МГТУ им. Н.Э. Баумана
ТЕХНОПАРК
Mail.Ru Group

# Углубленный Python

Лекция 8

Латкин Игорь

“

Не забудьте отметиться на занятии!

*Цитата великих*

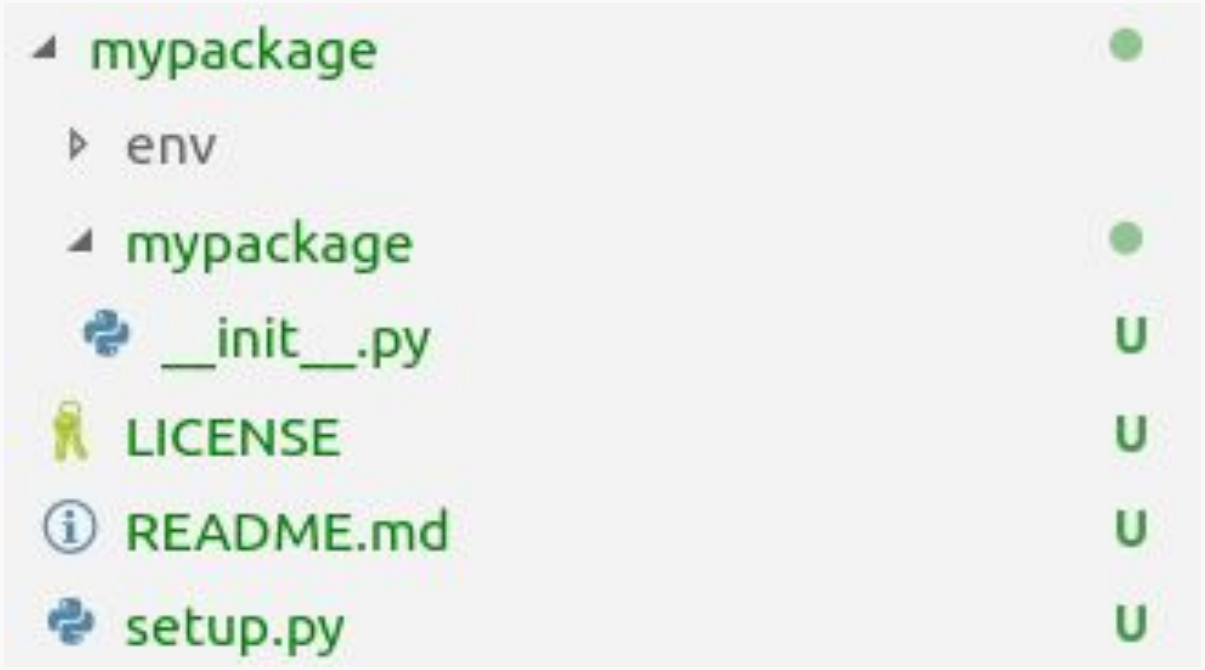
## Повестка дня

1. packaging
2. typing
3. decimal
4. logging

## Packaging

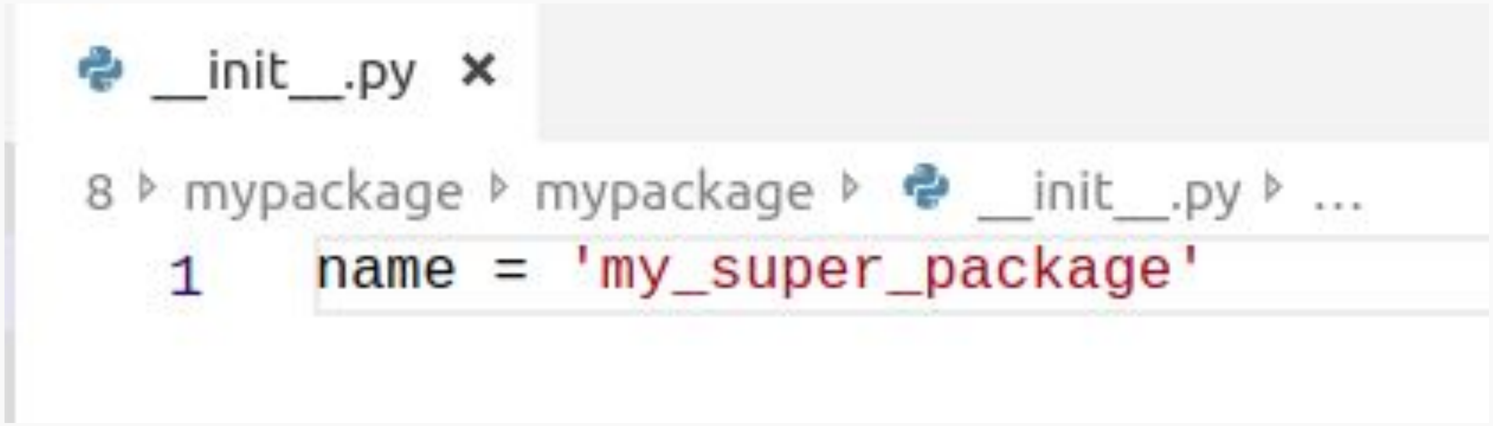
https://packaging.python.org/tutorials/packaging-projects/

