my_project_weather_api

Выпуск 1.1

Vitaliy Lozhnichenko

нояб. 08, 2025

Содержание:

Добро пожаловать в документацию к проекту my_project_weather_api!

Этот проект предоставляет инструменты для работы с погодными данными через API.

Глава 1

weather package

1.1 Submodules

1.2 weather.api module

Модуль для работы с Open-Meteo API и OpenStreetMap Reverse Geocoding с помощью requests.

`weather.api.get_coordinates`(*city: str*) → tuple[float, float]

Получает координаты города через Open-Meteo Geocoding API.

Параметры
`city` (*str*) – Название города для поиска координат

Результат
Кортеж с широтой и долготой города

Тип результата
tuple[float, float]

Исключение

- `ValueError` – Если город не найден
- `requests.RequestException` – При ошибках сетевого запроса

`weather.api.get_location_info`(*city: str | None = None*, *lat: float | None = None*, *lon: float | None = None*) → Dict[str, Any] | None

Определяет координаты и название города. Если указан город — используется Open-Meteo Geocoding API. Если указаны координаты — используется OpenStreetMap Reverse Geocoding.

Параметры

- `city` (*str, optional*) – Название города
- `lat` (*float, optional*) – Широта
- `lon` (*float, optional*) – Долгота

Результат
Словарь с ключами „city“, „lat“, „lon“ или None при ошибке

Тип результата
Dict[str, Any]

`weather.api.get_weather(`*city: str | None = None*, *lat: float | None = None*, *lon: float | None = None*`)` → Dict[str, Any]

Получает текущую погоду по названию города или координатам.

Параметры

- `city` (*str, optional*) – Название города
- `lat` (*float, optional*) – Широта
- `lon` (*float, optional*) – Долгота

Результат

Словарь с данными о погоде, включая:

- city: название города
- latitude, longitude: координаты
- current_weather: текущие погодные условия

Тип результата
Dict[str, Any]

Исключение

- `ValueError` – Если не удалось определить местоположение
- `ConnectionError` – При ошибках получения данных о погоде

1.3 weather.cache module

Модуль для простого кэширования ответов API в JSON-файл.

`weather.cache.read_cache(`*city: str*`)` → Dict[str, Any] | None

Читает кэшированные данные для указанного города, если они актуальны.

Параметры
`city` (*str*) – Ключ для поиска в кэше (название города или координаты)

Результат

Данные о погоде из кэша или None, если:

- файл кэша не существует
- запись для города не найдена
- запись устарела (превышен TTL)
- произошла ошибка чтения

Тип результата
Optional[Dict[str, Any]]

weather.cache.write_cache(*city: str*, *data: Dict[str, Any]*) → None

Сохраняет данные в кэш с текущей меткой времени.

Параметры

- city (*str*) – Ключ для сохранения (название города или координаты)
- data (*Dict[str, Any]*) – Данные о погоде для кэширования

Примечание

Если файл кэша не существует - создается новый. Если файл существует - данные обновляются/добавляются. Существующие записи для других городов сохраняются.

1.4 weather.commands module

Основной модуль для обработки команд и вывода погоды. Добавлен цветной вывод с помощью colorama.

weather.commands.handle_command(*args*) → None

Обрабатывает команду пользователя: получает или кэширует погоду.

Параметры

args – Объект с аргументами командной строки, содержащий: - city: название города - lat: широта - lon: долгота - refresh: флаг принудительного обновления кэша

weather.commands.print_weather(*weather_data*) → None

Форматированный и цветной вывод текущей погоды.

Параметры

weather_data (*dict*) – Словарь с данными о погоде, содержащий: - city: название города - latitude, longitude: координаты - current_weather: словарь с текущей погодой

1.5 weather.parser module

Модуль парсер для данных

weather.parser.create_parser() → ArgumentParser

Создаёт и возвращает объект парсера аргументов командной строки.

Результат

настроенный парсер для обработки аргументов погодного приложения.

Тип результата

argparse.ArgumentParser

Глава 2

Indices and tables

- genindex
- modindex
- search

Содержание модулей Python

Алфавитный указатель