
# Простые программы PYTHON

Урок 2

Актау, 2018


Синтаксис языка прост, понятен и нагляден. В некотором смысле его можно даже назвать по-спартански лаконичным.


# «Привет, Мир!»

```
>>> print("Hello, World!")
Hello, World!
```

# Математические операции

```
>>> print(9*9)
81
>>> 9*9
81
```

Посчитайте:

(8-4)*15 77/11-2+24

18*4-29 2+2*2

# Переменные

name = input("Как Ваше имя? \n")

Получится

```
>>> name = input("Как Ваше имя? \n")

Как Ваше имя?
Nata
```

Переменной присвоено значение.

Обратимся к нему

```
>>> name

'Nata'
```

Примечание: оператор \n переводит на новую строку

# Переменные

```
>>> a=10

>>> b=20

>>> c=a+b

>>> c

30
```

- Переменной a  присвоили значение 10
- Переменной b присвоили – 20
- Переменной c присвоили значение, равное сумме двух первых переменных

# Переменные

```
>>> A=11; B=17; print(A,B)

11 17
```

Оператор print без кавычек выведет значения переменных

python

# Комментарии

```
>>> #После решетки пишутся комментарии
```

# Строки

```
>>> x = 'Любая строка'

>>> x[5],x[7],x[11]

(' ', 'т', 'а')
>>> x[0]

'Л'
```

Выведем 5, 7 и 11 элементы строки

Затем выведем 0й элемент

# Строки

```
>>> x[2:7]

'бая с'
>>> x[:5]

'Любая'
>>> x[:-10]

'Лю'
>>> x[:-5], x[:-3]

('Любая с', 'Любая стр')
```

- Выведем элементы строки со второго по 7й
- Выведем элементы с нулевого по пятый
- Выведем все элементы строки без последних десяти
- Выведем элементы без последних пяти, затем без последних трех

# Строки

```
>>> len(x)  #работает со строкой

12
```

Посчитаем число элементов строки


# Строки

```
>>> 'склеиваем бук' + 'вы и числа 1532' #склеивание

'склеиваем буквы и числа 1532'
```

Склеим две строки

# Строки

```
>>> '123'*3   #перемножим строку - она дублируется

'123123123'
```

Строка, умноженная на Число, повторится Число раз

```
>>>  'школа'*4

'школашколашколашкола'
```


# Модуль

```
>>> import keyword

>>> keyword.kwlist

['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break',
ss', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for',
', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or',
s', 'raise', 'return', 'try', 'while', 'with', 'yield']
>>> #зарезервированные системой слова
```

Проверим, не назвали ли мы переменные зарезервированными системой именами


```
>>> keyword.iskeyword('except')

True
```

```
>>> keyword.iskeyword('ex')

False
```

# Дзен Питона


```
>>> import this

The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

Если проект большой, требует отдельного окна

File - New File