

Занятие 4

Словари

Начнем с ...

- 1. Разминка (вопросы из интервью)
- 2. Обсуждение домашнего задания

Что выведет код?

```
d = '4'
```

```
e = 'hi'
```

```
d, e = e, int(d)
```

```
print(d, e)
```

```
lang = 'Python'
```

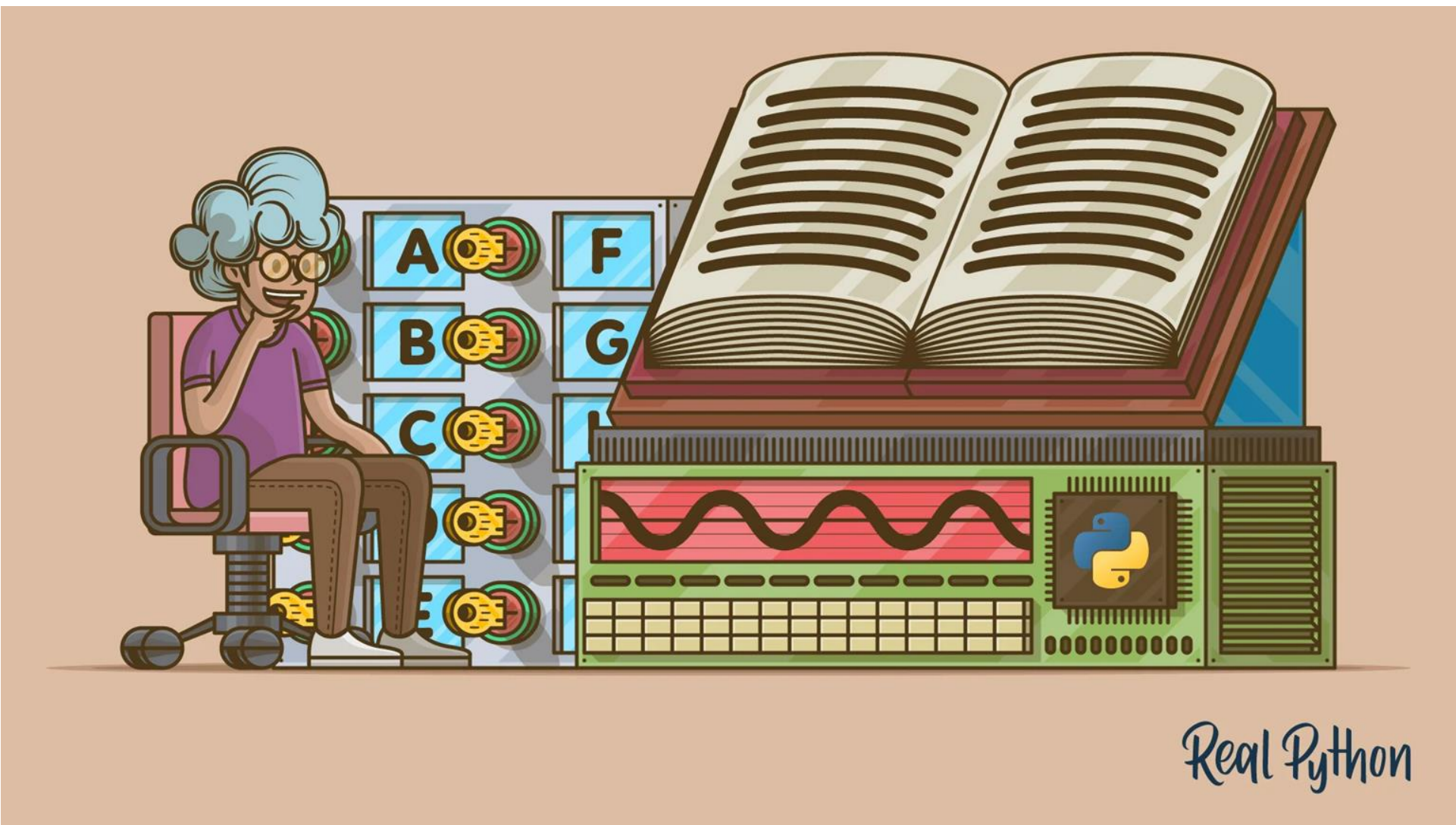
```
print(f"{'lang'} is the best!")
```

Коллекции

1. Строка (str) 'Hello world'
2. Список (list) [1, 100, 1, 'a', True]
3. Кортеж (tuple) (1, 100, 1, 'a', True)
4. **Словарь (dict) {1:1, 22:100, 123:1, 'a':'a', 5:True}**
5. Множество (set) {1, 100, 'a', True}

•

Словарь



Real Python

Определение словаря

```
dict = {k:v}
```

- Словарь задается парой **ключ:значение**, ключ – уникален!

