

Python

```
# sets upper servo to its designated (called) position
def moveToUpperPosition (self, upper):
    # finds out if old_upper is less than upper
    if self.old_upper < upper:
        # steps from small to large
        step = 5
    else:
        # steps backwards from large to small
        step = -5
    for i in range(self.old_upper, upper, step):
        print("in upper for loop at", i)
        self.upper.moveTo(i)
        time.sleep(0.1)
    print("\n")
    self.upper.moveTo(upper)
    # remember this position
    self.old_upper = upper
```

Среда обучения

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 26 2018, 23:26:24)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> 2**64
18446744073709551615
>>> 2**1000
10715086071862673209484250490600018105614048117055336074437503883703510511249361
22493198378815695858127594672917553146825187145285692314043598457757469857480393
4567748242309854210746050623711418779541821530464749835819412673987675591655439
46077062914571196477686542167660429831652624386837205668069376
>>> sum=0
>>> num=range(64)
>>> for i in num:
    sum=sum+2**i
>>> print(sum)
18446744073709551615
>>>
```

File -> New File



```
sum = 0
num = range(64)
for i in num:
    sum = sum + 2**i
print(sum)
```

Save As ...

Run Module (F5)



Области использования



PYTON_федоров.pdf

Синтаксис

Оператор	Описание
+	Сложение
-	Вычитание
/	Деление
//	Деление вещественное (в результате округлением вниз)
**	Деление в степени
%	Возведение в степень
	Остаток от деления

```
>>> myvar = 3
```

```
>>> myvar += 2
```

```
>>> myvar -= 1
```

```
"""«Это многострочный комментарий
```

```
Строки заключенные в три двойные кавычки игнорируются»"""
```

```
>>> mystring = «Hello»
```

```
>>> mystring += " world."
```

```
>>> print mystring
```

```
Hello world.
```

```
# Следующая строка меняет
```

```
значения переменных местами. (Всего одна строка!)
```

```
>>> myvar, mystring = mystring, myvar
```

Структуры данных

Списки (**lists**), кортежи (**tuples**), словари (**dictionaries**) и множества (**set**).

```
>>> sample = [1, [«another», «list»], («a», «tuple»)] #Список состоит из целого числа,  
другого списка и кортежа  
>>> mylist = [«List item 1», 2, 3.14] #Этот список содержит строку, целое и дробное число  
>>> mylist[0] = «List item 1 again» #Изменяем первый (нулевой) элемент листа mylist  
>>> mylist[-1] = 3.14 #Изменяем последний элемент листа  
>>> mydict = {«Key 1»: «Value 1», 2: 3, «pi»: 3.14} #Создаем словарь, с числовыми и  
целочисленным индексами  
>>> mydict[«pi»] = 3.15 #Изменяем элемент словаря под индексом «pi».  
>>> mytuple = (1, 2, 3) #Задаем кортеж  
>>> myfunction = len #Python позволяет таким образом объявлять синонимы функции  
>>> print myfunction(list)  
3
```

Списки — похожи на массивы (но могут включать данные разного типа), кортежи — неизменяемые списки, словари — списки с индексами любого типа, а не только числовые.

Структуры данных

```
>>> mylist = [«List item 1», 2, 3.14]
>>> print mylist[:] #Считываются все элементы массива
['List item 1', 2, 3.1400000000000001]
>>> print mylist[0:2] #Считываются нулевой и первый элемент массива.
['List item 1', 2]
>>> print mylist[-3:-1] #Считываются элементы от нулевого (-3) до второго (-1) (не
включительно)
['List item 1', 2]
>>> print mylist[1:] #Считываются элементы от первого, до последнего
[2, 3.14]
```

Можно использовать часть массива, задавая первый и последний индекс через двоеточие «:». В таком случае вы получите часть массива, от первого индекса до второго не включительно. Если не указан первый элемент, то отсчет начинается с начала массива, а если не указан последний — то массив считывается до последнего элемента. Отрицательные значения определяют положение элемента с конца.

Строки

Обособляются кавычками двойными «"» или одинарными «'». Внутри двойных кавычек могут присутствовать одинарные или наоборот. К примеру строка «Он сказал 'привет'!» будет выведена на экран как «Он сказал 'привет'!». Если нужно использовать строку из несколько строчек, то эту строку надо начинать и заканчивать тремя двойными кавычками «"""».

```
>>> print «Name: %s\nNumber: %s\nString: %s» % (myclass.name, 3, 3 * "-")
```

```
Name: Poromenos
```

```
Number: 3
```

```
String: —
```

```
strString = """«Этот текст расположен  
на нескольких строках»"""
```

```
>>> print «This %(verb)s a %(noun)s.» % {«noun»: «test», «verb»: «is»}
```

```
This is a test.
```

Можно подставить в шаблон строки элементы из кортежа или словаря. Знак процента «%» между строкой и кортежем, заменяет в строке символы «%s» на элемент кортежа. Словари позволяют вставлять в строку элемент под заданным индексом.

Операторы

```
rangelist = range(10)
>>> print(rangelist)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

for number in rangelist:
    # Check if number is one of
    # the numbers in the tuple.
    if number in (3, 4, 7, 9):
        # "Break" terminates a for without
        # executing the "else" clause.
        break
    else:
        # "Continue" starts the next iteration
        # of the loop. It's rather useless here,
        # as it's the last statement of the loop.
        continue
else:
    # The "else" clause is optional and is
    # executed only if the loop didn't "break".
    pass # Do nothing

if rangelist[1] == 2:
    print("The second item (lists are 0-based) is 2")
elif rangelist[1] == 3:
    print("The second item (lists are 0-based) is 3")
else:
    print("Dunno")

while rangelist[1] == 1:
    pass
```