## Структура Python проекта



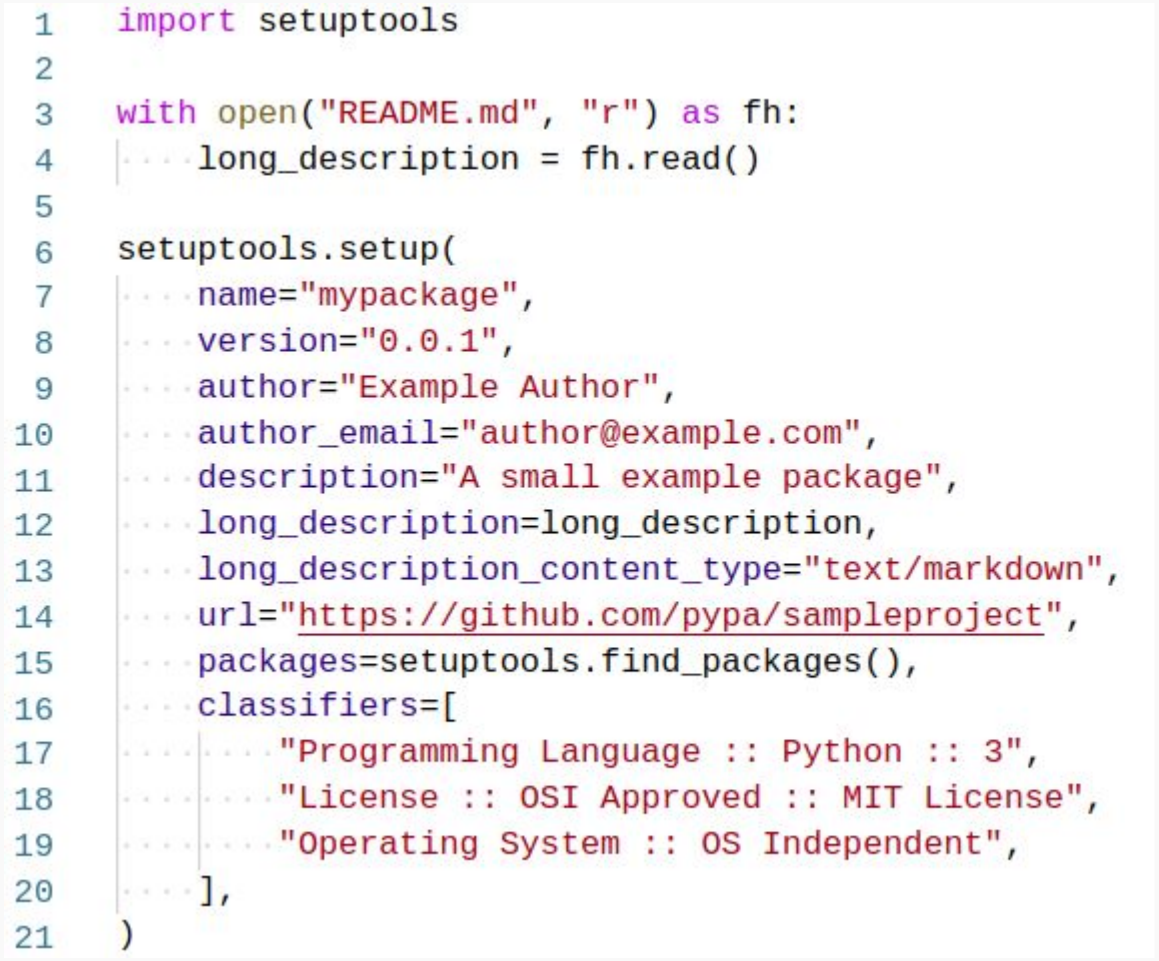
```
◢ mypackage                ●
  ▷ env
  ◢ mypackage              ●
      __init__.py          U
  LICENSE                  U
  README.md                U
  setup.py                 U
```

## Структура Python проекта.



```python
name = 'my_super_package'
```

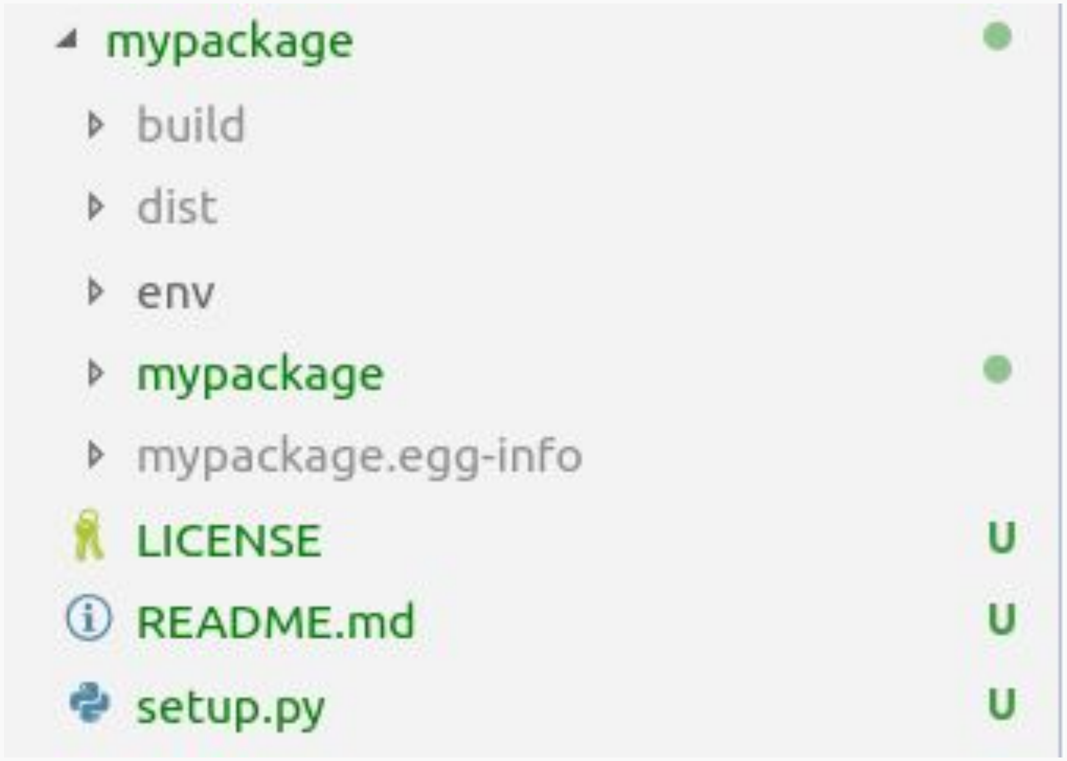
## Структура Python проекта. setup.py



```python
import setuptools

with open("README.md", "r") as fh:
    long_description = fh.read()

setuptools.setup(
    name="mypackage",
    version="0.0.1",
    author="Example Author",
    author_email="author@example.com",
    description="A small example package",
    long_description=long_description,
    long_description_content_type="text/markdown",
    url="https://github.com/pypa/sampleproject",
    packages=setuptools.find_packages(),
    classifiers=[
        "Programming Language :: Python :: 3",
        "License :: OSI Approved :: MIT License",
        "Operating System :: OS Independent",
    ],
)
```

