https://dfedorov.spb.ru

Словари также нередко называют **ассоциативными массивами** (associative arrays), **ассоциативными хеш-таблицами** (hashmaps), **поисковыми таблицами** (lookup tables) или **таблицами преобразования**. Они допускают эффективный поиск, вставку и удаление любого объекта, связанного с заданным ключом.

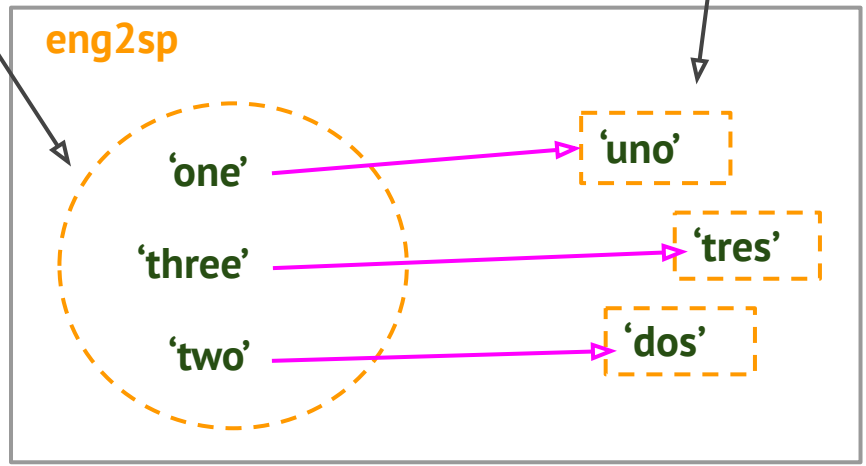
Словари (dict)

Неупорядоченная изменяемая коллекция.

“Список” с произвольными ключами, неизменяемого типа.

Множество ключей неизменяемого типа

Набор значений (из произвольных элементов)




```
>>> eng2sp = dict()
>>> eng2sp['one'] = 'uno'
>>> eng2sp['one']
>>> eng2sp['two'] = 'dos'
>>> eng2sp['three'] = 'tres'
>>> eng2sp
{'one': 'uno', 'two': 'dos', 'three': 'tres'}
```

Подробнее

Словари Python индексируются ключами, у которых может быть любой хешируемый тип: хешируемый объект имеет хеш-значение, которое никогда не меняется в течение его жизни и его можно сравнивать с другими объектами. Кроме того, эквивалентные друг другу хешируемые объекты должны иметь одинаковое хеш-значение.
Неизменяемые типы, такие как строковые значение и числа, являются хешируемыми объектами и хорошо работают в качестве ключей словаря. В качестве ключей словаря также можно использовать объекты-кортежи — при условии, что они сами содержат только хешируемые типы.

```
hash('string')
hash((1, 2, (2, 3)))
# fails because lists are mutable
hash((1, 2, [2, 3]))
```

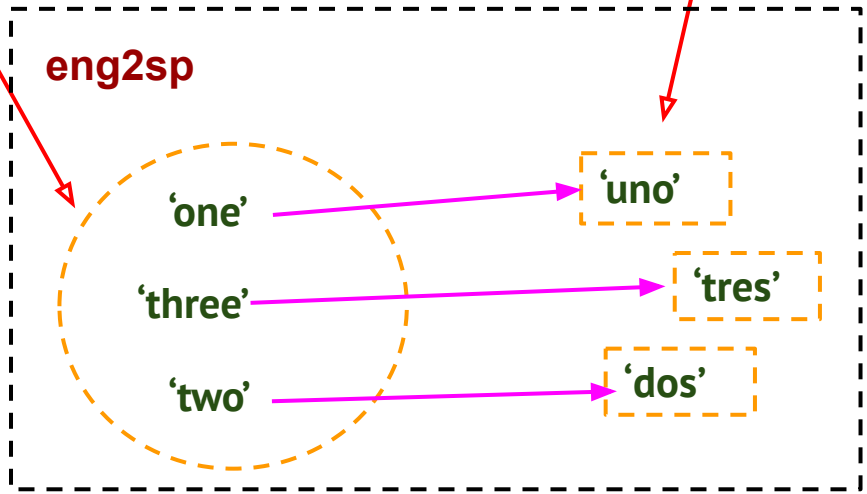
```
>>> eng2sp['four']
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    eng2sp['four']
KeyError: 'four'
```

