```
!pip install groq
```

```
vivek
 vivek is 42 years old
Collecting groq
  Downloading groq-0.30.0-py3-none-any.whl.metadata (16 kB)
Requirement already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.11/dist-packages (from groq) (4.9.0)
Requirement already satisfied: distro<2,>=1.7.0 in /usr/local/lib/python3.11/dist-packages (from groq) (1.9.0)
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.11/dist-packages (from groq) (0.28.1)
Requirement already satisfied: pydantic<3,>=1.9.0 in /usr/local/lib/python3.11/dist-packages (from groq) (2.11.7)
Requirement already satisfied: sniffio in /usr/local/lib/python3.11/dist-packages (from groq) (1.3.1)
Requirement already satisfied: typing-extensions<5,>=4.10 in /usr/local/lib/python3.11/dist-packages (from groq) (4.14.1)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packages (from anyio<5,>=3.5.0->groq) (3.10)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from httpx<1,>=0.23.0->groq) (2025.7.9)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packages (from httpx<1,>=0.23.0->groq) (1.0.9)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packages (from httpcore==1.*->httpx<1,>=0.23.0->groq) (0
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<3,>=1.9.0->groq) (0
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dist-packages (from pydantic<3,>=1.9.0->groq) (2.3
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<3,>=1.9.0->groq)
Downloading groq-0.30.0-py3-none-any.whl (131 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 131.1/131.1 kB 4.1 MB/s eta 0:00:00
Installing collected packages: groq
Successfully installed groq-0.30.0
Collecting pyjokes
  Downloading pyjokes-0.8.3-py3-none-any.whl.metadata (3.4 kB)
Downloading pyjokes-0.8.3-py3-none-any.whl (47 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 47.6/47.6 kB 2.1 MB/s eta 0:00:00
Installing collected packages: pyjokes
Successfully installed pyjokes-0.8.3
```

```
!pip install groq
from groq import Groq
import os
# Ask for the API key securely (won't be visible in Colab output)
api_key = input("Enter your Groq API key: ")

# Initialize the client
client = Groq(api_key=api_key)
def chat_with_groq(prompt):
    response = client.chat.completions.create(
        model="llama3-70b-8192",  # or "mixtral-8x7b-32768"
        messages=[
            {"role": "system", "content": "You are a helpful assistant."},
            {"role": "user", "content": prompt}
        ],
        temperature=0.7
    )
    return response.choices[0].message.content
while True:
    user_input = input("You: ")
    if user_input.lower() in ['exit', 'quit']:
        print("Goodbye!")
        break
    reply = chat_with_groq(user_input)
    print("Assistant:", reply)
```

```
Requirement already satisfied: groq in /usr/local/lib/python3.11/dist-packages (0.30.0)
Requirement already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.11/dist-packages (from groq) (4.9.0)
Requirement already satisfied: distro<2,>=1.7.0 in /usr/local/lib/python3.11/dist-packages (from groq) (1.9.0)
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.11/dist-packages (from groq) (0.28.1)
Requirement already satisfied: pydantic<3,>=1.9.0 in /usr/local/lib/python3.11/dist-packages (from groq) (2.11.7)
Requirement already satisfied: sniffio in /usr/local/lib/python3.11/dist-packages (from groq) (1.3.1)
Requirement already satisfied: typing-extensions<5,>=4.10 in /usr/local/lib/python3.11/dist-packages (from groq) (4.14.1)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packages (from anyio<5,>=3.5.0->groq) (3.10)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from httpx<1,>=0.23.0->groq) (2025.7.9)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packages (from httpx<1,>=0.23.0->groq) (1.0.9)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packages (from httpcore==1.*->httpx<1,>=0.23.0->groq) (0
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<3,>=1.9.0->groq) (0
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dist-packages (from pydantic<3,>=1.9.0->groq) (2.3
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<3,>=1.9.0->groq)
```

```
!pip install groq
from groq import Groq
import os
# Ask for the API key securely (won't be visible in Colab output)
api_key = input("Enter your Groq API key: ")
```

```
# Initialize the client
client = Groq(api_key=api_key)
def chat_with_groq(prompt):
    response = client.chat.completions.create(
        model="llama3-70b-8192",  # or "mixtral-8x7b-32768"
        messages=[
