



Модули и пакеты

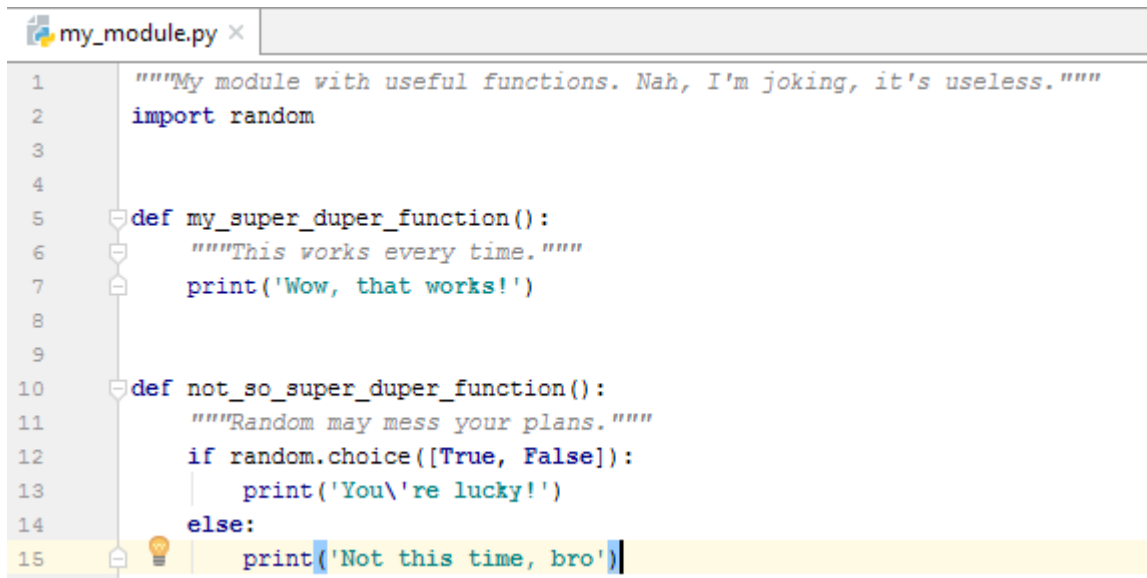


Темы для изучения

- Модуль
- Пакет
- Как Python находит что импортировать
- Пакетный менеджер
- Зависимости
- Виртуальное окружение

Что такое модуль?

Модуль – это любой файл .py, в котором содержится код.
Вы можете импортировать этот код из одного модуля в другой.



```
1  """My module with useful functions. Nah, I'm joking, it's useless."""
2  import random
3
4
5  def my_super_duper_function():
6      """This works every time."""
7      print('Wow, that works!')
8
9
10 def not_so_super_duper_function():
11     """Random may mess your plans."""
12     if random.choice([True, False]):
13         print('You\'re lucky!')
14     else:
15         print('Not this time, bro')
```

Как импортировать?

- Целиком: `import my_module`
- Частично: `from my_module import my_super_duper_function`
- Целиком с переименованием: `import my_module as mm`
- Частично с переименованием: `from my_module import my_super_duper_function as msdf`

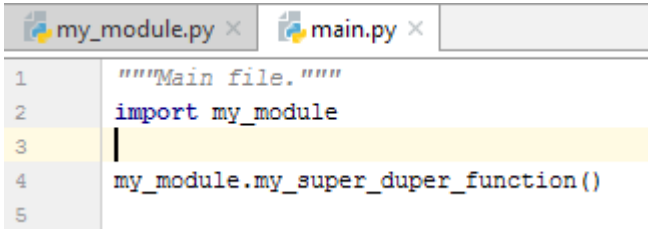
Как делать не нужно:

- `from my_module import *`

Если вам нужно всё содержимое модуля – импортируйте его целиком. Импорт через звездочку может превратить ваше пространство имён к хаос.

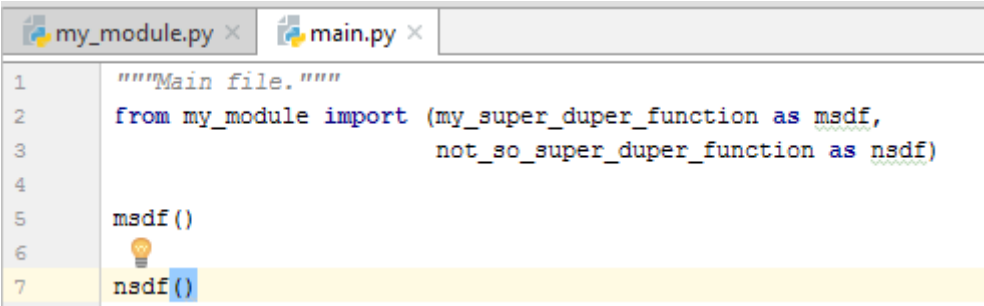
А как пользоваться?

