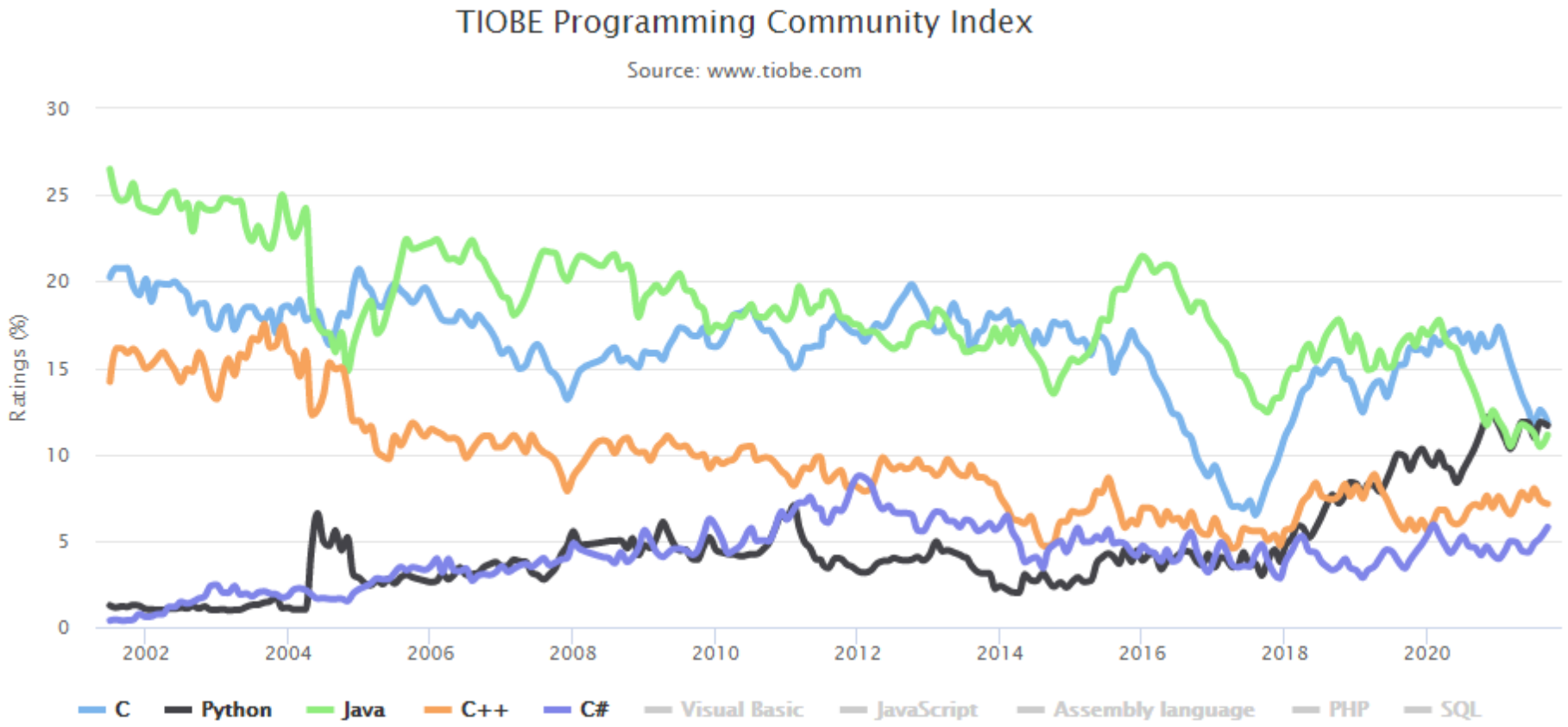
Статистика использования языков



https://www.tiobe.com/tiobe-index/

Статистика использования языков (2)

September 2017 = 2,98%
September 2018 = 7,65%
September 2019 = 9,88%
September 2020 = 10,47%
September 2021 = 11,67%

Sep 2021	Sep 2020	Change	Programming Language	Ratings	Change
1	1		C	11.83%	-4.12%
2	3	↑	Python	11.67%	+1.20%
3	2	↓	Java	11.12%	-2.37%
4	4		C++	7.13%	+0.01%
5	5		C#	5.78%	+1.20%
6	6		Visual Basic	4.62%	+0.50%
7	7		JavaScript	2.55%	+0.01%
8	14	⇈	Assembly language	2.42%	+1.12%
9	8	↓	PHP	1.85%	-0.64%
10	10		SQL	1.80%	+0.04%
11	22	⇈	Classic Visual Basic	1.52%	+0.77%
12	17	⇈	Groovy	1.46%	+0.48%
13	15	↑	Ruby	1.27%	+0.03%
14	11	↓	Go	1.13%	-0.33%
15	12	↓	Swift	1.07%	-0.31%

