OpenAI API Features

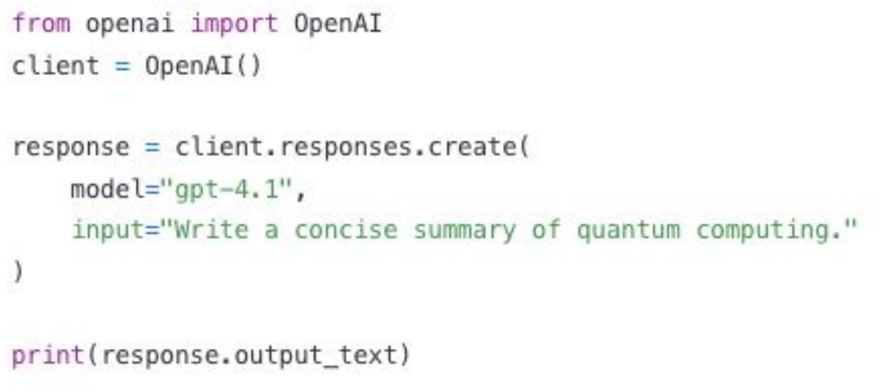
Explore all of the core features that OpenAI offers

Overview of OpenAI Platform

- **What is OpenAI?** A leading AI research and deployment company.
- **Developer Platform:** Comprehensive suite of AI models and tools.
- **Use Cases:** From simple text generation to complex reasoning and multimodal applications.
- **Integration Options:** SDKs available for Python, Node.js, and other languages.
- **API-First Design:** Build AI capabilities directly into your applications.

Core Language Capabilities

Text Generation & Completion

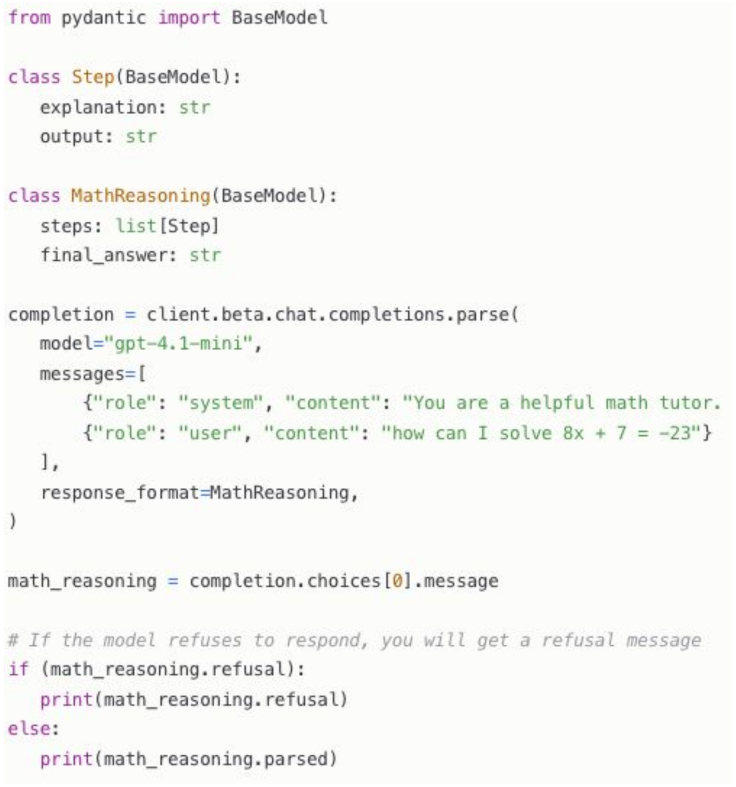


```python
from openai import OpenAI
client = OpenAI()

response = client.responses.create(
    model="gpt-4.1",
    input="Write a concise summary of quantum computing."
)

print(response.output_text)
```

- Models range from lightweight to advanced reasoning models.
- **Control parameters:** temperature, max tokens, top_p.
- Content filtering and safety measures built-in.