            {"role": "system", "content": "You are a helpful assistant."},
            {"role": "user", "content": prompt}
        ],
        temperature=0.7
    )
    return response.choices[0].message.content
while True:
    user_input = input("You: ")
    if user_input.lower() in ['exit', 'quit']:
        print("Goodbye!")
        break
    reply = chat_with_groq(user_input)
    print("Assistant:", reply)
```

```
Requirement already satisfied: groq in /usr/local/lib/python3.11/dist-packages (0.30.0)
Requirement already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.11/dist-packages (from groq) (4.9.0)
Requirement already satisfied: distro<2,>=1.7.0 in /usr/local/lib/python3.11/dist-packages (from groq) (1.9.0)
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.11/dist-packages (from groq) (0.28.1)
Requirement already satisfied: pydantic<3,>=1.9.0 in /usr/local/lib/python3.11/dist-packages (from groq) (2.11.7)
Requirement already satisfied: sniffio in /usr/local/lib/python3.11/dist-packages (from groq) (1.3.1)
Requirement already satisfied: typing-extensions<5,>=4.10 in /usr/local/lib/python3.11/dist-packages (from groq) (4.14.1)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packages (from anyio<5,>=3.5.0->groq) (3.10)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from httpx<1,>=0.23.0->groq) (2025.7.9)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packages (from httpx<1,>=0.23.0->groq) (1.0.9)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packages (from httpcore==1.*->httpx<1,>=0.23.0->groq) (
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<3,>=1.9.0->groq) (
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dist-packages (from pydantic<3,>=1.9.0->groq) (2
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<3,>=1.9.0->groq)
Enter your Groq API key: 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
You: what is mountain
Assistant: A mountain is a natural elevation of the Earth's surface rising more than 300 meters (1,000 feet) above the surrounding

Some key characteristics of mountains include:

1. Elevation: Mountains are higher than the surrounding terrain, with a minimum elevation of 300 meters (1,000 feet).
2. Slope: Mountains have a steep slope, which can be rocky, grassy, or forested.
3. Peak: The highest point of a mountain is called the peak or summit.
4. Terrain: Mountains can be rugged, rocky, or snow-capped, and may have valleys, canyons, or other landforms.

Mountains can be classified into different types based on their geological characteristics, such as:

1. Volcanic mountains: Formed by volcanic activity, such as Mount Fuji in Japan.
2. Fold mountains: Formed by the folding of the Earth's crust, such as the Himalayas.
3. Block mountains: Formed by the movement of large blocks of the Earth's crust, such as the Sierra Nevada in California.
4. Dome mountains: Formed by the upwelling of magma, such as the Ozark Mountains in the United States.

Mountains play a crucial role in the environment and human society, providing:

1. Habitat for diverse flora and fauna
2. Water sources, such as rivers and glaciers
3. Mineral resources, such as metals and gemstones
4. Recreation and tourism opportunities, such as hiking, skiing, and climbing
5. Climate regulation, with mountains influencing local and global weather patterns

I hope this helps you understand what a mountain is!
You: what is python
Assistant: Python is a high-level, interpreted programming language that is widely used for various purposes such as web developme

Here are some key features and reasons why Python is so popular:

**Easy to learn**: Python has a simple syntax and is relatively easy to learn, making it a great language for beginners.

** Versatile**: Python can be used for a wide range of applications, including:

        * Web development: Frameworks like Django and Flask make it easy to build web applications.
        * Data analysis and science: Libraries like NumPy, pandas, and scikit-learn provide efficient data manipulation and analys
        * Artificial intelligence and machine learning: TensorFlow, Keras, and scikit-learn provide tools for building AI and ML m
        * Automation: Python can be used to automate tasks, such as data scraping, file management, and system administration.
        * Scientific computing: Python is widely used in scientific computing for tasks like data analysis, numerical simulations,

**Large community**: Python has a large and active community, which means there are many resources available, including:
```