Языки программирования лидеров IT-рынка

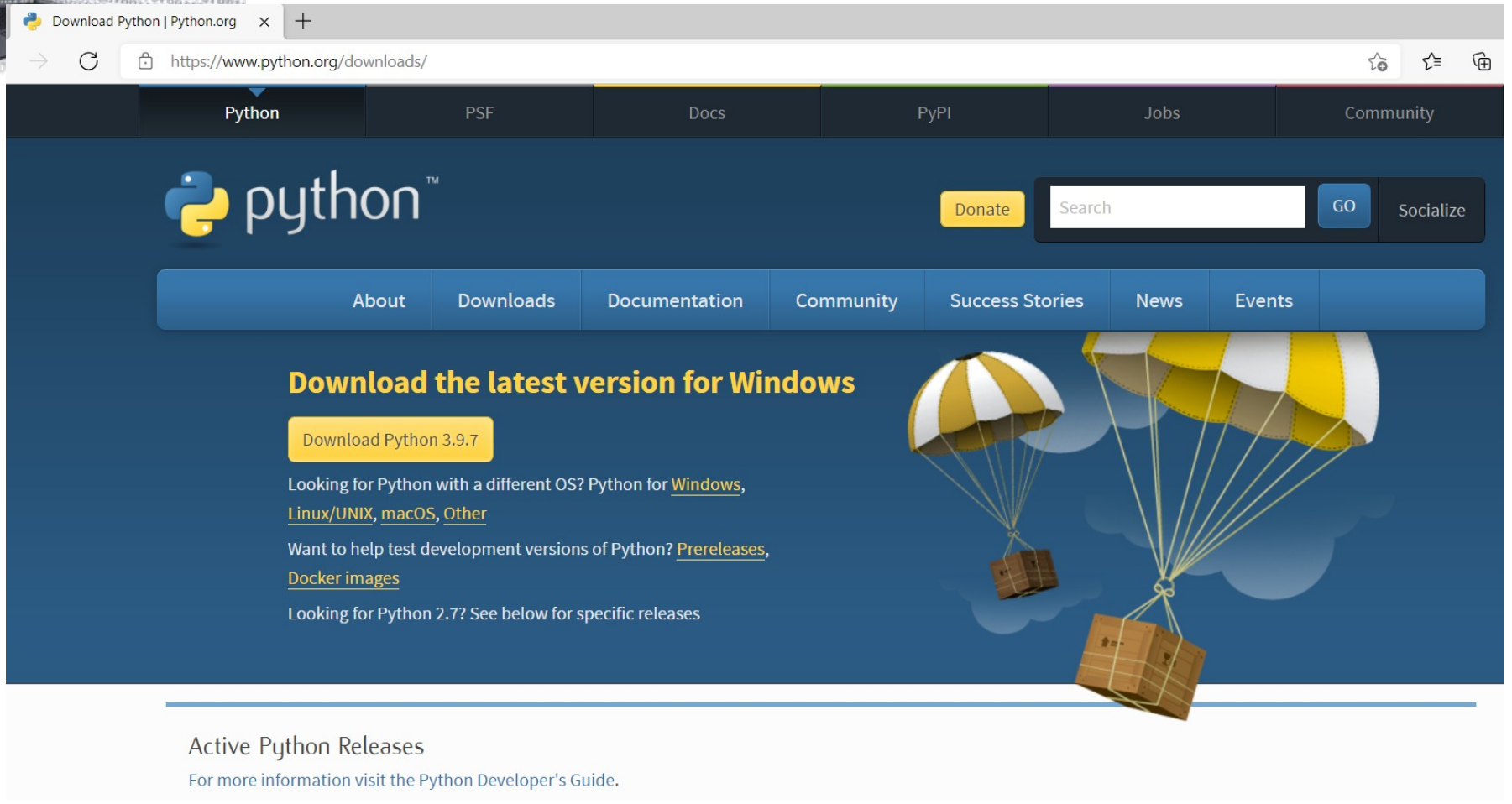
SAMSUNG	C, C++, Java, Python, JavaScript
Microsoft	C, C++, C#, HTML5/JavaScript
Google	C, C++, Java, Python, Go, HTML5/JavaScript
	Objective-C, Swift
facebook	PHP, HTML5/JavaScript, Hack
Интернет-стартапы	Python, Ruby

In [6]:

```
for i in range(20):
print (i)
```

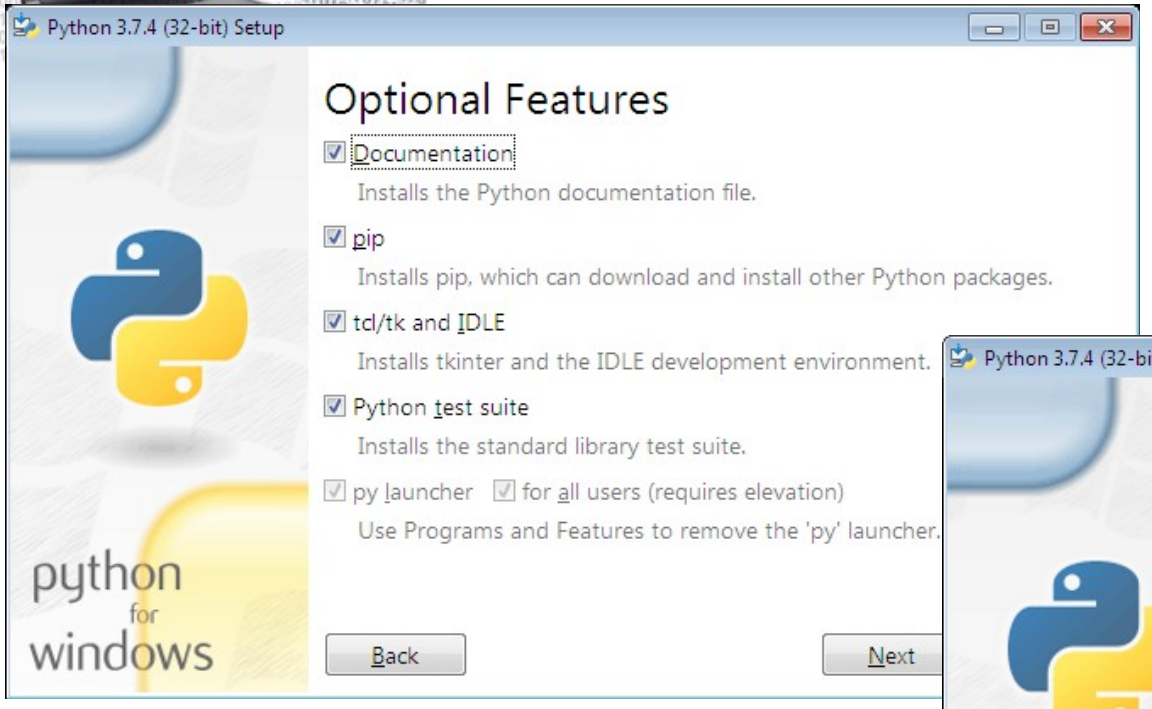
```
  File "<ipython-input-6-db022ee2e780>",
line 2
    print (i)
         ^
IndentationError: expected an indented b
lock
```