Обращение к несуществующему элементу словаря приводит к ошибке

Множество ключей неизменяемого типа

Набор значений (из произвольных элементов)

```
>>> eng2sp
>>> 'one' in eng2sp
True
>>>
```

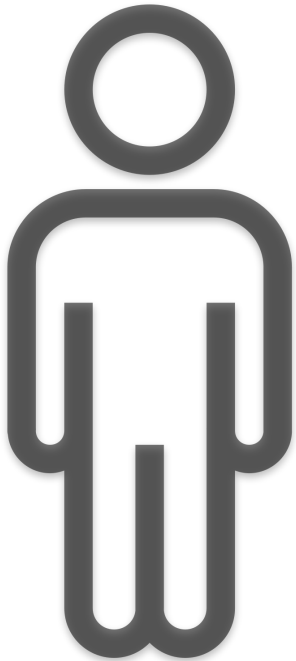


Проверяем наличие элемента в словаре

```
>>> eng2sp.get('four', 'нет перевода')
'нет перевода'
>>> eng2sp.setdefault('four', 'нет перевода')
'нет перевода'
>>> eng2sp
{'one': 'uno', 'two': 'dos', 'three': 'tres', 'four': 'нет перевода'}
```

Сведения о человеке:

- name
- age
- about



```
D = {'about': {'index': 67, 'id': 54, 'age': 43}, 'age': 45}
```

Используется для красивого отображения вложенных словарей на экране

```
from pprint import pprint
pprint(name_of_dict)
```

Наименование: ПЕППЕРОНИ
Состав: Пепперони, моцарелла, томатный соус

S: 395 руб
M: 545 руб

Наименование: МАРГАРИТА
Состав: Томаты, моцарелла, томатный соус

S: 395 руб
M: 545 руб

Наименование: ГАВАЙСКАЯ
Состав: Курица, ветчина, ананас, моцарелла, томатный соус

S: 415 руб
M: 595 руб

Представить в виде словаря

Затем сможем выполнять запросы к словарю:

```
>>> pizza['ГАВАЙСКАЯ']['consist']
['Курица', 'ветчина', 'ананас', 'моцарелла', 'томатный соус']

>>> pizza['ГАВАЙСКАЯ']['size_price']['M']
595
```

Реализация подсчета встречаемости элементов в последовательности

```
import collections
count = collections.Counter(['Ringo', 'Paul', 'John', 'Ringo'])
print(count)
print(count['Ringo'])
print(count['Fred'])
```

Методы словарей

```
>>> D = dict(name='mel', age=45)
>>> D
{'name': 'mel', 'age': 45}
>>> list(D.values())
['mel', 45]
>>> list(D.items())
[('name', 'mel'), ('age', 45)]
>>> D['about'] = {'index': 67, 'id': 54, 'age': 43}
>>> D
{'name': 'mel', 'about': {'index': 67, 'id': 54, 'age': 43}, 'age': 45}
>>> del D['name']
>>> D
{'about': {'index': 67, 'id': 54, 'age': 43}, 'age': 45}
>>> list(D.keys())
['about', 'age']
>>> D
{'about': {'index': 67, 'id': 54, 'age': 43}, 'age': 45}
>>> D['about']['index']
67
```

```
>>> d1 = {'a' : 'some value', 'b' : [1, 2, 3, 4]}

>>> d1.update({'b' : 'foo', 'c' : 12})
>>> d1

{'a': 'some value', 'b': 'foo', 'c': 12}
```

