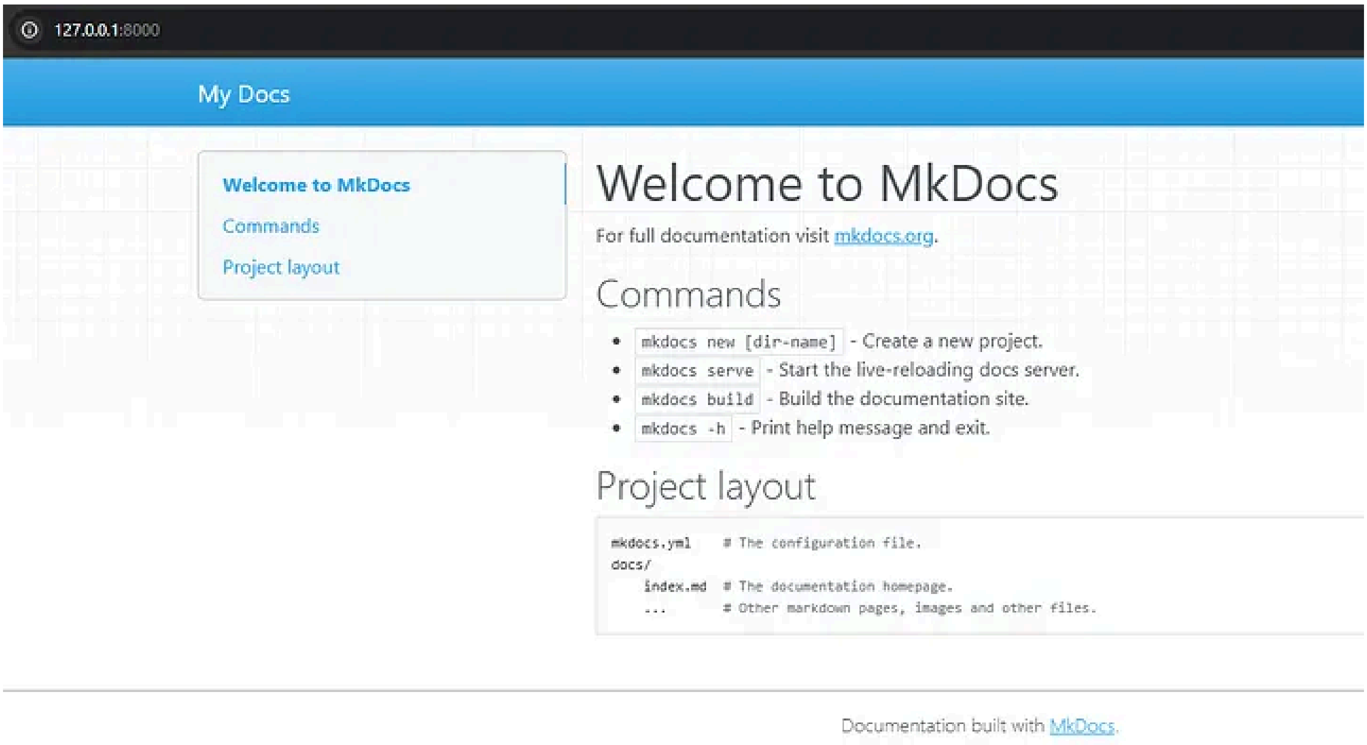
```
# Open a local server to display the docs
mkdocs serve
```



```
(p2) PS C:\MyDocuments\testes\p2> mkdocs serve
INFO    -  Building documentation...
INFO    -  Cleaning site directory
INFO    -  Documentation built in 0.05 seconds
INFO    -  [16:01:35] Watching paths for changes: 'docs', 'mkdocs.yml'
INFO    -  [16:01:35] Serving on http://127.0.0.1:8000/
```

Local version running on the port 8000. Image by the author.

Now, we can just copy + paste that HTTP into a browser (or Ctrl + click) to see how the documentation currently is.



Welcome to MkDocs

For full documentation visit mkdocs.org.

Commands

- `mkdocs new [dir-name]` - Create a new project.
- `mkdocs serve` - Start the live-reloading docs server.
- `mkdocs build` - Build the documentation site.
- `mkdocs -h` - Print help message and exit.

Project layout

```
mkdocs.yml    # The configuration file.
docs/
    index.md  # The documentation homepage.
    ...       # Other markdown pages, images and other files.
```

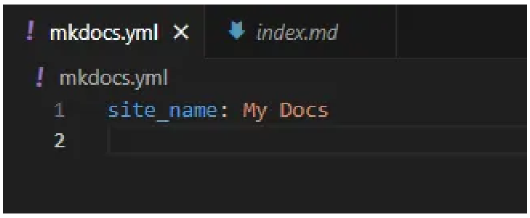
Documentation built with MkDocs.

MkDocs documentation “out of the box”. Image by the author.

Customization

It is time to customize our documentation.

Let’s start by changing the Title of the documentation page. Open the `mkdocs.yml` file. You will see only that `site_name` title in the default file.



```
site_name: My Docs

```

mkdocs.yml default file. Image by the author.

