```
In [1]: from IPython.core.interactiveshell import InteractiveShell
        InteractiveShell.ast_node_interactivity = "all"
```

PYTHON 3

Словари, множества, collections

MIPT 2030

Словари

```
In [2]: a = {'Key1' : 'Value1', 'Key2' : 'Value2'}
        a
        b = {}
        type(b)

Out[2]: {'Key1': 'Value1', 'Key2': 'Value2'}

Out[2]: dict
```

```
In [3]: a['Key1']

Out[3]: 'Value1'
```

```
In [4]: a['Key3']

        ---------------------------------------------------------------------------
        KeyError                                  Traceback (most recent call last)
        <ipython-input-4-a99dc9db84a5> in <module>
        ----> 1 a['Key3']

        KeyError: 'Key3'
```

```
In [6]: 'Key3' in a # a.keys()

Out[6]: False
```

```
In [7]: a.keys()
        a.values()
        a.items()

Out[7]: dict_keys(['Key1', 'Key2'])

Out[7]: dict_values(['Value1', 'Value2'])

Out[7]: dict_items([('Key1', 'Value1'), ('Key2', 'Value2')])
```

```
In [8]: a['Key3'] = 'Value3'
        a['Key3']

Out[8]: 'Value3'
```

```
In [9]: 'Key3' in a

Out[9]: True
```

```
In [10]: b = dict([(1, 1), (2, 4), (3, 9)])
         b

Out[10]: {1: 1, 2: 4, 3: 9}
```

Ключом словаря может быть любой хэшируемый объект (т.е. тот, от которого можно взять функцию **hash**)

```
In [11]: hash(343)

Out[11]: 343
```

```
In [12]: import sys

         sys.hash_info
         sys.maxsize
         hash(sys.maxsize - 3)

Out[12]: sys.hash_info(width=64, modulus=2305843009213693951, inf=314159, nan=0, imag=10
         00003, algorithm='siphash24', hash_bits=64, seed_bits=128, cutoff=0)

Out[12]: 9223372036854775807

Out[12]: 0
```

```
In [13]: hash(2305843009213693951)
         hash(2305843009213693951 + 16)
         hash(2305843009213693951 - 1)

Out[13]: 0

Out[13]: 16

Out[13]: 2305843009213693950
```

```
In [14]: hash(6.5)

Out[14]: 1152921504606846982
```

```
In [15]: hash('aaa')
         hash('aab')

Out[15]: -8624978997085350272

Out[15]: -2705635159545895278
```

```
In [16]: hash([1, 2])

         ---------------------------------------------------------------------------
         TypeError                                 Traceback (most recent call last)
         <ipython-input-16-4b420d0158ba> in <module>
         ----> 1 hash([1, 2])

         TypeError: unhashable type: 'list'
```

```
In [17]: hash((1, 2))

Out[17]: -3550055125485641917
```

По словарю можно итерироваться, причем как по ключам, так и по значениям

```
In [18]: # итерация по ключам
         apples = {'Bob': 5, 'Dave': 4, 'Charlie': 6, 'Alice': 3}

         for k in apples.keys():
             print(k)

         print()

         for k in apples: # лучше
             print(k)
```

```
Bob
Dave
Charlie
Alice

Bob
Dave
Charlie
Alice
```

```
In [19]: for v in apples.values(): # итерация по значениям
             print(v)
```

```
5
4
6
3
```

```
In [20]: for pair in apples.items(): # итерируемся сразу по парам (ключ: значение)
             print(pair)
```

```
('Bob', 5)
('Dave', 4)
('Charlie', 6)
('Alice', 3)
```

```
In [21]: for name, amount in apples.items():
             print(name, 'has', amount, 'apples')
```

```
Bob has 5 apples
Dave has 4 apples
Charlie has 6 apples
Alice has 3 apples
```

Функция **zip**:

```
In [22]: list(zip(['a', 'b', 'c'], [1, 2, 3]))
         list(zip(['a', 'b', 'c'], [1, 2]))
         list(zip(['a', 'b', 'c'], [1, 2, 3, 4]))
         zip([1], [2])
```

```
Out[22]: [('a', 1), ('b', 2), ('c', 3)]

Out[22]: [('a', 1), ('b', 2)]

Out[22]: [('a', 1), ('b', 2), ('c', 3)]

Out[22]: <zip at 0x7fbc28b8d800>
```

```
In [23]: l = list(zip(['a', 'b', 'c'], [1, 2, 3, 4], [9.5, 0.6, 4.5]))
         l
```

```
Out[23]: [('a', 1, 9.5), ('b', 2, 0.6), ('c', 3, 4.5)]
```

```
In [24]: list(zip(*l))

Out[24]: [('a', 'b', 'c'), (1, 2, 3), (9.5, 0.6, 4.5)]
```

```
In [25]: list(zip(('a', 1, 9.5), ('b', 2, 0.6), ('c', 3, 4.5)))

Out[25]: [('a', 'b', 'c'), (1, 2, 3), (9.5, 0.6, 4.5)]
```