Double-click (or enter) to edit

```
from google.colab import userdata
userdata.get('geminiapikey')
```

```
'AIzaSyAiYVDSubYfW2iBi8WfDRdJRp-yJwlJw88'
```

```
!pip install crewai==0.134.0
```

```
Collecting crewai==0.134.0
  Downloading crewai-0.134.0-py3-none-any.whl.metadata (35 kB)
Collecting appdirs>=1.4.4 (from crewai==0.134.0)
  Downloading appdirs-1.4.4-py2.py3-none-any.whl.metadata (9.0 kB)
Collecting auth0-python>=4.7.1 (from crewai==0.134.0)
  Downloading auth0_python-4.10.0-py3-none-any.whl.metadata (9.2 kB)
Requirement already satisfied: blinker>=1.9.0 in /usr/local/lib/python3.11/dist-packages (from crewai==0.134.0) (1.9.0)
Collecting chromadb>=0.5.23 (from crewai==0.134.0)
  Downloading chromadb-1.0.15-cp39-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (7.0 kB)
Requirement already satisfied: click>=8.1.7 in /usr/local/lib/python3.11/dist-packages (from crewai==0.134.0) (8.2.1)
Collecting instructor>=1.3.3 (from crewai==0.134.0)
  Downloading instructor-1.9.2-py3-none-any.whl.metadata (11 kB)
Collecting json-repair>=0.25.2 (from crewai==0.134.0)
  Downloading json_repair-0.47.7-py3-none-any.whl.metadata (12 kB)
Collecting json5>=0.10.0 (from crewai==0.134.0)
  Downloading json5-0.12.0-py3-none-any.whl.metadata (36 kB)
Collecting jsonref>=1.1.0 (from crewai==0.134.0)
  Downloading jsonref-1.1.0-py3-none-any.whl.metadata (2.7 kB)
Collecting litellm==1.72.0 (from crewai==0.134.0)
  Downloading litellm-1.72.0-py3-none-any.whl.metadata (39 kB)
Collecting onnxruntime==1.22.0 (from crewai==0.134.0)
  Downloading onnxruntime-1.22.0-cp311-cp311-manylinux_2_27_x86_64.manylinux_2_28_x86_64.whl.metadata (4.5 kB)
Requirement already satisfied: openai>=1.13.3 in /usr/local/lib/python3.11/dist-packages (from crewai==0.134.0) (1.94.0)
Requirement already satisfied: openpyxl>=3.1.5 in /usr/local/lib/python3.11/dist-packages (from crewai==0.134.0) (3.1.5)
Collecting opentelemetry-api>=1.30.0 (from crewai==0.134.0)
  Downloading opentelemetry_api-1.35.0-py3-none-any.whl.metadata (1.5 kB)
Collecting opentelemetry-exporter-otlp-proto-http>=1.30.0 (from crewai==0.134.0)
  Downloading opentelemetry_exporter_otlp_proto_http-1.35.0-py3-none-any.whl.metadata (2.3 kB)
Collecting opentelemetry-sdk>=1.30.0 (from crewai==0.134.0)
  Downloading opentelemetry_sdk-1.35.0-py3-none-any.whl.metadata (1.5 kB)
Collecting pdfplumber>=0.11.4 (from crewai==0.134.0)
  Downloading pdfplumber-0.11.7-py3-none-any.whl.metadata (42 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 42.8/42.8 kB 3.3 MB/s eta 0:00:00
Requirement already satisfied: pydantic>=2.4.2 in /usr/local/lib/python3.11/dist-packages (from crewai==0.134.0) (2.11.7)
Collecting python-dotenv>=1.0.0 (from crewai==0.134.0)
  Downloading python_dotenv-1.1.1-py3-none-any.whl.metadata (24 kB)
Collecting pyvis>=0.3.2 (from crewai==0.134.0)
  Downloading pyvis-0.3.2-py3-none-any.whl.metadata (1.7 kB)
Requirement already satisfied: regex>=2024.9.11 in /usr/local/lib/python3.11/dist-packages (from crewai==0.134.0) (2024.11.6)
Requirement already satisfied: tokenizers>=0.20.3 in /usr/local/lib/python3.11/dist-packages (from crewai==0.134.0) (0.21.2)
Collecting tomli-w>=1.1.0 (from crewai==0.134.0)
  Downloading tomli_w-1.2.0-py3-none-any.whl.metadata (5.7 kB)
Collecting tomli>=2.0.2 (from crewai==0.134.0)
  Downloading tomli-2.2.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (11 kB)
Collecting uv>=0.4.25 (from crewai==0.134.0)
  Downloading uv-0.7.21-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (11 kB)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-packages (from litellm==1.72.0->crewai==0.134.0) (3.11.15
Requirement already satisfied: httpx>=0.23.0 in /usr/local/lib/python3.11/dist-packages (from litellm==1.72.0->crewai==0.134.0) (0
Requirement already satisfied: importlib-metadata>=6.8.0 in /usr/local/lib/python3.11/dist-packages (from litellm==1.72.0->crewai=
Requirement already satisfied: jinja2<4.0.0,>=3.1.2 in /usr/local/lib/python3.11/dist-packages (from litellm==1.72.0->crewai==0.13
Requirement already satisfied: jsonschema<5.0.0,>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from litellm==1.72.0->crewai=
Requirement already satisfied: tiktoken>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from litellm==1.72.0->crewai==0.134.0)
Collecting coloredlogs (from onnxruntime==1.22.0->crewai==0.134.0)
  Downloading coloredlogs-15.0.1-py2.py3-none-any.whl.metadata (12 kB)
Requirement already satisfied: flatbuffers in /usr/local/lib/python3.11/dist-packages (from onnxruntime==1.22.0->crewai==0.134.0)