- dic = {
- <key>: <value>,
- <key>: <value>,
- .
- .
- .
- <key>: <value>
- }

Пример 1:

- `person = {`
- `'name': 'Маша',`
- `'login': 'masha',`
- `'age': 25,`
- `'email': 'masha@yandex.ru',`
- `'password': 'fhei23jj~'`
- `}`
- `print(type(person))`
- **`<class 'dict'>`**

Пример 2:

- #Словарь, где ключи являются целыми числами.
- dict_sample = {
 - 1: 'mango',
 - 2: 'coco'
 - }

Пример 3:

- # Hmm... если ключи состоят из примитивных типов то могу ли я сделать так ?
- dict_sample = {
 - **True:** 'mango',
 - **False:** 'coco'
 - }

Пример 4:

- # .. пойдём дальше
- dict_sample = {
- **None**: 'mango',
- **None**: 'coco'
- }

Задание

- Создайте словарь: номер месяца -> количество дней в месяце.
- После чего напишите программу, которая в бесконечном цикле вводит год и номер месяца и выводит количество дней в месяце.
- Будем считать, что если год делится на 4 ($\text{year \% 4} == 0$), то год високосный.
- Выход из цикла: ввод двух нулей.
- Ввод двух чисел можно реализовать функцией:
`y, m = int(input()), int(input())`



Другие способы создания


- `{'name': 'Маша', 'age': 16}` # литеральным выражением
- `person = {}` # динамическое присваивание по ключам
- `person['name'] = 'Маша'`
- `person['age'] = 16`
- **`dict(name='Маша', age=16)`** # через конструктор `dict`
- `letters = ['a', 'b', 'c', 'd']`
- `pronans = ['эй', 'би', 'си', 'ди']`
- `d = dict(zip(letters, pronans))` # используя функцию `zip`
- # `{'a': 'эй', 'b': 'би', 'c': 'си', 'd': 'ди'}`

Задание

- Вводится число, например: 1231
- Вывести строчку, например: один два три один.
- Подсказка: создайте словарь, где ключами являются цифры, а значениями являются слова их обозначающие.

Доступ к элементу по ключу. Замена значения

dict = { k : v }



- >>> person['name']
- Маша
- # Замена значения
- >>> person['name'] = 'Даша'