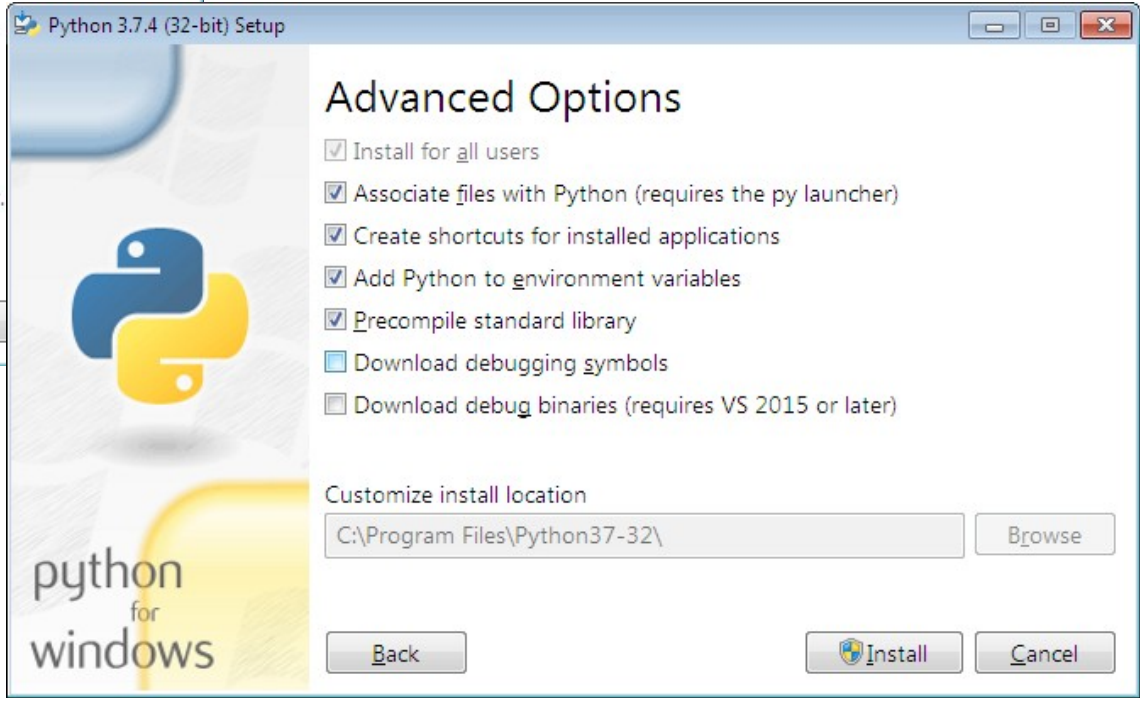

```
for i in range(20):
    print (i)
```



https://www.python.org/downloads/

Установка(2)





Важно установить pip для дальнейшего подключения пакетов/библиотек

Hello, world!

Python 3.7.0 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\Users\balap\Desktop\Hello_World.py ==========
Hello, World!
>>>
```

Hello_World.py - C:\Users\balap\Desktop\Hello_World.py (3.7.0)

File Edit Format Run Options Window Help

```
print('Hello, World!')
```

Ln: 2 Col: 0

Ln: 6 Col: 4

Environmnet Jupyter

Примеры команд

pip install --upgrade ipython jupyter
pip install jupyterlab

cd C:\Users\<USER_NAME>\AppData\Local\Programs\Python\Python37\Scripts

jupyter-notebook.exe

```
C:\>pip install numpy
Collecting numpy
  Downloading https://files.pythonhosted.org/packages/96/d6/53a59338c613e0c3ec7e3052bbf068a5457a005a5f7ad4ae005167c3597e
/numpy-1.15.2-cp37-none-win_amd64.whl (13.5MB)
    100% |                                | 13.5MB 1.4MB/s
Installing collected packages: numpy
Successfully installed numpy-1.15.2
You are using pip version 10.0.1, however version 18.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

Hello, world! (2)

Untitled1 - Jupyter Notebook

localhost:8888/notebooks/Untitled1.ipynb

jupyter Untitled1 Last Checkpoint: 42 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3

Run Code

```
In [2]: print("Hello, world!")
        Hello, world!

In [ ]:
```


```
In [18]: 255 + 34
Out[18]: 289

In [19]: 5 * 2
Out[19]: 10

In [20]: 20 / 3
Out[20]: 6.666666666666667

In [21]: 20 // 3
Out[21]: 6

In [22]: 20 % 3
Out[22]: 2

In [23]: 3 ** 4
Out[23]: 81

In [24]: pow(3, 4)
Out[24]: 81
```

Функции в Python(2)

In [25]:
```
n = -37
print (bin(n))
n.bit_length()
```

```
-0b100101
```

Out[25]: 6

In [26]:
```
print ((1024).to_bytes(2, byteorder='big'))
print (int.from_bytes(b'\x00\x10', byteorder='big'))
```

```
b'\x04\x00'
16
```

In [27]:
```
print (bin(19))
print (oct(19))
print (hex(19))
print (0b10011)
print (int('10011', 2))
```

```
0b10011
0o23
0x13
19
19
```

Функции в Python(3)

In [28]:
```
import math
print (math.pi)
print (math.sqrt(85))
```

```
3.141592653589793
9.219544457292887
```

In [29]:
```
x = complex(1, 2)
print (x)
```

```
(1+2j)
```

In [31]:
```
S1 = 'spam'
S2 = 'eggs'
print (S1 + S2)
print (len('spam'))

print (S1[0])
print (S1[1])
print (S1[-2])
```

```
spameggs
4
s
p
a
```

Функции в Python(4)

In [32]:
```
a = " Hello, World! "
print(a.strip())
print(a.lower())
print(a.upper())
print(a.replace("H", "J"))
print(a.split(","))
```

```
Hello, World!
 hello, world!
 HELLO, WORLD!
 Jello, World!
[' Hello', ' World! ']
```

In [34]:
```
age = 36
txt = "My name is John, and I am {}"
print(txt.format(age))
age = "36"
txt = "My name is John, I am " + age
print(txt)
```

```
My name is John, and I am 36
My name is John, I am 36
```

Функции в Python(5)

In [8]:
```
def sum (x, y):
    total = x + y
    return total
```

In [13]:
```
a = sum(1, 5)
print ("sum of 1 and 5 is: ", a)
b = sum(1.5, 1.023)
print ("sum of 1.5 and 1.023 is: ", b)
```

```
sum of 1 and 5 is:  6
sum of 1.5 and 1.023 is:  2.5229999999999997
```

Функции в Python(6)

In [15]:
```
a = int(input())
if a < -5:
    print('Low')
elif -5 <= a <= 5:
    print('Mid')
else:
    print('High')
```

```
15
High
```

In [16]:
```
for i in 'hello world':
    print(i * 2, end='')
```

```
hheelllloo  wwoorrlldd
```

In [17]:
```
for i in 'hello world':
    if i == 'a':
        break
else:
    print('There is no letter "a"')
```

```
There is no letter "a"
```