Еще способы создания словаря

```
In [26]: a = dict(a=1, b=2, c=3)
         a

         dct = {i : i ** 3 for i in range(5)}   # Dict comprehension
         dct

         keys = ["Petya", "Vasya", "Masha"]
         values = [20, 21, 22]

         dictionary = dict(zip(keys, values))
         dictionary

Out[26]: {'a': 1, 'b': 2, 'c': 3}

Out[26]: {0: 0, 1: 1, 2: 8, 3: 27, 4: 64}

Out[26]: {'Petya': 20, 'Vasya': 21, 'Masha': 22}
```

Изменение словаря

```
In [27]: d = {'a': 1, 'b': 2, 'c': 3, 'd': 5}
         print(d)

         {'a': 1, 'b': 2, 'c': 3, 'd': 5}
```

```
In [28]: d.update({'a': 6, 'e': 4})
         print(d)

         {'a': 6, 'b': 2, 'c': 3, 'd': 5, 'e': 4}
```

```
In [29]: print(d)
         x = d.get('f', 'default')
         print(x)

         {'a': 6, 'b': 2, 'c': 3, 'd': 5, 'e': 4}
         default
```

```
In [30]: del d['c']
         print(d)

         {'a': 6, 'b': 2, 'd': 5, 'e': 4}
```

```
In [31]: del d['c']

         ---------------------------------------------------------------------------
         KeyError                                  Traceback (most recent call last)
         <ipython-input-31-3b08e515963e> in <module>
         ----> 1 del d['c']

         KeyError: 'c'
```

In [32]:
```
print(d)
y = d.pop('d')
print(y)
print(d)
```

```
{'a': 6, 'b': 2, 'd': 5, 'e': 4}
5
{'a': 6, 'b': 2, 'e': 4}
```

In [33]:
```
d.pop('d')
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-33-48771fd952a5> in <module>
----> 1 d.pop('d')

KeyError: 'd'
```

In [34]:
```
d[('Composite', 'Key')] = [1, 2, 3]
print(d)
```

```
{'a': 6, 'b': 2, 'e': 4, ('Composite', 'Key'): [1, 2, 3]}
```

Множества (set)

В основе set тоже лежит хэш-таблица

In [35]:
```
a = {1, 2, 3}
b = set([2, 3, 4])
c = {i for i in range(10) if not i % 3}   # Set comprehension

print(a)
print(b)
print(c)
```

```
{1, 2, 3}
{2, 3, 4}
{0, 9, 3, 6}
```

In [36]:
```
2 in a
1 in b
```

```
Out[36]: True

Out[36]: False
```

In [37]:
```
a = {1, 2, 4}
print(a)
a.add(3)
print(a)
a.remove(4)
print(a)
a.remove(4)
```

```
{1, 2, 4}
{1, 2, 3, 4}
{1, 2, 3}

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-37-20364c760c4b> in <module>
      5 a.remove(4)
      6 print(a)
----> 7 a.remove(4)

KeyError: 4
```

In [5]:
```
a.discard(2)
a.discard(4)
a
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-5-3f23c975cd1d> in <module>
----> 1 a.discard(2)
      2 a.discard(4)
      3 a

AttributeError: 'dict' object has no attribute 'discard'
```

In [38]:
```
%%time
import random

s = set()
for i in range(10000):
    s.add(random.randint(1, 1000000))
```

```
CPU times: user 12.8 ms, sys: 54 µs, total: 12.8 ms
Wall time: 12.5 ms
```