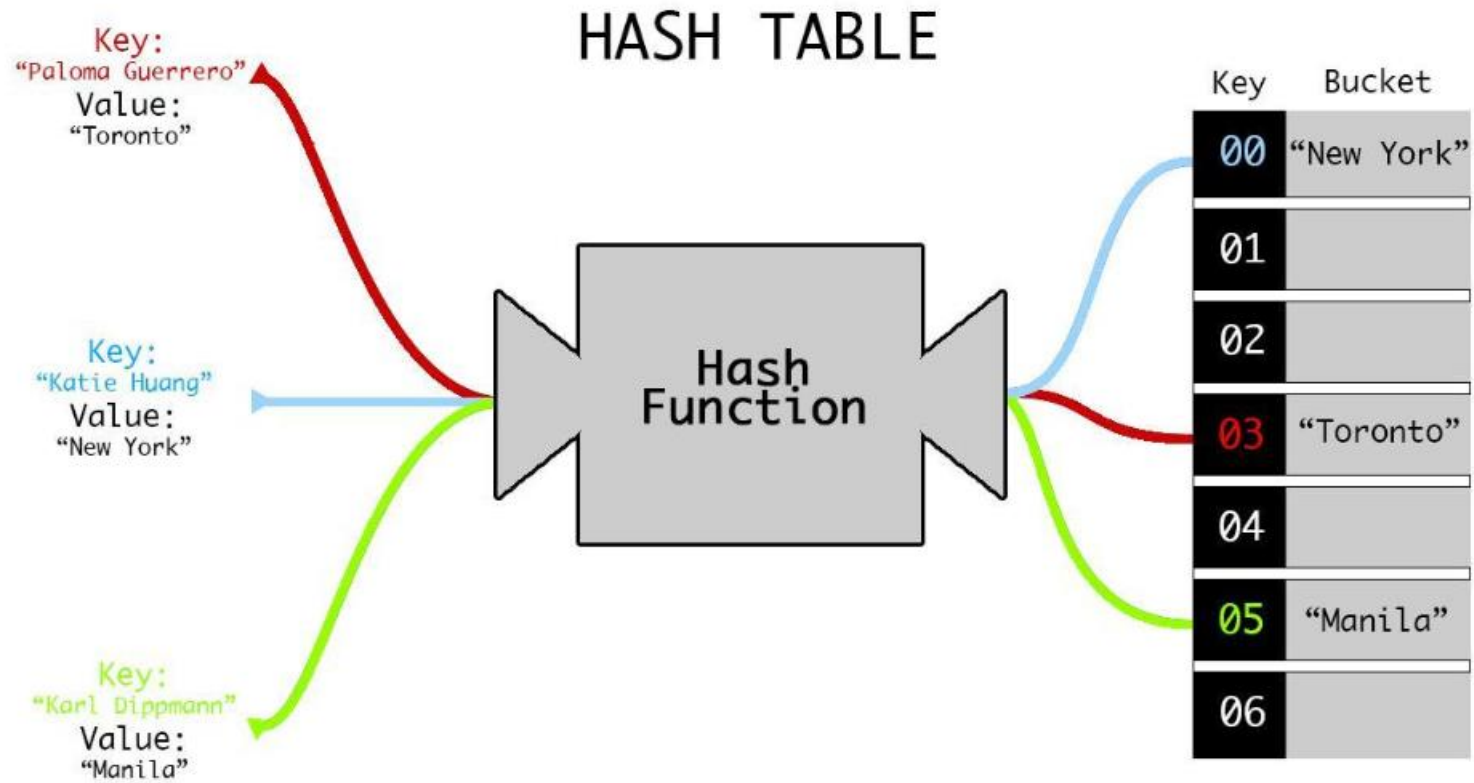
Добавление нового элемента

dict = { **k**: **v**, **k2**: **v2** }



- >>> person['surname']= 'Медведьева'
- {
 - 'name': 'Даша',
 - 'login': 'masha',
 - 'age': 25, 'email': 'masha@yandex.ru',
 - 'password': 'fhei23jj~',
 - 'surname': 'Медведьева'
- }

Хэш-таблицы – быстрый доступ



Удаление элемента

dict = { k : v ,  }

- >>> del person['login']
- {
 - 'name': 'Даша',
 - 'age': 25,
 - 'email': 'masha@yandex.ru',
 - 'password': 'fhei23jj~',
 - 'surname': 'Медведьева'
- }

Проверка на наличие ключа

```
dict = { "A": v }
```

"A"? ↗

- `>>> 'name' in person`
- `True`
- `print('Ключ есть') if ('name' in person) else print('Ключа нет')`
- А что будет, если все-таки попытаться обратиться к ключу, которого нет?
- `d = {1:123}`
- `print(d[2])`

Длина словаря в Python

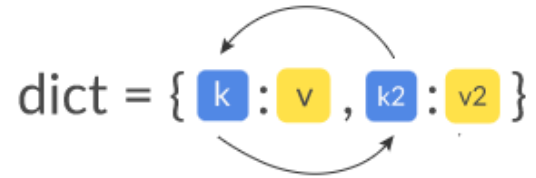
dict = { k : v , k2 : v2 }



len = 2

- Количество записей мы можем получить, воспользовавшись функцией len()
- >>> num_of_items = len(person)
- >>> print(num_of_items)
- >>> 5
-

Сортировка словаря



- statistic_dict = {'b': 10, 'd': 30, 'e': 15, 'c': 14, 'a': 33}
- for key in **sorted**(statistic_dict):
- print(key)
- a
- b
- c
- d
- e