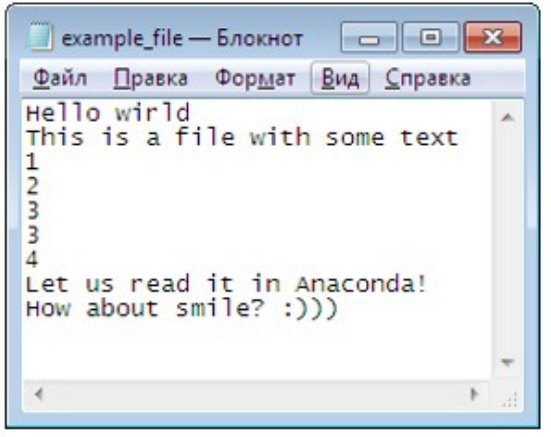
Работа с файлами в Python

In [44]:
```
address = 'D:\Jupiter\example_file.txt'
f = open(address, 'r')
print (f)
```

```
<_io.TextIOWrapper name='D:\\Jupiter\\example_file.txt' mode='r' encoding='c
p1251'>
```

In [45]:
```
print (f.read(1))

for line in f:
    print (line)
```

```
H
ello wirld

This is a file with some text

1

2

3

3

4

Let us read it in Anaconda!

How about smile? :)))
```



Работа с файлами в Python(2)

```
In [51]: l = [str(i)+str(i-1) for i in range(20)]
         print (l)

         f = open(address, 'w')

         for index in l:
              f.write(index + '\n')
         f.close()
```

```
['0-1', '10', '21', '32', '43', '54', '65', '76', '87', '98', '109', '1110',
'1211', '1312', '1413', '1514', '1615', '1716', '1817', '1918']
```

example_file — Блокнот

Файл Правка Формат Вид Справка

```
0-1
10
21
32
43
54
65
76
87
98
109
1110
1211
1312
1413
1514
1615
1716
1817
1918
```

Запуск из командной строки

D:\Jupiter\Hello_World.py - Notepad++

Файл Правка Поиск Вид Кодировки Синтаксис Опции Макросы
Запуск Плагины Окна ? X

Hello_World.py

```
1   print('Hello, World!')
```

Ln : 1 Col : 23 Sel : 0 Dos\Windows ANSI

C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

C:\Users\Aglaia>python
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> age = 36
>>> txt = "My name is John, and I am {}"
>>> print(txt.format(age))
My name is John, and I am 36
>>> age = "36"
>>> txt = "My name is John, I am " + age
>>> print(txt)
My name is John, I am 36
>>> exit()

C:\Users\Aglaia>D:

D:\>cd Jupiter\

D:\Jupiter>python Hello_World.py
Hello, World!

