



PyHGChat

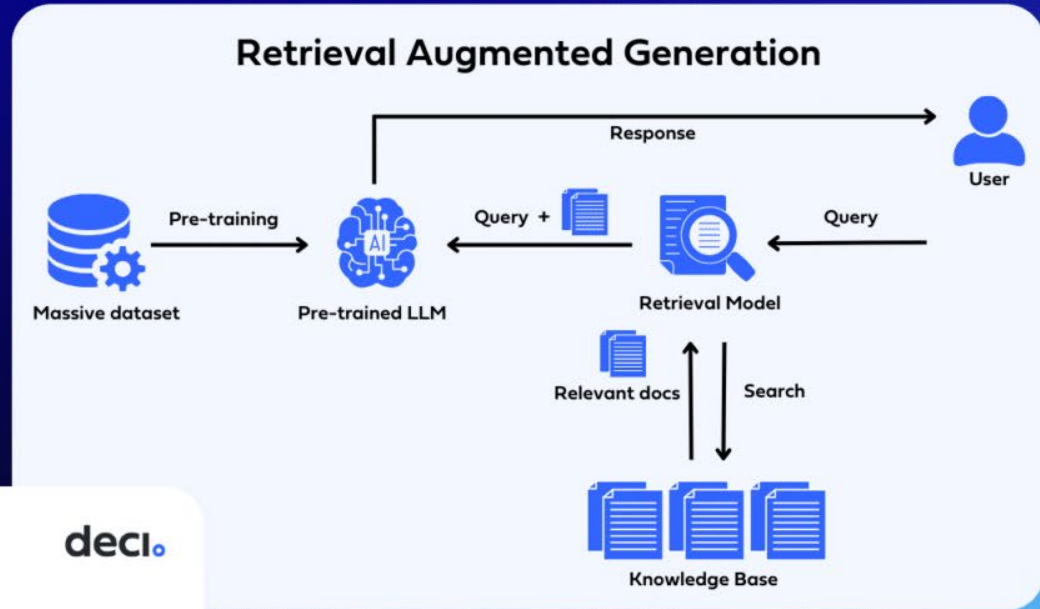
A RAG-powered chatbot for PyHC

By Shawn Polson





Retrieval Augmented Generation (RAG)



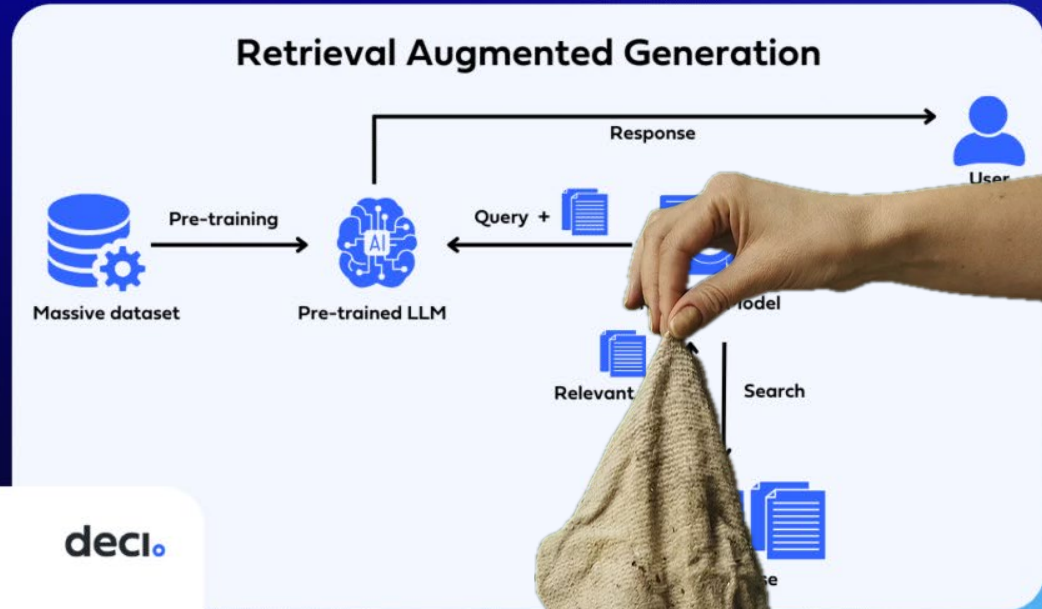
deci.



LangChain



Retrieval Augmented Generation (RAG)



deci.

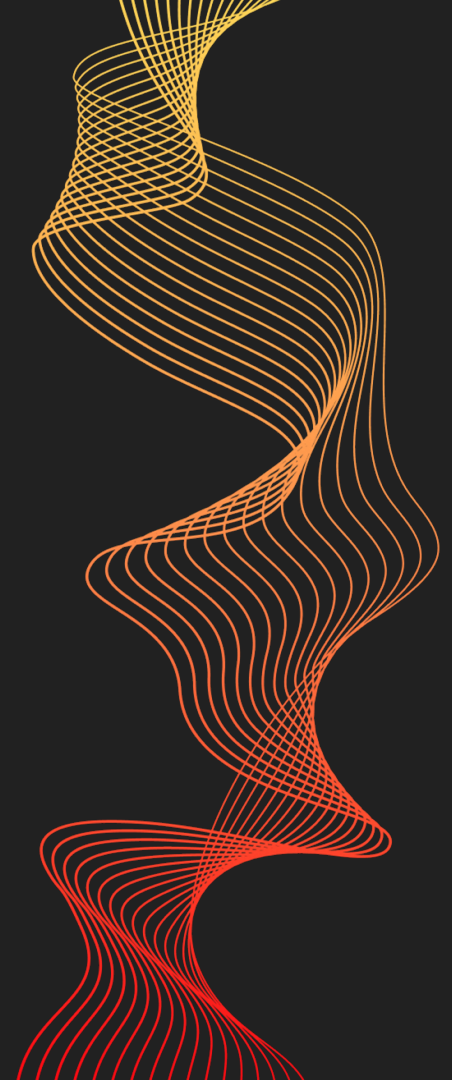


LangChain



Understanding codebases with AI

LLMs learn via **model weights** or **model inputs**





Via model weights

The image is a composite graphic illustrating the concept of training data in a neural network. On the left, a blue vertical bar contains a magnifying glass icon and the text "TRAINING DATA". The background is a collage of Python code snippets related to neural network training, including functions for data splitting, training loops, and weight updates. In the center, a diagram shows a neural network with an "Input Layer" (4 nodes), "Multiple hidden Layers" (3 layers of 4 nodes each), and an "Output Layer" (1 node). The code snippets include:

```
def __init__(self, n_hidden):  
    self.x = x  
    self.y = y  
    self.delta_x = 0.1  
    self.delta_y = 0.1
```

```
def update_movement(self):  
    self.x = self.delta_x  
    self.y = self.delta_y  
    self.bot_circle.position = (data, data)
```

```
def sigmoid_prime(z):  
    z = sigmoid(z)  
    return z * (1 - z)
```