Let’s change it.

```
site_name: P2 Project Documentation
```

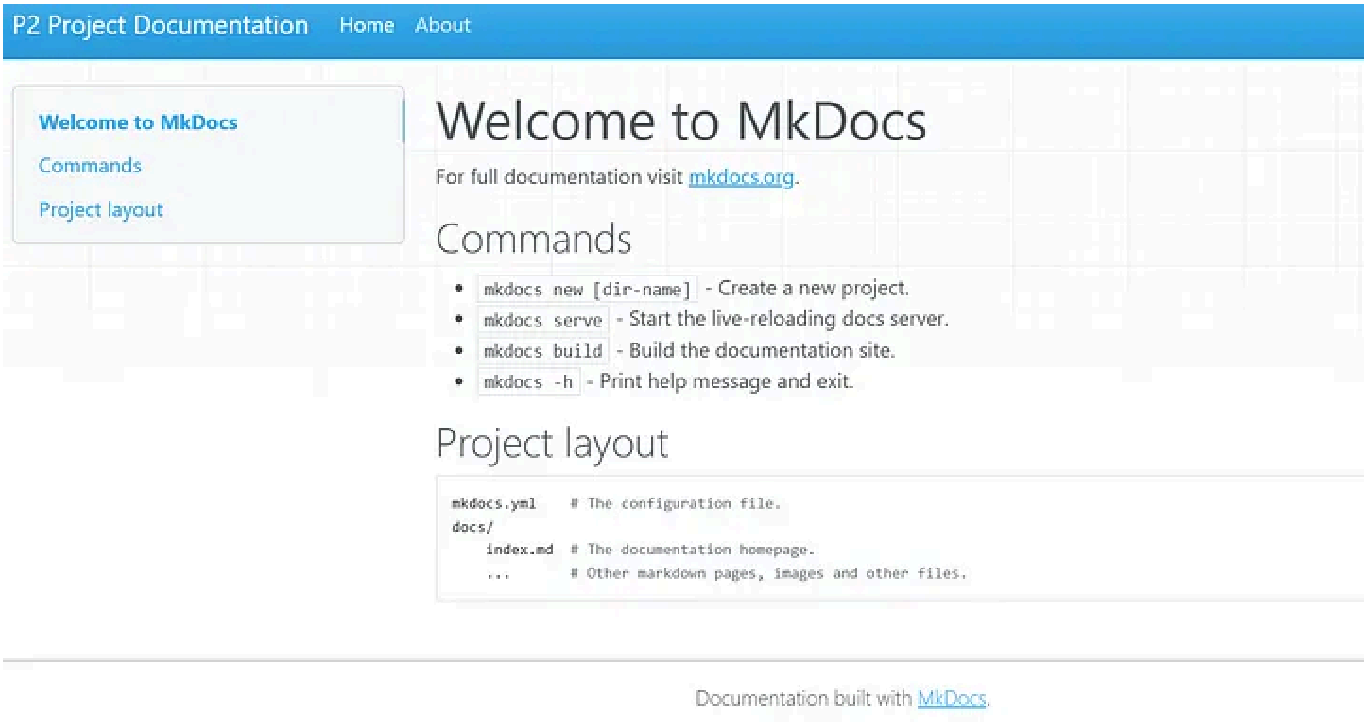
We can add a new tab `About` with the information about the project. For that to actually work, we also need to add a markdown file `about.md` to the folder **docs**.

```
site_name: P2 Project Documentation
nav:
  - Home: index.md
  - About: about.md
```

And we can change the theme if we want to. Check for built-in available themes here. Or for installable themes gallery here.

```
site_name: P2 Project Documentation
nav:
  - Home: index.md
  - About: about.md
theme: mkdocs
```

Here is the result, so far.



Welcome to MkDocs

For full documentation visit mkdocs.org.

Commands

- `mkdocs new [dir-name]` - Create a new project.
- `mkdocs serve` - Start the live-reloading docs server.
- `mkdocs build` - Build the documentation site.
- `mkdocs -h` - Print help message and exit.

Project layout

```
mkdocs.yml    # The configuration file.
docs/
    index.md  # The documentation homepage.
    ...       # Other markdown pages, images and other files.
```

Documentation built with MkDocs.

Mkdocs is easily customizable. Image by the author.

Next, let us start writing the documentation. This should be done in a markdown file within the folder **docs.**

I will write the whole example documentation to the file `index.md` and the project meta information will go to the file `about.md` .

- File **index.md**

We will erase the sample text that is in there and write our documentation instead.

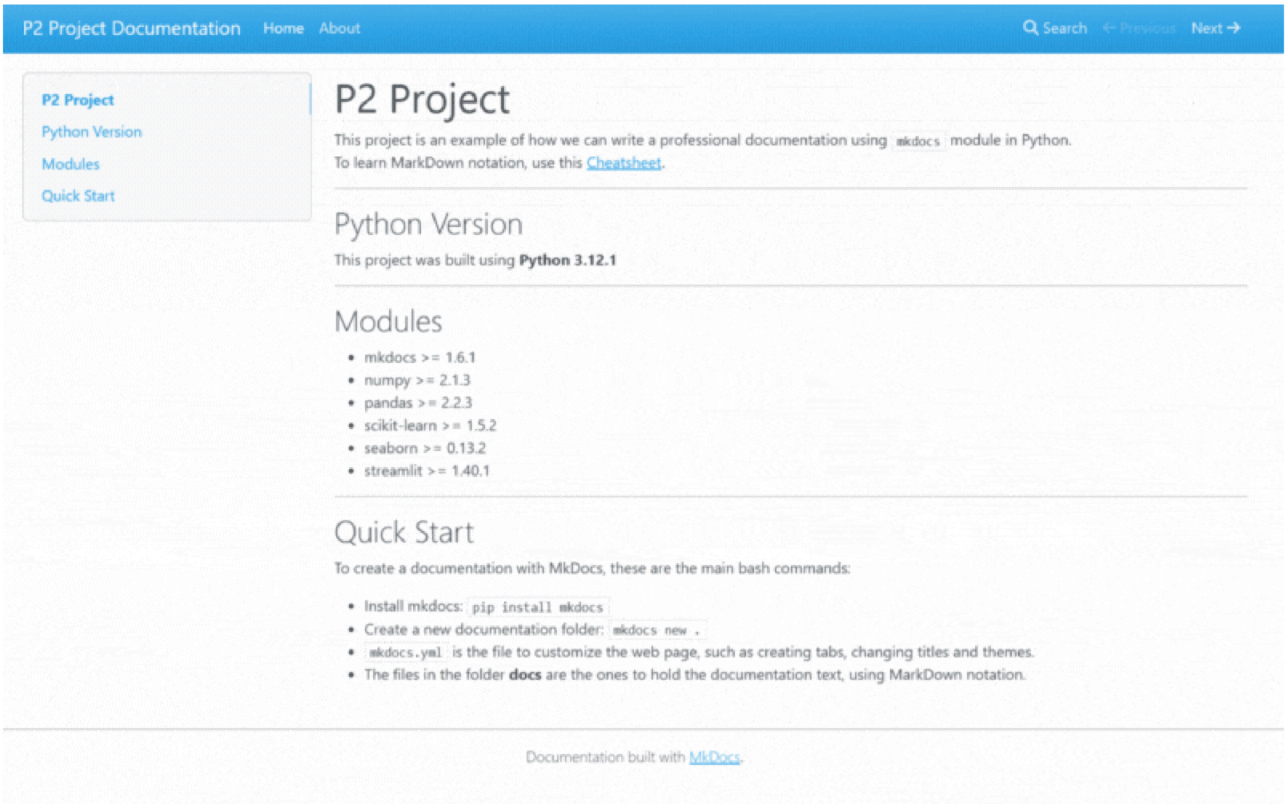
```
# P2 Project

This project is an example of how we can write a professional documentation using `mkdocs` module in
To learn MarkDown notation, use this [Cheatsheet](https://github.com/adam-p/markdown-here/wiki/Markdo

---

## Python Version
```

```
This project was built using **Python 3.12.1**

---

## Modules

* mkdocs >= 1.6.1
* numpy >= 2.1.3
* pandas >= 2.2.3
* scikit-learn >= 1.5.2
* seaborn >= 0.13.2
* streamlit >= 1.40.1

---

## Quick Start

To create a documentation with MkDocs, these are the main bash commands:

* Install mkdocs: `pip install mkdocs`
* Create a new documentation folder: `mkdocs new .`
* `mkdocs.yml` is the file to customize the web page, such as creating tabs, changing titles and them
* The files in the folder **docs** are the ones to hold the documentation text, using MarkDown notati
```