Structured Outputs

Getting Predictable JSON Responses

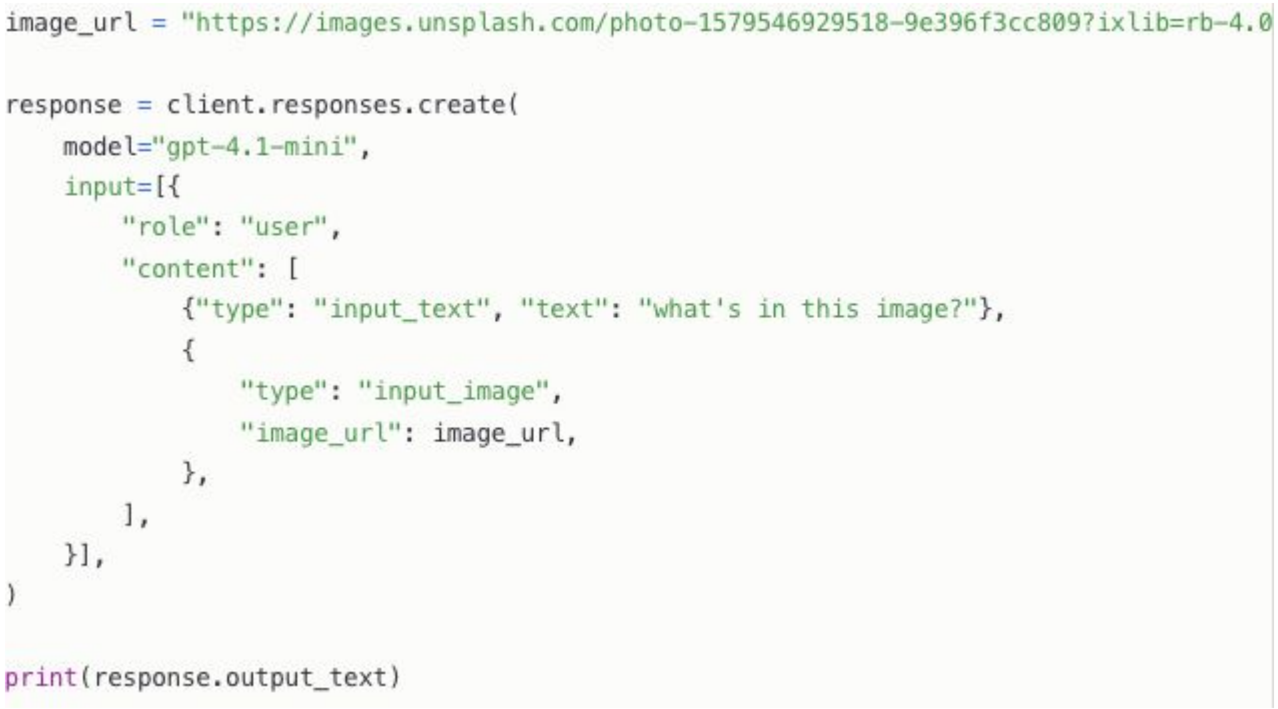


```python
from pydantic import BaseModel

class Step(BaseModel):
   explanation: str
   output: str

class MathReasoning(BaseModel):
   steps: list[Step]
   final_answer: str

completion = client.beta.chat.completions.parse(
   model="gpt-4.1-mini",
   messages=[
       {"role": "system", "content": "You are a helpful math tutor.
       {"role": "user", "content": "how can I solve 8x + 7 = -23"}
   ],
   response_format=MathReasoning,
)

math_reasoning = completion.choices[0].message

# If the model refuses to respond, you will get a refusal message
if (math_reasoning.refusal):
   print(math_reasoning.refusal)
else:
   print(math_reasoning.parsed)
```

- Perfect for data extraction and structured information retrieval.
- Enforces output format validation.
- Reduces post-processing effort.

Multimodal Capabilities

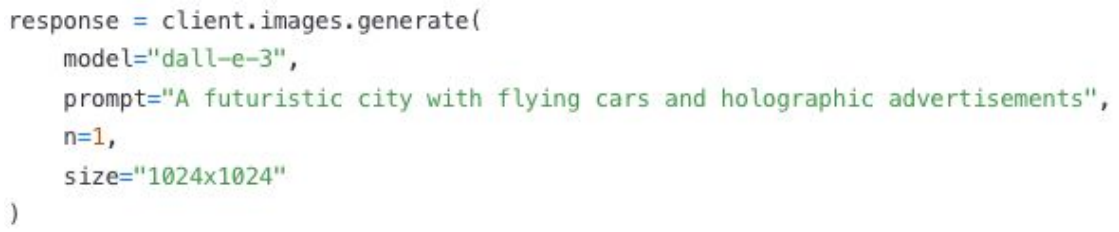
Vision & Image Analysis



```
image_url = "https://images.unsplash.com/photo-1579546929518-9e396f3cc809?ixlib=rb-4.0

response = client.responses.create(
    model="gpt-4.1-mini",
    input=[{
        "role": "user",
        "content": [
            {"type": "input_text", "text": "what's in this image?"},
            {
                "type": "input_image",
                "image_url": image_url,
            },
        ],
    }],
)

print(response.output_text)
```