## Создание пакета

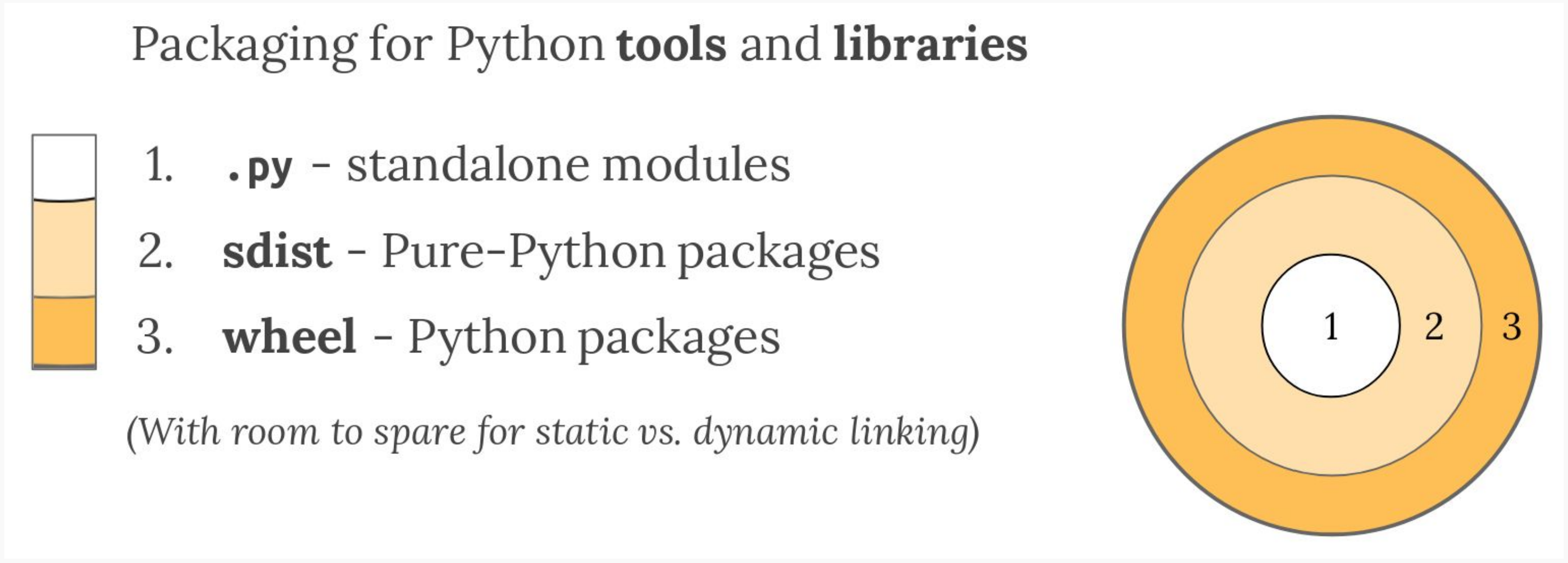
➜ `python setup.py sdist bdist_wheel`



```
mypackage          ●
  build
  dist
  env
  mypackage        ●
  mypackage.egg-info
  LICENSE          U
  README.md        U
  setup.py         U
```

## Создание пакета

```
➜ python setup.py sdist bdist_wheel
```



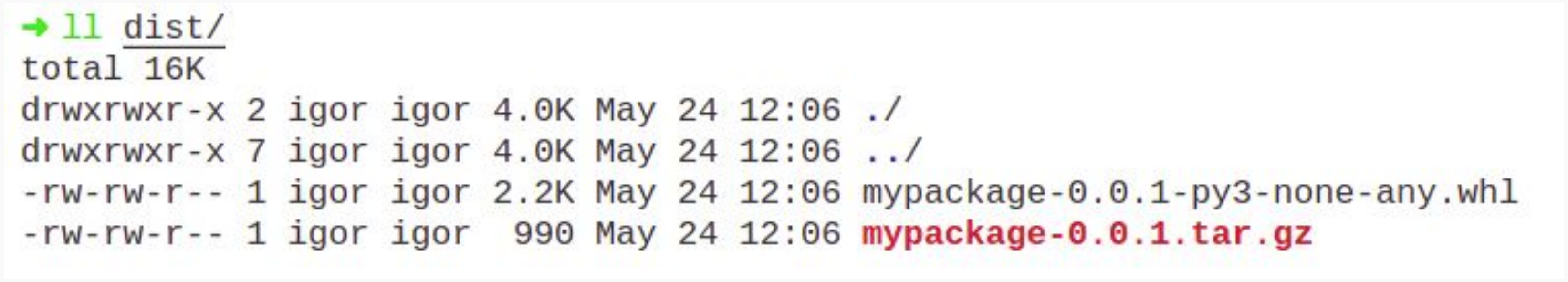
Packaging for Python **tools** and **libraries**

1. **.py** - standalone modules
2. **sdist** - Pure-Python packages
3. **wheel** - Python packages

*(With room to spare for static vs. dynamic linking)*

## Создание пакета

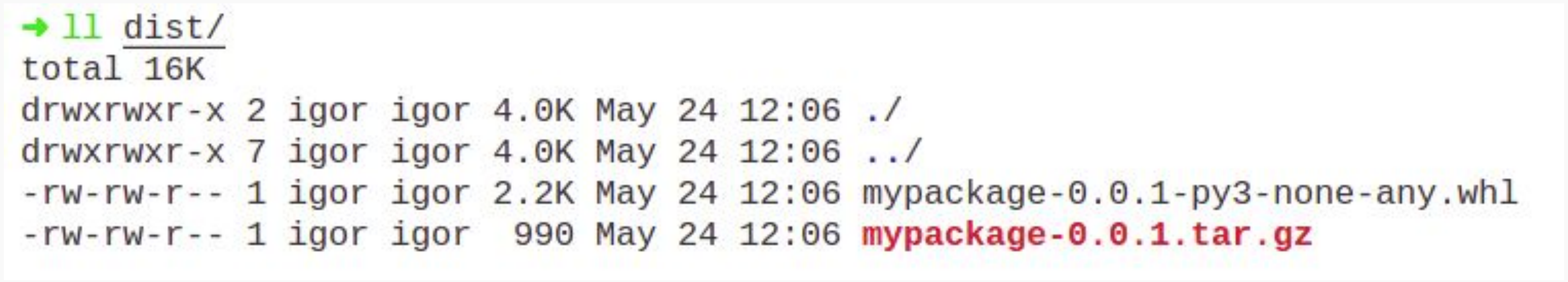
```
➜ python setup.py sdist bdist_wheel
```



```
➜ ll dist/
total 16K
drwxrwxr-x 2 igor igor 4.0K May 24 12:06 ./
drwxrwxr-x 7 igor igor 4.0K May 24 12:06 ../
-rw-rw-r-- 1 igor igor 2.2K May 24 12:06 mypackage-0.0.1-py3-none-any.whl
-rw-rw-r-- 1 igor igor  990 May 24 12:06 mypackage-0.0.1.tar.gz
```