Requirement already satisfied: numpy>=1.21.6 in /usr/local/lib/python3.11/dist-packages (from onnxruntime==1.22.0->crewai==0.134.0
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from onnxruntime==1.22.0->crewai==0.134.0) (2
Requirement already satisfied: protobuf in /usr/local/lib/python3.11/dist-packages (from onnxruntime==1.22.0->crewai==0.134.0) (5
```

```
!pip install crewai==0.134.0
!pip install 'crewai[tools]'
```

```
Requirement already satisfied: opentelemetry-instrumentation-fastapi>=0.41b0 in /usr/local/lib/python3.11/dist-packages (from chro
Requirement already satisfied: pypika>=0.48.9 in /usr/local/lib/python3.11/dist-packages (from chromadb>=0.5.23->crewai==0.134.0)
Requirement already satisfied: tqdm>=4.65.0 in /usr/local/lib/python3.11/dist-packages (from chromadb>=0.5.23->crewai==0.134.0) (4
Requirement already satisfied: overrides>=7.3.1 in /usr/local/lib/python3.11/dist-packages (from chromadb>=0.5.23->crewai==0.134.0
Requirement already satisfied: importlib-resources in /usr/local/lib/python3.11/dist-packages (from chromadb>=0.5.23->crewai==0.13
Requirement already satisfied: grpcio>=1.58.0 in /usr/local/lib/python3.11/dist-packages (from chromadb>=0.5.23->crewai==0.134.0)
Requirement already satisfied: bcrypt>=4.0.1 in /usr/local/lib/python3.11/dist-packages (from chromadb>=0.5.23->crewai==0.134.0) (
Requirement already satisfied: typer>=0.9.0 in /usr/local/lib/python3.11/dist-packages (from chromadb>=0.5.23->crewai==0.134.0) (0
Requirement already satisfied: kubernetes>=28.1.0 in /usr/local/lib/python3.11/dist-packages (from chromadb>=0.5.23->crewai==0.134
Requirement already satisfied: tenacity>=8.2.3 in /usr/local/lib/python3.11/dist-packages (from chromadb>=0.5.23->crewai==0.134.0)
Requirement already satisfied: PyYAML>=6.0.0 in /usr/local/lib/python3.11/dist-packages (from chromadb>=0.5.23->crewai==0.134.0) (
Requirement already satisfied: mmh3>=4.0.1 in /usr/local/lib/python3.11/dist-packages (from chromadb>=0.5.23->crewai==0.134.0) (5.
Requirement already satisfied: orjson>=3.9.12 in /usr/local/lib/python3.11/dist-packages (from chromadb>=0.5.23->crewai==0.134.0)
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.11/dist-packages (from chromadb>=0.5.23->crewai==0.134.0) (
Requirement already satisfied: docstring-parser<1.0,>=0.16 in /usr/local/lib/python3.11/dist-packages (from instructor>=1.3.3->cre
Requirement already satisfied: jiter<0.11,>=0.6.1 in /usr/local/lib/python3.11/dist-packages (from instructor>=1.3.3->crewai==0.13
Requirement already satisfied: mkdocs-material>=9.5.49 in /usr/local/lib/python3.11/dist-packages (from instructor>=1.3.3->crewai=
Requirement already satisfied: mkdocs>=1.6.1 in /usr/local/lib/python3.11/dist-packages (from instructor>=1.3.3->crewai==0.134.0)
```

```
Requirement already satisfied: pre-commit>=4.2.0 in /usr/local/lib/python3.11/dist-packages (from instructor>=1.3.3->crewai==0.134
Requirement already satisfied: pydantic-core<3.0.0,>=2.18.0 in /usr/local/lib/python3.11/dist-packages (from instructor>=1.3.3->cr
Requirement already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.11/dist-packages (from openai>=1.13.3->crewai==0.134.0) (
Requirement already satisfied: distro<2,>=1.7.0 in /usr/local/lib/python3.11/dist-packages (from openai>=1.13.3->crewai==0.134.0)
Requirement already satisfied: sniffio in /usr/local/lib/python3.11/dist-packages (from openai>=1.13.3->crewai==0.134.0) (1.3.1)
Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.11/dist-packages (from openpyxl>=3.1.5->crewai==0.134.0) (2.0.
Requirement already satisfied: googleapis-common-protos~=1.52 in /usr/local/lib/python3.11/dist-packages (from opentelemetry-expor
Requirement already satisfied: opentelemetry-exporter-otlp-proto-common==1.35.0 in /usr/local/lib/python3.11/dist-packages (from o
Requirement already satisfied: opentelemetry-proto==1.35.0 in /usr/local/lib/python3.11/dist-packages (from opentelemetry-exporter
Requirement already satisfied: opentelemetry-semantic-conventions==0.56b0 in /usr/local/lib/python3.11/dist-packages (from opentel
ib/python3.11/dist-packages (from kubernetes>=28.1.0->chromadb>=0.5.23->crewai==0.134.0) (3.3.1)
Requirement already satisfied: durationpy>=0.7 in /usr/local/lib/python3.11/dist-packages (from kubernetes>=28.1.0->chromadb>=0.5.
Requirement already satisfied: ghp-import>=1.0 in /usr/local/lib/python3.11/dist-packages (from mkdocs>=1.6.1->instructor>=1.3.3->
Requirement already satisfied: markdown>=3.3.6 in /usr/local/lib/python3.11/dist-packages (from mkdocs>=1.6.1->instructor>=1.3.3->
Requirement already satisfied: mergedeep>=1.3.4 in /usr/local/lib/python3.11/dist-packages (from mkdocs>=1.6.1->instructor>=1.3.3-