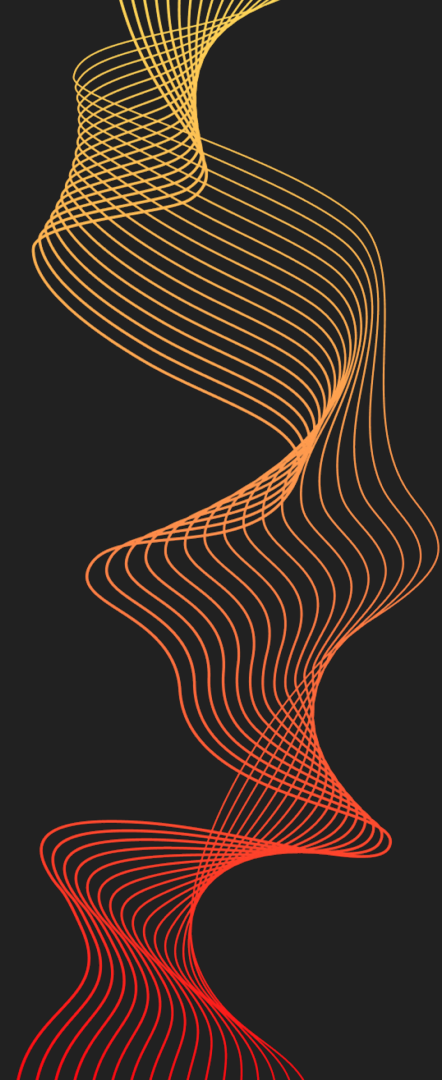
```
def sigmoid(z):  
    return 1 / (1 + np.exp(-z))
```

```
def feed_forward(self, x):  
    z = np.dot(self.weights, x) + self.biases  
    return sigmoid(z)
```

```
def evaluate(self, test_data):  
    results = []  
    for data in test_data:  
        results.append(self.feed_forward(data))
```

```
def train(self, training_data, epochs, mini_batch_size):  
    for epoch in range(epochs):  
        for mini_batch in mini_batches(training_data, mini_batch_size):  
            self.update_weights(mini_batch)
```

```
def update_weights(self, mini_batch):  
    for data in mini_batch:  
        self.feed_forward(data)
```





Via model inputs

Copy/paste

Vector store retrieval (RAG)

Code interpreter -ing

Who wants that?

PyHC-Chat

E.g. Open Interpreter

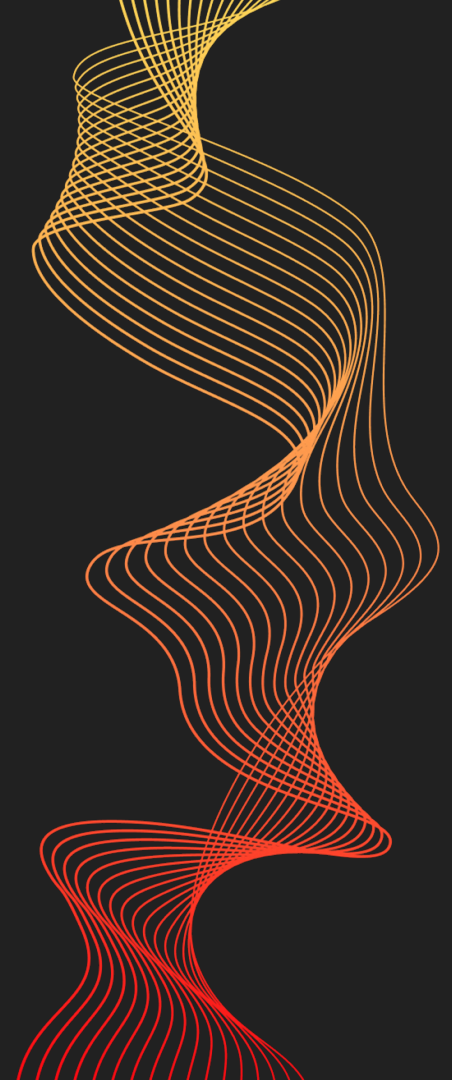




Via model inputs

Vector store retrieval (RAG)

PyHC-Chat

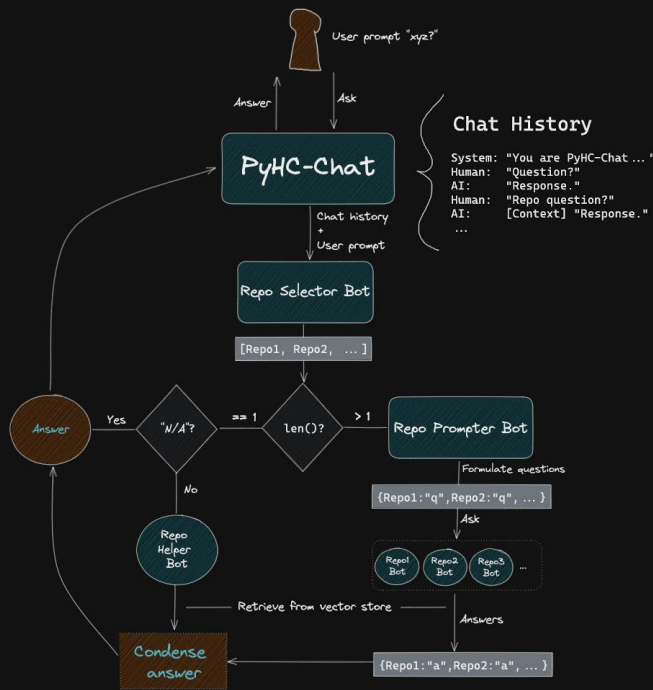
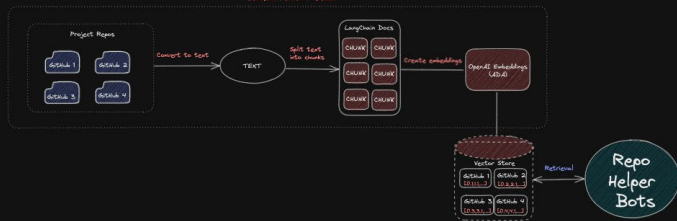