## Создание пакета

```
➜ python setup.py sdist bdist_wheel
```



```
➜ ll dist/
total 16K
drwxrwxr-x 2 igor igor 4.0K May 24 12:06 ./
drwxrwxr-x 7 igor igor 4.0K May 24 12:06 ../
-rw-rw-r-- 1 igor igor 2.2K May 24 12:06 mypackage-0.0.1-py3-none-any.whl
-rw-rw-r-- 1 igor igor  990 May 24 12:06 mypackage-0.0.1.tar.gz
```

## Публикация пакета

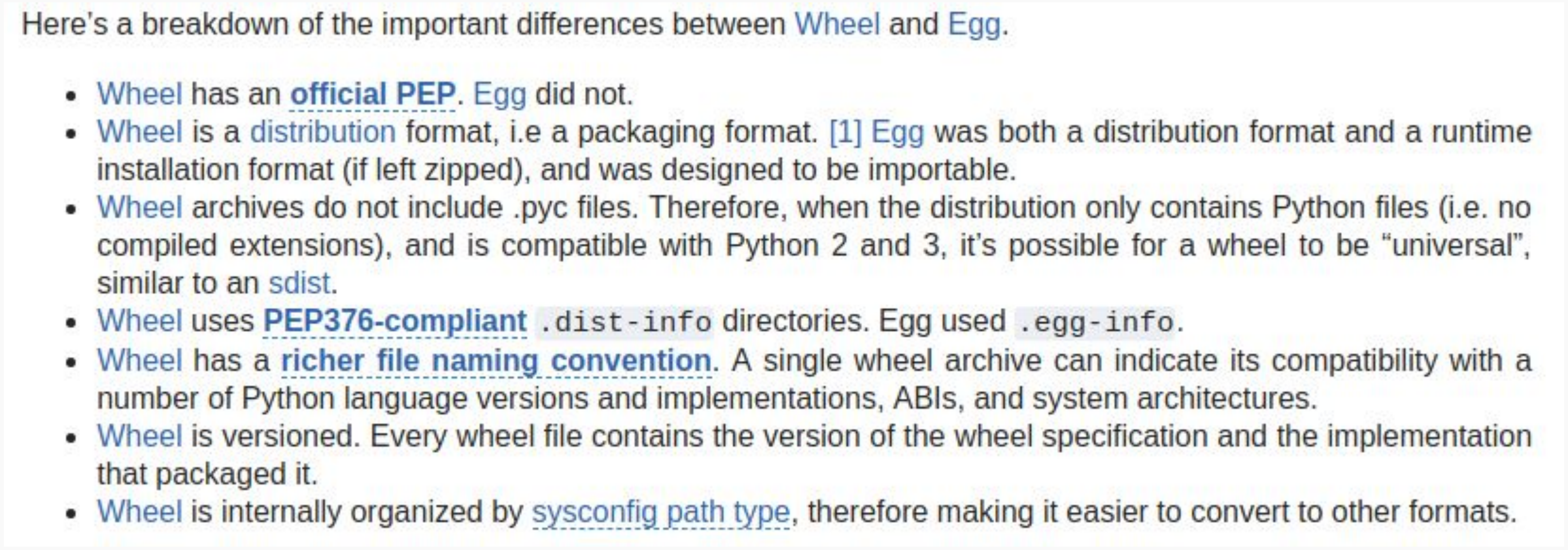
https://test.pypi.org/

https://pypi.org/

```
➜ pip install twine
➜ twine upload dist/*
```

## Eggs vs Wheels

https://packaging.python.org/discussions/wheel-vs-egg/



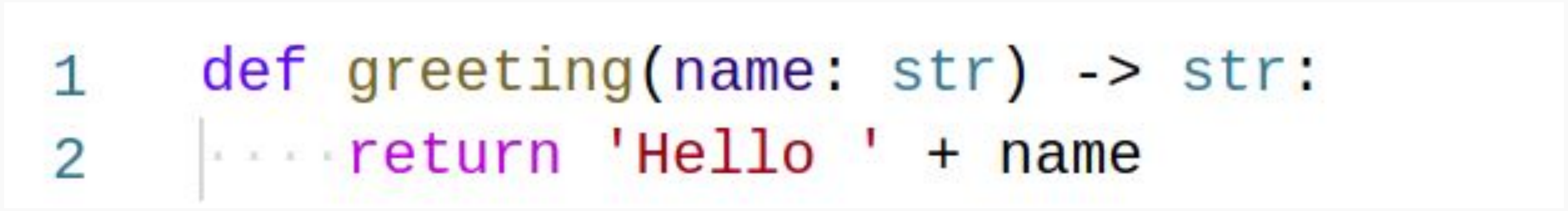
Here's a breakdown of the important differences between Wheel and Egg.

- Wheel has an **official PEP**. Egg did not.
- Wheel is a distribution format, i.e a packaging format. [1] Egg was both a distribution format and a runtime installation format (if left zipped), and was designed to be importable.
- Wheel archives do not include .pyc files. Therefore, when the distribution only contains Python files (i.e. no compiled extensions), and is compatible with Python 2 and 3, it's possible for a wheel to be "universal", similar to an sdist.
- Wheel uses **PEP376-compliant** `.dist-info` directories. Egg used `.egg-info`.
- Wheel has a **richer file naming convention**. A single wheel archive can indicate its compatibility with a number of Python language versions and implementations, ABIs, and system architectures.
- Wheel is versioned. Every wheel file contains the version of the wheel specification and the implementation that packaged it.
- Wheel is internally organized by sysconfig path type, therefore making it easier to convert to other formats.

