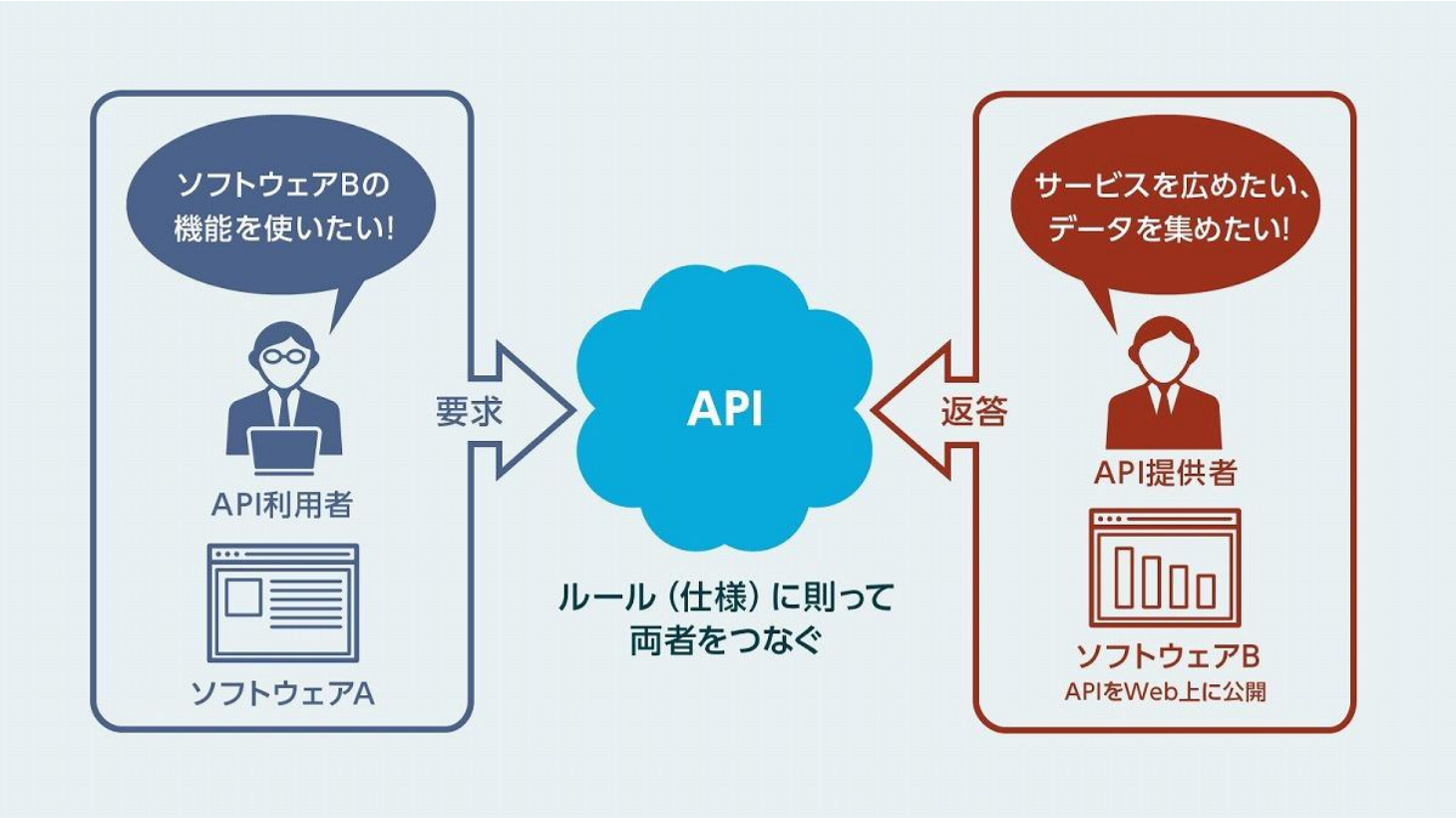
生成AI基礎

3 生成AIのAPIを用いたコーディングができるようになる

APIとは?