D:\Jupiter>
```

Полезные функции для работы со строками

capitalize()	Converts the first character to upper case	ljust()	Returns a left justified version of the string
casefold()	Converts string into lower case	lower()	Converts a string into lower case
center()	Returns a centered string	lstrip()	Returns a left trim version of the string
count()	Returns the number of times a specified value occurs in a string	maketrans()	Returns a translation table to be used in translations
encode()	Returns an encoded version of the string	partition()	Returns a tuple where the string is parted into three parts
endswith()	Returns true if the string ends with the specified value	replace()	Returns a string where a specified value is replaced with a specified value
expandtabs()	Sets the tab size of the string	rfind()	Searches the string for a specified value and returns the last position of where it was found
find()	Searches the string for a specified value and returns the position of where it was found	rindex()	Searches the string for a specified value and returns the last position of where it was found
format()	Formats specified values in a string	rjust()	Returns a right justified version of the string
format_map()	Formats specified values in a string	rpartition()	Returns a tuple where the string is parted into three parts
index()	Searches the string for a specified value and returns the position of where it was found	rsplit()	Splits the string at the specified separator, and returns a list
isalnum()	Returns True if all characters in the string are alphanumeric	rstrip()	Returns a right trim version of the string
isalpha()	Returns True if all characters in the string are in the alphabet	split()	Splits the string at the specified separator, and returns a list
isdecimal()	Returns True if all characters in the string are decimals	splitlines()	Splits the string at line breaks and returns a list

Полезные функции для работы со строками(2)

isdigit()	Returns True if all characters in the string are digits	startswith()	Returns true if the string starts with the specified value
isidentifier()	Returns True if the string is an identifier	strip()	Returns a trimmed version of the string
islower()	Returns True if all characters in the string are lower case	swapcase()	Swaps cases, lower case becomes upper case and vice versa
isnumeric()	Returns True if all characters in the string are numeric	title()	Converts the first character of each word to upper case
isprintable()	Returns True if all characters in the string are printable	translate()	Returns a translated string
isspace()	Returns True if all characters in the string are whitespaces	upper()	Converts a string into upper case
istitle()	Returns True if the string follows the rules of a title	zfill()	Fills the string with a specified number of 0 values at the beginning
isupper()	Returns True if all characters in the string are upper case	ljust()	Returns a left justified version of the string
join()	Joins the elements of an iterable to the end of the string	lower()	Converts a string into lower case
capitalize()	Converts the first character to upper case	lstrip()	Returns a left trim version of the string
casefold()	Converts string into lower case	maketrans()	Returns a translation table to be used in translations
center()	Returns a centered string	partition()	Returns a tuple where the string is parted into three parts
count()	Returns the number of times a specified value occurs in a string	replace()	Returns a string where a specified value is replaced with a specified value
encode()	Returns an encoded version of the string	rfind()	Searches the string for a specified value and returns the last position of where it was found
endswith()	Returns true if the string ends with the specified value	rindex()	Searches the string for a specified value and returns the last position of where it was found

Полезные функции для работы со строками(3)

expandtabs()	Sets the tab size of the string	rjust()	Returns a right justified version of the string
find()	Searches the string for a specified value and returns the position of where it was found	rpartition()	Returns a tuple where the string is parted into three parts
format()	Formats specified values in a string	rsplit()	Splits the string at the specified separator, and returns a list
format_map()	Formats specified values in a string	rstrip()	Returns a right trim version of the string
index()	Searches the string for a specified value and returns the position of where it was found	split()	Splits the string at the specified separator, and returns a list

Дополнительные библиотеки и пакеты

python™

NumPy

IPython

SciPy

matplotlib

SymPy

Mayavi

pandas

Cython

StatsModels
Statistics in Python

scikits
learn
machine learning in Python

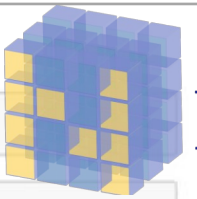
scikits-image
image processing in python

PyTables

NetworkX

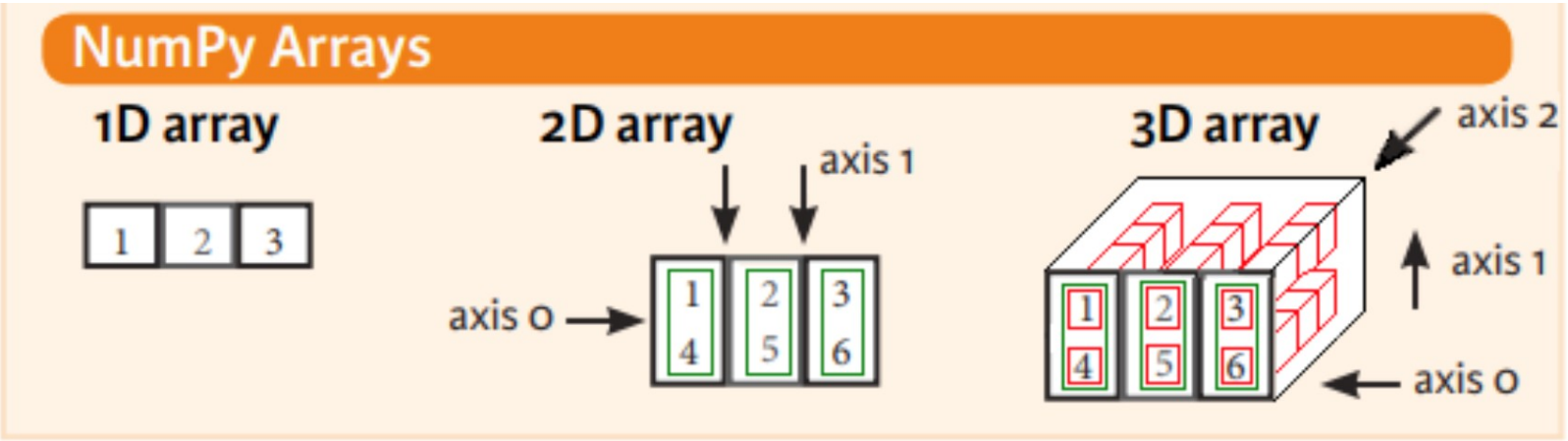
По материалам Жумагулова Я.В.

Дополнительные библиотеки и пакеты(2)

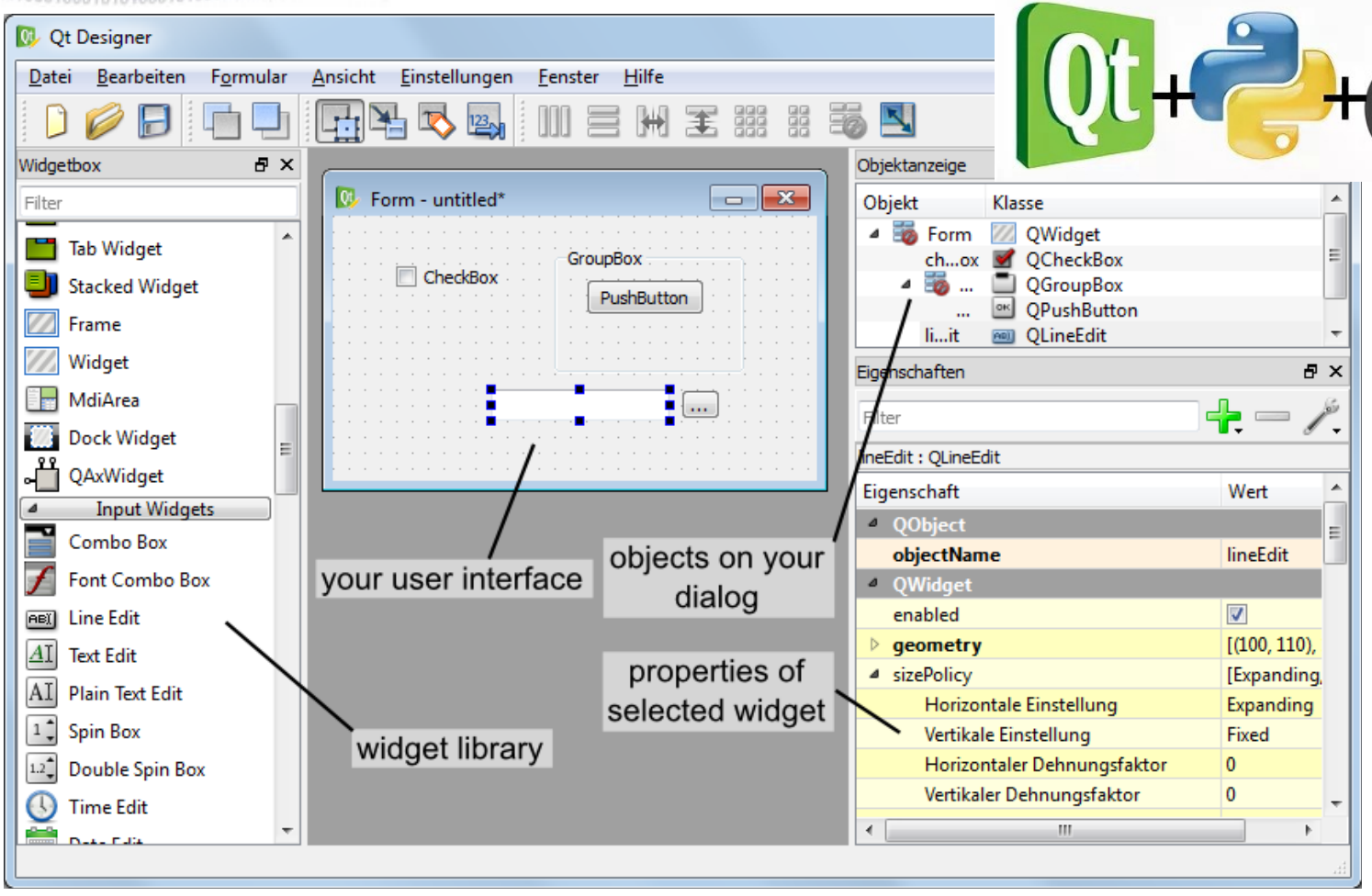


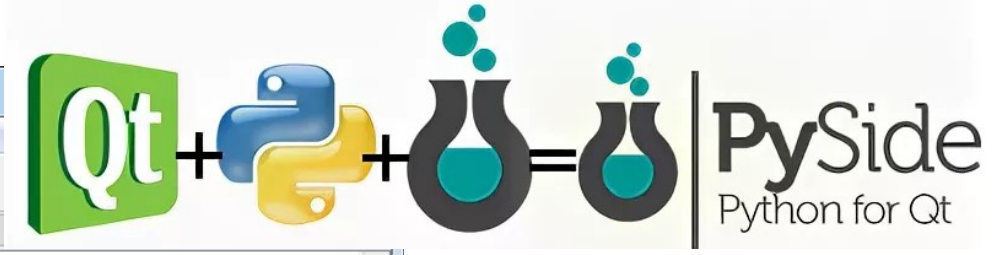
NumPy

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

In [2]: a = np.arange(12).reshape(2, 2, 3)

In [3]: a

Out[3]: array([[[ 0,  1,  2],
                [ 3,  4,  5]],

               [[ 6,  7,  8],
                [ 9, 10, 11]]])
```