Requirement already satisfied: mkdocs-get-deps>=0.2.0 in /usr/local/lib/python3.11/dist-packages (from mkdocs>=1.6.1->instructor>=
Requirement already satisfied: pathspec>=0.11.1 in /usr/local/lib/python3.11/dist-packages (from mkdocs>=1.6.1->instructor>=1.3.3-
Requirement already satisfied: pyyaml-env-tag>=0.1 in /usr/local/lib/python3.11/dist-packages (from mkdocs>=1.6.1->instructor>=1.3
Requirement already satisfied: watchdog>=2.0 in /usr/local/lib/python3.11/dist-packages (from mkdocs>=1.6.1->instructor>=1.3.3->cr
Requirement already satisfied: babel~=2.10 in /usr/local/lib/python3.11/dist-packages (from mkdocs-material>=9.5.49->instructor>=1
Requirement already satisfied: backrefs~=5.7.post1 in /usr/local/lib/python3.11/dist-packages (from mkdocs-material>=9.5.49->instr
Requirement already satisfied: colorama~=0.4 in /usr/local/lib/python3.11/dist-packages (from mkdocs-material>=9.5.49->instructor>
Requirement already satisfied: mkdocs-material-extensions~=1.3 in /usr/local/lib/python3.11/dist-packages (from mkdocs-material>=9
Requirement already satisfied: paginate~=0.5 in /usr/local/lib/python3.11/dist-packages (from mkdocs-material>=9.5.49->instructor>
Requirement already satisfied: pymdown-extensions~=10.2 in /usr/local/lib/python3.11/dist-packages (from mkdocs-material>=9.5.49->
Requirement already satisfied: opentelemetry-instrumentation-asgi==0.56b0 in /usr/local/lib/python3.11/dist-packages (from opentel
Requirement already satisfied: opentelemetry-instrumentation==0.56b0 in /usr/local/lib/python3.11/dist-packages (from opentelemetr
Requirement already satisfied: opentelemetry-util-http==0.56b0 in /usr/local/lib/python3.11/dist-packages (from opentelemetry-inst
Requirement already satisfied: wrapt<2.0.0,>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from opentelemetry-instrumentation=
Requirement already satisfied: asgiref~=3.0 in /usr/local/lib/python3.11/dist-packages (from opentelemetry-instrumentation-asgi==0
Requirement already satisfied: monotonic>=1.5 in /usr/local/lib/python3.11/dist-packages (from posthog>=2.4.0->chromadb>=0.5.23->c
Requirement already satisfied: backoff>=1.10.0 in /usr/local/lib/python3.11/dist-packages (from posthog>=2.4.0->chromadb>=0.5.23->
Requirement already satisfied: cfgv>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from pre-commit>=4.2.0->instructor>=1.3.3->
Requirement already satisfied: identify>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from pre-commit>=4.2.0->instructor>=1.3
Requirement already satisfied: nodeenv>=0.11.1 in /usr/local/lib/python3.11/dist-packages (from pre-commit>=4.2.0->instructor>=1.3
Requirement already satisfied: virtualenv>=20.10.0 in /usr/local/lib/python3.11/dist-packages (from pre-commit>=4.2.0->instructor>
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->chromadb>=0.5
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from typer>=0.9.0->chromadb>=0.5.23-
Requirement already satisfied: httptools>=0.6.3 in /usr/local/lib/python3.11/dist-packages (from uvicorn[standard]>=0.18.3->chroma
```

```
import os
from crewai import Agent, Task, Crew, Process, LLM
from crewai_tools import SerperDevTool

import warnings
warnings.filterwarnings('ignore')


from google.colab import userdata

os.environ["geminiapikey"] = userdata.get('geminiapikey')
os.environ["geminiapikey"] = userdata.get('geminiapikey')
# os.environ["OPENAI_API_KEY"] = userdata.get('OPENAI_API_KEY')

os.environ["GROQAPIKEY"] = userdata.get('GROQAPIKEY')


Start coding or generate with AI.


question_researcher = Agent(
    role='Interview Question Research Expert',
    goal='Find the most commonly asked and important interview questions for any given topic',
    backstory="""You are an expert recruiter and interview specialist with years of experience
    conducting technical interviews. You know exactly what questions are most frequently asked
    by top companies and can identify the most critical questions that candidates need to prepare for.
    You stay updated with the latest interview trends and have access to real interview experiences
    from various companies.""",
    tools=[],
    verbose=True,
    llm=gemini_llm
)

# Agent 2: DUBEXA Answer Expert
dubexa_expert = Agent(
    role='DUBEXA Format Answer Expert',
    goal='Create comprehensive, well-structured answers in DUBEXA format that help candidates thoroughly understand and explain concepts
    backstory="""You are a master educator and technical communication expert who specializes