- File **about.md**

```
# About This Project
 <br>

* **Author:** Your Name
* **Purpose:** Exemplify how to create a professional looking documentation for your projects using M

---

### Contact

Find me on [Linkedin](https://www.linkedin.com/in/profile)
```

The final result is this beautiful documentation site.



P2 Project

This project is an example of how we can write a professional documentation using `mkdocs` module in Python.
To learn MarkDown notation, use this Cheatsheet.

Python Version

This project was built using **Python 3.12.1**

Modules

- mkdocs >= 1.6.1
- numpy >= 2.1.3
- pandas >= 2.2.3
- scikit-learn >= 1.5.2
- seaborn >= 0.13.2
- streamlit >= 1.40.1

Quick Start

To create a documentation with MkDocs, these are the main bash commands:

- Install mkdocs: `pip install mkdocs`
- Create a new documentation folder: `mkdocs new .`
- `mkdocs.yml` is the file to customize the web page, such as creating tabs, changing titles and themes.
- The files in the folder **docs** are the ones to hold the documentation text, using MarkDown notation.

Documentation built with MkDocs.

Final documentation website. Image by the author.

Adding Functions and Docstrings

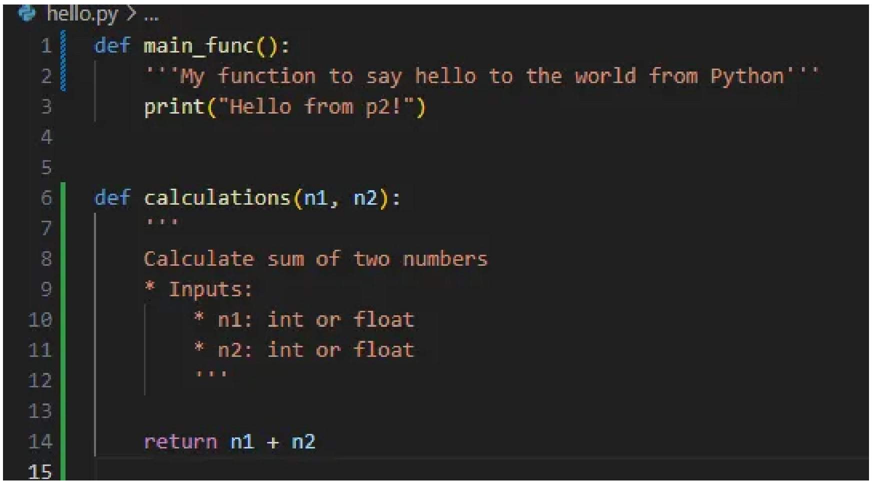
MkDocs also has ability to pull functions and respective docstrings from the code. To do that, first add the module MkDocstrings-Python using `pip install mkdocstrings-python` . In our case, I am using `uv` .

```
uv add mkdocstrings-python
```

Next, adjust the `mkdocs.yml` file to add the plugin. Add the following lines to the end of the file and save it.

```
plugins:
- mkdocstrings:
    default_handler: python
```

Now, let's look at our code. In this example project, we have only the file `hello.py` with two functions.



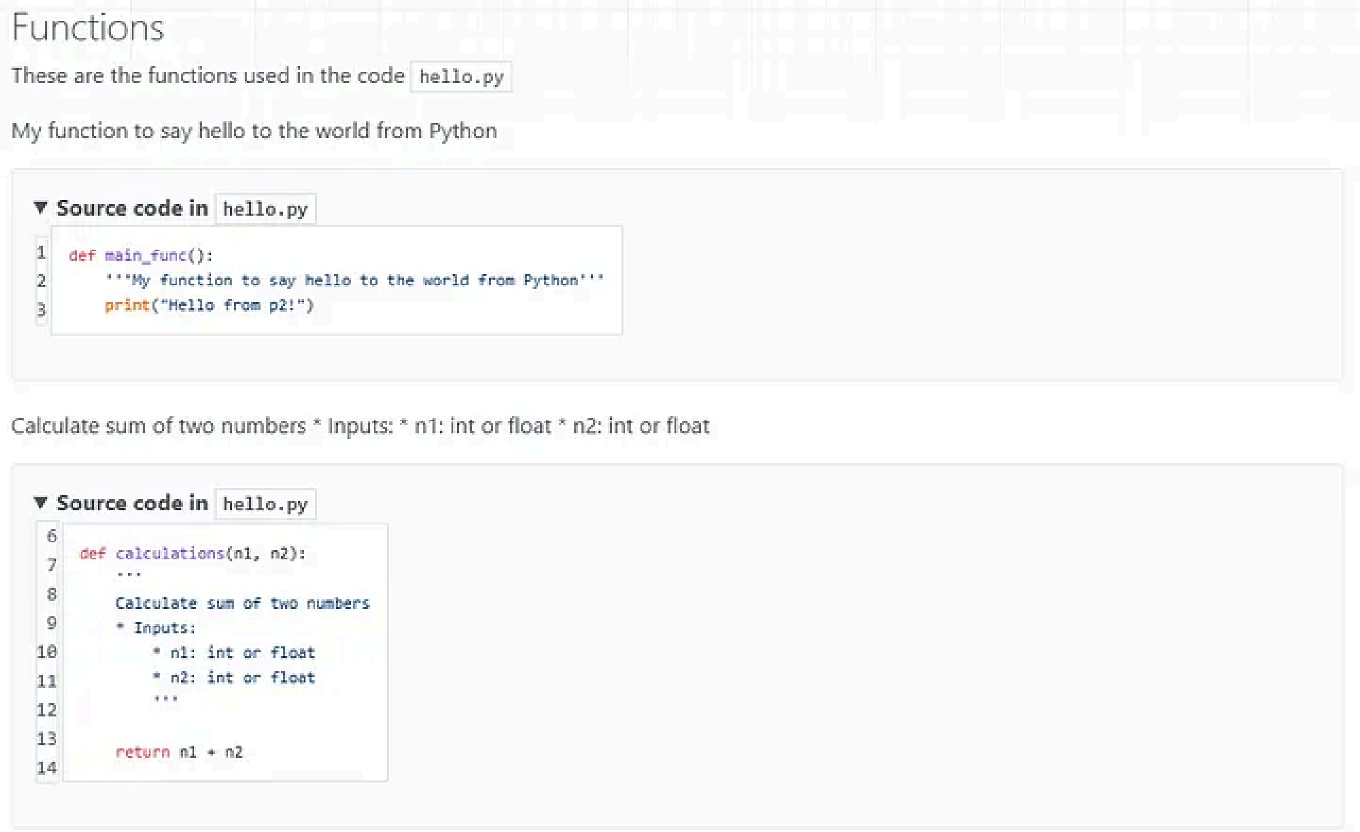
```
def main_func():
    '''My function to say hello to the world from Python'''
    print("Hello from p2!")


def calculations(n1, n2):
    '''
    Calculate sum of two numbers
    * Inputs:
        * n1: int or float
        * n2: int or float
        '''

    return n1 + n2
```