Итерирование словаря

dict = {  :  ,  :  ,  :  }

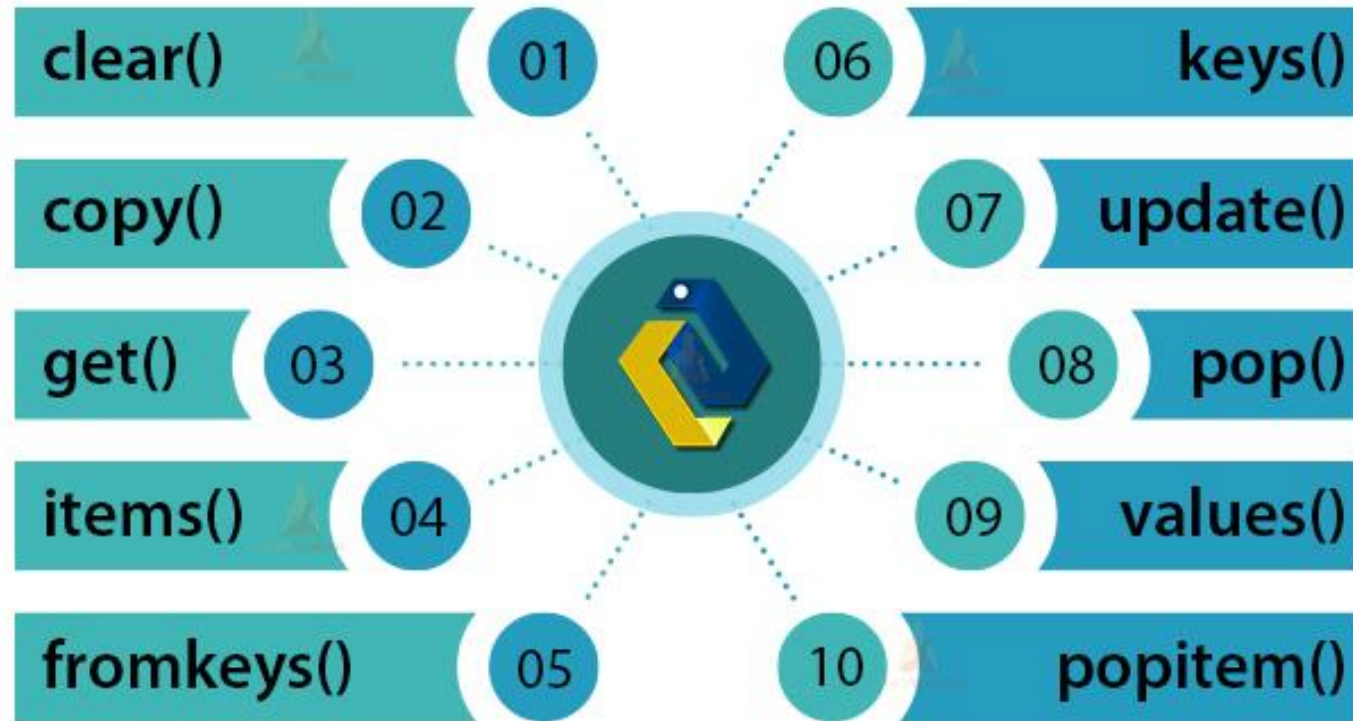


- statistic_dict = {'b': 10, 'd': 30, 'e': 15, 'c': 14, 'a': 33}
- for key, val in statistic_dict.items():
 - print(key)
 - print(val)
- for key in statistic_dict:
 - print(key, statistic_dict[key])

Задание

- Вводится строка букв, например: abracadabra
- Составьте словарь, который для каждой буквы хранит количество ее вхождений в введенное слово.