    in the DUBEXA format. You excel at breaking down complex concepts into clear, structured
    explanations that follow the DUBEXA methodology:

    - D: Definition using key technical terms
    - U: Use case scenarios and real-world examples
    - B: Benefits of using the concept
    - E: Example code (when applicable)
    - X: Extra information, edge cases, and limitations
```

```
        - A: Analogies and metaphors for better understanding

        You make technical concepts accessible while maintaining accuracy and depth.""",
        tools=[],
        verbose=True,
        llm=gemini_llm
    )

    # Agent 3: Interview Coach
    interview_coach = Agent(
        role='Senior Interview Coach',
        goal='Provide additional interview tips, delivery advice, and presentation strategies for each answer',
        backstory="""You are a seasoned interview coach who has helped thousands of candidates
        succeed in their interviews. You understand not just what to say, but how to say it
        effectively. You provide practical tips on delivery, common follow-up questions,
        and strategies to make answers more impactful during actual interviews.""",
        tools=[],
        verbose=True,
        llm=gemini_llm
    )


    Start coding or generate with AI.


    def create_question_research_task(topic):
        return Task(
            description=f"""Research and identify the top 3 most commonly asked interview questions about {topic}.

            **Requirements:**
            1. Focus on questions that are frequently asked in technical interviews
            2. Prioritize questions that test fundamental understanding
            3. Include a mix of basic and intermediate level questions
            4. Ensure questions are relevant to current industry standards
            5. Look for questions from reputable sources like:
               - Top tech companies (Google, Amazon, Microsoft, etc.)
               - Popular interview preparation platforms
               - Technical forums and communities
               - Recent interview experiences shared online

            **Research Focus:**
            - Find actual interview questions, not generic study topics
            - Prioritize questions that require clear conceptual understanding
            - Look for questions that allow for DUBEXA format explanations
            - Consider questions that might have follow-ups or variations

            Topic: {topic}""",

            expected_output="""A list of exactly 3 interview questions about the topic, formatted as:

            **Question 1:** [Exact question text]
            **Source/Context:** [Where this question is commonly asked - company types, interview rounds, etc.]
            **Difficulty Level:** [Beginner/Intermediate/Advanced]

            **Question 2:** [Exact question text]
            **Source/Context:** [Where this question is commonly asked]
            **Difficulty Level:** [Beginner/Intermediate/Advanced]

            **Question 3:** [Exact question text]
            **Source/Context:** [Where this question is commonly asked]
            **Difficulty Level:** [Beginner/Intermediate/Advanced]

            Each question should be specific, commonly asked.""",

            agent=question_researcher
        )

    def create_dubexa_answer_task(topic):
        return Task(
            description=f"""Create comprehensive DUBEXA format answers for each of the 3 interview questions about {topic}.

            **DUBEXA Format Requirements:**

            **D - Definition:**
            - Provide a clear, technically accurate definition
            - Use proper technical terminology
            - Keep it concise but complete

            **U - Use Cases & Why We Need It:**
            - Explain real-world scenarios where this is used
            - Provide specific, practical examples
            - Explain WHY this concept/technology is important
            - Include industry context
```

```
            **B - Benefits:**
            - List clear advantages and benefits
            - Compare with alternatives when relevant
            - Explain what problems it solves

            **E - Example Code:**
            - Provide practical, working code examples
            - Include comments explaining key parts
            - Use modern, best-practice syntax
            - Make examples relevant to the concept

            **X - Extra Information:**
            - Cover edge cases and limitations
            - Mention common pitfalls or misconceptions
            - Include performance considerations
            - Discuss when NOT to use it

