

Основы языка программирования Python¹

Модули

```
def add(a, b):  
    return a + b  
  
def mul(a, b):  
    return a * b  
  
def add_abs(a, b):  
    return abs(a + b)  
  
def spam(a, b, action):  
    return action(a, b) * 2 + 1  
  
foo = spam(2, 3, add)  
bar = spam(2, 3, mul)  
baz = spam(2, -5, add_abs)  
  
print(f"{foo=}")  
print(f"{bar=}")  
print(f"{baz=}")
```

```
foo=11  
bar=13  
baz=7
```

```
# tools.py
```

```
def add(a, b):  
    return a + b
```

```
def mul(a, b):  
    return a * b
```

```
def add_abs(a, b):  
    return abs(a + b)
```

```
# example_modules_01.py
```

```
import tools
```

```
def spam(a, b, action):  
    return action(a, b) * 2 + 1
```

```
foo = spam(2, 3, tools.add)  
bar = spam(2, 3, tools.mul)  
baz = spam(2, -5, tools.add_abs)
```

```
print(f"{foo=}")  
print(f"{bar=}")  
print(f"{baz=}")
```

```
foo=11  
bar=13  
baz=7
```

```
# tools.py
```

```
def add(a, b):  
    return a + b
```

```
def mul(a, b):  
    return a * b
```

```
def add_abs(a, b):  
    return abs(a + b)
```

```
# example_modules_02.py
```

```
import tools as tl
```

```
def spam(a, b, action):  
    return action(a, b) * 2 + 1
```

```
foo = spam(2, 3, tl.add)  
bar = spam(2, 3, tl.mul)  
baz = spam(2, -5, tl.add_abs)
```

```
print(f"foo={}")  
print(f"bar={}")  
print(f"baz={}")
```

```
foo=11  
bar=13  
baz=7
```

```
# tools.py
```

```
def add(a, b):  
    return a + b
```

```
def mul(a, b):  
    return a * b
```

```
def add_abs(a, b):  
    return abs(a + b)
```

```
# example_modules_03.py
```

```
from tools import add, add_abs, mul
```

```
def spam(a, b, action):  
    return action(a, b) * 2 + 1
```

```
foo = spam(2, 3, add)  
bar = spam(2, 3, mul)  
baz = spam(2, -5, add_abs)
```

```
print(f"{foo=}")  
print(f"{bar=}")  
print(f"{baz=}")
```

```
foo=11  
bar=13  
baz=7
```

```
# tools.py
```

```
def add(a, b):  
    return a + b
```

```
def mul(a, b):  
    return a * b
```

```
def add_abs(a, b):  
    return abs(a + b)
```

```
# example_modules_04.py
```

```
# Не рекомендуемый способ импорта!
```

```
from tools import *
```

```
def spam(a, b, action):  
    return action(a, b) * 2 + 1
```

```
foo = spam(2, 3, add)  
bar = spam(2, 3, mul)  
baz = spam(2, -5, add_abs)
```

```
print(f"{foo=}")  
print(f"{bar=}")  
print(f"{baz=}")
```

```
foo=11  
bar=13  
baz=7
```

```
# tools_debug.py
```

```
print("Run tool_debug - 1")
```

```
def add(a, b):  
    return a + b
```

```
print("Run tool_debug - 2")
```

```
def mul(a, b):  
    return a * b
```

```
print("Run tool_debug - 3")
```

```
def add_abs(a, b):  
    return abs(a + b)
```

```
print("Run tool_debug - 4")
```

```
# example_modules_05.py
```

```
print("Main script - 1")
```

```
import tools_debug
```

```
print("Main script - 2")
```

```
def spam(a, b, action):  
    return action(a, b) * 2 + 1
```

```
foo = spam(2, 3, tools_debug.add)  
bar = spam(2, 3, tools_debug.mul)  
baz = spam(2, -5, tools_debug.add_abs)
```

```
print(f"{foo=}")
```

```
print(f"{bar=}")
```

```
print(f"{baz=}")
```

```
Main script - 1  
Run tool_debug - 1  
Run tool_debug - 2  
Run tool_debug - 3  
Run tool_debug - 4  
Main script - 2  
foo=11  
bar=13  
baz=7
```

```
# example_modules_01.py
```

```
import tools
```

```
def spam(a, b, action):  
    return action(a, b) * 2 + 1
```

```
foo = spam(2, 3, tools.add)  
bar = spam(2, 3, tools.mul)  
baz = spam(2, -5, tools.add_abs)
```

```
print(f"{foo=}")  
print(f"{bar=}")  
print(f"{baz=}")
```

```
# example_modules_06.py
```

```
import tools  
from example_modules_01 import spam
```

```
foo = spam(3, 4, tools.add)  
bar = spam(5, 6, tools.mul)  
baz = spam(7, -8, tools.add_abs)
```

```
print(f"{foo=}")  
print(f"{bar=}")  
print(f"{baz=}")
```

```
foo=11  
bar=13  
baz=7  
foo=15  
bar=61  
baz=3
```



```
# tools_vars.py
```

```
def add(a, b):  
    return a + b
```

```
def mul(a, b):  
    return a * b
```

```
def add_abs(a, b):  
    return abs(a + b)
```

```
print(f"tools_vars.py: {__name__=}")
```

```
# example_modules_07.py
```

```
import tools_vars
```

```
print(f"example_modules_07.py: {__name__=}")
```

```
def spam(a, b, action):  
    return action(a, b) * 2 + 1
```

```
foo = spam(2, 3, tools_vars.add)
```

```
bar = spam(2, 3, tools_vars.mul)
```

```
baz = spam(2, -5, tools_vars.add_abs)
```

```
print(f"{foo=}")
```

```
print(f"{bar=}")
```

```
print(f"{baz=}")
```

```
tools_vars.py: __name__='tools_vars'
```

```
example_modules_07.py: __name__='__main__'
```

```
foo=11
```

```
bar=13
```

```
baz=7
```

```
# example_modules_08.py
```

```
import tools
```

```
def spam(a, b, action):  
    return action(a, b) * 2 + 1
```

```
if __name__ == "__main__":  
    foo = spam(2, 3, tools.add)  
    bar = spam(2, 3, tools.mul)  
    baz = spam(2, -5, tools.add_abs)  
  
    print(f"{foo=}")  
    print(f"{bar=}")  
    print(f"{baz=}")
```

```
# example_modules_09.py
```

```
import tools  
from example_modules_08 import spam
```

```
foo = spam(3, 4, tools.add)  
bar = spam(5, 6, tools.mul)  
baz = spam(7, -8, tools.add_abs)
```

```
print(f"{foo=}")  
print(f"{bar=}")  
print(f"{baz=}")
```

```
foo=15  
bar=61  
baz=3
```