## Typing

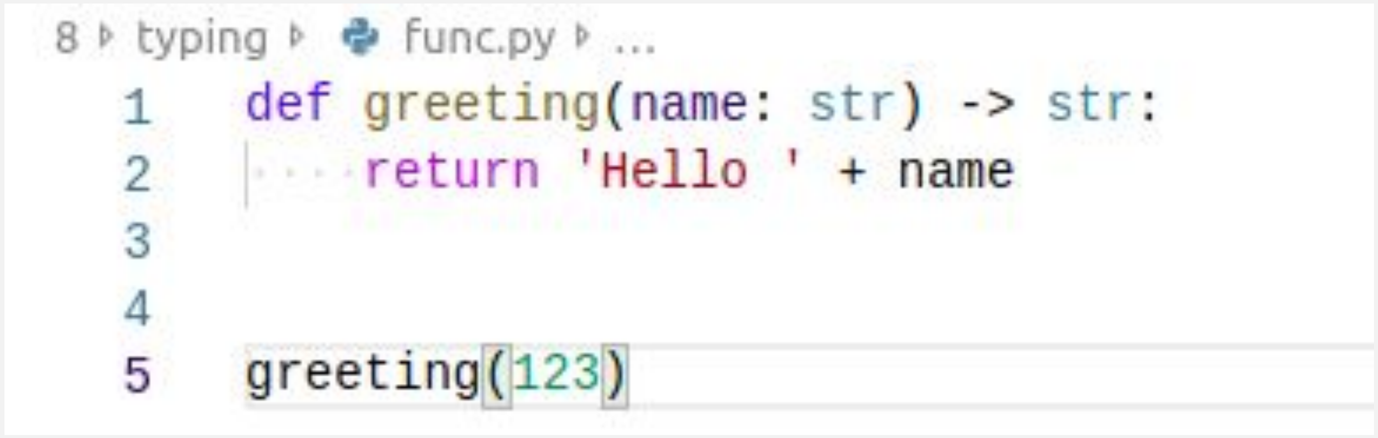
https://docs.python.org/3/library/typing.html

## Базовое использование

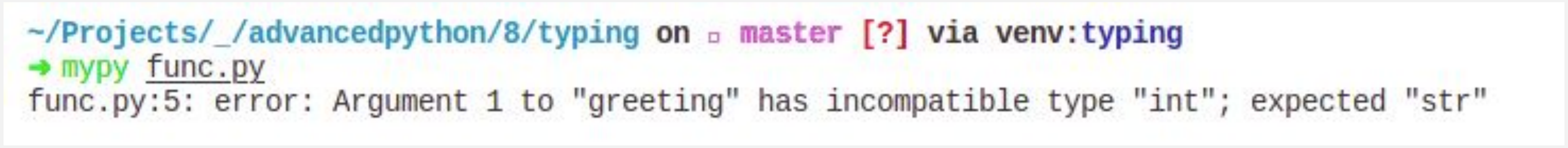


```python
def greeting(name: str) -> str:
    return 'Hello ' + name
```

## Базовое использование



```python
def greeting(name: str) -> str:
    return 'Hello ' + name


greeting(123)
```



```
~/Projects/_/advancedpython/8/typing on ▫ master [?] via venv:typing
➜ mypy func.py
func.py:5: error: Argument 1 to "greeting" has incompatible type "int"; expected "str"
```

## Более сложные структуры

- `Any` - произвольный тип
- `List[int]`
- `Tuple[int, str]`
- `Union[str, bytes]` - допустим любой из перчисленных типов
- `Callable[[int, int], float]` - “функция”, принимающая 2 целых числа и возвращающая float
- `Iterable[T]`
- `Mapping[K, V], Dict[K, V]`
- `Awaitable[T_co]`
- `Type[T]`

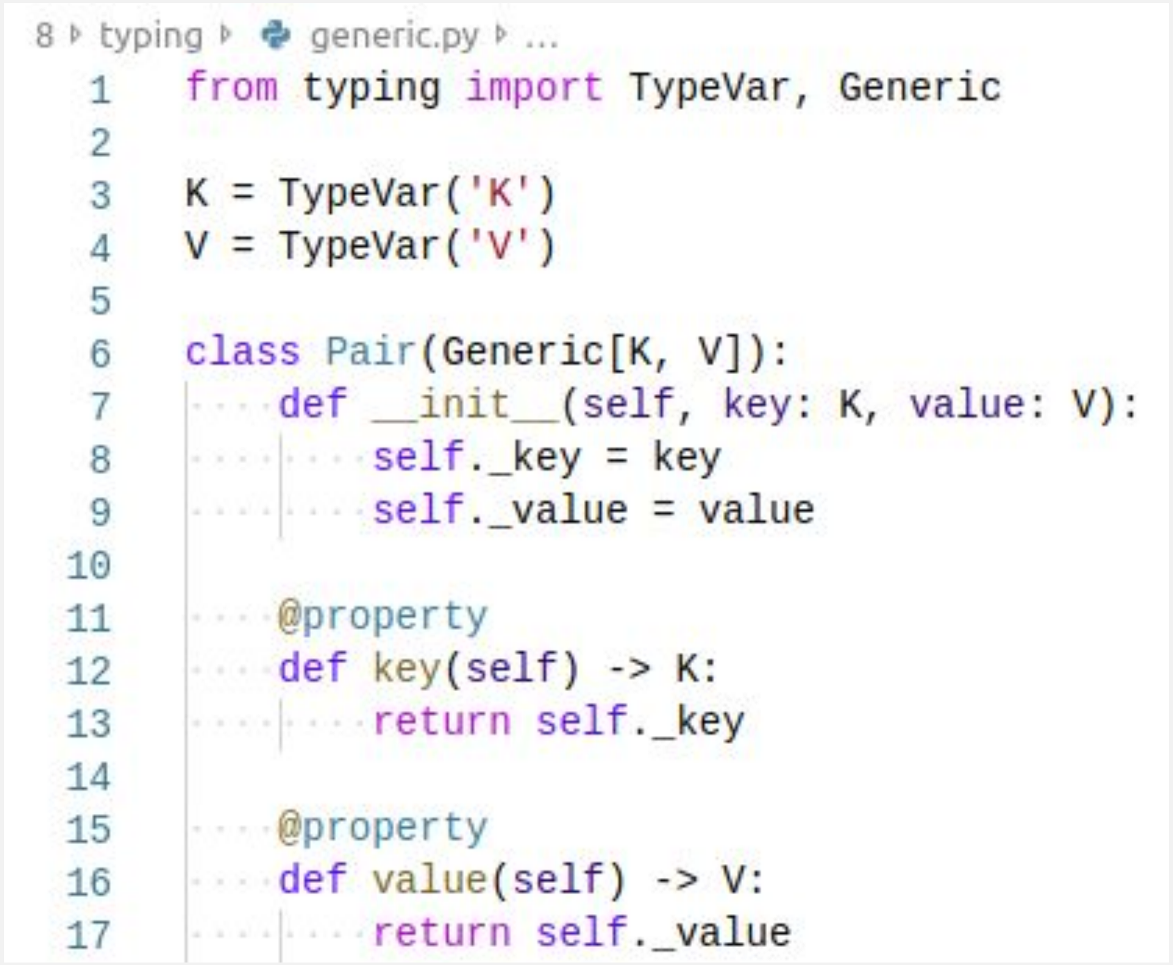
## Алиасы типов

```python
from typing import List
Vector = List[float]

def scale(scalar: float, vector: Vector) -> Vector:
    return [scalar * num for num in vector]

# typechecks; a list of floats qualifies as a Vector.
new_vector = scale(2.0, [1.0, -4.2, 5.4])
```