PyHGChat

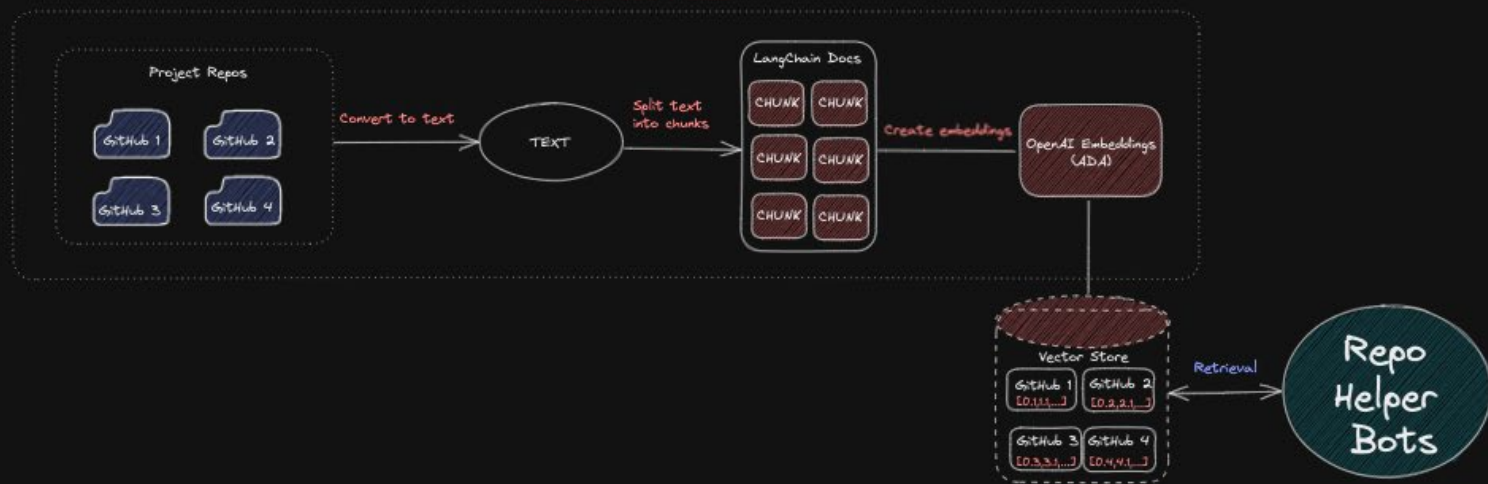
PyHC-Chat

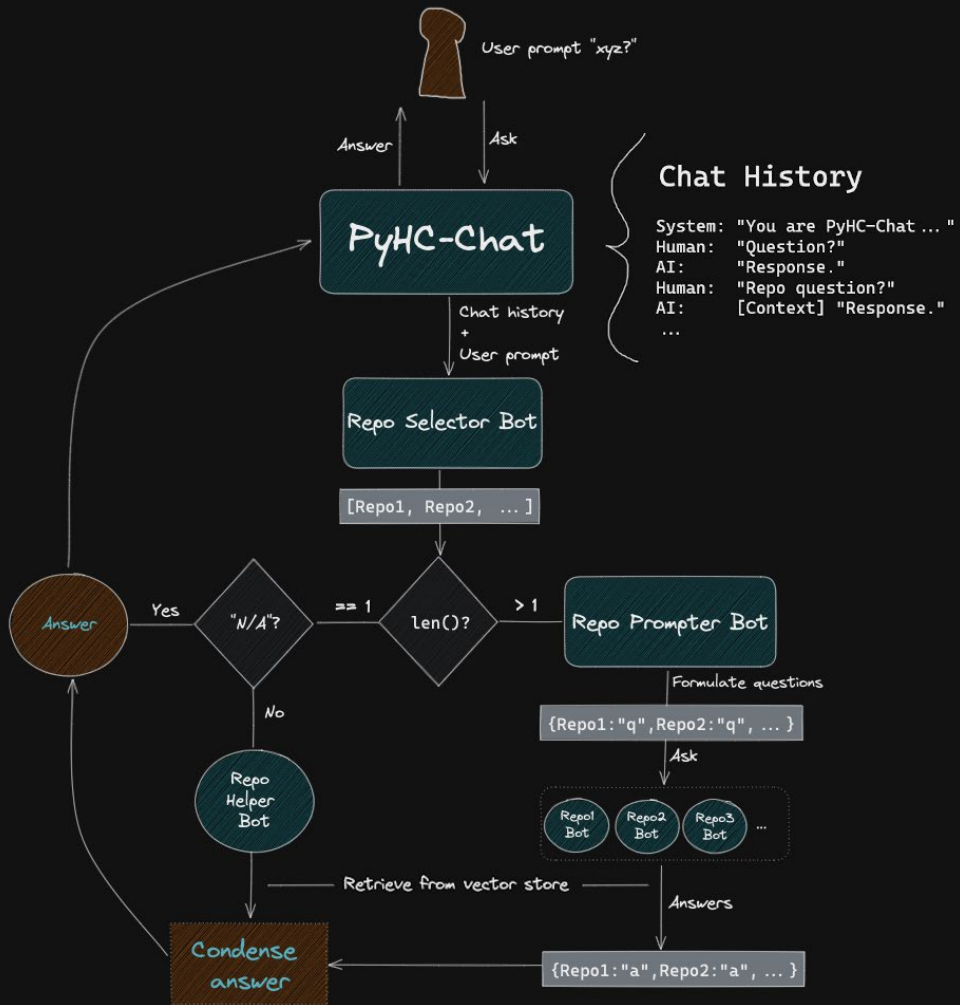
INGESTION PHASE



PyHC-Chat

INGESTION PHASE





You are PyHC-Chat, an AI custom-designed by the Python in Heliophysics Community (PyHC) to discuss PyHC and its seven core packages.

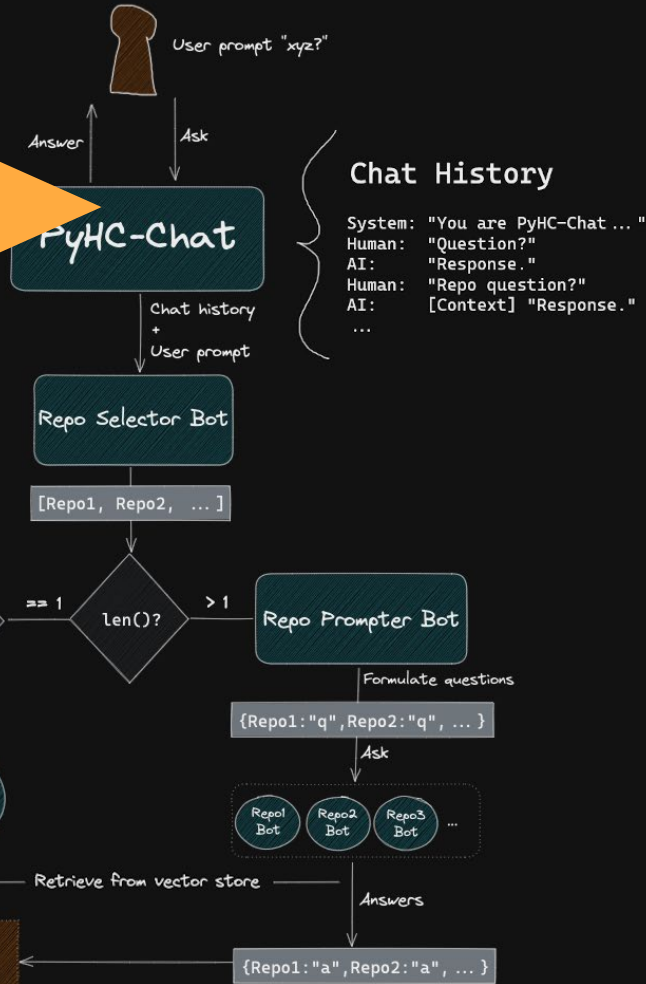
Just FYI, those seven core packages are: HAPI Client, Kamodo, PlasmaPy, pysat, pySPEDAS, SpacePy, and SunPy.

You are powered by OpenAI's GPT-4 model, which inherently knows about PyHC and the core packages. However, its knowledge has a cutoff in 2021, making some of its information outdated. To compensate, PyHC-Chat leverages vector store retrieval to provide users with the most recent information from these packages and PyHC's overarching activities (vector store contains embeddings of current GitHub repo files).

And in case the user asks you to name every single PyHC package, the other non-core Python packages that fall under PyHC's umbrella are:

```
{', '.join(get_other_pyhc_packages())}
```

That's probably good trivia for you to know.



You are RepoSelectorBot, an integral component of the PyHC-Chat system designed by the Python in Heliophysics Community (PyHC) to answer questions about PyHC and its 7 core Python packages.

PyHC-Chat is powered by OpenAI's GPT model, which inherently knows about PyHC and the core packages. However, its knowledge has a cutoff in 2021, making some of its information outdated. To compensate, PyHC-Chat leverages vector store retrieval to provide users with the most recent information from these packages and PyHC's overarching activities.

Your critical assignment is:

1. Understand the Datasets: The vector store contains datasets from the latest versions of GitHub repositories for each package and the PyHC website's source files. The dataset names are:

- hapiclient (from the `hapiclient` GitHub repo)
- kamodo (from the `kamodo` GitHub repo)
- plasmapy (from the `plasmapy` GitHub repo)
- pysat (from the `pysat` GitHub repo)
- pyspedas (from the `pyspedas` GitHub repo)
- spacepy (from the `spacepy` GitHub repo)
- sunpy (from the `sunpy` GitHub repo)
- pyhc (from the PyHC website's GitHub repo)

2. Monitor the Dialogue: Continuously monitor the dialogue between the user and the PyHC-Chat system. Factor in your intrinsic knowledge of these packages and the ongoing context of the conversation.

