

Астрономическое сообщество

БФУ им.И. Канта

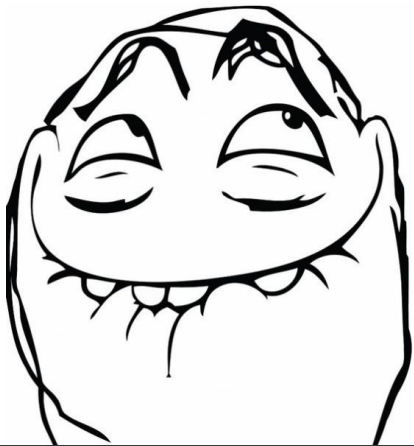
МОДУЛИ И БИБЛИОТЕКИ



Работа с МОДУЛЬНЫМИ

ОПРЕДЕЛЕНИЕ

Модуль в Рубле - самая крупная организационная программная единица, которая вмещает в себя программный код и данные, готовые для многократного ~~использования~~ использования программа/файл/скрипт, подгружаемый другой программой



СОЗДАНИЕ МОДУЛЯ

**любой файл с расширением «.py»
автоматически считается
модулем Python**

«*my_module.py*»

```
In [1]: a = 3 #Атрибут модуля
```

ИНСТРУКЦИЯ import

```
In [1]: import my_module #Инструкция, целиком загружающая модуль
```

```
In [2]: print(my_module.a)
```

3

```
In [3]: b = my_module.a**2
```

```
In [4]: print(b)
```

9

**оператор,
предоставляющий
доступ к атрибутам
объектов (модулей,
библиотек и т.п.)**

КОНСТРУКЦИЯ `import` - `as`

```
In [1]: import my_module as mm #Команда создания псевдонима для имени модуля
```

```
In [2]: print(mm.a)
```

3

```
In [3]: print(mm.a/(1 - mm.a**3))
```

-0.11538461538461539

ИНСТРУКЦИЯ FROM

```
In [1]: from my_module import a #Инструкция, копирующая атрибуты модуля
```

```
In [2]: print(a)
```

3

ИНСТРУКЦИЯ FROM *

«*my_module.py*»

```
In [1]: a = 3  
In [2]: b = 55  
In [3]: c = 1
```

} #Атрибуты модуля

#-----

```
In [1]: from my_module import a, b, c #Можно так
```

#-----

```
In [1]: from my_module import * #Инструкция, копирующая все атрибуты модуля  
In [2]: print(a*b - c)
```

164

КОНСТРУКЦИЯ FROM – import - as

«*my_module.py*»

```
In [1]: earth_mass = 5.94*10**24
```

```
In [2]: gravity_constant = 6.67*10**(-11)
```

```
In [3]: your_mam_mass = 100000**1000000
```

```
#-----
```

```
In [1]: from my_module import gravity_constant as G
```

```
In [2]: g = 500*G/10*2
```

```
In [3]: print(g)
```

```
6.6699999999999995e-09
```

КОГДА НЕОБХОДИМО ИСПОЛЬЗОВАТЬ ИНСТРУКЦИЮ `import`

«*my_module_1.py*»

```
In [1]: a = 10
```

#-----

«*my_module_2.py*»

```
In [1]: a = 5
```

#-----

```
In [1]: from my_module_1 import a
```

```
In [2]: from my_module_2 import a
```

```
In [3]: print(a)
```

5

КОГДА НЕОБХОДИМО ИСПОЛЬЗОВАТЬ ИНСТРУКЦИЮ `import`

```
In [4]: import my_module_1
```

```
In [5]: import my_module_2
```

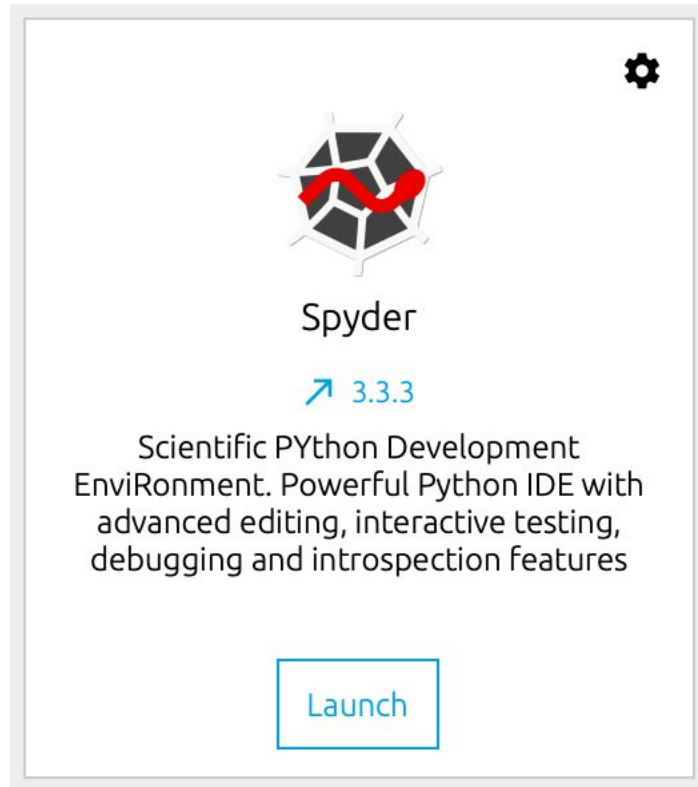
```
In [6]: print(my_module_1.a * my_module_2.a)
```

```
50
```