APIは、異なるソフトウェア間で情報を交換するためのインターフェースです



OpenAI API

OpenAI APIの概要を説明します

OpenAI APIでできること

1. GPT-4などのモデルを使ったテキスト生成
    - 入力に基づいたテキスト生成が可能
    - コンテンツ作成、クリエイティブライティング、コード生成など幅広い用途
2. APIを使ったチャットボットとアシスタントの開発
    - 対話、質問応答、タスク実行などが可能な対話型エージェントを作成
    - リアルタイムで双方向のコミュニケーションを提供

APIキーについて

- APIキーは公開してはいけません。リポジトリや公開される可能性のある場所には保管しないでください。
- APIキーはソースコード内に直接書かず、環境変数などを通じて安全に管理し、アプリケーションから読み込むようにします。

OpenAI APIのコスト

- OpenAIのAPIの料金は、選択した言語モデルのトークン料金に基づいて決まる
- トークンとは、モデルが処理するテキストの単位で、単語の一部に相当する
- 英語の場合、1000トークンは約750語に相当するが、日本語の場合は異なる

https://openai.com/pricing

Chat Completions APIを使ったコーディング

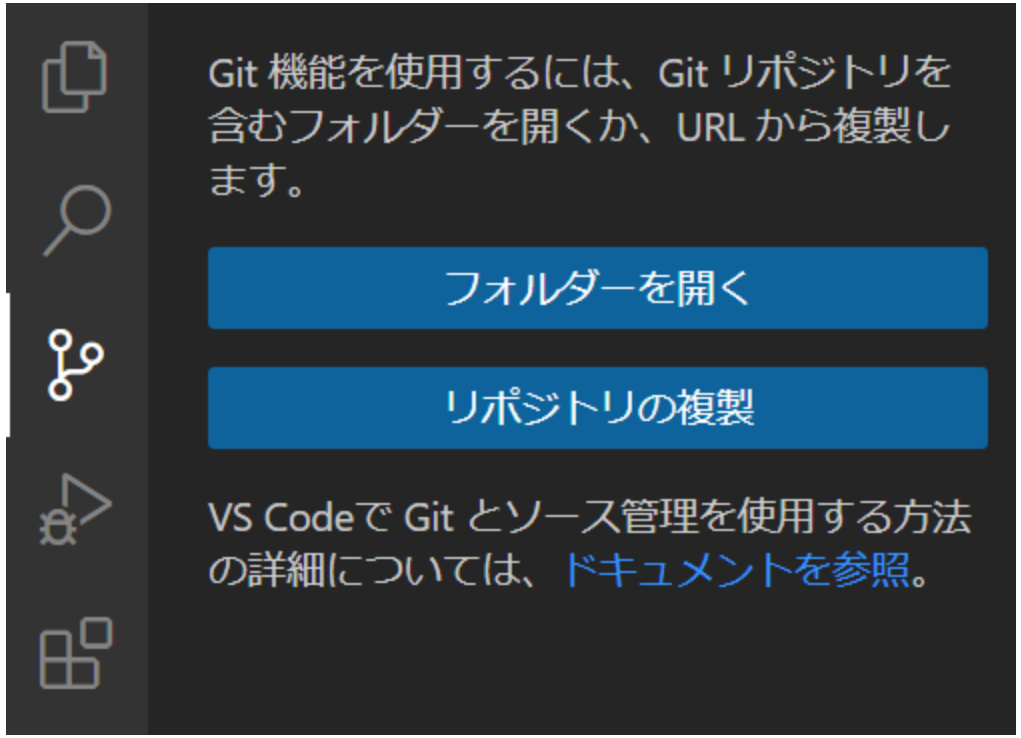
一番基本的なAPIの使い方を説明します

準備

1. 配布されたIPアドレスを確認
   例: 111.111.111.111
2. ブラウザで `http://{IPアドレス}:8000/` にアクセス
   例: http://111.111.111.111:8000/