## Алиасы типов

```python
from typing import Dict, Tuple, Sequence

ConnectionOptions = Dict[str, str]
Address = Tuple[str, int]
Server = Tuple[Address, ConnectionOptions]

def broadcast_message(message: str, servers: Sequence[Server]) -> None:
    ...

# The static type checker will treat the previous type signature as
# being exactly equivalent to this one.
def broadcast_message(
        message: str,
        servers: Sequence[Tuple[Tuple[str, int], Dict[str, str]]]) -> None:
    ...
```

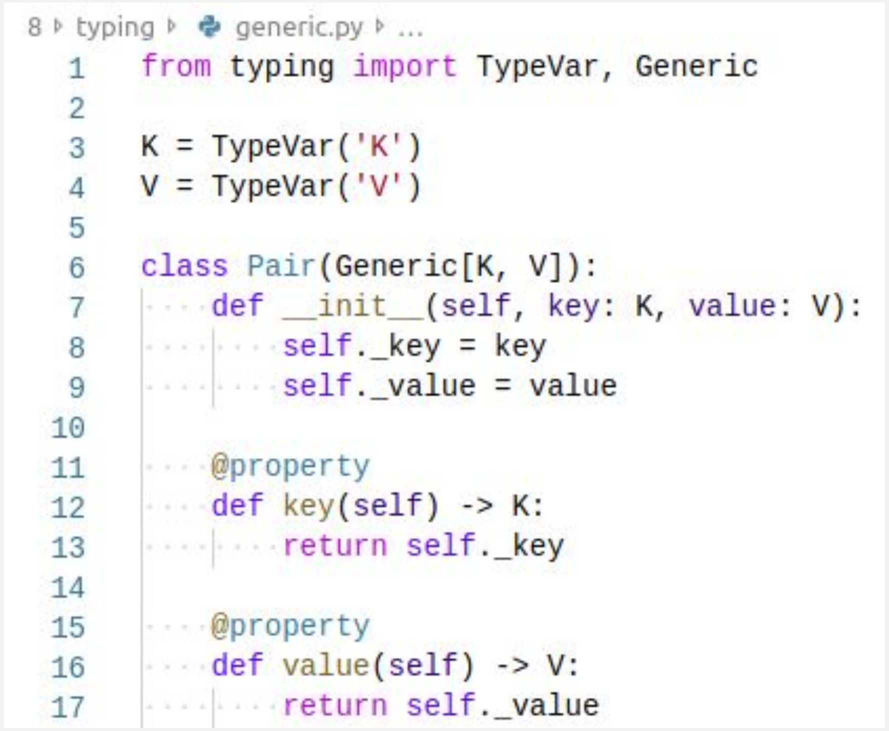
## Generic



8 ▸ typing ▸ generic.py ▸ ...

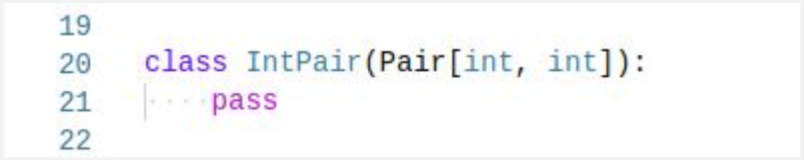
```python
from typing import TypeVar, Generic

K = TypeVar('K')
V = TypeVar('V')

class Pair(Generic[K, V]):
    def __init__(self, key: K, value: V):
        self._key = key
        self._value = value

    @property
    def key(self) -> K:
        return self._key

    @property
    def value(self) -> V:
        return self._value
```

## Generic



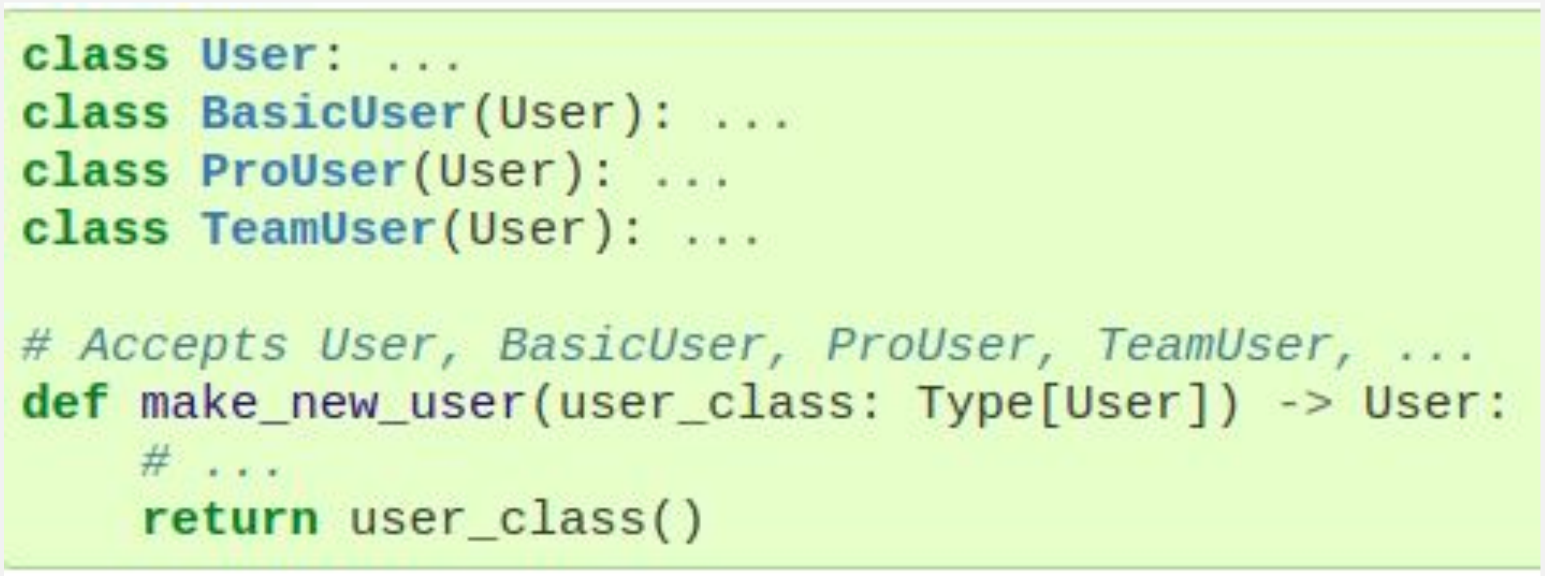
8 ▸ typing ▸ generic.py ▸ ...

```python
from typing import TypeVar, Generic

K = TypeVar('K')
V = TypeVar('V')

class Pair(Generic[K, V]):
    def __init__(self, key: K, value: V):
        self._key = key
        self._value = value

    @property
    def key(self) -> K:
        return self._key

    @property
    def value(self) -> V:
        return self._value
```



```python

class IntPair(Pair[int, int]):
    pass

