Основы языка программирования Python

Модули

```python
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
```

```python
# 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)
```

```python
# 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
```

```python
# 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)
```

```python
# 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
```

```python
# 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)
```

```python
# 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
```

```python
# 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)
```

```python
# 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
```

```python
# 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")
```

```python
# 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
```

```python
# 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=}")
```

```python
# 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
```

```python
# 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__=}")
```

```python
# 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
```

```python
# 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=}")
```

```python
# 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
```

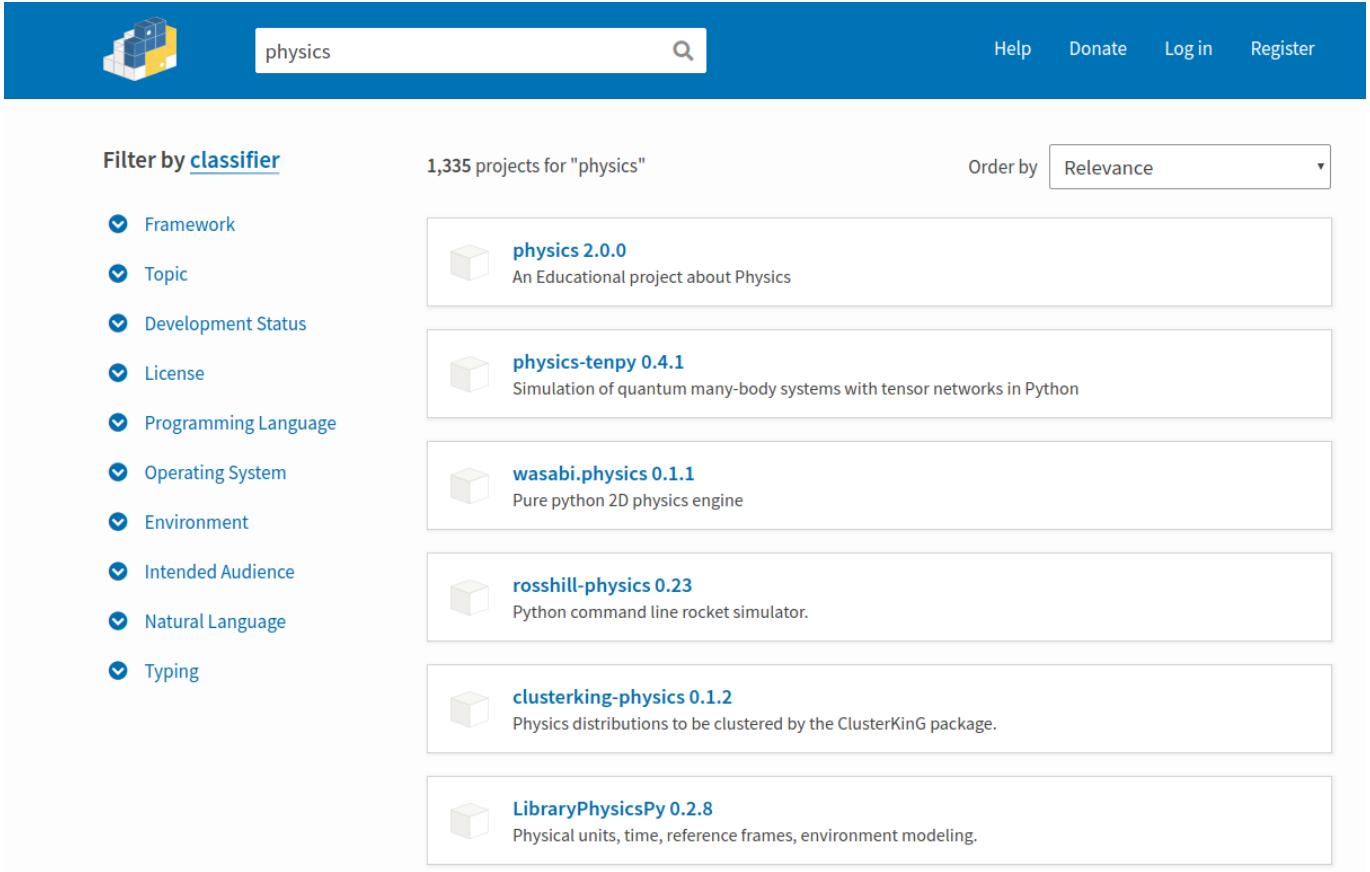
Пакеты модулей

`src/11. Modules/example_modules_10.py`

`src/11. Modules/example_modules_11.py`

Установка библиотек с помощью pip

https://pypi.org



Установка библиотек с помощью pip

Установка пакетов для всех пользователей ОС (требуются 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

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 (требуются 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

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

pip install --user numpy matplotlib scipy pandas