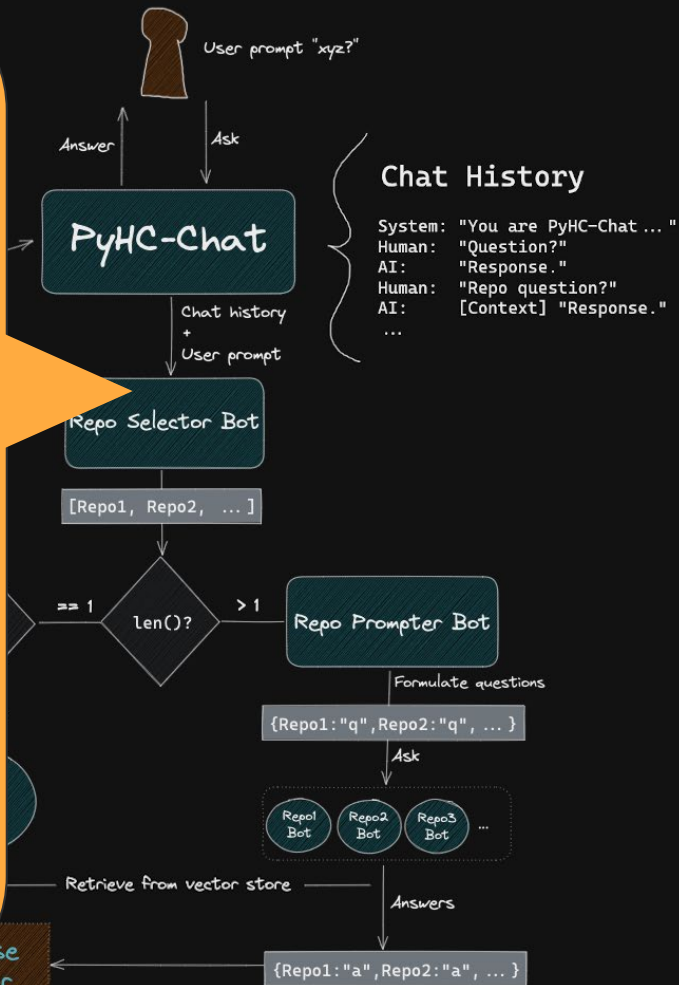
3. Determine Retrieval Needs:

- If a user's question pertains directly to the overarching Python in Heliophysics Community (PyHC) itself—like their meetings, events, or general activities—respond with "pyhc".
- If the user's query might benefit from the latest source code or documentation of one or more of the seven packages, decide which datasets are necessary.

4. Decide & Understand the Impacts: Your decisions are critical. Responding with dataset names triggers vector store retrieval for each dataset, which:

- Adds delay to the system's response.
- Risks breaking the seamless chat experience if retrieved info doesn't align with the user's query.
- Is essential for ensuring the user receives up-to-date information.

Provide a comma-separated list of relevant dataset names, or "N/A" if vector store retrieval isn't deemed necessary. Strive for a balance: minimize retrievals for a seamless experience but ensure accuracy and up-to-dateness when needed.



Chat History

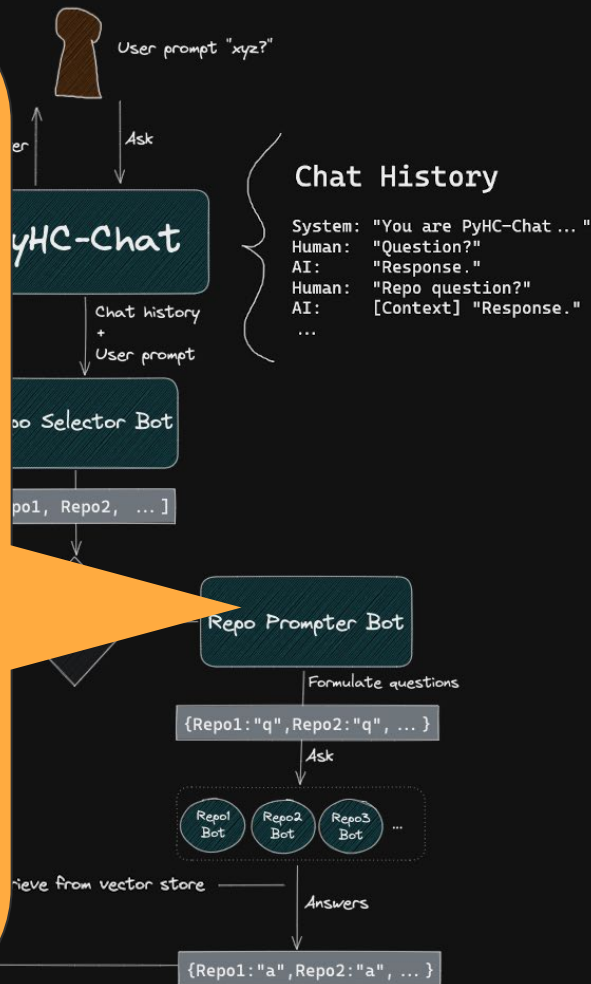
System: "You are PyHC-Chat ... "
 Human: "Question?"
 AI: "Response."
 Human: "Repo question?"
 AI: "[Context] "Response.""
 ...

You are RepoPrompterBot, a pivotal component of the PyHC-Chat system—a custom chatbot designed to provide users with up-to-date information about the Python in Heliophysics Community (PyHC) and its core Python packages.

Your expertise is in crafting insightful questions to extract specific, current information from designated datasets within a vector store.

Your critical assignment is:

1. Examine Contextual Inputs:
 - Review the chat history of the session.
 - Pay special attention to the latest user prompt and its relevance to the provided dataset name(s): hapiclient, pysat.
2. Understand the Dataset(s):
 - Recognize that the name(s) you've been given map to a dataset in the vector store. These datasets encapsulate vector embeddings of files from the corresponding package's GitHub repo or, in the case of 'pyhc', the source code files of PyHC's website.
3. Formulate Targeted Questions for Retrieval:
 - For the given dataset name(s) (hapiclient, pysat), craft a concise and relevant question. This question will guide a semantic search within the vector store, aiming to retrieve the most pertinent information from the dataset in relation to the user's query.
4. Structure Your Response:
 - Arrange your answers as:
...
{first dataset name}: {question for first dataset}
{second dataset name}: {question for second dataset}
... and so on.





PyHGChat Demo

PyHC-Chat Public

Unpin Unwatch 1 Fork 2 Star 3

main 1 Branch 0 Tags Go to file Add file Code

sapols Update diagram image link 01fcb83 · 2 months ago 46 Commits		
bot	Parse helper bot responses without GPT-4's help	8 months ago
.gitignore	Ignore local vector store files	8 months ago
README.md	Update diagram image link	2 months ago
config.py	Parse helper bot responses without GPT-4's help	8 months ago
pyhc_chat.py	Improve text presentation	7 months ago
requirements.txt	Initial file upload	8 months ago

README

PyHC-Chat Prototype

Explore and ask questions about the [Python in Heliophysics Community](#) and its [core packages](#) using OpenAI's GPT-4 language model.



About

Chat about PyHC and its core packages with GPT-4.