```

## Type[T]



```
class User: ...
class BasicUser(User): ...
class ProUser(User): ...
class TeamUser(User): ...

# Accepts User, BasicUser, ProUser, TeamUser, ...
def make_new_user(user_class: Type[User]) -> User:
    # ...
    return user_class()
```

## Decimal

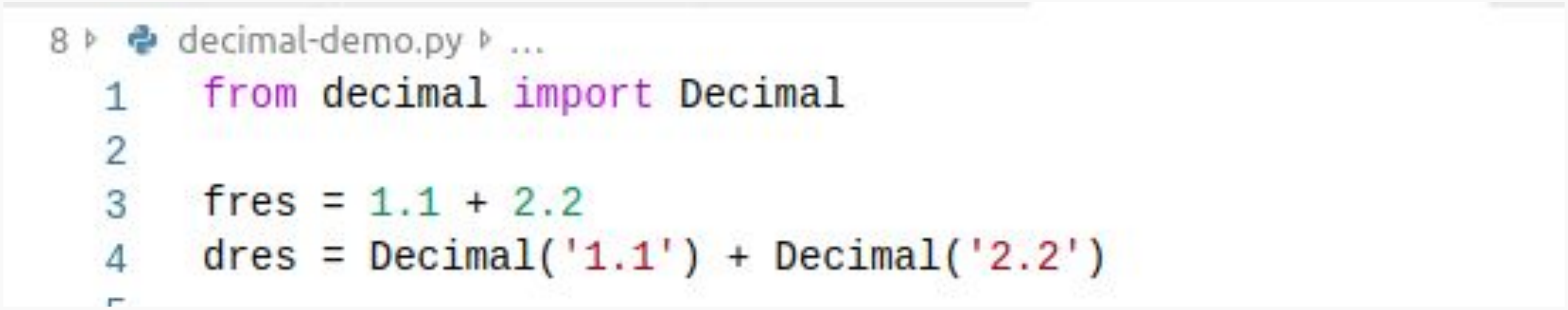
https://docs.python.org/3/library/decimal.html

## Decimal. Зачем?

- float: 1.1 + 2.2 == 3.3000000000000003
- decimal: 1.1 + 2.2 == 3.3

- float: 0.1 + 0.1 + 0.1 - 0.3 == 5.55111512312578e-017
- decimal: 0.1 + 0.1 + 0.1 - 0.3 == 0

## Decimal



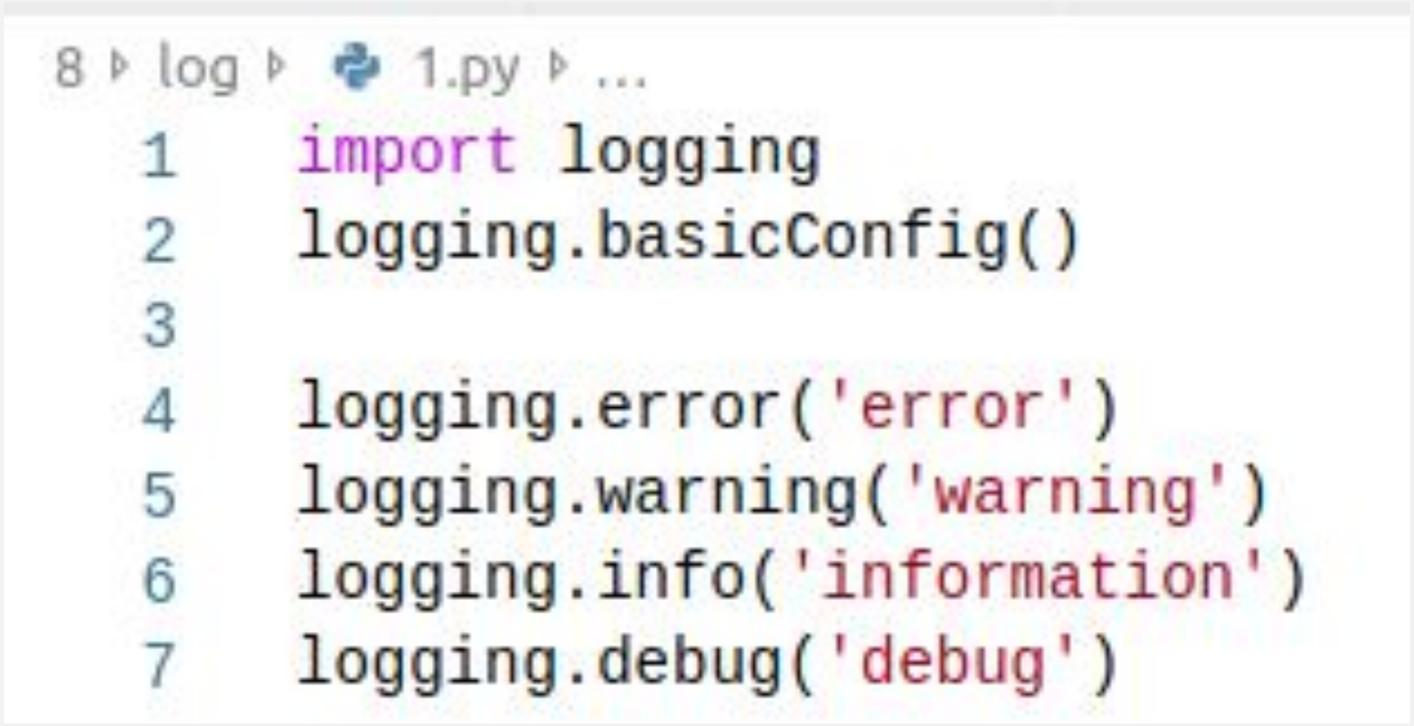
8 decimal-demo.py …

```python
from decimal import Decimal

fres = 1.1 + 2.2
dres = Decimal('1.1') + Decimal('2.2')
```