Functions in the python file. Image by the author.

Adding them to the documentation is pretty simple. Use three `:::` followed by the path to your file. Since this file is in the main folder of the project, we simply add the `file_name.function` . If it is within a folder, you can use something like `folder.file.function` .

```
### Function

These are the functions used in the code `hello.py`

::: hello.main_func
::: hello.calculations
```

After saving the file, we can look at the result `mkdocs serve` .



Functions

These are the functions used in the code `hello.py`

My function to say hello to the world from Python

▼ **Source code in** `hello.py`

```
def main_func():
    '''My function to say hello to the world from Python'''
    print("Hello from p2!")
```

Calculate sum of two numbers * Inputs: * n1: int or float * n2: int or float

▼ **Source code in** `hello.py`

```
def calculations(n1, n2):
    '''
    Calculate sum of two numbers
    * Inputs:
        * n1: int or float
        * n2: int or float
        '''

    return n1 + n2
```

Functions pulled directly from the .py file. Image by the author.

Now let's deploy the docs.

Deploying

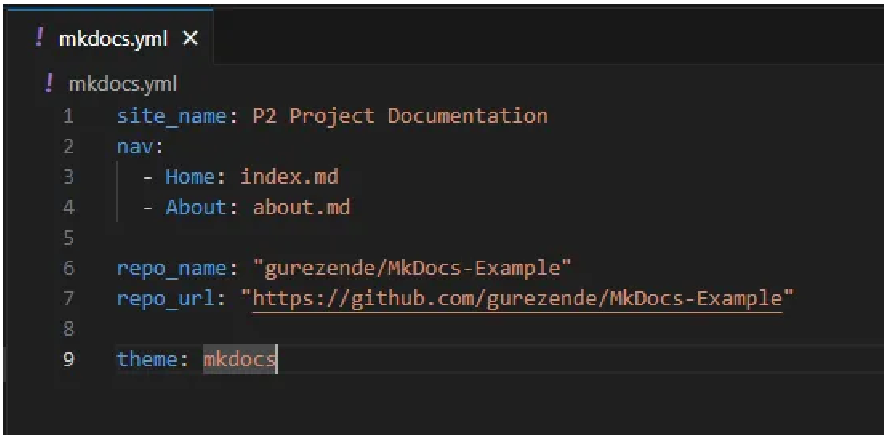
Deploying our documentation page is simple.

First, we must create a GitHub page for the project, if we already don't have one.

Next, go back to the IDE terminal and we will build our page with the next command. This command will create the folders and files necessary to deploy the documentation website.

```
mkdocs build

[OUT]:
INFO  -  Cleaning site directory
INFO  -  Building documentation to directory: C:\MyDocuments\testes\p2\site
INFO  -  Documentation built in 0.06 seconds
```

Now, need to add the GitHub repository in the `mkdocs.yml` file, so the module knows where to deploy the documentation to.



```
site_name: P2 Project Documentation
nav:
  - Home: index.md
  - About: about.md

repo_name: "gurezende/MkDocs-Example"
repo_url: "https://github.com/gurezende/MkDocs-Example"

theme: mkdocs
```

Adding the repository name to the mkdocs.yml. Image by the author.

Then we open a Git Bash Terminal to initialize Git and commit.

```
# Initialize Git
git init

# Add the reference repository
git remote add origin https://github.com/gurezende/MkDocs-Example.git

# Add files from the project
git add .

# Commit the files
git commit -m "Project Code and documentation"

# Create Branch Main
git branch -M main
```

```
# Push files to GitHub
git push -u origin main
```

And then we can deploy the documentation with the following bash code in a Powershell terminal.

```
mkdocs gh-deploy

## Output ##
INFO    -  Cleaning site directory
INFO    -  Building documentation to directory: C:\MyDocuments\testes\p2\site
INFO    -  Documentation built in 0.08 seconds
WARNING -  Version check skipped: No version specified in previous deployment.
INFO    -  Copying 'C:\MyDocuments\testes\p2\site' to 'gh-pages' branch and pushing to GitHub.
Enumerating objects: 39, done.
Counting objects: 100% (39/39), done.
Delta compression using up to 12 threads
Compressing objects: 100% (37/37), done.
Writing objects: 100% (39/39), 829.12 KiB | 11.68 MiB/s, done.
Total 39 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
remote:
remote: Create a pull request for 'gh-pages' on GitHub by visiting:
remote:      https://github.com/gurezende/MkDocs-Example/pull/new/gh-pages
remote:
To https://github.com/gurezende/MkDocs-Example.git
 * [new branch]      gh-pages -> gh-pages
INFO - Your documentation should shortly be available at: https://gurezende.github.io/MkDocs-Example/
```

Notice that on the last line, we have the URL where the documentation was deployed. This address can be added to your GitHub readme file.