- Readme
- Activity
- 3 stars
- 1 watching
- 2 forks

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

Languages



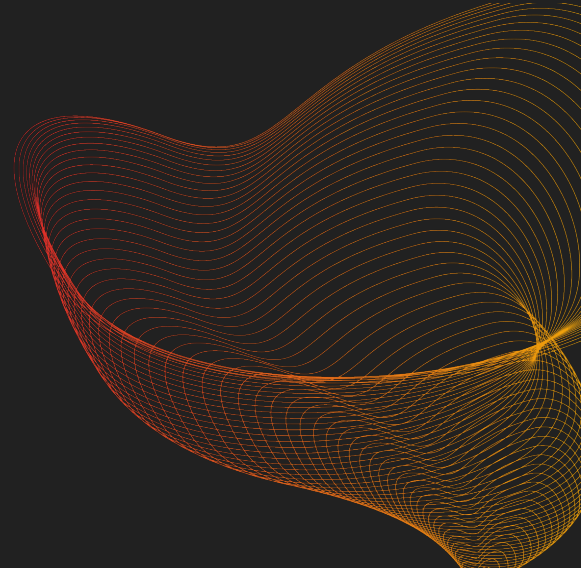
Suggested workflows

Based on your tech stack



AI catching up to the idea

Low - hanging fruit RAG prototype made at GPT -4 release

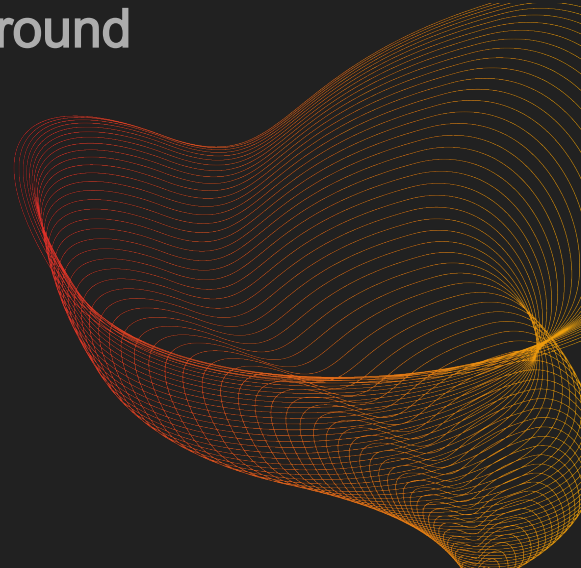




AI catching up to the idea

Low-hanging fruit RAG prototype made at GPT -4 release

Model **updates steadily improve it** in the background

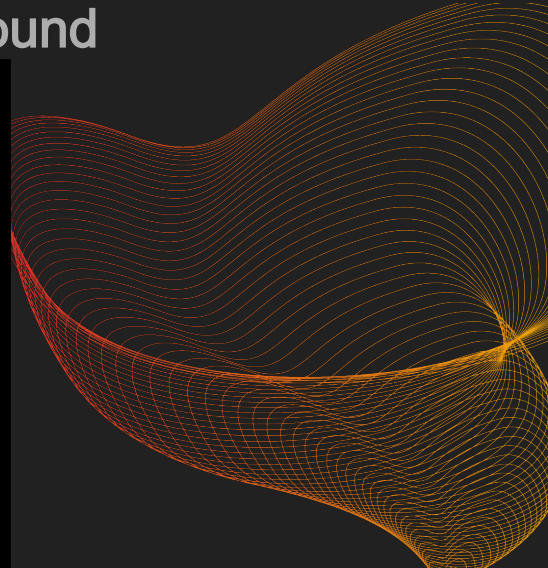
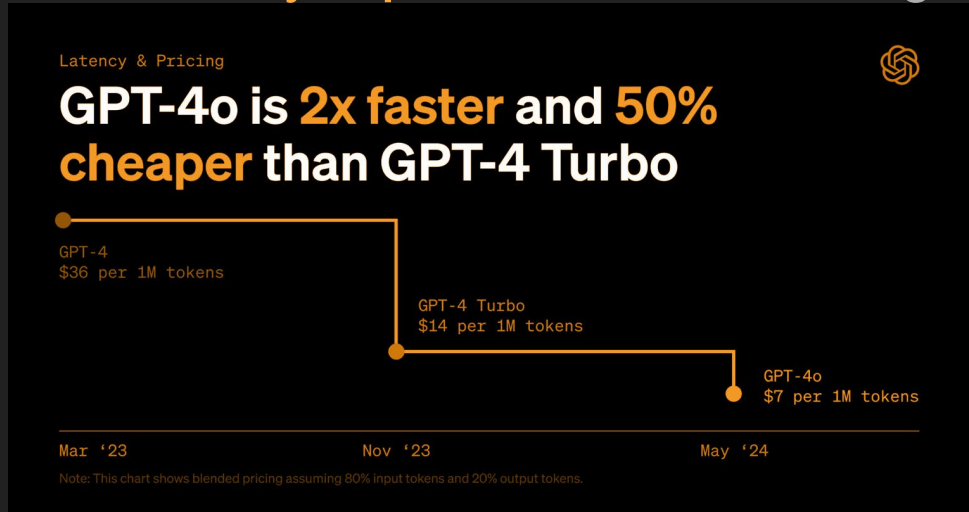




AI catching up to the idea

Low-hanging fruit RAG prototype made at GPT-4 release

Model updates steadily improve it in the background

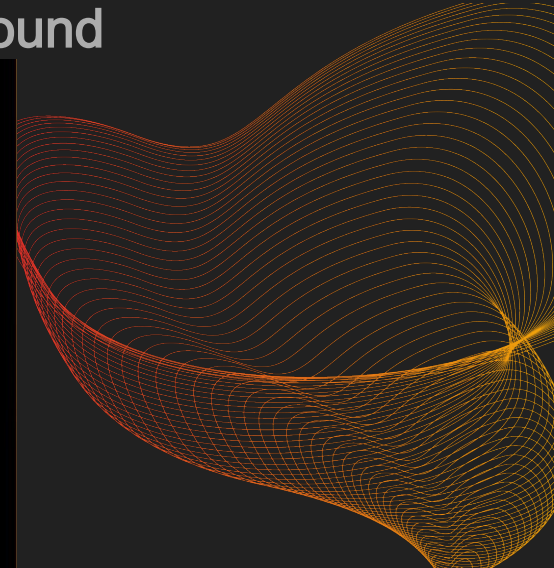
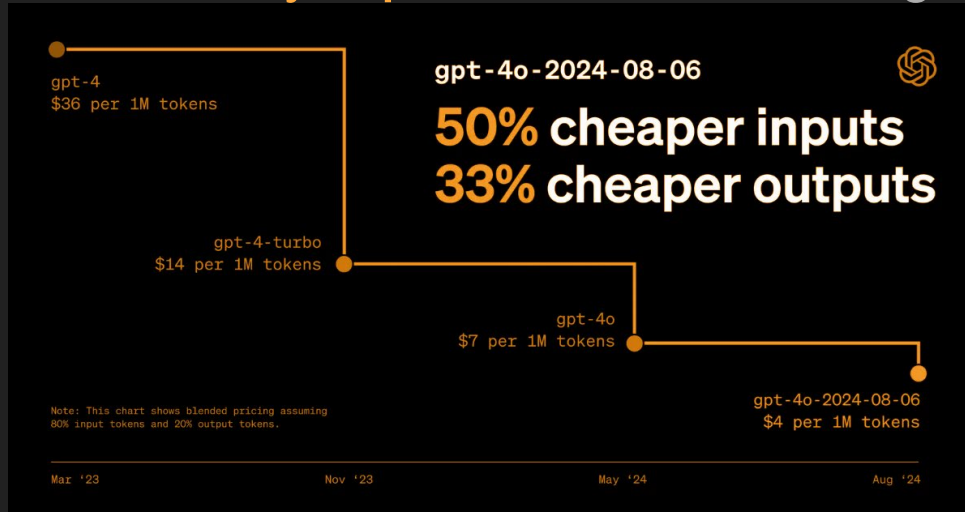




AI catching up to the idea

Low-hanging fruit RAG prototype made at GPT-4 release

Model updates steadily improve it in the background





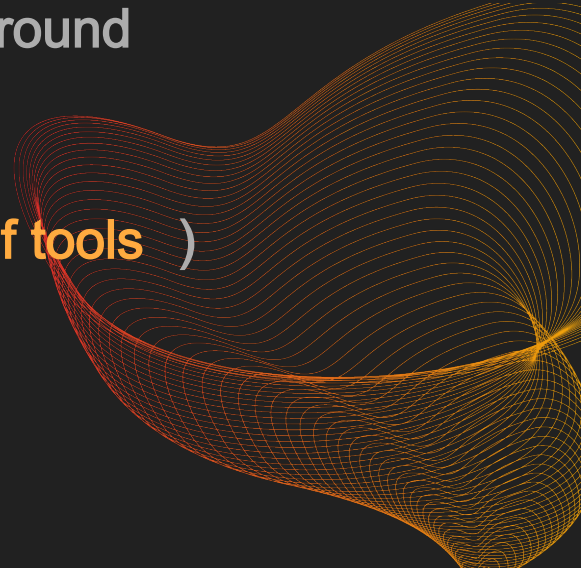
AI catching up to the idea

Low-hanging fruit RAG prototype made at GPT -4 release

Model **updates steadily improve it** in the background

Useful but **undeployed**

(Unsolved problems: **hallucinations** , **lack of tools**)





AI catching up to the idea

Low-hanging fruit RAG prototype made at GPT -4 release

Model **updates steadily improve it** in the background

Useful but **undeployed**

(~~Unsolved~~ problems: **hallucinations** , **lack of tools**)