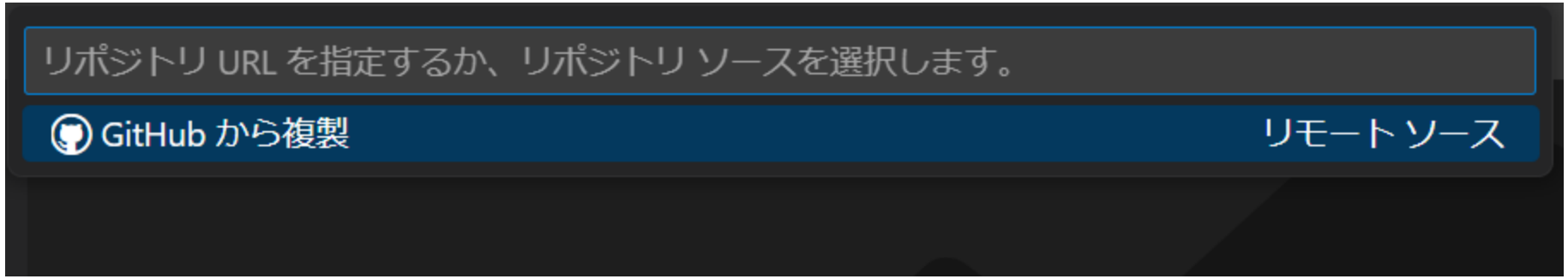
準備

3. 左のメニューの「ソース管理」をクリック



準備

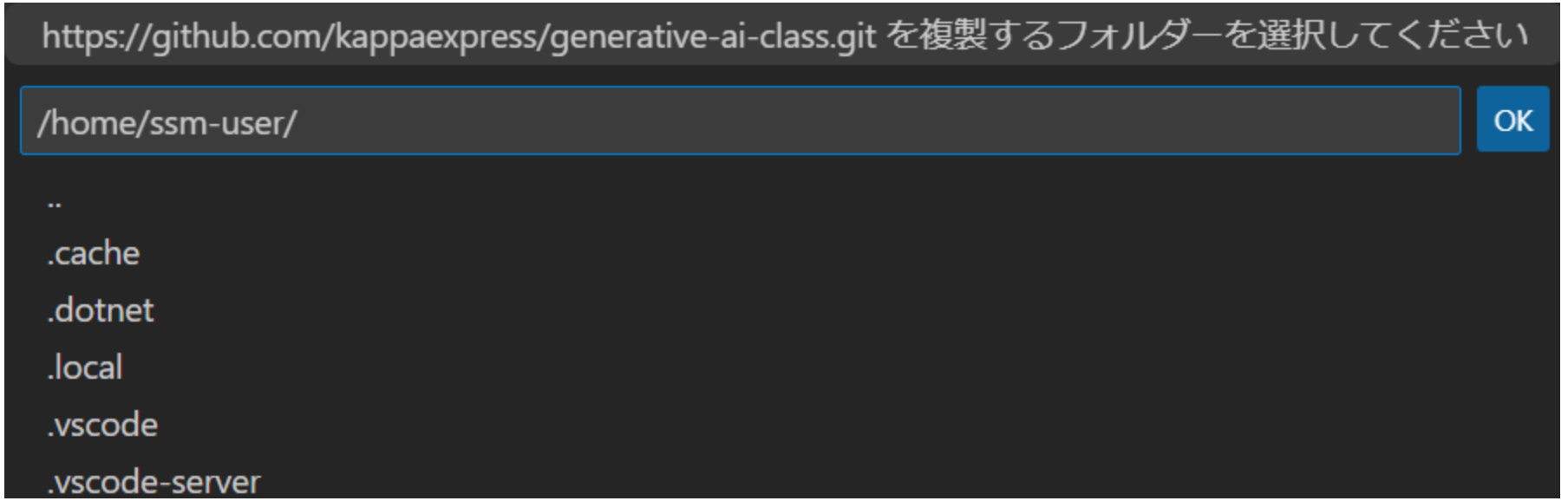
4. 「リポジトリの複製」をクリック



5. `https://github.com/kappaexpress/generative-ai-class.git` を入力して「レポジトリのURL」をクリック