P2 Project

None

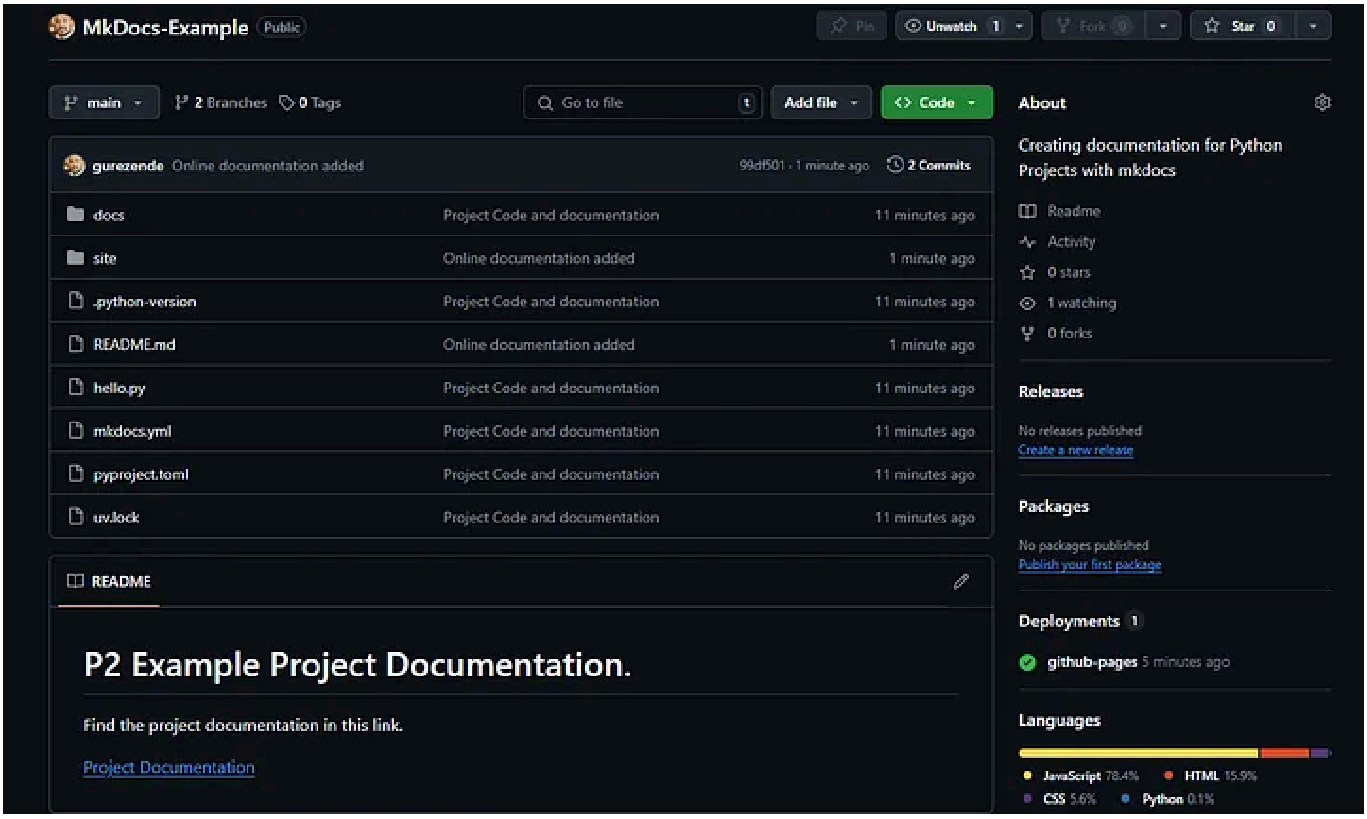
gurezende.github.io

After deployment, we just need to push the updates again to update GitHub, using the following commands on a Git Bash terminal.

```
git add .
git commit -m "Online documentation added"
git push origin main
```

That's it! The documentation is live!

From now on, every time we update the markdown files from our project and command `mkdocs gh-deploy`, the web page is updated and our documentation stays up to date. Easy like that!



gurezende Online documentation added 99df501 · 1 minute ago 2 Commits

docs	Project Code and documentation	11 minutes ago
site	Online documentation added	1 minute ago
.python-version	Project Code and documentation	11 minutes ago
README.md	Online documentation added	1 minute ago
hello.py	Project Code and documentation	11 minutes ago
mkdocs.yml	Project Code and documentation	11 minutes ago
pyproject.toml	Project Code and documentation	11 minutes ago
uv.lock	Project Code and documentation	11 minutes ago

P2 Example Project Documentation.

Find the project documentation in this link.

Project Documentation

About

Creating documentation for Python Projects with mkdocs

Readme
Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Deployments 1

github-pages 5 minutes ago

Languages

JavaScript 78.4% HTML 15.9%
CSS 5.6% Python 0.1%

GitHub page of the project. Image by the author.

Before You Go

Documenting your projects is important.

After all, nobody knows what was in your head when you developed something. Therefore, documenting is like showing your line of thought, the steps used to reach an end.

Open a window in your mind to show other people how you created that product and how to use it.

MkDocs make it so easy and looks super professional. I am sure it will help a lot in documenting your projects at work, helping fellow colleagues to navigate your code, as well as positively impacting anyone who looks at your portfolio from now on.

If you liked this content, follow me for more.

Gustavo R Santos

I am a Data Scientist specializing in data analysis, machine learning, and visualization using Python, R, SQL, and...

gustavorsantos.me

GitHub Repository

Here is the GitHub Repository for this article.

GitHub - gurezende/MkDocs-Example: Creating documentation for Python Projects with mkdocs

Creating documentation for Python Projects with mkdocs - gurezende/MkDocs-Example

github.com

Learn More

If you want to see this content in video, here’s a product from my Gumroad page.

