**How to use Fibers:**

1. Data has to be in the format of python functions, each node in the knowledge tree need to be one python function. Note that each function will require comments describing the function in order for the llm to summarize them and creating the tree, below are guides on how to add comments, note that they have to follow the format correctly to be parsed by the program:

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图形用户界面, 文本, 应用程序, 电子邮件, Teams

描述已自动生成

Details can be found here: <https://github.com/EvoEvolver/Moduler/blob/main/README.md>

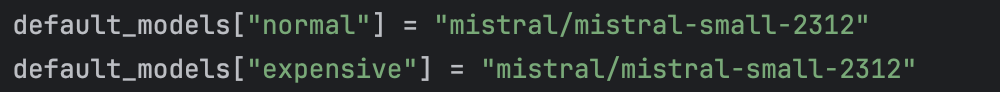
1. Fibers use litellm which supports a range of LLM providers such as openai, replicate, mistral etc. All available providers can be found here: <https://litellm.vercel.app/docs/providers/>. each provider will require a different api key to run their model, and you should add the api key of that provider in the environment before running the queries
2. The code below is used to create the knowledge tree and then run the user query, note that “photonic\_component” is the name of the python file that the llm will use to generate the tree. You could also change the LLM by changing the “default\_models” parameter, or use a function to automatically switch LLM

Also note that if the results weren’t good enough, or there are errors when executing the code, try delete “.llm\_cache” folder and re-run the program

图形用户界面, 文本

中度可信度描述已自动生成

To manually change LLM:



By default, the program runs the “expensive” model, and you can check on litellm (<https://litellm.vercel.app/docs/providers/>) for a specific model

Python functions that switch to different LLMs:

| def set\_default\_to\_google():  default\_models["normal"] = "gemini-1.0-pro"  default\_models["expensive"] = "gemini-1.0-pro"  default\_models["vision"] = "gemini-1.0-pro-vision"  default\_models["embedding"] = "vertex\_ai/textembedding-gecko"  def set\_default\_to\_openai(use\_gpt\_4=False):  check\_openai\_env()  default\_models["normal"] = "gpt-3.5-turbo"  default\_models["expensive"] = "gpt-4-turbo-preview" if not use\_gpt\_4 else "gpt-4"  default\_models["vision"] = "gpt-4-vision-preview"  default\_models["embedding"] = "text-embedding-3-large"  def set\_default\_to\_anthropic(expensive\_vision\_model=False):  check\_anthropic\_env()  default\_models["normal"] = "claude-3-sonnet-20240229"  default\_models["expensive"] = "claude-3-opus-20240229"  default\_models["vision"] = "claude-3-sonnet-20240229" if not expensive\_vision\_model else "claude-3-opus-20240229"  default\_models["embedding"] = "text-embedding-3-large"  def set\_default\_to\_llama():  check\_replicate\_env()  default\_models["normal"] = "replicate/meta/llama-2-13b-chat"  default\_models["expensive"] = "replicate/meta/llama-2-70b-chat"  default\_models["vision"] = "gpt-4-vision-preview"  default\_models["embedding"] = "text-embedding-3-large"  def set\_default\_to\_mistral():   check\_replicate\_env()  default\_models["normal"] = "replicate/mistralai/mixtral-8x7b-instruct-v0.1"  default\_models["expensive"] = "replicate/mistralai/mixtral-8x7b-instruct-v0.1"  default\_models["vision"] = "gpt-4-vision-preview"  default\_models["embedding"] = "text-embedding-3-large" |
| --- |