ЗНАКОМСТВО С НЕКОТОРЫМИ БЛИЗКОЛЕЖАЩИМИ ЗВЕЗДАМИ

ПОЧЕМУ SPYDER?

Библиотека – коллекция модулей



The image shows a window for the Spyder application. At the top right is a gear icon. In the center is the Spyder logo, which consists of a white hexagonal grid with a red wavy line passing through it. Below the logo, the text "Spyder" is displayed. Underneath that, a blue arrow points to the version number "3.3.3". Below the version number, there is a description: "Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features". At the bottom center, there is a blue button with the text "Launch".

Библиотека math

```
In [1]: from math import sin, tan, sqrt, pi
```

```
In [2]: sin(
```

```
Signature: sin(x, /)
```

```
Docstring: Return the sine of x (measured in radians).
```

```
Type:      builtin_function_or_method
```

```
In [2]: sin(2*pi)
```

```
-2.4492935982947064e-16
```

```
In [3]: sin(pi)
```

```
1.2246467991473532e-16
```

```
In [4]: sqrt(3**2 - 5*sin(pi/2))
```

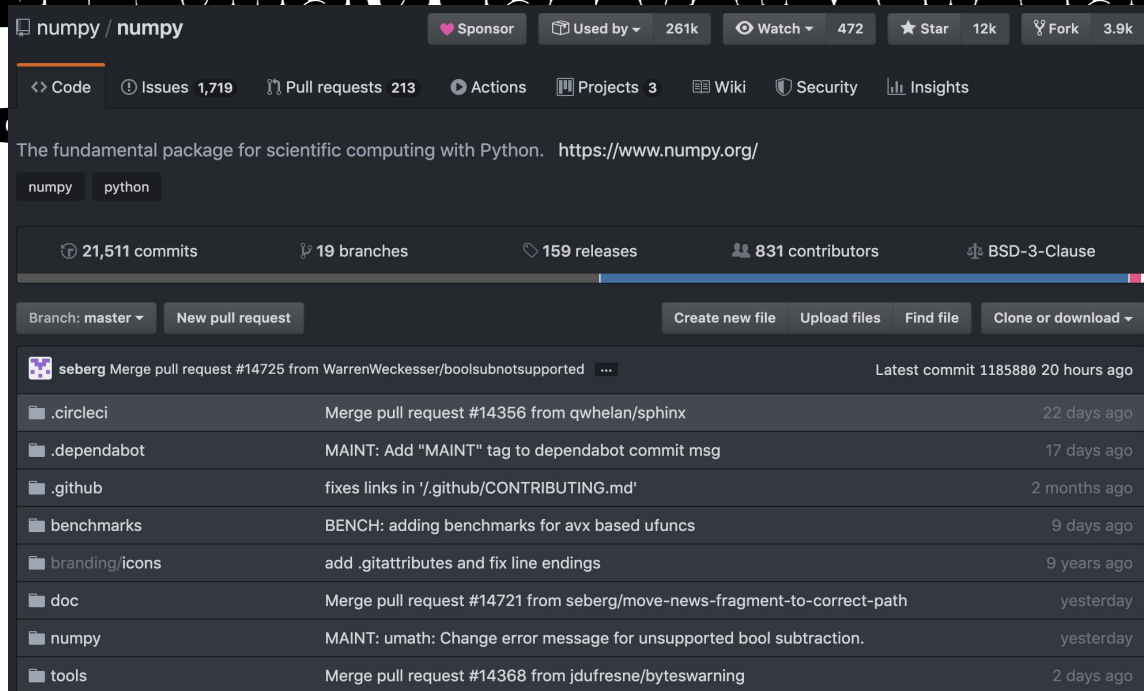
```
2.0
```

```
In [5]: sin(pi/2)
```

```
1.0
```

