https://dfedorov.spb.ru

Создание списка с помощью range

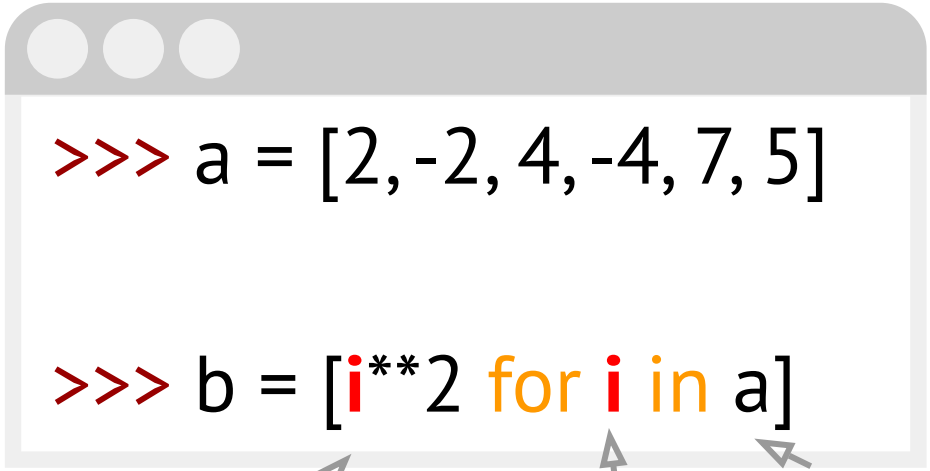
```
>>> a = []
>>> for i in range(1, 15):
        a.append(i)

>>> a = list(range(1, 15))

>>> a = [i for i in range(1, 15)]
```

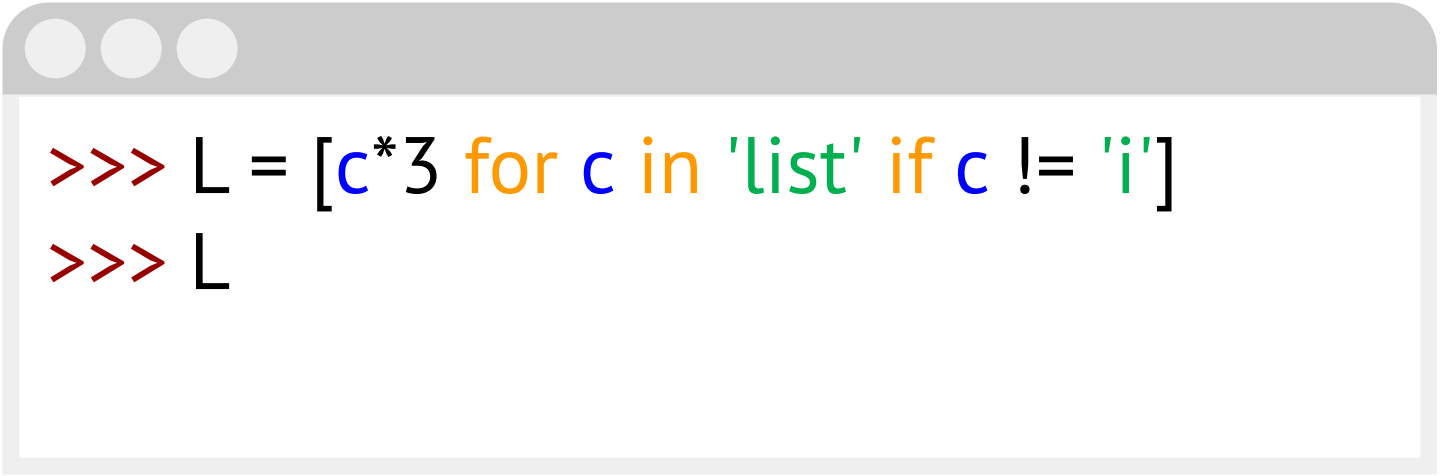
Списковое включение
(list comprehension)

что делаем с элементом

что берем

откуда берем


>>> a = [2, -2, 4, -4, 7, 5]
>>> b = [i**2 for i in a]
что делаем с элементом
что берем
откуда берем

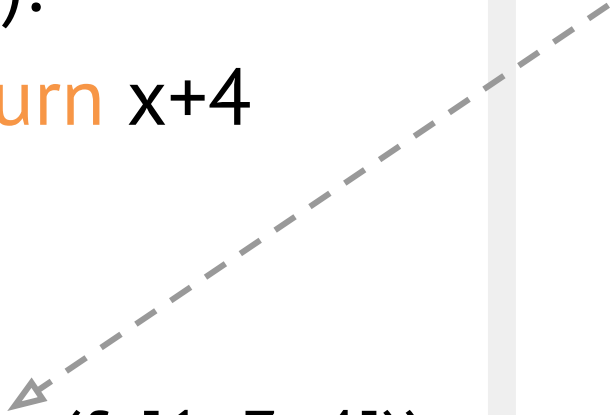

>>> L = [c*3 for c in 'list' if c != 'i']
>>> L

Рассмотрим пример

Создать новый список, элементами которого будут элементы списка [1, 3, 4], увеличенные на 4.