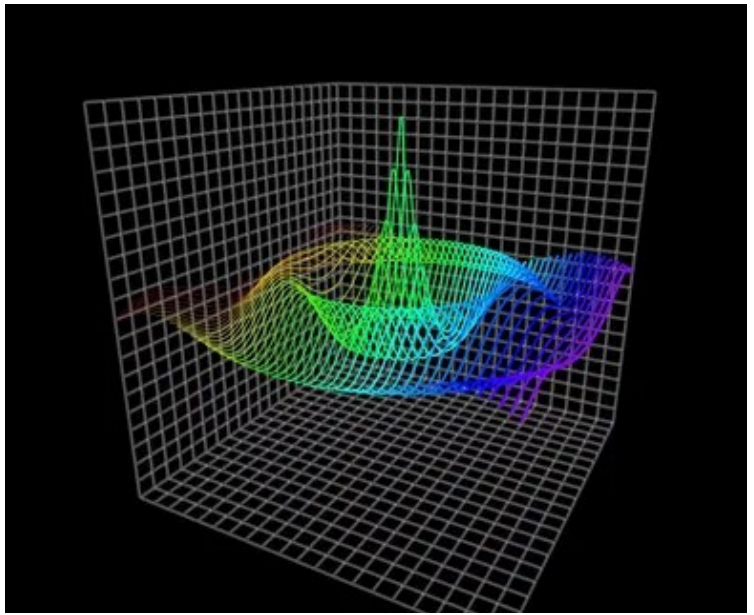
Дополнительные библиотеки и пакеты(3)

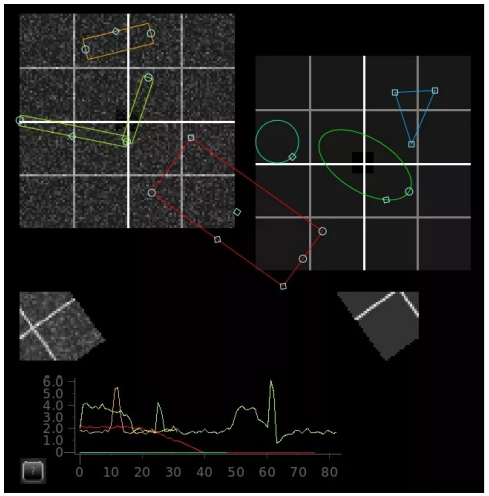


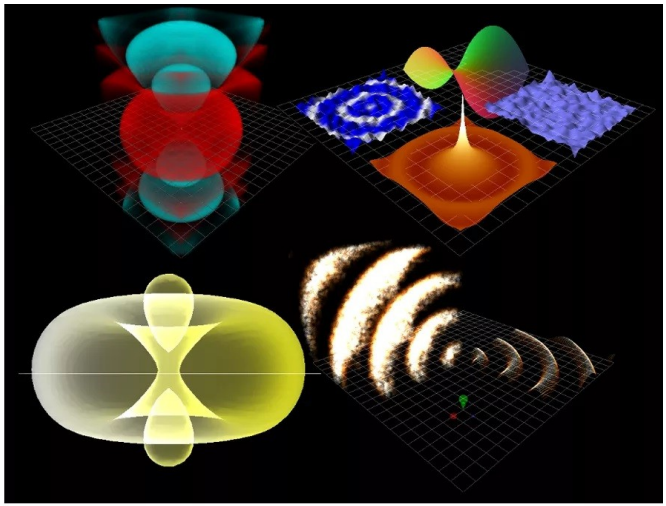


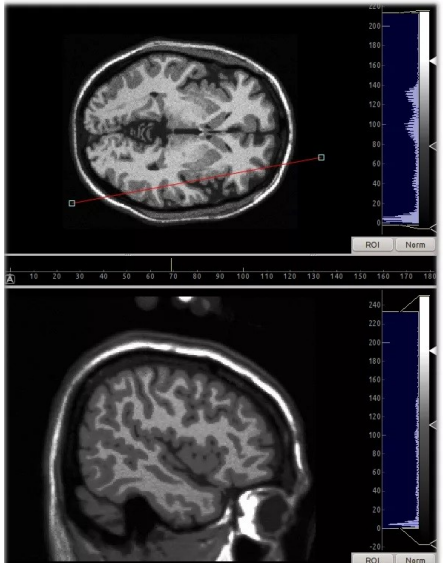
Дополнительные библиотеки и пакеты(4)