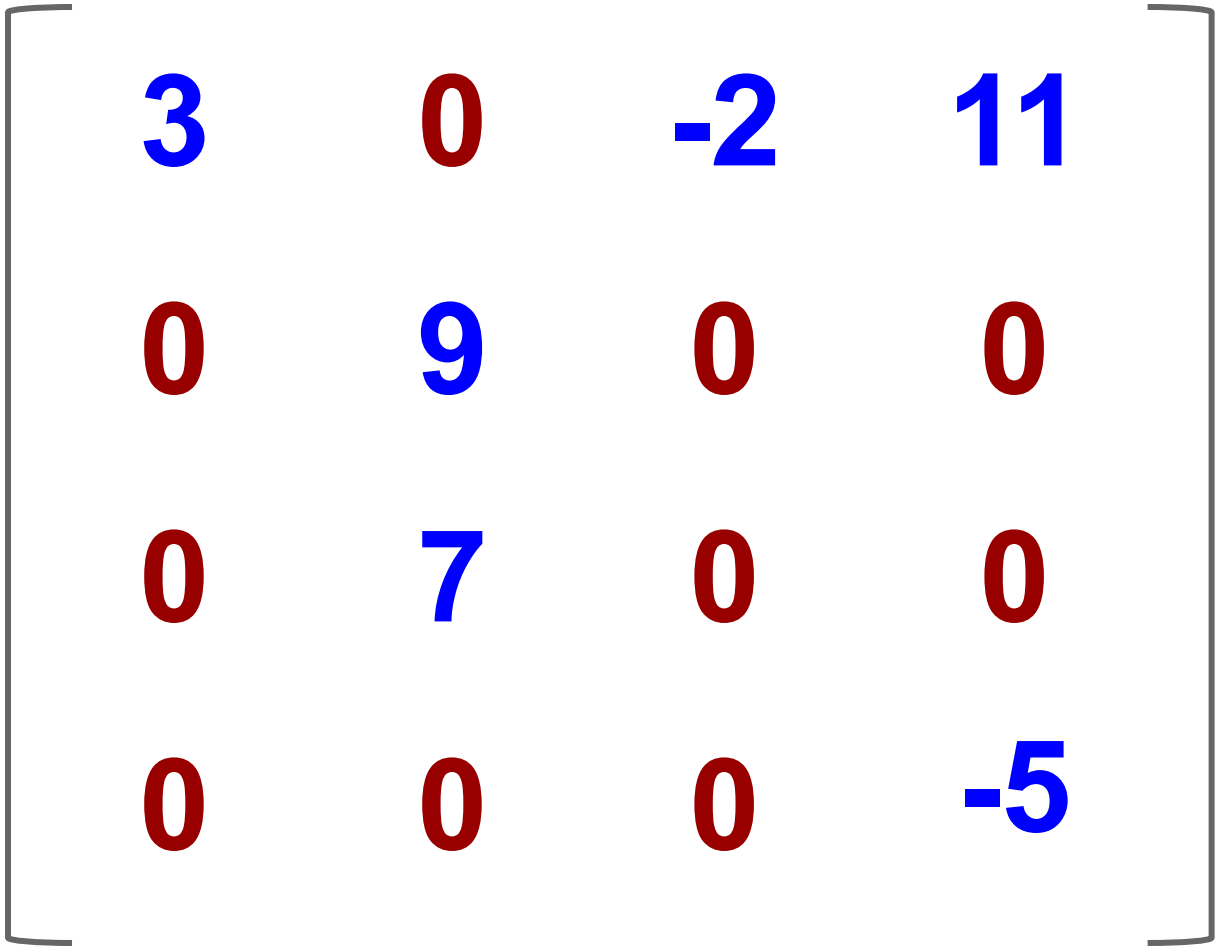
Повсюду словари...
На словарях построено внутреннее представление типов данных

```
>>> def foo():
        x = 1
        y = 'string'
        print(locals())


>>> foo()
```