Create Stunning Documentation with Python and MkDocs

Creating project documentation is essential! But do you know how to create a professional page using only Markdown…

gustavorsantos.gumroad.com

References

MkDocs

Project documentation with Markdown.

www.mkdocs.org

Usage - mkdocstrings-python

A Python handler for mkdocstrings.

mkdocstrings.github.io

PYTHON — Project Documentation with MkDocs and Python

Without requirements or design, programming is the art of adding bugs to an empty text file. — Louis Srygley

laxfed.dev

Markdown Here Cheatsheet

Google Chrome, Firefox, and Thunderbird extension that lets you write email in Markdown and render it before sending. …

github.com

Choosing Your Theme - MkDocs

Edit description

www.mkdocs.org

Gallery

Edit description

pawamoy.github.io

Documentation Python Mkdocs Project Management Tips And Tricks

Data Science

Following

Published in TDS Archive

814K Followers · Last published Feb 3, 2025

An archive of data science, data analytics, data engineering, machine learning, and artificial intelligence writing from the former Towards Data Science Medium publication.

Edit profile

Written by Gustavo R Santos

3K Followers · 34 Following

Data Scientist | I solve business challenges through the power of data. | Visit my site: https://gustavorsantos.me

Responses (5)

Gustavo R Santos

What are your thoughts?

David Bui
Nov 22, 2024

Nice. Thanks 🙏

👏 2 1 reply Reply

Ar Kulakov
Feb 12

No documentation is still better than outdated/incorrect one. There is only one source of truth - the code. I trust tests more than any docs because I'm yet to see at least couple years old project that has up to date docs. Developers hate to write... more

3 1 reply Reply

Brad Brown
Nov 26, 2024

Thanks for this; I hadn't heard of this tool but I like the idea a lot. Does it have the ability to collect docstrings from code and include them in the generated web site? I believe in hand-written documentation, but I also like module and function documentation collected into the documentation package.

1 1 reply Reply

See all responses

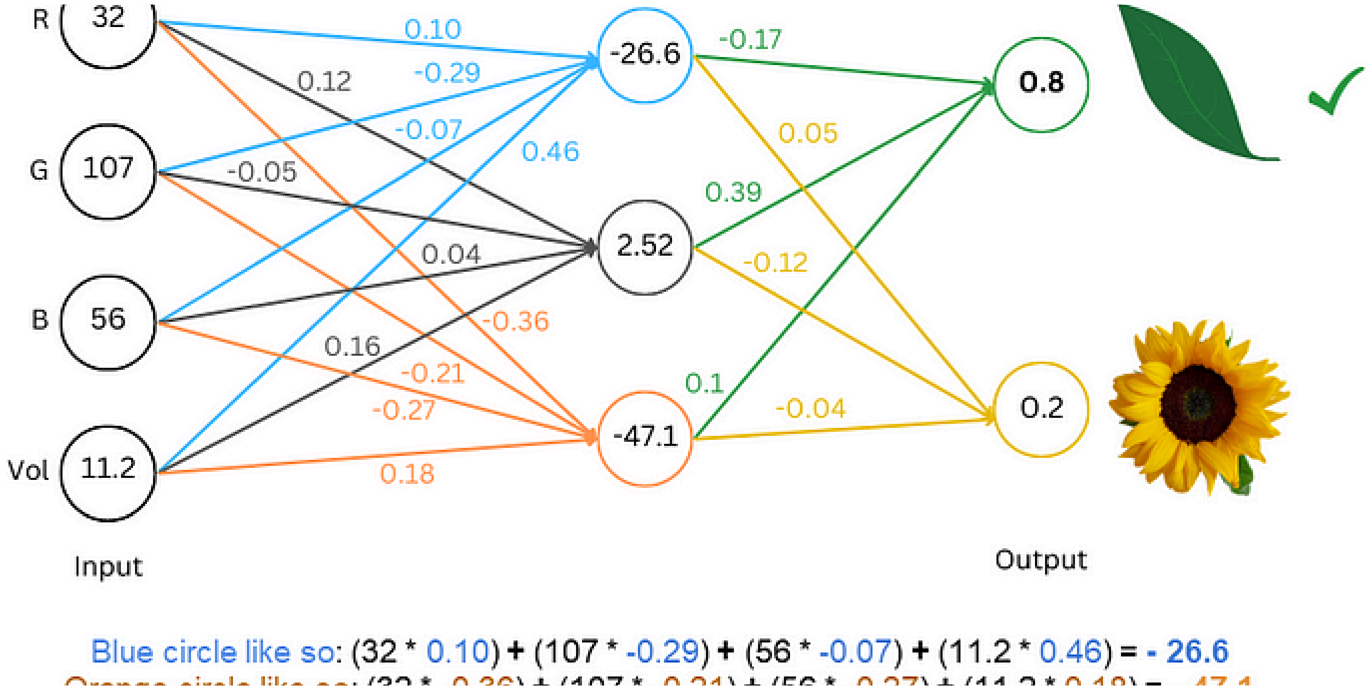
More from Gustavo R Santos and TDS Archive

In Data Science Collective by Gustavo R Santos

Modern NLP: Tokenization, Embedding, and Text Classification

Learning modern Natural Language Processing with Python code.