Multimodal Capabilities

Image Generation with Dalle



```
response = client.images.generate(
    model="dall-e-3",
    prompt="A futuristic city with flying cars and holographic advertisements",
    n=1,
    size="1024x1024"
)
```

Audio & Speech Capabilities

Text-to-Speech

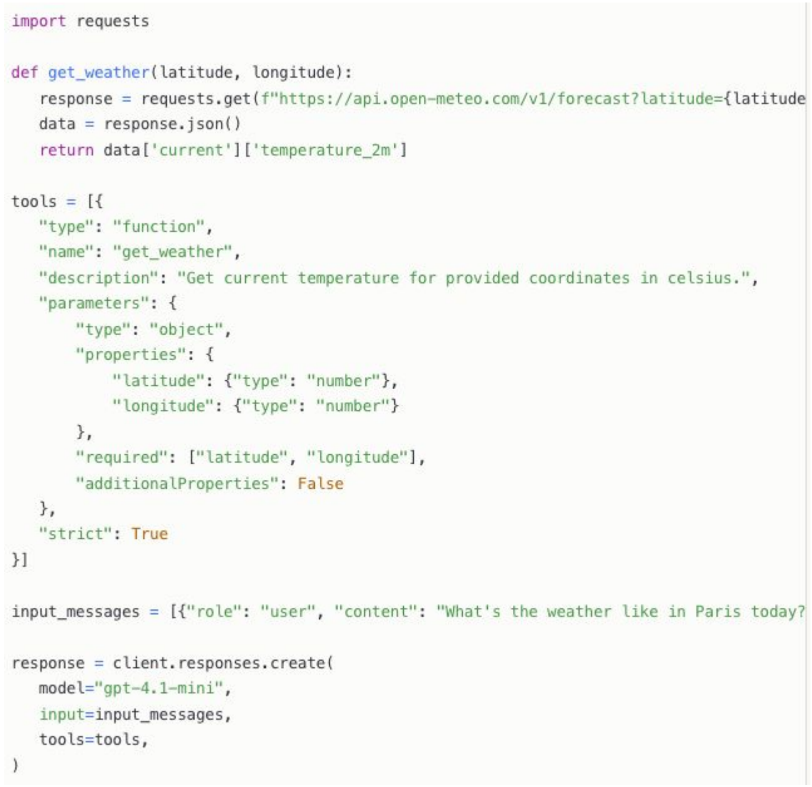
```python
response = client.audio.speech.create(
    model="tts-1",
    voice="alloy",
    input="Hello world! This is a demonstration of text to speech."
)

# Save to file
response.stream_to_file("speech.mp3")
```

Speech-to-Text (Transcription)

```python
with open("audio_file.mp3", "rb") as audio_file:
    transcript = client.audio.transcriptions.create(
        model="whisper-1",
        file=audio_file
    )
print(transcript.text)
```

Advanced Features - Function Calling & Tools

Building Tool-Using Applications

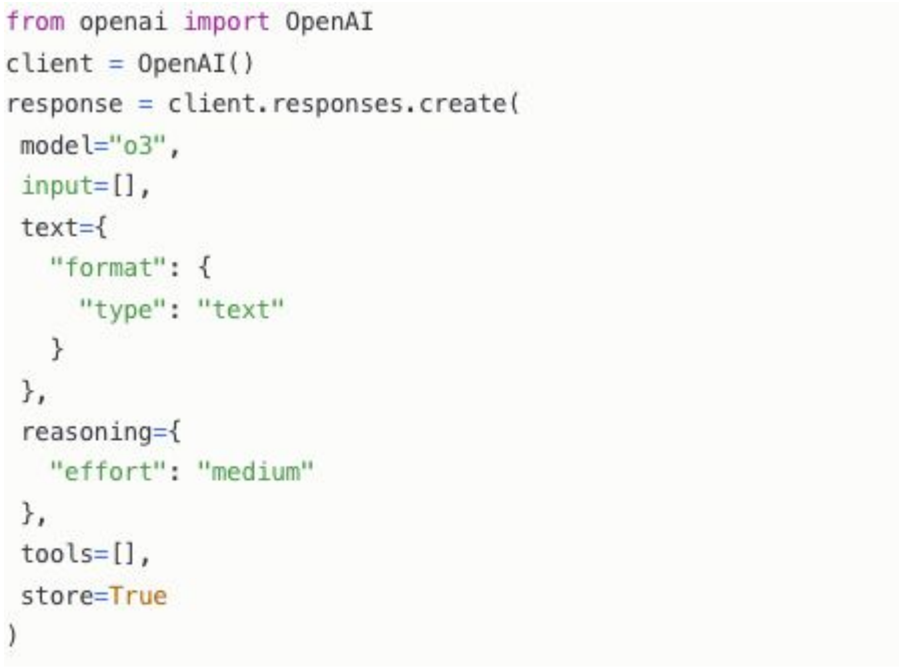


```python
import requests

def get_weather(latitude, longitude):
   response = requests.get(f"https://api.open-meteo.com/v1/forecast?latitude={latitude
   data = response.json()
   return data['current']['temperature_2m']

tools = [{
   "type": "function",
   "name": "get_weather",
   "description": "Get current temperature for provided coordinates in celsius.",
   "parameters": {
       "type": "object",
       "properties": {
           "latitude": {"type": "number"},
           "longitude": {"type": "number"}
       },
       "required": ["latitude", "longitude"],
       "additionalProperties": False
   },
   "strict": True
}]

input_messages = [{"role": "user", "content": "What's the weather like in Paris today?

response = client.responses.create(
   model="gpt-4.1-mini",
   input=input_messages,
   tools=tools,
)
```