Python Dictionary Methods



clear()

- Метод производит удаление всех элементов из словаря.
- ```
>>> x = {'one': 0, 'two': 20, 'three': 3, 'four': 4}
```
- ```
>>> x.clear()
```
- ```
>>> x
```
- ```
# {}
```


copy()

- Метод создает копию словаря.
- `>>> x = {'one': 0, 'two': 20, 'three': 3, 'four': 4}`
- `>>> y = x.copy()`
- `>>> y`
- `{'one': 0, 'two': 20, 'three': 3, 'four': 4}`

get(*key*[, *default*])

- Метод dict.get() возвращает значение для ключа key, если ключ находится в словаре, если ключ отсутствует то вернет значение default.
- Если значение default не задано и ключ key не найден, то метод вернет значение None.
- Метод dict.get() никогда не вызывает исключение KeyError, как это происходит в операции получения значения словаря по ключу [dict[key]].

```
x = {'one': 1, 'two': 2, 'three': 3, 'four': 4}
```

```
x.get('two', 0) # 2
```

```
x.get('ten', 0) # 0
```

```
print(x)
```

setdefault(*key*[, *default*])

- Похож на get, но есть отличие.
- `dct = {1:111, 2:222, 3:333}`
- `print(dct.get(4, 0))`
- `print(dct)`
- `print(dct.setdefault(4, 0))`
- `print(dct)`

Задание

- Введите длинный текст с пробелами
- Напечатайте наиболее часто встречающееся слово
- Если таких слов несколько, то создайте список из этих слов и напечатайте его.

items()

- Метод `dict.items()` возвращает новый список кортежей вида `(key, value)`, состоящий из элементов словаря.
- ```
>>> x = {'one': 1, 'two': 2, 'three': 3, 'four': 4}
```
- ```
>>> items = x.items()
```
- ```
>>> items
```
- ```
dict_items([('one', 1), ('two', 2), ('three', 3), ('four', 4)])
```

fromkeys(iterable[, value])

- Метод dict.fromkeys() встроенного класса dict() создает новый словарь с ключами из последовательности iterable и значениями, установленными в value.
- ```
>>> x = dict.fromkeys(['one', 'two', 'three', 'four'])
```
- ```
>>> x
```
- ```
{'one': None, 'two': None, 'three': None, 'four': None}
```
- ```
>>> x = dict.fromkeys(['one', 'two', 'three', 'four'], 0)
```
- ```
>>> x
```
- ```
{'one': 0, 'two': 0, 'three': 0, 'four': 0}
```

keys()

- Метод `dict.keys()` возвращает список-представление всех ключей , содержащихся в словаре `dict`.
- ```
>>> x = {'one': 1, 'two': 2, 'three': 3, 'four': 4}
```
- ```
>>> keys = x.keys()
```
- ```
>>> keys
```
- ```
dict_keys(['one', 'two', 'three', 'four'])
```

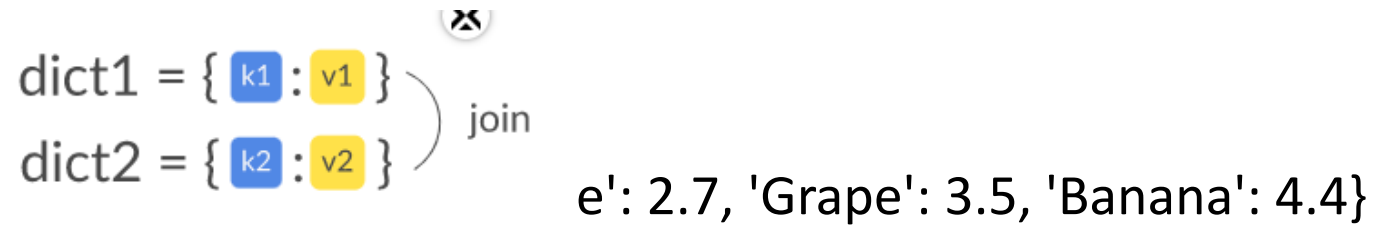
keys()

- Список-представление ключей `dict_keys`, является динамичным объектом. Это значит, что все изменения, такие как удаление или добавление ключей в словаре сразу отражаются на этом представлении.
- # Производим операции со словарем 'x', а все
- # отражается на списке-представлении `'keys'`
- `>>> x = {'one': 1, 'two': 2, 'three': 3, 'four': 4}`
- `>>> keys = x.keys()`
- `>>> del x['one']`
- `>>> keys`
- `dict_keys(['two', 'three', 'four'])`
- `>>> x`
- `{'two': 2, 'three': 3, 'four': 4}`

