skypro

Модули и пакеты. Шпаргалка

Модули и пакеты

Как использовать модуль в другом файле Python с помощью оператора `import`:

```
# Модуль example.py
def hello():
    print("Hello, world!")

# Использование модуля в другом файле
import example

example.hello()  # Вывод: Hello, world!
```

Чтобы создать пакет, необходимо создать директорию с файлом `__init__.py`:

```
mypackage/
    __init__.py
    module1.py
    module2.py
```

Команда для установки пакета с помощью `pip`:

```
pip install package-name
```

Основные команды `pip`:

- `pip install package-name` — устанавливает последнюю версию пакета;
- `pip install package-name==4.8.2` — устанавливает пакет версии 4.8.2;
- `pip install package-name --upgrade` — обновляет версию пакета;
- `pip install --user package-name` — устанавливает пакет в глобальное окружение текущего пользователя;
- `pip install -r requirements.txt` — устанавливает библиотеки, перечисленные в TXT файле;
- `pip uninstall package-name` — удаляет пакеты;
- `pip freeze` — выводит список установленных пакетов в необходимом формате (обычно в requirements.txt);
- `pip list` — выводит список установленных пакетов;
- `pip list --outdated` — выводит список устаревших пакетов;
- `pip show package-name` — показывает информацию об установленном пакете.

Способы импорта модулей и пакетов

1. Импорт модуля целиком (квалифицированный импорт):

```
# greeting.py
def say_hi():  # Определяем функцию
    print('Hi!')

name = 'Bob'  # Определяем переменную


# main.py
import greeting
```

2. Импорт отдельных определений:

```
# main.py
from greeting import say_hi, name
print(name)
say_hi()
```

3. Импорт всего содержимого модуля:

```
from some_module import *
from another_module import *
```

> Этот способ не рекомендуется использовать, так как он может усложнить чтение кода и затруднить поиск ошибок.

Типы импорта модулей в Python

- Абсолютный импорт (примеры выше).
- Относительный импорт.

Примеры относительных импортов:

```
from . import load
```

```
from .load import function
```

```
from .Sound.load import function
```

Виртуальное окружение

Команда для создания виртуального окружения с помощью встроенного модуля `venv`:

```
# Для Windows
python -m venv venv
# Для Linux, macOS
python3 -m venv venv
```

Активация виртуального окружения

Стандартная команда для активации виртуальных окружений из корневой директории проекта:

```
# Для Windows
venv\Scripts\activate
# Для Linux, macOS
source ./venv/bin/activate
```

Команда для выхода из виртуального окружения (доступно только после активации окружения):

```
deactivate
```

Библиотека os

`os.getcwd()` — возвращает текущую рабочую директорию в виде строки:

```
import os

current_directory = os.getcwd()
print(current_directory)

>>> /path/to/directory
```

`os.listdir()` — возвращает список файлов и директорий в указанной директории:

```
import os

directory_files = os.listdir('/path/to/directory')
print(directory_files)

>>> ['file1.txt', 'dir1', 'file2.txt', 'file3.py', '.dir2', 'dir3']
```

`os.path.join()` — используется для объединения путей, возвращает строку, содержащую объединенный путь:

```
import os

path1 = '/path/to/directory'
path2 = 'file.txt'

joined_path = os.path.join(path1, path2)
print(joined_path)

>>> /path/to/directory/file.txt
```

`os.path.dirname()` — возвращает имя директории из указанного пути:

```
import os

path = '/path/to/directory/file.txt'

directory_name = os.path.dirname(path)
print(directory_name)

>>> /path/to/directory
```

Конструкция, чтобы сформировать абсолютный путь к файлу:

```
os.path.join(os.path.dirname(__file__), "/path/to/file.txt")
```

Работа с файлами

Открытие файла:

```
file = open('example.txt', 'r')
```

Где:

- `'example.txt'` — это путь к файлу,
- `'r'` — режим открытия файла на чтение.

Закрытие файла:

```
file.close()
```

Чтение файлов:

- Метод `read()` — читает весь файл целиком:

  ```
  file = open('example.txt', 'r')
  content = file.read()
  ```

- Метод `readline()` — читает одну строку за раз:

```
file = open('example.txt', 'r')
line = file.readline()
```

- Метод `readlines()` — читает все строки и возвращает их в виде списка:

```
file = open('example.txt', 'r')
lines = file.readlines()
```

Запись в файлы:

```
file.write('Hello, World!')
```

Контекстный менеджер

> **Контекстный менеджер** в Python — это конструкция, которая управляет началом и завершением блока операций, обеспечивая выполнение некоторых действий до и после блока кода. Контекстный менеджер используется для автоматического управления ресурсами, такими как файлы, сетевые соединения и т. д.
> Он гарантирует, что ресурсы будут освобождены, например файл будет закрыт, даже если в блоке кода возникнет исключение.

```
with open('example.txt', 'r') as file:
    content = file.read()
    print(content)
```

Режимы работы с файлами

1. Режим чтения (Read Mode — `'r'`) — открывает файл только для чтения.

   Файл должен существовать, иначе возникнет ошибка `FileNotFoundError`.

   ```
   with open('example.txt', 'r') as file:
       content = file.read()
       print(content)
   ```

2. Режим записи (Write Mode — `'w'`) — открывает файл для записи.

   Если файл существует, его содержимое будет удалено. Если файл не существует, он будет создан.

   ```
   with open('example.txt', 'w') as file:
       file.write('Hello, world!')
   ```

3. Режим добавления (Append Mode — `'a'`) — открывает файл для добавления данных.

   Новый контент будет записан в конец файла. Если файл не существует, он будет создан.

   ```
   with open('example.txt', 'a') as file:
       file.write('Adding a new line.\n')
   ```

4. Режим чтения и записи (Read and Write Mode — `'r+'`) — открывает файл для чтения и записи.

   Файл должен существовать, иначе будет вызвана ошибка `FileNotFoundError`.

   ```
   with open('example.txt', 'r+') as file:
       content = file.read()
       file.write('\nNew content')
   ```

Параметр encoding

Параметр `encoding` указывает, какую кодировку использовать для чтения или записи текстовых данных:

```
file = open('example.txt', 'r', encoding='utf-8')
```

1. Создание файла и запись данных:

   ```
   with open('example.txt', 'w', encoding='utf-8') as file:
       file.write("Привет, мир!")
   ```

2. Чтение из файла:

   ```
   with open('example.txt', 'r', encoding='utf-8') as file:
       content = file.read()
       print(content)
   ```

Структура простого Python-проекта

```
.
├── src
│   ├── __init__.py
│   ├── helloworld.py
│   └── helpers.py
├── tests
│   ├── __init__.py
│   └── test_demo.py
├── .venv/
├── .git
├── .idea/
├── requirements.txt
└── README.md
```

Расширенная структура Python-проекта

```
.
├── src
│   ├── __init__.py
│   ├── helloworld.py
│   └── helpers.py
├── data
│   ├── processed
│   │   └── processeddata.csv
│   └── raw
│       └── sampledata.csv
├── tests
│   ├── __init__.py
│   └── test_demo.py
├── docs/
│   ├── hello.md
│   └── world.md
├── .venv/
├── .env
├── .git
├── .idea/
├── .flake8
├── requirements.txt
├── LICENSE
├── README.md
└── setup.cfg
```