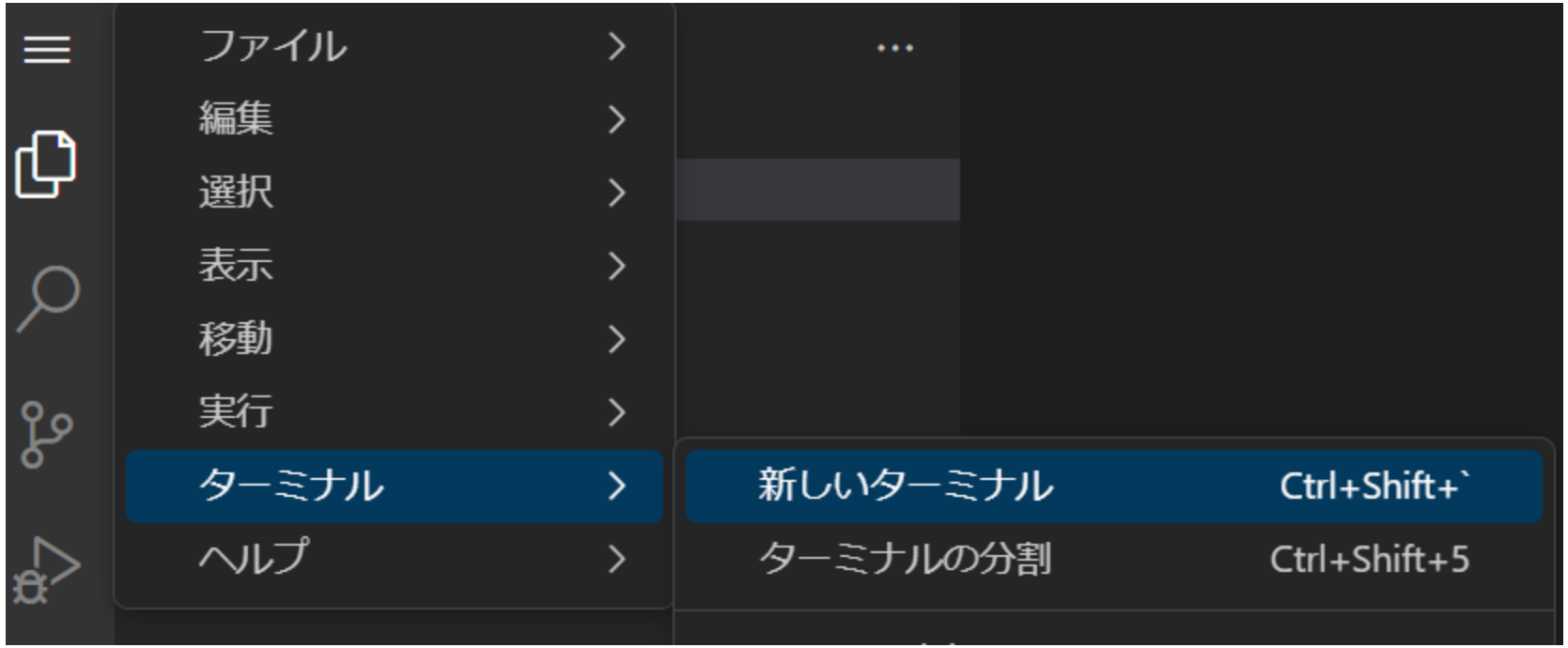
準備

6. 「OK」をクリック


https://github.com/kappaexpress/generative-ai-class.git を複製するフォルダーを選択してください
/home/ssm-user/
OK
..
.cache
.dotnet
.local
.vscode
.vscode-server

