مع دعاء استدعاء الوظائف

```
import os
import requests
from dotenv import load_dotenv
import argparse
import json

load_dotenv()

def call_mistral_api(prompt, model="mistral-small-2501", use_function_calling=False):
    api_key = os.environ.get("MISTRAL_API_KEY")
    if not api_key:
        print(:  "                    MISTRAL_API_KEY.")
        return None

    url = "https://api.mistral.ai/v1/chat/completions"
    headers = {
        "Content-Type": "application/json",
        "Accept": "application/json",
        "Authorization": f"Bearer {api_key}"
    }

    if use_function_calling:
        data = {
            "model": model,
            "messages": [
                {
                    "role": "user",
                    "content": prompt
                }
            ],
            "tools": [
                {
                    "type": "function",
                    "function": {
                        "name": "get_current_weather",
                        "description": "Get the current weather in a given location",
                        "parameters": {
                            "type": "object",
```

```
                        "properties": {
                            "location": {
                                "type": "string",
                                "description": "The city and state, e.g. San Francisco, CA",
                            },
                            "unit": {"type": "string", "enum": ["celsius", "fahrenheit"]},
                        },
                        "required": ["location"],
                    },
                }
            }
        ],
        "tool_choice": "auto",
    }
else:
    data = {
        "model": model,
        "messages": [
            {
                "role": "user",
                "content": prompt
            }
        ]
    }

print(f"Mistral API URL: {url}")
print(f"Mistral API Headers: {headers}")
print(f"Mistral API Data: {data}")
try:
    response = requests.post(url, headers=headers, json=data)
    response.raise_for_status()
    response_json = response.json()
    print(response_json)
    if response_json and response_json['choices']:
        choice = response_json['choices'][0]
        if 'message' in choice and 'content' in choice['message']:
            content = choice['message']['content']
            return content
        elif 'message' in choice and 'tool_calls' in choice['message']:
            tool_calls = choice['message']['tool_calls']
```

```
                print(f"Tool calls: {tool_calls}")
                return tool_calls # Return the tool calls for processing
            else:
                print(f"Mistral API Error: Invalid response format: {response_json}")
                return None
        else:
            print(f"Mistral API Error: Invalid response format: {response_json}")
            return None
    except requests.exceptions.RequestException as e:
        print(f"Mistral API Error: {e}")
        if e.response:
            print(f"Response status code: {e.response.status_code}")
            print(f"Response content: {e.response.text}")
        return None

def call_codestral_api(prompt, model="codestral-latest"):
    api_key = os.environ.get("MISTRAL_API_KEY")
    if not api_key:
        print(:  "                    MISTRAL_API_KEY.")
        return None

    url = "https://api.mistral.ai/v1/fim/completions"
    headers = {
        "Content-Type": "application/json",
        "Accept": "application/json",
        "Authorization": f"Bearer {api_key}"
    }
    data = {
        "model": model,
        "prompt": prompt,
        "suffix": "return a + b",
        "max_tokens": 64,
        "temperature": 0
    }
    print(f"Codestral API URL: {url}")
    print(f"Codestral API Headers: {headers}")
    print(f"Codestral API Data: {json.dumps(data)}")
    try:
        response = requests.post(url, headers=headers, json=data)
        response.raise_for_status()
```

```
        response_json = response.json()
        print(response_json)
        if response_json and response_json['choices']:
            content = response_json['choices'][0]['message']['content']
            return content
        else:
            print(f"Codestral API Error: Invalid response format: {response_json}")
            return None
    except requests.exceptions.RequestException as e:
        print(f"Codestral API Error: {e}")
        if e.response:
            print(f"Response status code: {e.response.status_code}")
            print(f"Response content: {e.response.text}")
        return None

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Test Mistral and Codestral APIs.")
    parser.add_argument("--type", type=str, default="mistral", choices=["mistral", "codestral"], help="Type of
    parser.add_argument("--function_calling", action="store_true", help="Enable function calling for Mistral A
    args = parser.parse_args()

    if args.type == "mistral":
        prompt =  "          "
        response = call_mistral_api(prompt, use_function_calling=args.function_calling)
        if response:
            print(f"Response: {response}")
    elif args.type == "codestral":
        prompt = "def f("
        response = call_codestral_api(prompt, model="codestral-latest")
        if response:
            print(f"Response: {response}")
```