- Models can determine when to call functions.
- Enables connection to external APIs and databases.
- Build complex workflows and agents.

Reasoning Models

Complex Problem-Solving with o3 and o4-mini

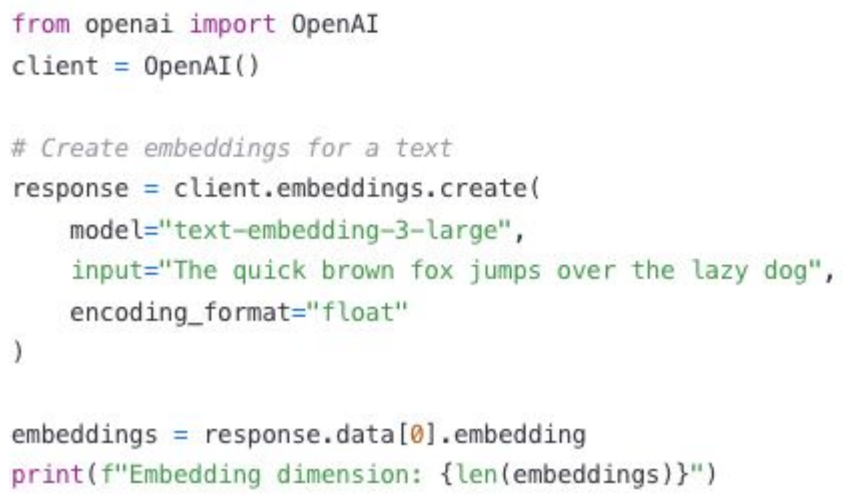


```python
from openai import OpenAI
client = OpenAI()
response = client.responses.create(
 model="o3",
 input=[],
 text={
   "format": {
     "type": "text"
   }
 },
 reasoning={
   "effort": "medium"
 },
 tools=[],
 store=True
)
```

- **o3:** High-intelligence reasoning model.
- **o4-mini:** Fast, flexible intelligence.
- **Tackle complex tasks** requiring multi-step reasoning.
- Suitable for technical problem-solving and analysis.

Embeddings

Vector Representations for Semantic Search and Similarity



```python
from openai import OpenAI
client = OpenAI()

# Create embeddings for a text
response = client.embeddings.create(
    model="text-embedding-3-large",
    input="The quick brown fox jumps over the lazy dog",
    encoding_format="float"
)

embeddings = response.data[0].embedding
print(f"Embedding dimension: {len(embeddings)}")
```

- Convert text into numerical vector representations
- Enable semantic search and content recommendation

Applications:

- Similarity matching
- Content retrieval
- Clustering and classification
- Knowledge bases and RAG (Retrieval Augmented Generation)

Customization Options

Fine-tuning

- Train models on your specific data.
- Improve performance on domain-specific tasks.
- Reduce prompt engineering requirements.

Evals

- Systematically evaluate model performance.
- Create benchmarks for your specific use cases.
- Identify areas for improvement.

Distillation

- Create smaller, specialized models.
- Reduce inference costs and latency.

Developer Resources

- **Documentation:** Comprehensive guides and API references
- **Cookbook:** Open-source code examples and tutorials
- **Developer Forum:** Community support and discussions
- **SDKs:** Official libraries for multiple programming languages
- **Rate Limits & Quotas:** Understand usage limitations

Getting Started

1. **Create Account:** Sign up at platform.openai.com.
2. **Generate API Key:** Secure authentication for API requests.
3. **Choose a Model:** Select based on your use case and budget.
4. **Install SDK:** `pip install openai` or use another language SDK.
5. **Make Your First Call:** Follow quickstart guide documentation.
6. **Scale Gradually:** Monitor usage and optimize as you grow.