Feb 28 475 5

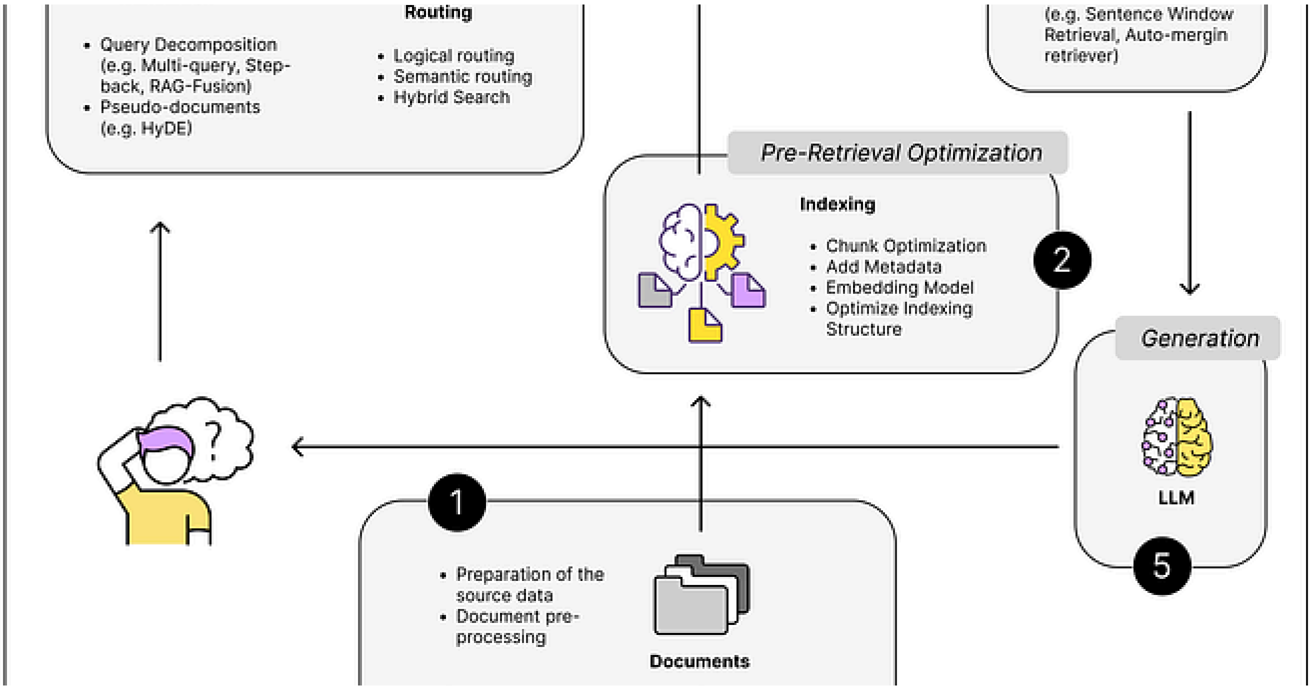


In TDS Archive by Rohit Patel

Understanding LLMs from Scratch Using Middle School Math

In this article, we talk about how LLMs work, from scratch — assuming only that you know how to add and multiply two numbers. The article…

Oct 19, 2024 7K 88


Routing
Query Decomposition (e.g. Multi-query, Step-back, RAG-Fusion)
Pseudo-documents (e.g. HyDE)
Logical routing
Semantic routing
Hybrid Search
(e.g. Sentence Window Retrieval, Auto-mergin retriever)
Pre-Retrieval Optimization
Indexing
Chunk Optimization
Add Metadata
Embedding Model
Optimize Indexing Structure
2
Generation
LLM
5
1
Preparation of the source data
Document pre-processing
Documents

In TDS Archive by Dominik Polzer

17 (Advanced) RAG Techniques to Turn Your LLM App Prototype into a Production-Ready Solution

A collection of RAG techniques to help you develop your RAG app into something robust that will last

Jun 26, 2024 3K 31

In Data Science Collective by Gustavo R Santos

Vector Databases and Search By Similarity for NLP

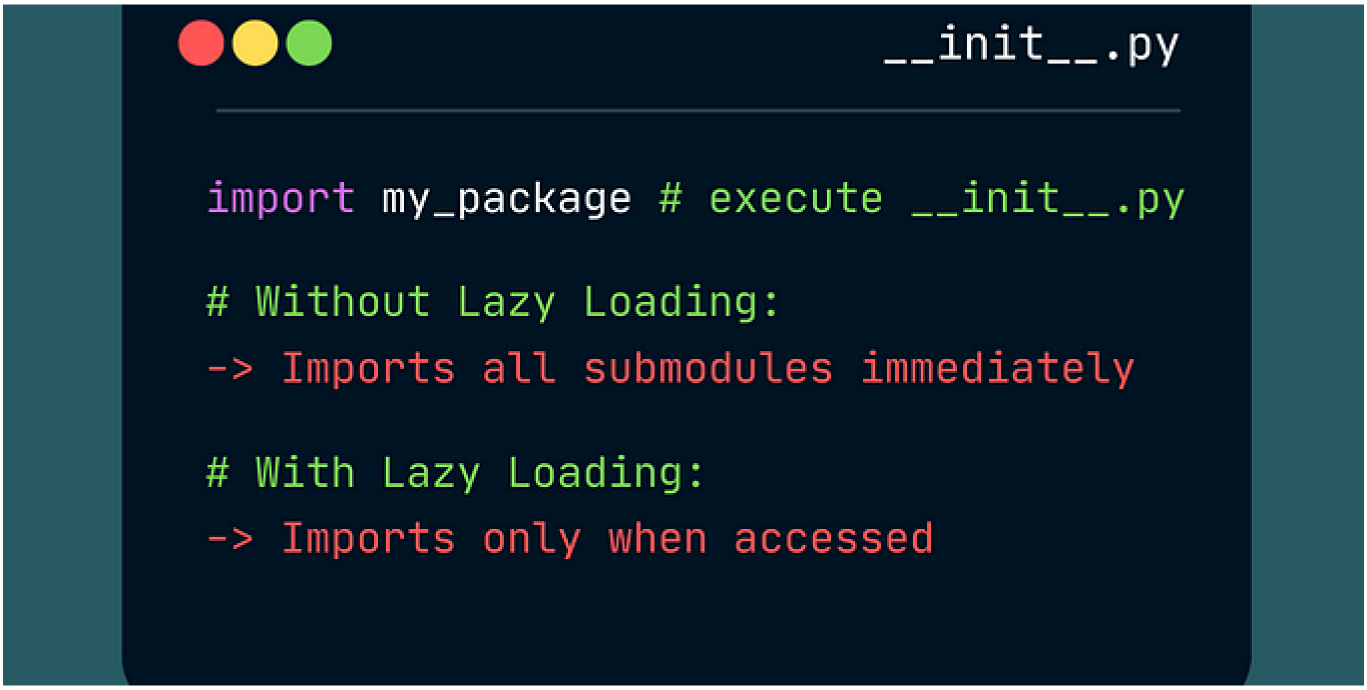
Learn about vector databases and how they can help your data science projects.

✦ Mar 14 100

See all from Gustavo R Santos

See all from TDS Archive

Recommended from Medium



```
import my_package # execute __init__.py

# Without Lazy Loading:
-> Imports all submodules immediately

# With Lazy Loading:
-> Imports only when accessed
```

In Level Up Coding by Joseph Robinson, Ph.D.

Mastering __init__.py in Python: A Complete Guide to Imports, Packages, and Best Practices

Structure Python packages, control imports, and write a clean, efficient __init__.py. Machine learning example included!