Библиотека NumPy

NumPy – библиотека с открытым исходным кодом для языка программирования Python, с возможностями поддержки многомерных массивов (включая матрицы) и высокопроизводительных



np.array

```
In [1]: import numpy as np
```

```
In [2]: a = [1,2,4]
```

```
In [3]: b = np.array(a) #Команда создания массивов из списков Python
```

```
In [4]: print(type(a))
```

```
<class 'list'>
```

```
In [5]: print(type(b))
```

```
<class 'numpy.ndarray'>
```

```
In [6]: print(b*b)
```

```
[ 1  4 16]
```

```
In [7]: print(b/b)
```

```
[1.  1.  1.]
```

```
In [8]: print(b - b)
```

```
[0 0 0]
```

```
In [9]: print(a/a)
```

TypeError

Traceback (most recent call last)

np.zeros, np.ones и np.ndarray

```
In [10]: A = np.zeros((2, 3))
```

```
In [11]: print(A)
```

```
[[0. 0. 0.]  
 [0. 0. 0.]
```

```
In [12]: A[0, 2] = 5
```

```
In [13]: print(A)
```

```
[[0. 0. 5.]  
 [0. 0. 0.]
```

```
In [14]: B = np.ones((3, 2))
```

```
In [15]: print(B)
```

```
[[1. 1.]  
 [1. 1.]  
 [1. 1.]
```

```
In [16]: A = np.ndarray(shape=(2, 3))
```

```
In [17]: print(A)
```

```
[[0. 0. 0.]  
 [0. 0. 0.]
```

np.arange

```
In [18]: A = np.arange(0, 10, 1)
```

```
In [19]: print(A)
```

```
[0 1 2 3 4 5 6 7 8 9]
```

```
In [20]: B = np.arange(0, 5, 0.5)
```

```
In [21]: print(B)
```

```
[0.  0.5 1.  1.5 2.  2.5 3.  3.5 4.  4.5]
```

```
In [22]: print(A[-1])
```

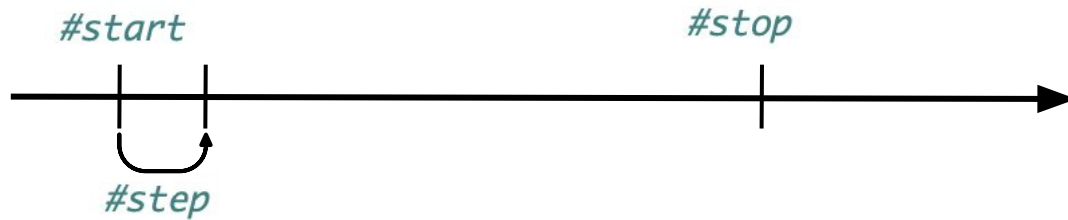
```
9
```

```
In [23]: print(len(A)) #Оператор определения длины массива/списка
```

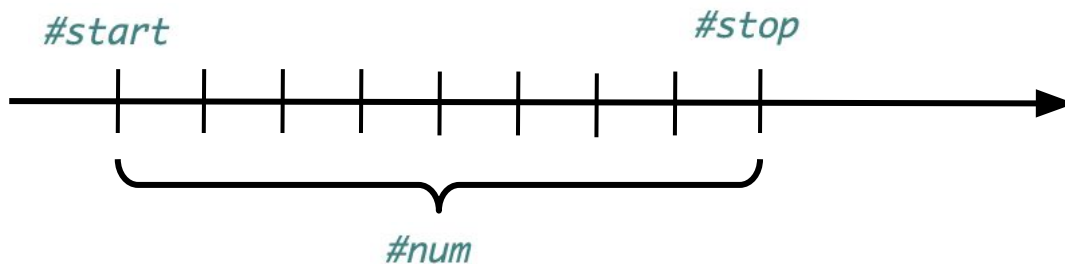
```
10
```

np.linspace

`np.arange(start, stop, step)`



`np.linspace(start, stop, num)`



np.linspace

```
In [24]: A = np.linspace(0, 10, 10)
```

```
In [25]: print(A)
```

```
[ 0.          1.11111111  2.22222222  3.33333333  4.44444444  5.55555556  
 6.66666667  7.77777778  8.88888889 10.         ]
```

СРЕЗЫ

```
In [26]: A = np.array([[1, 2, 3], [4, 5, 6]])
```

```
In [27]: print(A[1, :])
```

```
[4 5 6]
```

```
In [28]: print(A[:, 1])
```

```
[2 5]
```

```
In [29]: B = A[:, ::-1]
```

```
In [30]: print('A', A)
```

```
A [[1 2 3]
    [4 5 6]]
```

```
In [31]: print('B', B)
```

```
B [[3 2 1]
    [6 5 4]]
```

```
In [32]: print(A[:, :])
```

```
[[1 2 3]
 [4 5 6]]
```

ОПРЕДЕЛЕНИЕ ПОЗИЦИЙ