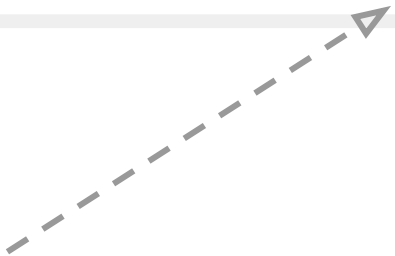

>>> b = [i + 4 for i in [1, 3, 4]]
>>> b

```
>>> def f(x):
        return x+4


>>> list(map(f, [1, 3, 4]))
```



функция высшего порядка
(на вход подается функция) - возвращает последовательность

```
>>> counter = [1, 3, 4]

>>> list(map((lambda x: x + 4), counter))
```



lambda-функция используется, если **нет потребности в создании отдельной функции**

```
lambda arguments: expression

def _(arguments):
    return expression
```

```
print((lambda x: x + 3)(3))
```

```
>>> print([x.lower() for x in ['I', 'AM', 'NOT', 'SHOUTING']])
```

```
>>> [x for x in range(5) if x%2 == 0]
[0, 2, 4]


>>> res = []
>>> for x in range(5):
        if x%2 == 0:
            res.append(x)
>>> res
[0, 2, 4]
```

Функция делает буквально то, о чем говорит ее название: она фильтрует элементы в последовательности.

Каждый элемент передается функции, которая включает его в последовательность, если по условию получает `True`, и отбрасывает в случае `False`:

```
def add_three(x):
    if x % 2 == 0:
        return True
    else:
        return False

li = [1,2,3,4,5,6,7,8]

[i for i in filter(add_three, li)]
#=> [2, 4, 6, 8]
```

Обратите внимание, как удалены все элементы, которые не делятся на 2.

```
>>> any([])
False
>>> any([0, None])
False
>>> any([0, None, 1])
True
>>> all([])
True
>>> all(["str", None])
False
>>>
```

any возвращает True, если в последовательности существует хоть один элемент, который в логическом контексте возвращает значение True

all возвращает True, если все элементы последовательности в логическом контексте возвращают значение True или последовательность не содержит элементов

`reduce` принимает функцию и последовательность — и проходит по этой последовательности. На каждой итерации в функцию передаются как текущий элемент, так и выходные данные предыдущего элемента. В конце концов, возвращается одно значение:

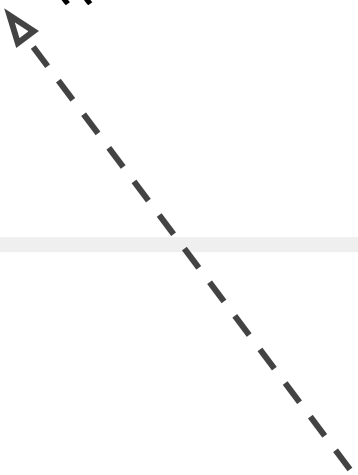
```
from functools import reduce
def add_three(x,y):
    return x + y
li = [1,2,3,5]
reduce(add_three, li)
#=> 11
```

Возвращается 11, что является суммой 1+2+3+5.

```
In [16]: # find the first 10 square numbers
         square = lambda x: x ** 2
         for val in map(square, range(10)):
             print(val, end=' ')
```

```
0 1 4 9 16 25 36 49 64 81
```

```
>>> list(filter((lambda x: x > 0), range(-5, 5)))

[1, 2, 3, 4]
```



Элементы последовательности, для которых применение функции возвращает истину, добавляются в список результатов

```
In [17]: # find values up to 10 for which x % 2 is zero
         is_even = lambda x: x % 2 == 0
         for val in filter(is_even, range(10)):
             print(val, end=' ')

0 2 4 6 8
```

для любителей функционального программирования

```
>>> from random import randint
>>> A = [randint(1, 9) for _ in range(5)]
```