            **A - Analogies & Metaphors:**
            - Create relatable analogies that simplify the concept
            - Use everyday examples that non-technical people could understand
            - Make the analogy memorable and accurate

            **Instructions:**
            - Answer each question thoroughly in DUBEXA format
            - Ensure answers are interview-appropriate (not too long, but comprehensive)
            - Use proper formatting and structure
            - Include all 6 DUBEXA components for each question""",

            expected_output="""Complete DUBEXA format answers for all 3 questions, structured as:

            ## Question 1: [Question text]

            **Definition:**
            [Clear technical definition]

            **Use Cases & Why We Need It:**
            [Real-world examples and importance]

            **Benefits:**
            [Advantages and problem-solving capabilities]

            **Example Code:**
            ```[language]
            [Working code example with comments]
            ```

            **Extra Information:**
            [Edge cases, limitations, best practices]

            **Analogies & Metaphors:**
            [Relatable analogies for better understanding]

            ---

            [Repeat same format for Questions 2 and 3]

            Each answer should be comprehensive yet concise enough for interview delivery.""",

            agent=dubexa_expert,
            context=[create_question_research_task(topic)]
        )

    def create_interview_coaching_task(topic):
        return Task(
            description=f"""Provide interview coaching and delivery tips for the DUBEXA answers about {topic}.

            **Coaching Focus:**
            1. **Delivery Tips:** How to present each answer effectively
            2. **Time Management:** How long to spend on each part of DUBEXA
            3. **Follow-up Questions:** Anticipate likely follow-up questions
            4. **Common Mistakes:** What to avoid when answering
            5. **Confidence Building:** Tips to sound knowledgeable and prepared
            6. **Adaptation:** How to adjust answers based on interviewer's background

            **For Each Question Provide:**
            - Key points to emphasize during delivery
            - Suggested time allocation for each DUBEXA component
            - Potential follow-up questions interviewers might ask
            - Red flags or common mistakes to avoid
            - Tips for making the answer more engaging
            - How to gauge if the interviewer wants more or less detail""",
```

```
            expected_output="""Interview coaching guide with delivery tips for each question:

            ## Overall Interview Strategy for {topic}
            [General tips for approaching these types of questions]

            ## Question 1 Coaching: [Question text]
            **Delivery Tips:** [How to present this answer effectively]
            **Time Management:** [Suggested timing for each DUBEXA component]
            **Likely Follow-ups:** [Questions interviewer might ask next]
            **Common Pitfalls:** [What to avoid]
            **Engagement Tips:** [How to make answer more compelling]

            ## Question 2 Coaching: [Question text]
            [Same structure as above]

            ## Question 3 Coaching: [Question text]
            [Same structure as above]

            ## Final Interview Tips
            - Body language and confidence tips
            - How to handle if you don't know something
            - Ways to show enthusiasm and deeper knowledge
            - How to connect answers to real experience""",

            agent=interview_coach,
            context=[create_question_research_task(topic), create_dubexa_answer_task(topic)]
        )


    def create_interview_prep_crew(topic):
        """Create an interview preparation crew for a specific topic"""

        # Create all tasks for this topic
        research_task = create_question_research_task(topic)
        dubexa_task = create_dubexa_answer_task(topic)
        coaching_task = create_interview_coaching_task(topic)

        # Create the crew
        crew = Crew(
            agents=[question_researcher, dubexa_expert, interview_coach],
            tasks=[research_task, dubexa_task, coaching_task],
            process=Process.sequential,
            verbose=True,
        )

        return crew


    example_topic = "JavaScript Closures"
    interview_crew = create_interview_prep_crew(example_topic)
    result = interview_crew.kickoff()
```