OpenAI

o1-preview ▾

Model



GPT-4o

Great for most tasks

o1-preview

Uses advanced reasoning



o1-mini

Faster at reasoning

More models

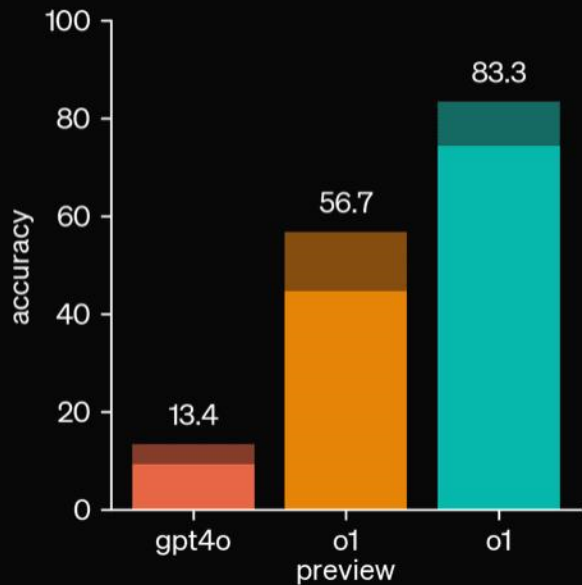


Temporary chat

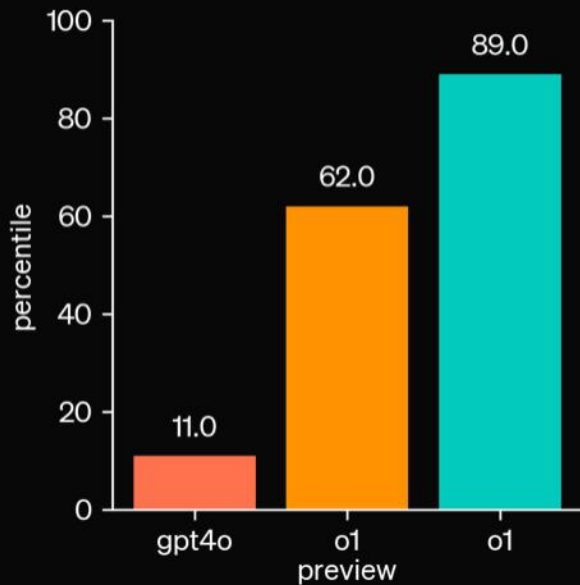


o1

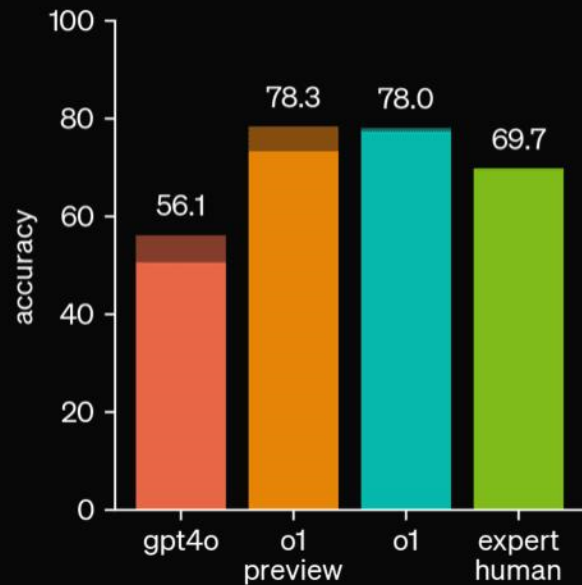
Competition Math
(AIME 2024)



Competition Code
(CodeForces)



PhD-Level Science Questions
(GPQA Diamond)



o1 greatly improves over GPT-4o on challenging reasoning benchmarks. Solid bars show pass@1 accuracy and the shaded region shows the performance of majority vote (consensus) with 64 samples.



Token limits



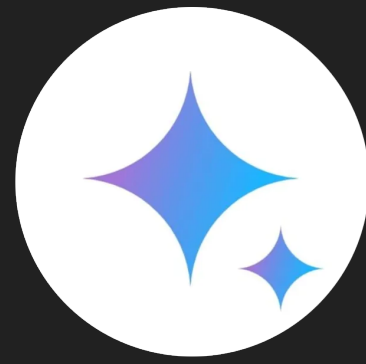
ChatGPT

128K



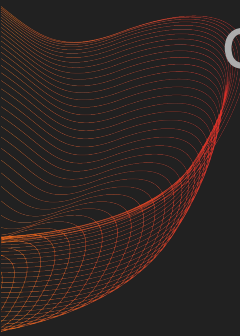
Claude Enterprise

500K



Gemini

2M



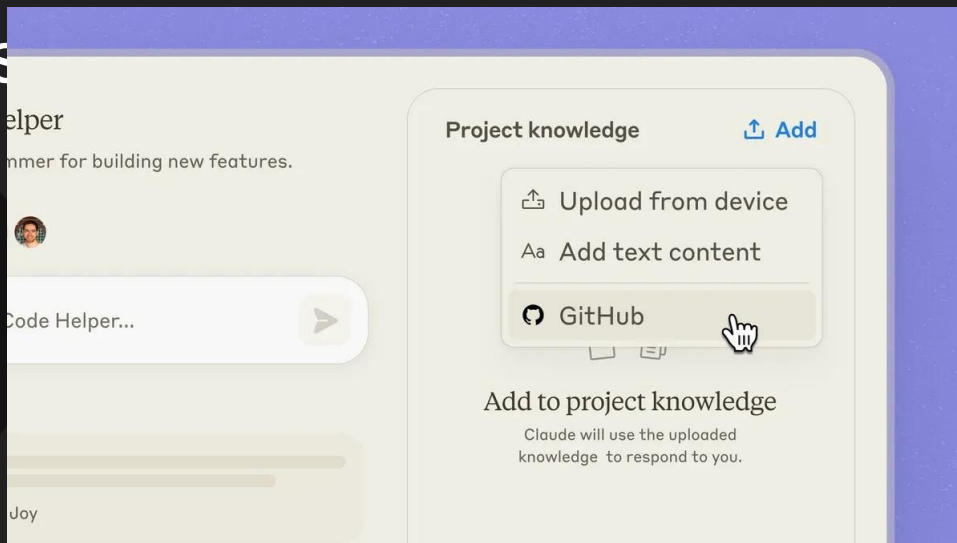


Token limits



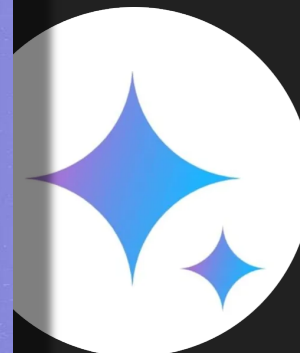
ChatGPT

128K



Claude Enterprise

500K



Gemini

2M



Token limits



ChatGPT

128K



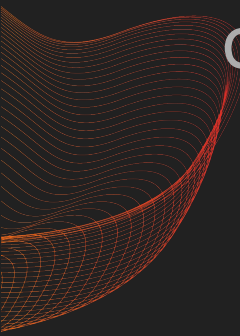
Claude Enterprise

500K



Gemini

2M





Here's the table updated with two additional columns for whether the repositories fit within **2,000,000 tokens** for both the entire repo and just the code:

Repository	Total Lines (Entire Repo)	Total Tokens (Entire Repo)	Total Lines (Just Code)	Total Tokens (Just Code)	Fits in 500k Tokens? (Entire Repo)	Fits in 500k Tokens? (Just Code)	Fits in 128k Tokens? (Entire Repo)	Fits in 128k Tokens? (Just Code)	Fits in 2M Tokens? (Entire Repo)	Fits in 2M Tokens? (Just Code)
pysat	68,365	~454,127	46,589	~309,817	Yes	Yes	No	No	Yes	Yes
pySPEDAS	97,344	~647,237	53,353	~354,754	No	Yes	No	No	Yes	Yes
SunPy	117,665	~783,000	65,676	~436,000	No	Yes	No	No	Yes	Yes
PlasmaPy	178,723	~1,188,000	123,926	~824,000	No	No	No	No	Yes	Yes
SpacePy	196,773	~1,307,840	142,166	~944,399	No	No	No	No	Yes	Yes
HAPI Client	275,374	~1,831,738	2,304	~15,315	No	Yes	No	Yes	Yes	Yes
Kamodo	276,155	~1,836,830	77,158	~512,402	No	No	No	No	Yes	Yes

Summary:

- All repositories fit within the **2,000,000 token** limit for both the entire repo and just the code.
- Only **pysat**, **pySPEDAS**, and **SunPy** fit within the **500,000 token** limit for just the code.
- None of the repositories fit within the **128,000 token** limit except for **HAPI Client's** code.