Получить доступ к внутренностям модуля, если вы импортировали его целиком, можно через точку



```
1 """Main file."""
2 import my_module
3
4 my_module.my_super_duper_function()
5
```

Если вы импортировали что-то конкретное, то это уже можно вызывать



```
1 """Main file."""
2 from my_module import (my_super_duper_function as msdf,
3                         not_so_super_duper_function as nsdf)
4
5 msdf()
6
7 nsdf()
```

А зачем так сложно?

Такой подход позволит вам правильно структурировать код, легче находить нужные части. Он используется повсеместно – в стандартной библиотеке Python, в сторонних библиотеках, во фреймворках типа Django, и т.д.

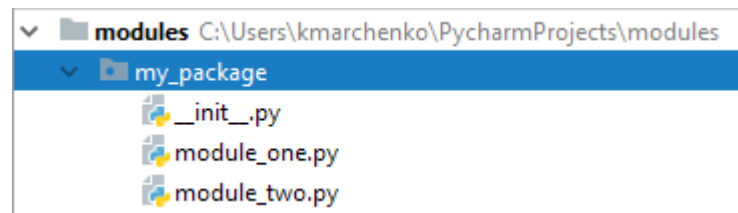
Что такое пакет модулей?

Как и пакет с пакетами, пакет модулей в Python – это возможность еще раз упростить себе жизнь. Модули со сходным функционалом вы можете объединить в пакет, и держать в одном месте. При импорте пакета вы можете обращаться к нужным модулям.



Как создать пакет?

Пакет в Python – это обычная директория, в которой хранятся модули с кодом. Для того, чтобы директория стала пакетом, туда необходимо добавить файл с именем `__init__.py`



Импорт модуля из пакета

Импортируется он как обычно:

- `import my_package.module_one`
- `from my_package import module_one`
- `import my_package.module_two as mt`

Получить доступ к нужному коду можно через точку:

```
my_package.module_one.my_function()  
mt.my_func()
```

Где Python их находит?

Python ищет модули и пакеты используя системную переменную PYTHONPATH.

Посмотреть ее содержимое внутри Python можно, если импортировать модуль `sys` и обратиться к переменной `path`:

```
>>> import sys  
>>> print(sys.path)
```

В первую очередь python ищет имена модулей в той директории, откуда вы вызвали код. Дальше он идет по системным директориям, которые, как правило, устанавливаются автоматически, когда вы впервые устанавливаете Python.

Например, так он находит всё то, что содержится в стандартной библиотеке Python, или в библиотеках, которые вы установили пакетным менеджером.

Пакетный менеджер

Нет смысла изобретать велосипед, т.к. множество разработчиков создают свои собственные библиотеки, которые могут решать нужные вам задачи.

Пожалуй, самый распространенный пакетный менеджер сейчас, это `pip`.

Для того чтобы установить нужный пакет, нужно использовать команду
`pip install package_name`

Он проверит все зависимости пакета, установит их, а потом установит сам пакет.

После этого вы можете импортировать и использовать его в своих программах.

Зависимости

Если вы собираетесь где-то опубликовать свой проект (например, на гитхабе), или написали новую библиотеку, то чтобы люди не получали ошибку импорта, вам нужно указать зависимости кода.

Их перечисляют в файле `requirements.txt`

Для того чтобы установить их вручную, можно также воспользоваться `pip`:

```
pip install -r requirements.txt
```

В зависимостях лучше указывать конкретные версии библиотек, которые вы использовали. В этом поможет команда `pip freeze`.

```
pip freeze > requirements.txt
```

Виртуальное окружение

Поскольку разный код может требовать разные версии библиотек, вы можете использовать виртуальное окружение. У вас будет отдельный интерпретатор и набор библиотек. Установите пакет для создания виртуального окружения командой `pip install virtualenv`.


Создать окружение можно командой `virtualenv .` (точка тут обязательна, либо нужен путь до директории)

Активировать – `source Scripts/activate`

Дополнительные материалы

- <https://docs.python.org/3/library/> - стандартная библиотека Python
- <https://pypi.org/project/pip/> - документация по pip
- <https://virtualenv.pypa.io/en/latest/> - документация по virtualenv

Thanks for your
attention

 [artezio_software](#)

 info@artezio.com

www.artezio.com