Пакеты модулей

`src/11. Modules/example_modules_10.py`

`src/11. Modules/example_modules_11.py`

Установка библиотек с помощью pip

https://pypi.org




[Help](#) [Donate](#) [Log in](#) [Register](#)

Filter by [classifier](#)


- Framework
- Topic
- Development Status
- License
- Programming Language
- Operating System
- Environment
- Intended Audience
- Natural Language
- Typing

1,335 projects for "physics"


Order by Relevance




physics 2.0.0
An Educational project about Physics




physics-tenpy 0.4.1
Simulation of quantum many-body systems with tensor networks in Python




wasabi.physics 0.1.1
Pure python 2D physics engine



rosshill-physics 0.23
Python command line rocket simulator.



clusterking-physics 0.1.2
Physics distributions to be clustered by the ClusterKinG package.



LibraryPhysicsPy 0.2.8
Physical units, time, reference frames, environment modeling.

Установка библиотек с помощью pip

14

Установка пакетов для всех пользователей ОС
(требуется права администратора):

```
pip install пакет_1 пакет_2 ... пакет_N
```

ИЛИ

```
python -m pip install пакет_1 пакет_2 ... пакет_N
```

Установка библиотек с помощью pip

15

Установка пакетов для текущего пользователя ОС
(права администратора не требуются):

```
pip install --user пакет_1 пакет_2 ... пакет_N
```

ИЛИ

```
python -m pip install --user пакет_1 пакет_2 ... пакет_N
```

Обновление библиотек с помощью pip

Обновление библиотек для всех пользователей ОС
(требуется права администратора):

```
pip install --upgrade пакет_1 пакет_2 ... пакет_N
```

Обновление библиотек для текущего пользователя ОС
(права администратора не требуются):

```
pip install --upgrade --user пакет_1 пакет_2 ... пакет_N
```


Удаление библиотек с помощью `pip`

```
pip uninstall пакет_1 пакет_2 ... пакет_N
```

Получение списка установленных библиотек с помощью pip

18

Получить список всех установленных пакетов:

```
pip list
```

Получить список установленных пакетов, для которых
появились новые версии:

```
pip list --outdated
```

Установка библиотек для инженерных и научных вычислений

19

```
pip install --user numpy matplotlib scipy pandas
```