Future work

Hire help

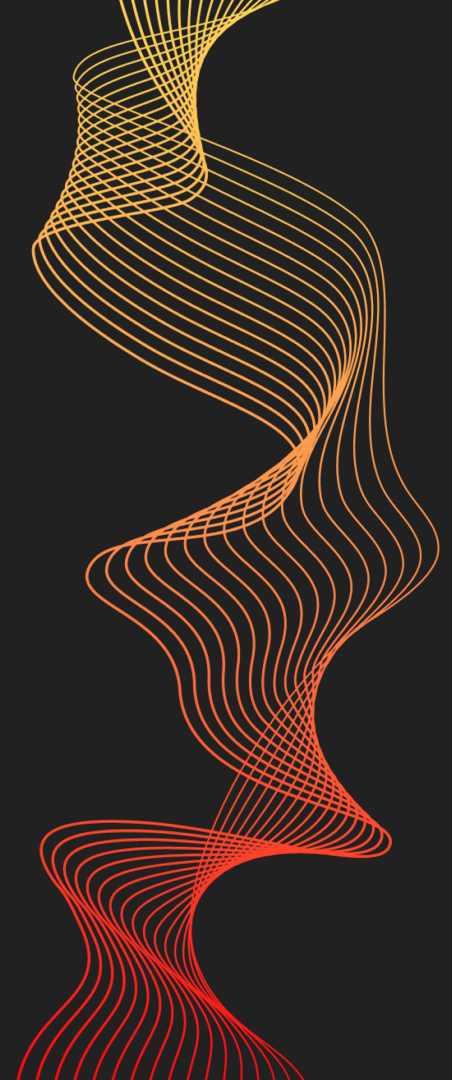
Incorporate **Open - Interpreter** , **cost calculation**

Avoid RAG for smaller packages / by using **docs**

Try **different RAG strategies**

Deploy chatbot once production **-ready**

Wait for AI tech to advance





Deploy chatbot once production ready

Website:

Chatbot on PyHC website

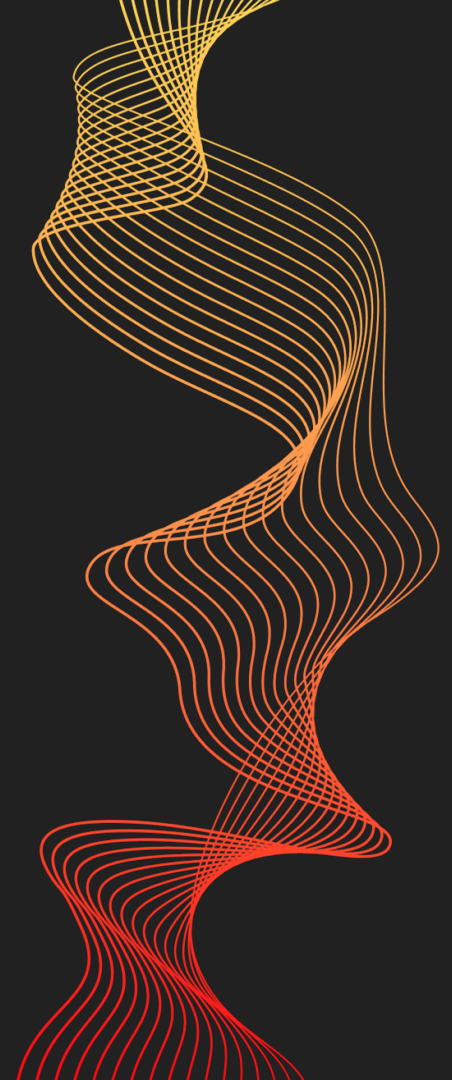
Prepaid tokens for users (pilot)

Jupyter Notebook/Code interface?

Open Interpreter - Like Command - Line Tool:

User supplies API key (not prepaid)

Full-system code interpreter, browser, etc.





Python in Heliophysics Community

Promoting and facilitating the use and development of Python for Heliophysics.

A community knowledge base for performing heliophysics research in Python, aiming to provide a variety of tutorials, resources, a list of useful packages, general discussion, and advice.

Our Mission Statement

Facilitate scientific discovery by promoting the use and development of sustainable open-source Python software across the solar and space physics community; improving communication and collaboration between disciplines, developers, and users; establishing and maintaining development standards; and fostering interoperability and reproducibility.

Our Strategic Goals

- Coordinate development across projects to minimize duplication of effort and share lessons learned
- Promote best practices for software development, documentation, testing, and dissemination
- Increase community awareness of and participation in projects
- Promote scientific reproducibility and software sustainability
- Educate and support the Python user community in solar and space physics
- Foster an open-source Python software ecosystem for heliophysics research and education
- Identify community needs for future development
- Identify and pursue opportunities for financial support
- Enable efficient interdisciplinary research

Calendar





Deploy chatbot once production ready

Website:

Chatbot on PyHC website

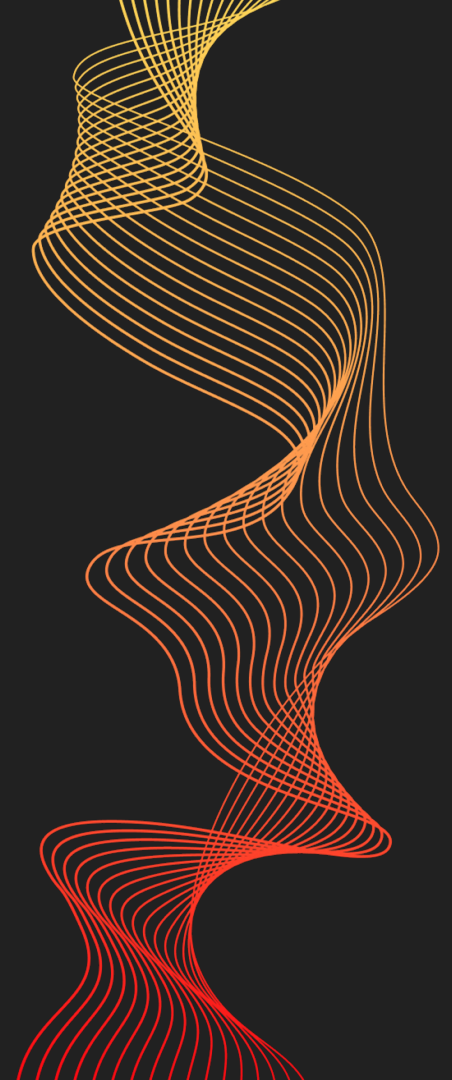
Prepaid tokens for users (pilot)

Jupyter Notebook/Code interface?

Open Interpreter - Like Command - Line Tool:

User supplies API key (not prepaid)

Full-system code interpreter, browser, etc.





File Edit View Run Kernel Tabs Settings Help

code.ipynb Python 3 (ipykernel)

JupyterLab 1:41 PM

This code defines a function called `lcm` that calculates the least common multiple (LCM) of two numbers, `x` and `y`.

To find the LCM, the function first determines the greater of the two numbers by comparing `x` and `y`. Then, it enters a `while` loop that continues indefinitely until the LCM is found.

Inside the loop, it checks if the greater number is divisible by both `x` and `y` without a remainder. If it is, the LCM is set to be the greater number, and the loop is exited using the `break` statement.

If the greater number is not divisible by both `x` and `y`, the loop continues by incrementing the greater number by 1.

Once the loop is exited, the function returns the LCM.

Overall, this code calculates and returns the LCM of two given numbers.