準備

7. 左のメニューの「三」をクリック
8. 「ターミナル」をクリック
9. 「新しいターミナル」をクリック



準備

10. 下記のコマンドをターミナルで実行してライブラリをインストール

```
bash
poetry install
poetry shell
```

準備完了です！

Chat Completions APIの基本的な使い方

```python
# openaiのライブラリをインポート
from openai import OpenAI

# OpenAIクライアントを作成
client = OpenAI()

# クライアントを使ってChat Completions APIを呼び出す
response = client.chat.completions.create(
  model="gpt-3.5-turbo",
  messages=[
    {"role": "system", "content": "あなたはうさぎです。語尾にぴょんをつけてください"},
    {"role": "user", "content": "藤井聡太は初めてのタイトルをいつ獲得しましたか？"},
    {"role": "assistant", "content": "藤井聡太は2020年7月16日に棋聖戦のタイトルを獲得しました。"},
    {"role": "user", "content": "2020年7月16日に藤井聡太は何のタイトルを獲得しましたか?"},
  ],
  temperature=0.4
)

# Chat Completions APIの結果を表示
print(response.choices[0].message.content)
```

Chat Completions APIの基本的な使い方

実行方法

1. ターミナルでAPIキーを設定

```
bash
export OPENAI_API_KEY="sk-XXXXXXXXXXXXXXXXXXXXXXX"
```

2. ターミナルで`main.py`を作成

```
bash
touch main.py
```

3. `main.py`を左のメニューから開く

Chat Completions APIの基本的な使い方

4. `main.py` にコードを貼り付ける
5. `main.py` を実行

```
bash
python main.py
```

role（system, user, assistant）の使い分け方

- system: AIアシスタントの設定を書きます
- user: ユーザーとしてのメッセージ
- assistant: AIアシスタントとしてのメッセージ

assistantのメッセージを使って、文脈を与えることができます

temperatureパラメーターについて

- temperatureが小さい場合：
  - 出力が確定的になり、最も確率の高い次のトークンが選択されます
  - 予測可能で一貫した応答が得られるため、事実に基づくQAなどのタスクに適しています。
- temperatureが大きい場合：
  - ランダム性が増し、出力が多様で創造的になります
  - 他の可能なトークンの重みが増えるため、創造的なタスクに適しています。

0.4程度がバランスが取れた値です
1.5くらいまで増やすと、支離滅裂になることがあります

GPT-4 Turbo with Visionを使ったコーディング

画像を認識させるAPIの使い方

マルチモーダル

- マルチモーダルAIは、異なる種類の情報（例えば、画像、音声、テキスト）を組み合わせて処理するAIのことです。
- 複数のデータタイプを同時に処理し、複雑なタスクを解析する能力を持っています
- 人間の五感に似た方法で情報を同時に取り込み、瞬時に処理することができます

GPT-4 Turbo with Visionの基本的な使い方

画像を読み込んでAPIに渡す部分が追加されます
コードは次のページ

```python
from openai import OpenAI
import base64
client = OpenAI()
# 画像を読み込む関数 画像を文字列に変換する
def encode_image(image_path):
    with open(image_path, "rb") as image_file:
        return base64.b64encode(image_file.read()).decode('utf-8')
# 画像のパス
image_path = "image.png"
# 画像が返還された文字列の取得
base64_image = encode_image(image_path)
# Chat Completions APIを呼び出す
response = client.chat.completions.create(
  model="gpt-4-turbo",
  messages=[
    {
      "role": "user",
      "content": [
        {"type": "text", "text": "この画像は何を意味していますか?"},
        {
          "type": "image_url",
          "image_url": {
            "url": f"{base64_image}",
          },
        },
      ],
    }
  ]
)
# Chat Completions APIの結果を表示
print(response.choices[0].message.content)
```

不得意なこと

- 非英語：日本語や韓国語などのラテン文字以外のテキストが含まれる画像を処理する際、最適なパフォーマンスを発揮しない場合があります。
- 回転：回転または上下逆のテキストや画像の解釈を誤ることがあります。
- 視覚要素：色やスタイル（実線、破線、点線など）が異なるグラフやテキストの理解に苦労することがあります。
- 空間推理：チェスのポジションを特定するなど、正確な空間的位置付けを要求されるタスクに苦労します。
- 画像形状：パノラマや魚眼レンズの画像を扱う際に苦労します。
- オブジェクトのカウント：画像内のオブジェクトの数をおおよそ数えます。

まとめ

- OpenAI APIを使ったコーディングの基本
- Chat Completions APIの使い方
- GPT-4 Turbo with Visionの使い方