## Logging

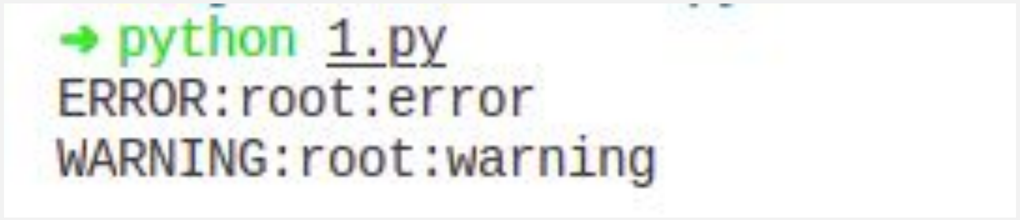
https://docs.python.org/3/library/logging.html

## Простое использование



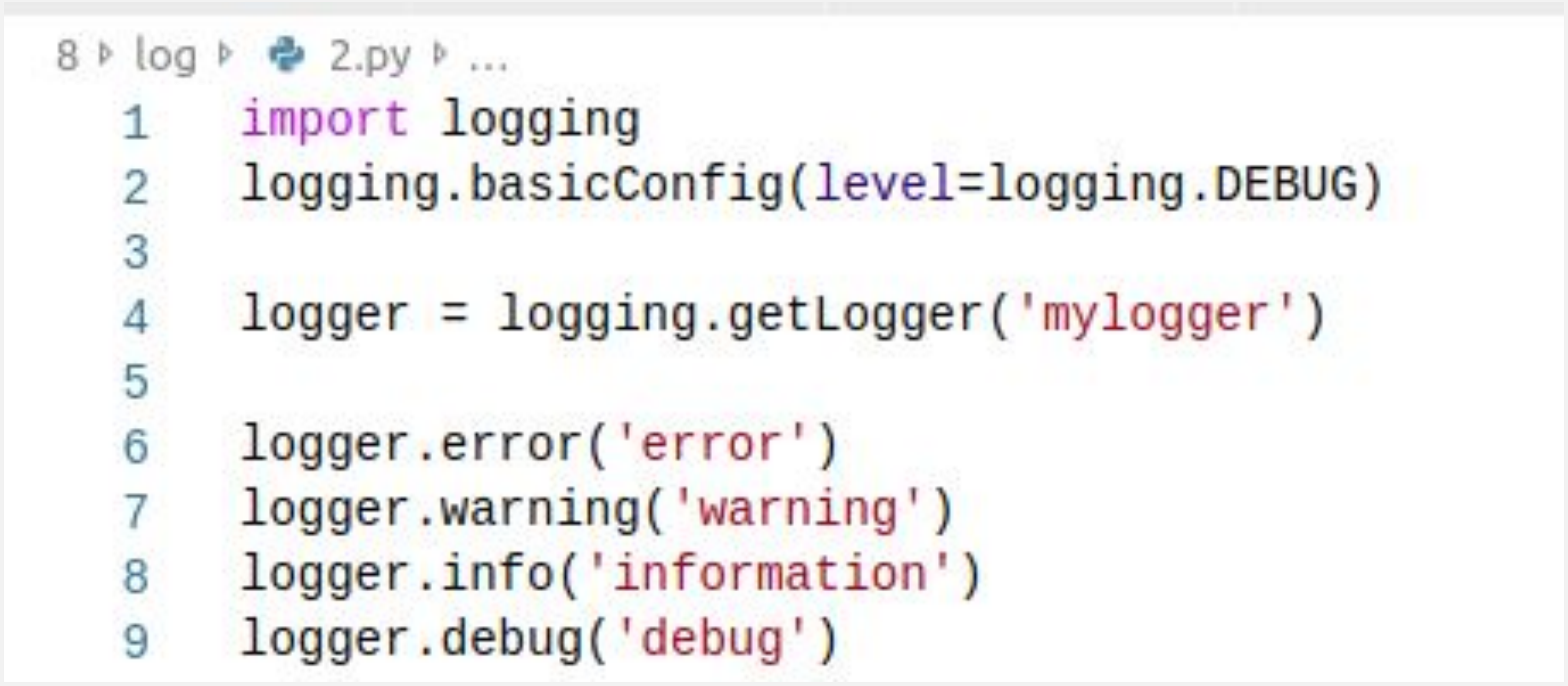
8 ▸ log ▸ 1.py ▸ ...

```python
import logging
logging.basicConfig()

logging.error('error')
logging.warning('warning')
logging.info('information')
logging.debug('debug')
```

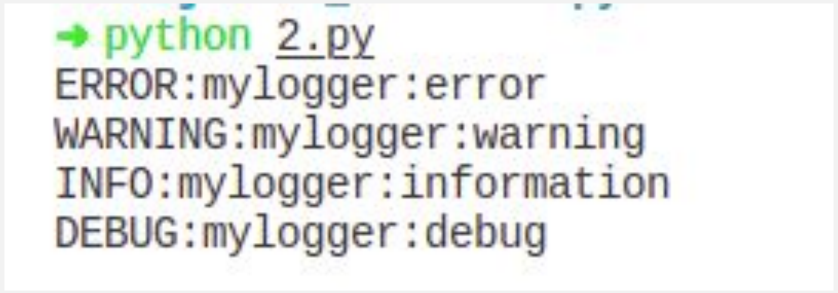


```
➜ python 1.py
ERROR:root:error
WARNING:root:warning
```

## Простое использование


8 ▸ log ▸ 2.py ▸ ...

```python
import logging
logging.basicConfig(level=logging.DEBUG)

logger = logging.getLogger('mylogger')

logger.error('error')
logger.warning('warning')
logger.info('information')
logger.debug('debug')
```



```
➜ python 2.py
ERROR:mylogger:error
WARNING:mylogger:warning
INFO:mylogger:information
DEBUG:mylogger:debug
```

## Logging. Components

1. Formatter
2. Handler
3. Logger
4. LogRecord

https://realpython.com/python-logging/

## Logger. Extra.

```
FORMAT = '%(asctime)-15s %(clientip)s %(user)-8s %(message)s'
logging.basicConfig(format=FORMAT)
d = {'clientip': '192.168.0.1', 'user': 'fbloggs'}
logger = logging.getLogger('tcpserver')
logger.warning('Protocol problem: %s', 'connection reset', extra=d)
```

That's all Folks!

МГТУ
им. Н. Э. Баумана
ТЕХНОПАРК
Mail.Ru Group

telegram: alexopryshko
email: alexopryshko@gmail.com

telegram: igorcoding
email: igor.latkin@outlook.com

Спасибо за внимание!