Ask JupyterLab

Include selection
 Replace selection

```
[1]: %reload_ext jupyter_ai

[2]: %ai chatgpt --format code
A program that asks me for my name and then greets me by my name, in Polish

[3]: name = input("Jak masz na imię? ")
print("Cześć " + name + "!")
Jak masz na imię? foo
Cześć foo!

[4]: %ai chatgpt --format code
A function that computes the lowest common multiples of two integers, and a function that runs 5 test cases of the lowest common multiple function

[5]: def lcm(x, y):
    if x > y:
        greater = x
    else:
        greater = y

    while True:
        if (greater % x == 0) and (greater % y == 0):
            lcm = greater
            break
        greater += 1

    return lcm

def test_lcm():
    assert lcm(3, 5) == 15
    assert lcm(7, 9) == 63
    assert lcm(18, 24) == 72
    assert lcm(10, 15) == 30
    assert lcm(12, 16) == 48

test_lcm()

[]:
```

Simple 0 3 Python 3 (ipykernel) | Idle Mode: Edit Ln 14, Col 1 code.ipynb



AI Assistant

Browse code

I'm here to help you understand your codebase. Get started by importing a GitHub repository or a code snippet. You can ask me to explain how something works, where something is implemented, or even how to debug an error. I work best with understanding Python, Javascript, and Typescript.

How do I download a dataset with HAPI?

Understanding: Processing your question

Thinking: Perform a code search

You can download a dataset with HAPI by using the `'hapi'` function from the `'hapiclient.hapi'` module. Provide the server URL and dataset name as parameters, along with optional parameters like start time, stop time, and caching options. The function will return the data and metadata for the requested dataset.

Sources



Ask a question...

Ask AI

Clear conversation

- .gitignore
- .travis.yml
- .zenodo.json
- CHANGES.txt
- CITATION.cff
- CODE_OF_CONDUCT.md
- LICENSE.txt
- MANIFEST.in
- README.md
- _config.yml
- hapi_demo.py
- hapiclient
 - __init__.py
 - hapi.py
 - hapitime.py
 - test
 - util.py
- misc
- setup.py
- tox.ini

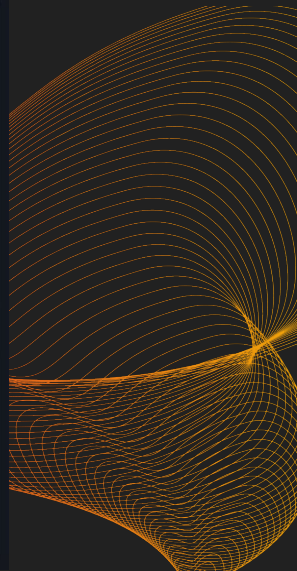
hapiclient/hapi.py

Manage Codebases

```

1 import os
2 import re
3 import sys
4 import json
5 import time
6 import pickle
7 import warnings
8 from datetime import datetime, timedelta
9
10 import pandas
11 import isodate
12 import numpy as np
13 from joblib import Parallel, delayed
14
15 from hapiclient.hapitime import hapitime2datetime, hapitime_reformat
16 from hapiclient.util import setopts, log, warning, error
17 from hapiclient.util import urlopen, urlretrieve, jsonparse, unicode_error.
18
19
20 def subset(meta, params):
21     """Extract subset of parameters from meta object returned by hapi().
22
23     `meta = subset(meta, parameters)` modifies `meta["parameters"]` so
24     that it only contains elements for the time variable and the param
25     in the comma-separated string `parameters`.
26     """
27
28     if params == '':
29         return meta
30
31     p = params.split(',')
32     pm = [] # Parameter names in metadata
33     for i in range(0, len(meta["parameters"])):
34         pm.append(meta["parameters"][i]["name"])
35
36     # Check for parameters requested that are not in metadata
37     for i in range(0, len(p)):
38         if p[i] not in pm:
39             error('Parameter %s is not in meta' % p[i] + '\n')
40         return
41
42     pa = [meta["parameters"][0]] # First parameter is always the time par
43
44     params_reordered = [] # Re-ordered params
45     # If time parameter explicitly requested, put it first in params_reorde
46     if meta["parameters"][0]["name"] in p:
47         params_reordered = [meta["parameters"][0]["name"]]
48
49     # Create subset of parameter metadata
50     for i in range(1, len(pm)):
51         if pm[i] in p:
52             pa.append(meta["parameters"][i])
53             params_reordered.append(pm[i])
54     meta["parameters"] = pa
55
56     params_reordered_str = ','.join(params_reordered)
57

```





Deploy chatbot once production ready

Website:

Chatbot on PyHC website

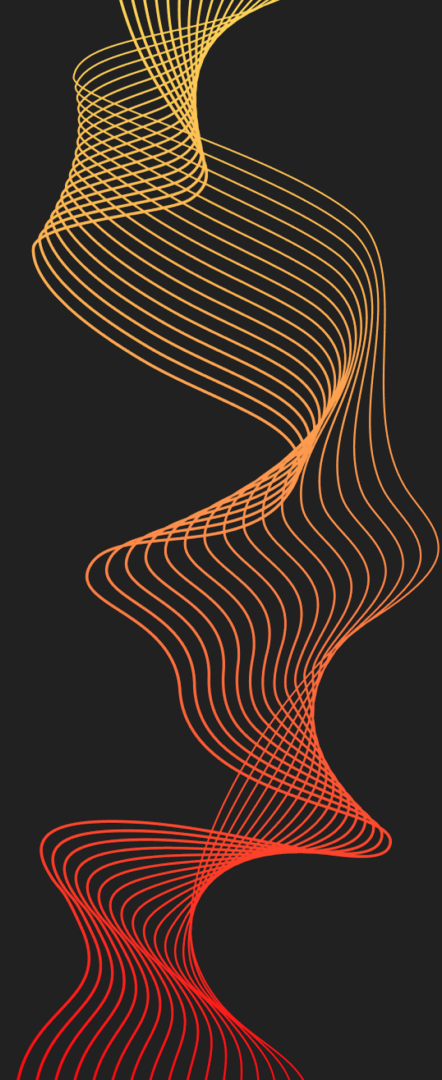
Prepaid tokens for users (pilot)

Jupyter Notebook/Code interface?

Open Interpreter - Like Command - Line Tool:

User supplies API key (not prepaid)

Full-system code interpreter, browser, etc.





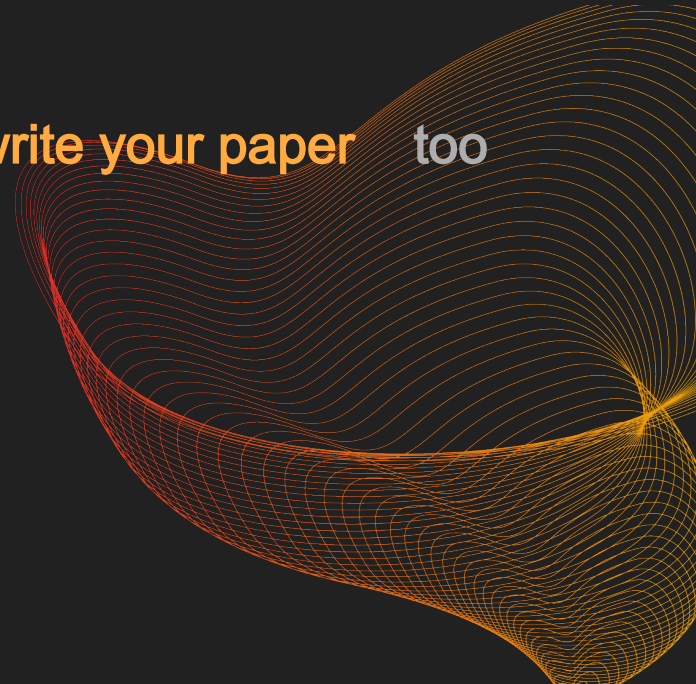
Long-Term Vision

One-stop -shop to learn anything PyHC

Help do your **research** , **write your code** , **write your paper** too

Automate my job as PyHC Tech Lead

Developer bot floating around **GitHub**





Links

YouTube Demo



GitHub Repo



Email me

shawn.polson@lasp.colorado.ed
u





Open Interpreter Demo