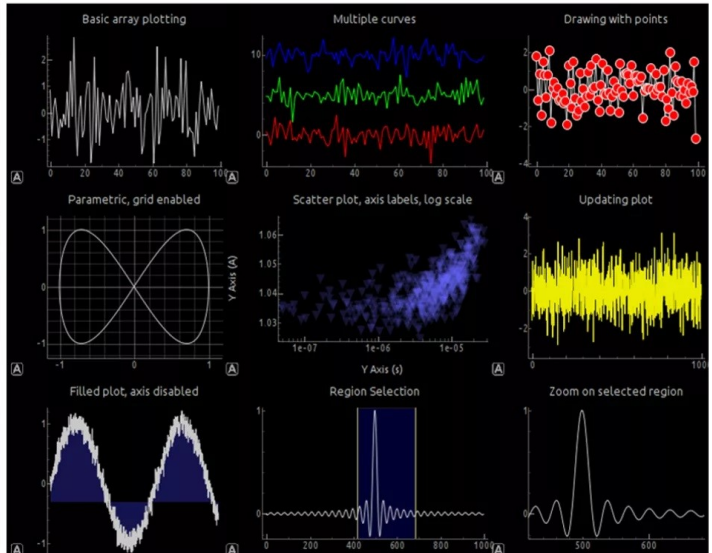
Pyqtgraph











Полезные ссылки (2)

https://ru.wikiversity.org/wiki/Программирование_и_научные_вычисления_на_языке_Python

https://realpython.com/ - Простые примеры

https://habr.com/post/352678/ - Установка и использование NumPy

https://www.lfd.uci.edu/~gohlke/pythonlibs/ - Набор готовых библиотек

https://tproger.ru/translations/jupyter-notebook-python-3/ - Командная оболочка Jupyter для интерактивных вычислений

https://www.jetbrains.com/pycharm/ - Интегрированная среда разработки

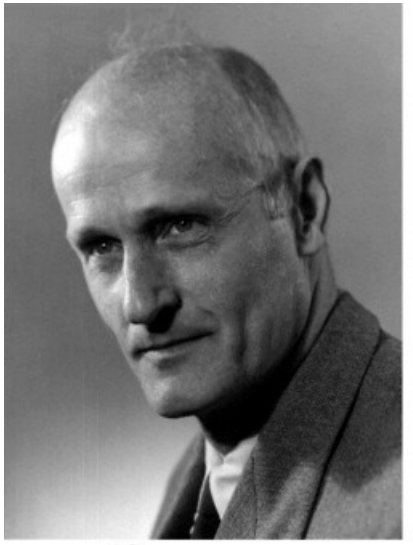
https://books.ifmo.ru/file/pdf/2256.pdf - Методическое пособие Лямина А.В.

Определение

Регулярные выражения (regular expressions) — последовательность символов, определяющая шаблон для поиска в строках.

Их поддерживают языки Python, Perl, R, C++, Java.

https://regex101.com/



Stephen Cole Kleene
(1909-1994)

Примеры регулярных выражений

REGULAR EXPRESSION

348 matches (709 steps, 0.3ms)

/ . / gm

TEST STRING

```
My IP-address \home\: 192.168.1.0
My IP-address \home\: 192.168.1.1
My IP-address \home\: 192.168.1.2
My IP-address \home\: 192.168.1.3
...
My IP-address \work\: 192.168.1.100
My IP-address \work\: 192.168.1.101
My IP-address \work\: 192.168.1.102
...
My IP-address \home\: 192.168.1.253
My IP-address \home\: 192.168.1.254
My IP-address \home\: 192.168.1.255
```

REGULAR EXPRESSION

92 matches (184 steps, 0.2ms)

/ \d / gm

TEST STRING

```
My IP-address \home\: 192.168.1.0
My IP-address \home\: 192.168.1.1
My IP-address \home\: 192.168.1.2
My IP-address \home\: 192.168.1.3
...
My IP-address \work\: 192.168.1.100
My IP-address \work\: 192.168.1.101
My IP-address \work\: 192.168.1.102
...
My IP-address \home\: 192.168.1.253
My IP-address \home\: 192.168.1.254
My IP-address \home\: 192.168.1.255
```

Примеры регулярных выражений (2)

REGULAR EXPRESSION

4 matches (138 steps, 0.2ms)

/ 192\.168\.1\.1 / gm

TEST STRING

```
My IP-address \home\: 192.168.1.0
My IP-address \home\: 192.168.1.1
My IP-address \home\: 192.168.1.2
My IP-address \home\: 192.168.1.3
...
My IP-address \work\: 192.168.1.100
My IP-address \work\: 192.168.1.101
My IP-address \work\: 192.168.1.102
...
My IP-address \home\: 192.168.1.253
My IP-address \home\: 192.168.1.254
My IP-address \home\: 192.168.1.255
```

REGULAR EXPRESSION

10 matches (120 steps, 0.3ms)

/ 192\.168\.1\.\d{1,3} / gm

TEST STRING

```
My IP-address \home\: 192.168.1.0
My IP-address \home\: 192.168.1.1
My IP-address \home\: 192.168.1.2
My IP-address \home\: 192.168.1.3
...
My IP-address \work\: 192.168.1.100
My IP-address \work\: 192.168.1.101
My IP-address \work\: 192.168.1.102
...
My IP-address \home\: 192.168.1.253
My IP-address \home\: 192.168.1.254
My IP-address \home\: 192.168.1.255
```

Примеры регулярных выражений (3)

REGULAR EXPRESSION — 29 matches (511 steps, 6.2ms)

/ \w\s / gm

TEST STRING

```
My IP-address \home\: 192.168.1.0
My IP-address \home\: 192.168.1.1
My IP-address \home\: 192.168.1.2
My IP-address \home\: 192.168.1.3
...
My IP-address \work\: 192.168.1.100
My IP-address \work\: 192.168.1.101
My IP-address \work\: 192.168.1.102
...
My IP-address \home\: 192.168.1.253
My IP-address \home\: 192.168.1.254
My IP-address \home\: 192.168.1.255
```

Буквенный/цифровой + пробельный символ

REGULAR EXPRESSION — 40 matches (80 steps, 0.5ms)

/ [ds] / gm

TEST STRING

```
My IP-address \home\: 192.168.1.0
My IP-address \home\: 192.168.1.1
My IP-address \home\: 192.168.1.2
My IP-address \home\: 192.168.1.3
...
My IP-address \work\: 192.168.1.100
My IP-address \work\: 192.168.1.101
My IP-address \work\: 192.168.1.102
...
My IP-address \home\: 192.168.1.253
My IP-address \home\: 192.168.1.254
My IP-address \home\: 192.168.1.255
```

Примеры регулярных выражений (4)

REGULAR EXPRESSION

10 matches (95 steps, 0.1ms)