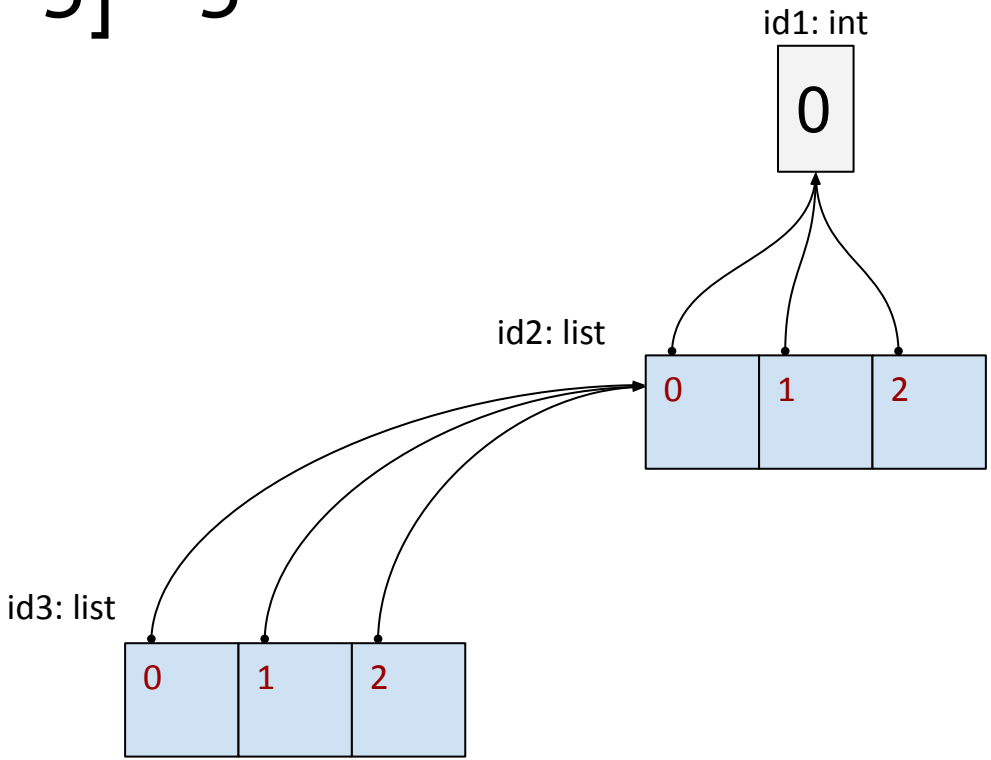
создание списка, состоящего из случайных значений

Особенность работы со списками и ссылки на объекты

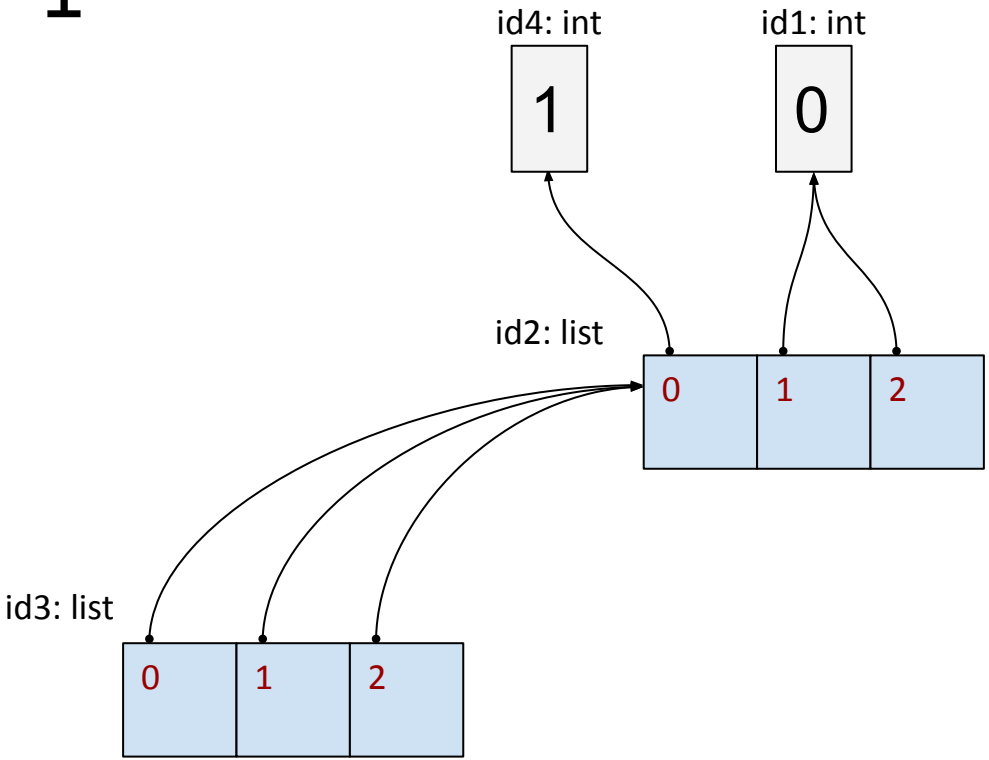
```
>>> xs = [[0] * 3] * 3
>>> xs
>>> xs[0][0] = 1
>>> xs
?????
```

https://nedbatchelder.com/blog/201308/names_and_values_making_a_game_board.html

```
>>> xs = [[0] * 3] * 3
```



```
>>> xs[0][0] = 1
```



Правильный код:

```
>>> xs = [[0] * 3 for _ in range(3)]
>>> xs[0][0] = 1
>>> xs
```

https://dfedorov.spb.ru