Задание

- Вводится строка, состоящая из чисел, разделенных пробелом.
- Превращаем ее в список (`lst = list(map(int, input().split()))`)
- Составьте и напечатайте словарь, ключи – числа из списка, а значения списки из индексов, на которых эти числа стоят.
- Например:
- Ввод: 1 1 1 22 1 1 22 33 4 5 1
- Вывод: {1:[0, 1, 2, 4, 5, 10], 22:[3, 6], 33:[7], 4:[8], 5:[9]}

update() - объединение словарей


dict1 = { k1: v1 }
dict2 = { k2: v2 }
e': 2.7, 'Grape': 3.5, 'Banana': 4.4}

- showcase_2 = {'Orange': 1.9, 'Coconut': 10}
- showcase_1.update(showcase_2)
- print(showcase_1)
- > {'Apple': 2.7, 'Grape': 3.5, 'Banana': 4.4, 'Orange': 1.9, 'Coconut': 10}

pop(key[, default])

- Метод dict.pop() вернет значение ключа key, а также удалит его из словаря dict. Если ключ не найден, то вернет значение по умолчанию default.
- >>> x = {'one': 0, 'two': 20, 'three': 3}
- >>> x.pop('three')
- 3
- >>> x
- {'one': 0, 'two': 20}
- >>> x.pop('three', 150)
- 150
- >>> x.pop('three')
- # Traceback (most recent call last):
- # File "<stdin>", line 1, in <module>
- # KeyError: 'ten'

values()

- Метод `dict.values()` возвращает новый список-представление всех значений `dict_values`, содержащихся в словаре `dict`.
 - Список-представление значений `dict_values`, является динамичным объектом. Это значит, что все изменения, такие как удаление, изменение или добавление значений в словаре сразу отражаются на этом представлении.
-
- `>>> x = {'one': 1, 'two': 2, 'three': 3, 'four': 4}`
 - `>>> values = x.values()`
 - `>>> values`
 - `# dict_values([1, 2, 3, 4])`

popitem()

- Метод dict.popitem() удалит и вернет двойной кортеж (key, value) из словаря dict. Пары возвращаются с конца словаря, в порядке **LIFO** (последним пришёл - первым ушёл)
- ```
>>> x = {'one': 0, 'two': 20, 'three': 3}
```
- ```
>>> x.popitem()
```
- ```
('four', 4)
```
- ```
>>> x.popitem()
```
- ```
('three', 3)
```
- ```
>>> x.popitem()
```
- ```
('two', 20)
```
- ```
>>> x.popitem()
```
- ```
Traceback (most recent call last):
```
- ```
#   File "<stdin>", line 1, in <module>
```
- ```
KeyError: 'popitem(): dictionary is empty'
```

# Задание

Ввод: 2 слова, разделенных пробелами.

Для ввода используем функцию `s = input().split()`

Определить, являются ли эти слова анаграммами (словами с одинаковым набором букв).

Если да, то `True`

Если нет, то `False`

(Примеры: АКВАРЕЛИСТ-КАВАЛЕРИСТ, АНТИМОНИЯ-АНТИНОМИЯ, АНАКОНДА-КАНОНАДА, ВЕРНОСТЬ-РЕВНОСТЬ, ВЛАДЕНИЕ-ДАВЛЕНИЕ, ЛЕПЕСТОК-ТЕЛЕСКОП)

# Задание 4-1

- Напишите калькулятор (простой).
- На вход подается строка, например:
- $1 + 2$  или  $5 - 3$  или  $3 * 4$  или  $10 / 2$ .
- Вывод: сосчитать и напечатать результат операции.
- Гарантируется, что два операнда и операция есть в каждой строчке, и все они разделены пробелами.

## Задача 4-2

- Вводим натуральное число  $n$ .
- Напечатайте спираль из чисел  $1, 2, 3, \dots, n * n$
- Например для  $n = 4$ :

```
1 2 3 4
12 13 14 5
11 16 15 6
10 9 8 7
```

Можно использовать словарь с двумя индексами  $d[x, y]$



## Задача 4-3

Ввод: 2 предложения, содержащие пробелы, знаки препинания.

Определить, являются ли эти предложения анаграммами (т.е. имеют одинаковый набор букв).

Игнорируем пробелы, знаки препинания, цифры и т.д.

Вывод: Если да, то True, если нет, то False