Mar 17 469 4

In The Pythoneers by Aashish Kumar

How I Automate 5 Annoying Tasks with Python (So I Never Have to Do Them Again!)

Here’s how I use Python to automate 5 annoying tasks—so I never have to do them manually again!

Mar 24 187 1

In Python in Plain English by Coders Stop

7 Python Libraries That Will Make Your Code Worse — But Your Life Better

Ever found yourself staring at elegant, production-ready code and thinking, "This is beautiful… but it took me three days to write"? You're…

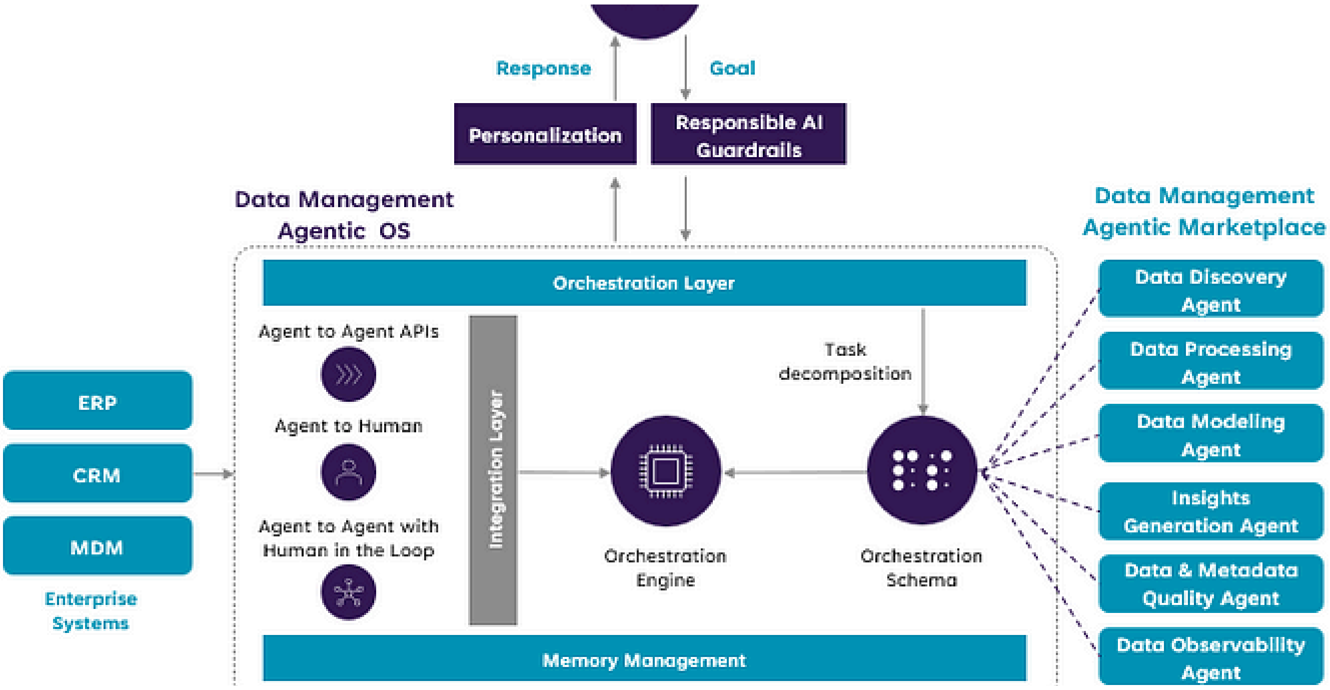
6d ago 112 1

Amos Gyamfi

The 3 Best Python Frameworks To Build UIs for AI Apps

Build user-friendly AI chat interfaces crucial for seamless interactions, demo presentation, and testing using the leading Python…

Mar 17 316 7

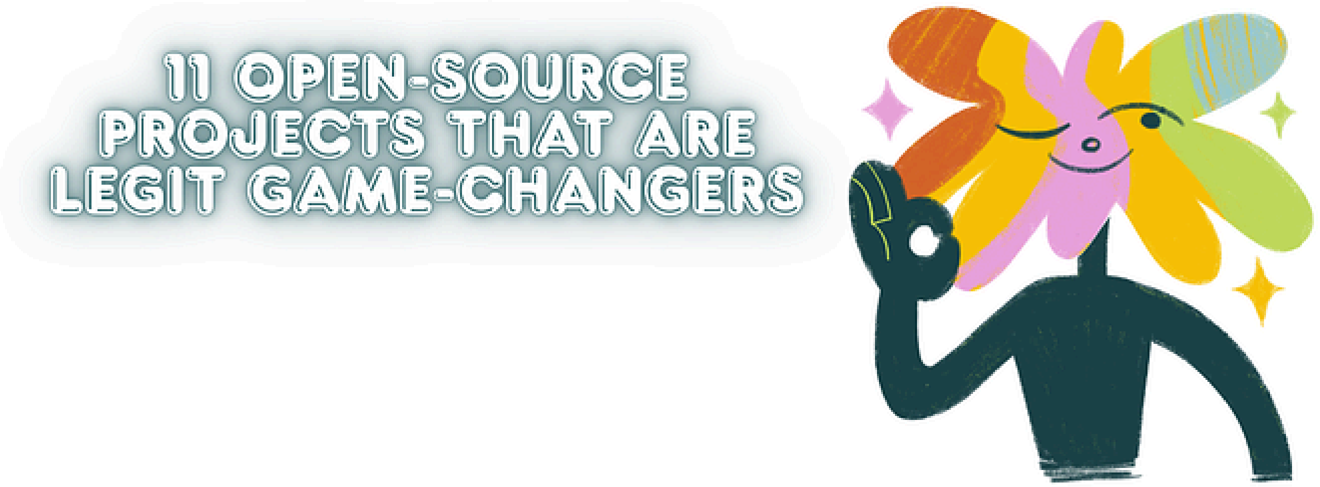


In AI Advances by Debmalya Biswas

Agentic AI for Data Engineering

Reimagining Enterprise Data Management leveraging AI Agents

Mar 23 514 12



In Let's Code Future by Let's Code Future

11 Open-Source Projects That are Legit Game-Changers

Really unique and valuable

Mar 24 968 14

See more recommendations