Crew Execution Started

Crew Execution Started
Name: crew
ID: f851b3f9-9210-4c4a-9ae8-7f678aeec710
Tool Args:

Agent Started

Agent: **Interview Question Research Expert**

Task: Research and identify the top 3 most commonly asked interview questions about JavaScript Closures.

```
**Requirements:**
1. Focus on questions that are frequently asked in technical interviews
2. Prioritize questions that test fundamental understanding
3. Include a mix of basic and intermediate level questions
4. Ensure questions are relevant to current industry standards
5. Look for questions from reputable sources like:
   - Top tech companies (Google, Amazon, Microsoft, etc.)
   - Popular interview preparation platforms
   - Technical forums and communities
   - Recent interview experiences shared online

**Research Focus:**
- Find actual interview questions, not generic study topics
- Prioritize questions that require clear conceptual understanding
- Look for questions that allow for DUBEXA format explanations
- Consider questions that might have follow-ups or variations

Topic: JavaScript Closures
```

Crew: crew
└── Task: 493230b3-557d-43f2-97ba-936a36f47882
Status: Executing Task...

Agent Final Answer

Agent: **Interview Question Research Expert**

Final Answer:
**

Question 1: What is a closure in JavaScript? Explain with a simple example, and describe how it can be useful.
Source/Context: Commonly asked in initial technical interviews at various companies, including startups and mid-sized tech firms. Often appears in phone screenings or the first round of on-site interviews.
Difficulty Level: Beginner

Question 2: Describe a scenario where using closures could lead to a memory leak in JavaScript. How can you prevent this?
Source/Context: Frequently asked in intermediate-level JavaScript interviews, especially at companies like Google, Microsoft, and those emphasizing performance and optimization. Often shows up in second-round interviews or when discussing performance-related topics.
Difficulty Level: Intermediate

```
example_topic = "Ufc fighter"
interview_crew = create_interview_prep_crew(example_topic)
result = interview_crew.kickoff()
```

Source/Context: This is a very common coding challenge-style question asked in technical interviews across different companies, including Amazon, Facebook (Meta), and others focusing on practical JavaScript skills. It assesses the candidate's ability to apply closure concepts in a practical setting. Often appears in coding interview rounds.
Difficulty Level: Intermediate

Task Completion

Task Completed
Name: 493230b3-557d-43f2-97ba-936a36f47882
Agent: Interview Question Research Expert
Tool Args:

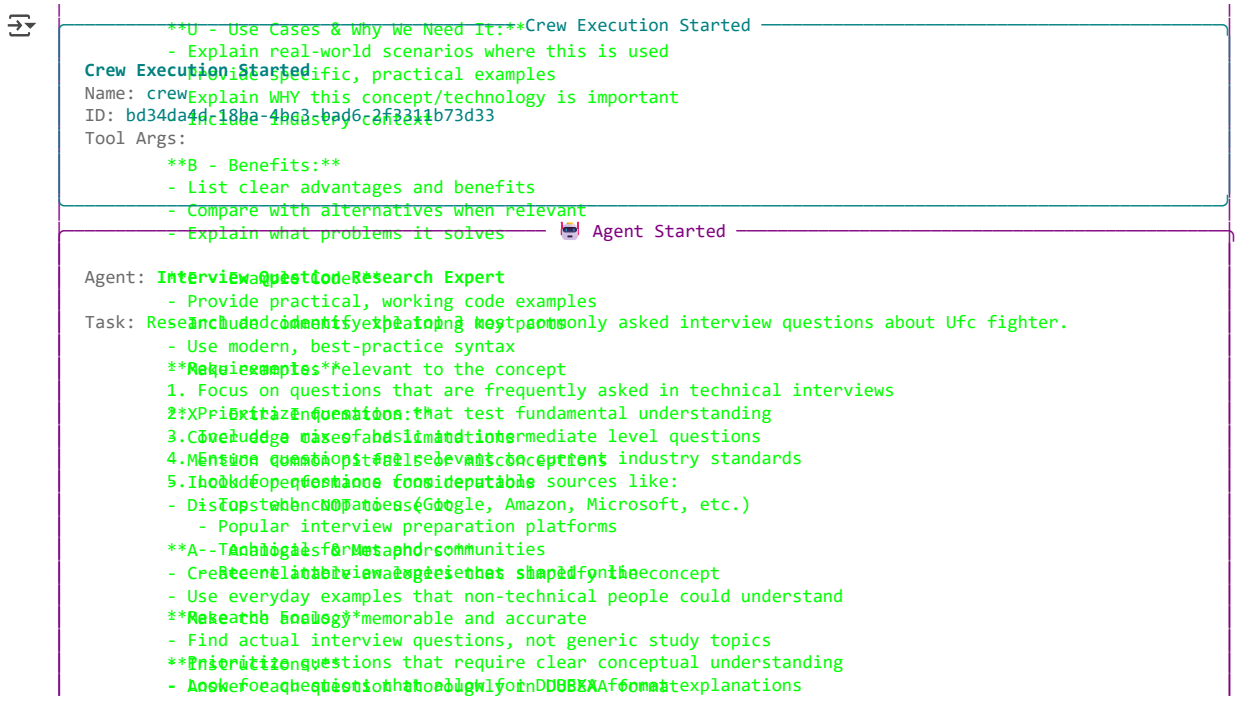
Agent Started

Agent: **DUBEXA Format Answer Expert**

Task: Create comprehensive DUBEXA format answers for each of the 3 interview questions about JavaScript Closures.

```
**DUBEXA Format Requirements:**

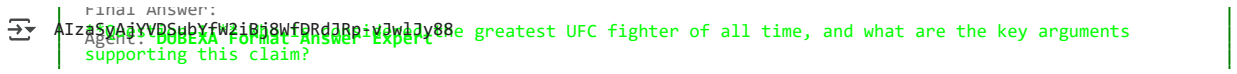
**D - Definition:**
- Provide a clear, technically accurate definition
- Use proper technical terminology
- Keep it concise but complete
```



V

```
example_topic = "jujus"
interview_crew = create_interview_prep_crew(example_topic)
result = interview_crew.kickoff()
```

```
from google.colab import userdata
gemini_api_key =  userdata.get('geminiapikey')
print(gemini_api_key)
```




```
# ===== Create LLMs
# groq_llm = LLM(
#     model="groq/llama-3.3-70b-versatile",
#     temperature=0.1
# )

gemini_llm = LLM(
    model="gemini/gemini-2.0-flash",
    api_key=gemini_api_key,
```