In [39]:
```
%%time
s = set()
for i in range(10000):
    s.add(2305843009213693951 * i)
```

```
CPU times: user 986 ms, sys: 0 ns, total: 986 ms
Wall time: 984 ms
```

```
In [40]: a = {1, 2, 3}
         b = set([2, 3, 4])

         print (a - b)
         print (b - a)
         print (a | b) # объединение
         print (a & b) # пересечение
         print (a ^ b)
```

```
{1}
{4}
{1, 2, 3, 4}
{2, 3}
{1, 4}
```

```
In [41]: a.difference(b)
         b.difference(a)
         a.union(b)
         a.intersection(b)
         a.symmetric_difference(b)
```

```
Out[41]: {1}

Out[41]: {4}

Out[41]: {1, 2, 3, 4}

Out[41]: {2, 3}

Out[41]: {1, 4}
```

```
In [42]: print(a)
         print(b)
         a -= b
         print(a)
         a |= b
         print(a)
         a &= b
         print(a)
         a ^= b
         print(a)
```

```
{1, 2, 3}
{2, 3, 4}
{1}
{1, 2, 3, 4}
{2, 3, 4}
set()
```

```
In [43]: print(a)
         print(b)
         a.difference_update(b)
         print(a)
         a.update(b)
         print(a)
         a.intersection_update(b)
         print(a)
         a.symmetric_difference_update(b)
         print(a)
```

```
set()
{2, 3, 4}
set()
{2, 3, 4}
{2, 3, 4}
set()
```

```
In [44]: a = {1, 2}
         b = {1, 2, 3}
         c = {1, 2, 4}
         d = {3, 4}

         a.issubset(b), a < b
         b.issuperset(a), b > a
         b.isdisjoint(c), not b & c
         a.isdisjoint(d), not a & d

Out[44]: (True, True)

Out[44]: (True, True)

Out[44]: (False, False)

Out[44]: (True, True)
```

frozenset - неизменемый set

```
In [45]: a = frozenset([1, 2])
         print(a)

         a.add(4)

frozenset({1, 2})

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-45-c9f70c23b0ab> in <module>
      2 print(a)
      3 
----> 4 a.add(4)

AttributeError: 'frozenset' object has no attribute 'add'
```

collections

Объекты в collections - модифицированные для разных нужд словари и еще несколько удобных структур данных.

Хороший краткий обзор модуля collections можно почитать здесь (https://pythonworld.ru/moduli/modul-collections.html)

```
In [46]: from collections import defaultdict
         dct = defaultdict(float)
         print(dct[2]) # если ключа нет, то устанавливает дефолтное значение
         print(dct)

0.0
defaultdict(<class 'float'>, {2: 0.0})
```

```
In [47]: float()

Out[47]: 0.0
```

```
In [48]: dct = defaultdict(lambda: 45)
         print(dct[2])
         print(dct)

45
defaultdict(<function <lambda> at 0x7fbc28b1ac10>, {2: 45})
```

Пример: считаем количество вхождений чисел в список

```
In [49]: numbers = [1, 2, 3, 2, 1, 2, 3, 2, 4, 3, 4, 1, 2, 3, 4]
         counts = defaultdict(int)
         for x in numbers:
             counts[x] += 1

         print(counts)
```

```
defaultdict(<class 'int'>, {1: 3, 2: 5, 3: 4, 4: 3})
```

А теперь нормально через Counter

```
In [50]: from collections import Counter
         print(Counter(numbers))
```

```
Counter({2: 5, 3: 4, 1: 3, 4: 3})
```

```
In [51]: from collections import deque
         q = deque()

         for i in range(10):
             q.append(i)

         while len(q) > 5:
             print(q.pop(), q) # O(1)

         print()

         while q:  # пока дек не пуст
             print(q.popleft(), q) # O(1)
```

```
9 deque([0, 1, 2, 3, 4, 5, 6, 7, 8])
8 deque([0, 1, 2, 3, 4, 5, 6, 7])
7 deque([0, 1, 2, 3, 4, 5, 6])
6 deque([0, 1, 2, 3, 4, 5])
5 deque([0, 1, 2, 3, 4])

0 deque([1, 2, 3, 4])
1 deque([2, 3, 4])
2 deque([3, 4])
3 deque([4])
4 deque([])
```

```
In [52]: import queue
```

```
In [53]: import heapq

         a = [5, 2 , 1, 4, 9, 16, 3]
         heapq.heapify(a)
         a
```

```
Out[53]: [1, 2, 3, 4, 9, 16, 5]
```

```
In [54]: heapq.nlargest(2, a)
         heapq.nsmallest(3, a)
```

```
Out[54]: [16, 9]

Out[54]: [1, 2, 3]
```

```
In [55]: heapq.heappush(a, 25)
         a
Out[55]: [1, 2, 3, 4, 9, 16, 5, 25]
```

```
In [56]: heapq.heappop(a)
Out[56]: 1
```

```
In [57]: a
Out[57]: [2, 4, 3, 25, 9, 16, 5]
```