/ address|addrass / gm

TEST STRING

```
My IP-address \home\: 192.168.1.0
My IP-address \home\: 192.168.1.1
My IP-address \home\: 192.168.1.2
My IP-address \home\: 192.168.1.3
...
My IP-addrass \work\: 192.168.1.100
My IP-addrass \work\: 192.168.1.101
My IP-addrass \work\: 192.168.1.102
...
My IP-address \home\: 192.168.1.253
My IP-address \home\: 192.168.1.254
My IP-address \home\: 192.168.1.255
```

REGULAR EXPRESSION

10 matches (107 steps, 0.1ms)

/ addr(a|e)ss / gm

TEST STRING

```
My IP-address \home\: 192.168.1.0
My IP-address \home\: 192.168.1.1
My IP-address \home\: 192.168.1.2
My IP-address \home\: 192.168.1.3
...
My IP-addrass \work\: 192.168.1.100
My IP-addrass \work\: 192.168.1.101
My IP-addrass \work\: 192.168.1.102
...
My IP-address \home\: 192.168.1.253
My IP-address \home\: 192.168.1.254
My IP-address \home\: 192.168.1.255
```

Примеры регулярных выражений (5)

REGULAR EXPRESSION

13 matches (26 steps, 0.1ms)

/ M / gm

TEST STRING

```
My IP-address \home\: 192.168.1.0
My IP-address \home\: 192.168.1.1
My IP-address \home\: 192.168.1.2
My IP-address \home\: 192.168.1.3
...
My IP-address \work\: 192.168.1.100
My IP-address \work\: 192.168.1.101
My IP-address \work\: 192.168.1.102
...
My IP-address \HOME\: 192.168.1.253
My IP-address \HOME\: 192.168.1.254
My IP-address \HOME\: 192.168.1.255
```

REGULAR EXPRESSION

10 matches (44 steps, 0.0ms)

/ ^M / gm

TEST STRING

```
My IP-address \home\: 192.168.1.0
My IP-address \home\: 192.168.1.1
My IP-address \home\: 192.168.1.2
My IP-address \home\: 192.168.1.3
...
My IP-address \work\: 192.168.1.100
My IP-address \work\: 192.168.1.101
My IP-address \work\: 192.168.1.102
...
My IP-address \HOME\: 192.168.1.253
My IP-address \HOME\: 192.168.1.254
My IP-address \HOME\: 192.168.1.255
```

Примеры регулярных выражений (6)

REGULAR EXPRESSION

14 matches (28 steps, 0.0ms)

/ e / gm

TEST STRING

```
My IP-address \home\: 192.168.1.0
My IP-address \home\: 192.168.1.1
My IP-address \home\: 192.168.1.2
My IP-address \home\: 192.168.1.3
...
My IP-address \work\: 192.168.1.100
My IP-address \work\: 192.168.1.101
My IP-address \work\: 192.168.1.102
...
My IP-address \HOME\: 192.168.1.253
My IP-address \HOME\: 192.168.1.254
My IP-address \HOME\: 192.168.1.255
```

REGULAR EXPRESSION

4 matches (32 steps, 0.0ms)

/ e\b / gm

TEST STRING

```
My IP-address \home\: 192.168.1.0
My IP-address \home\: 192.168.1.1
My IP-address \home\: 192.168.1.2
My IP-address \home\: 192.168.1.3
...
My IP-address \work\: 192.168.1.100
My IP-address \work\: 192.168.1.101
My IP-address \work\: 192.168.1.102
...
My IP-address \HOME\: 192.168.1.253
My IP-address \HOME\: 192.168.1.254
My IP-address \HOME\: 192.168.1.255
```

Примеры регулярных выражений (7)

REGULAR EXPRESSION

pattern error

/ \ / gm

TEST STRING

```
My IP-address \home\: 192.168.1.0
My IP-address \home\: 192.168.1.1
My IP-address \home\: 192.168.1.2
My IP-address \home\: 192.168.1.3
...
My IP-address \work\: 192.168.1.100
My IP-address \work\: 192.168.1.101
My IP-address \work\: 192.168.1.102
...
My IP-address \home\: 192.168.1.253
My IP-address \home\: 192.168.1.254
My IP-address \home\: 192.168.1.255
```

REGULAR EXPRESSION

20 matches (40 steps, 0.2ms)

/ \\ / gm

TEST STRING

```
My IP-address \home\: 192.168.1.0
My IP-address \home\: 192.168.1.1
My IP-address \home\: 192.168.1.2
My IP-address \home\: 192.168.1.3
...
My IP-address \work\: 192.168.1.100
My IP-address \work\: 192.168.1.101
My IP-address \work\: 192.168.1.102
...
My IP-address \home\: 192.168.1.253
My IP-address \home\: 192.168.1.254
My IP-address \home\: 192.168.1.255
```

Примеры регулярных выражений (8)

REGULAR EXPRESSION

10 matches (90 steps, 2.4ms)

/ [ию!] / gm

TEST STRING

Прошу отчислить меня по собственному желанию!
Прошу отчислить меня по собственному желанию!

REGULAR EXPRESSION

2 matches (92 steps, 2.5ms)

/ [ию!]$ / gm

TEST STRING

Прошу отчислить меня по собственному желанию!
Прошу отчислить меня по собственному желанию!

Примеры регулярных выражений (9)

REGULAR EXPRESSION

15 matches (30 steps, 0.1ms)

/ н / gm

TEST STRING

Прошу отчислить меня по собственному желанию!
Прошу отчислить меня по собственнному желанию!
Прошу отчислить меня по собственнннному желанию!

REGULAR EXPRESSION

141 matches (532 steps, 3.2ms)

/ н? / gm

TEST STRING

Прошу отчислить меня по собственному желанию!
Прошу отчислить меня по собственнному желанию!
Прошу отчислить меня по собственнннному желанию!

0 или 1 символ

Примеры регулярных выражений (10)

REGULAR EXPRESSION

135 matches (520 steps, 0.3ms)

/ н* / gm

TEST STRING

Прошу отчислить меня по собственному желанию!
Прошу отчислить меня по собственнному желанию!
Прошу отчислить меня по собственнннному желанию!

0 или более символов

REGULAR EXPRESSION

9 matches (18 steps, 0.1ms)

/ н+ / gm

TEST STRING

Прошу отчислить меня по собственному желанию!
Прошу отчислить меня по собственнному желанию!
Прошу отчислить меня по собственнннному желанию!

1 или более символов

Пример для запоминания

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

```
import re
```

Основные причины использования:

- поиск в строке;
- разбиение строки на подстроки;
- замена части строки.