Возвращает словарь локальных переменных

Пример использования словаря



Разреженные матрицы

```
>>> matrix = [[3, 0, -2, 11], [0, 9, 0, 0], [0, 7, 0, 0], [0, 0, 0, -5]]

>>> matrix[2][2]
0

>>> matrix = {(0, 0): 3, (0, 2): -2, (0, 3): 11, (1, 1): 9, (2, 1): 7, (3, 3): -5}

>>> element = matrix.get((0, 1), 0)

>>> element
0
```

Аналог switch-case в Python

```
>>> def first():
    print('one')

>>> def second():
    print('two')

>>> def third():
    print('three')
```

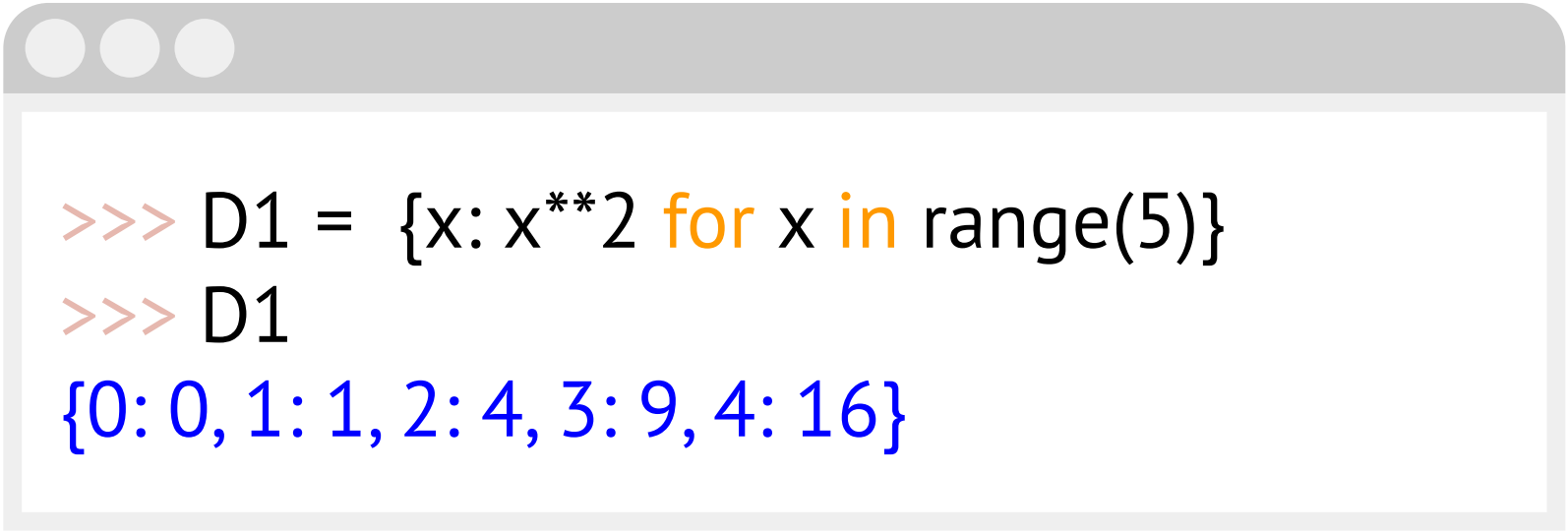
```
>>> x = 2

>>> if x == 1:
    first()
elif x == 2:
    second()
elif x == 3:
    third()

two
```

```
>>> dct = {1: first, 2: second, 3: third}

>>> dct[x]()
two
```


>>> D1 = {x: x**2 for x in range(5)}
>>> D1
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16}

Переменное число параметров

```
def catch_all(*args, **kwargs):
    print("args =", args)
    print("kwargs = ", kwargs)
```

Сортировка словарей

```
data = [{'first':'Guido', 'last':'Van Rossum', 'YOB':1956},
        {'first':'Grace', 'last':'Hopper', 'YOB':1906},
        {'first':'Alan', 'last':'Turing', 'YOB':1912}]

sorted(data, key=lambda item: item['first'])

sorted(data, key=lambda item: item['YOB'])
```

https://dfedorov.spb.ru