

Занятие 4

Словари

Что выведет код?

```
s = 'abracadabra'
```

```
for k in s:
```

```
    print(k, s.find(k))
```

```
#####
```

```
lang = 'Python'
```

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

```
#####
```

```
print('All    your    need    is    love'.split())
```

```
print('All    your    need    is    love'.split(' '))
```

Задача 3-1

- Вводить в бесконечном цикле зарплаты сотрудников.
- Окончание ввода – ввод 0.
- После чего напечатать среднюю зарплату.

Задача 3-2

Дано целое число.

Сосчитать и напечатать, сколько в его записи нулей, единиц, двоек и т.д.

Например:

Ввод: 133244459

Вывод:

0 - 0

1 – 1

2 – 1

3 – 2

4 – 3

5 – 1

6 - 0

и т.д.

Задача 3-3

На вход подается предложение из нескольких слов.

Слова разделены пробелами.

Напечатайте первое самое длинное слово в этом предложении.

Более сложный вариант, напечатать все самые длинные слова,

если их несколько с наибольшей длиной

```
# print(len(max(['I', 'send', 'you', 'a', 'message'], key = len)))
```

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}

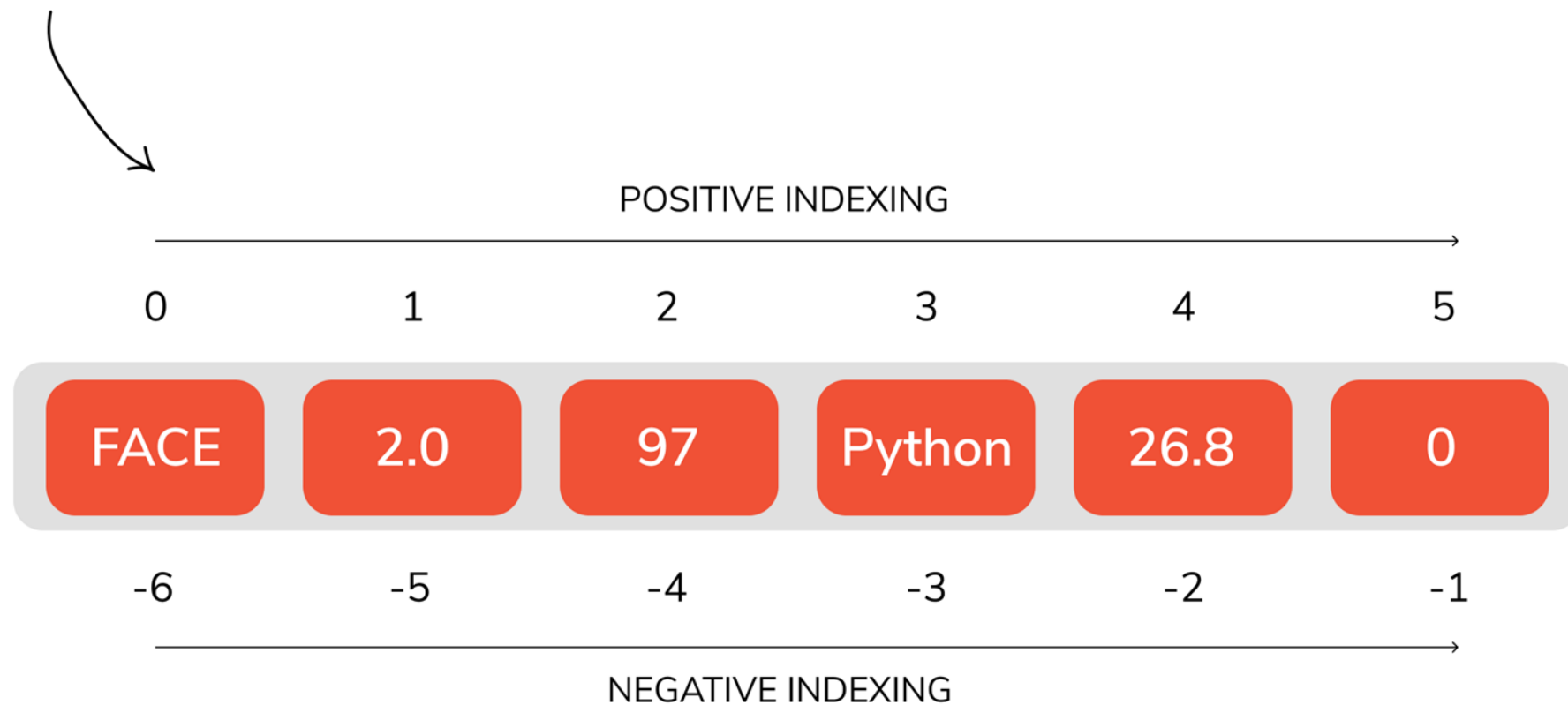
-

Множественное присваивание

- `x, y = 100, 200`
- `(x, y) = (100, 200)`
- `print(x)`
- `100`
- `print(y)`
- `200`

Индексация

Tuple = ('FACE', 2.0, 97, 'Python', 26.8, 0)



Python Tuple Methods

tuple •
 → count()
 → index()



Индекс заданного элемента `index(value, start, stop)`

- `rom = ('I', 'II', 'III', 'IV', 'V', 'VI', 'VII', 'VIII', 'IX', 'X')`
- `print(rom.index('X'))`
- 9
- `str = ('aa', 'bb', 'aa', 'cc')`
- `print(str.index('aa'))`
- 0
- `str = ('aa', 'bb', 'aa', 'cc')`
- `print(str.index('aa', 1, len(str)))`
- `print(str.index('aa', 1,))`
- 2

Число вхождений элемента
count()

- `t_str = ('aa', 'bb', 'aa', 'cc')`
- `print(t_str.count('aa'))`
- 2

Задание

Дан кортеж (123, 234, 345, 456, 567, 678, 789, 890).

Вводится еще одно целое число больше 0.

Создайте новый кортеж из первого кортежа и введенного числа, чтобы в новом кортеже числа не убывали.

Об одном свойстве кортежей

- `tpl = (1, 2, 3, [11, 22, 33])`
- `tpl[3].append(44)`
- `print(tpl)`
- Что будет напечатано?

Общие функции для list, tuple

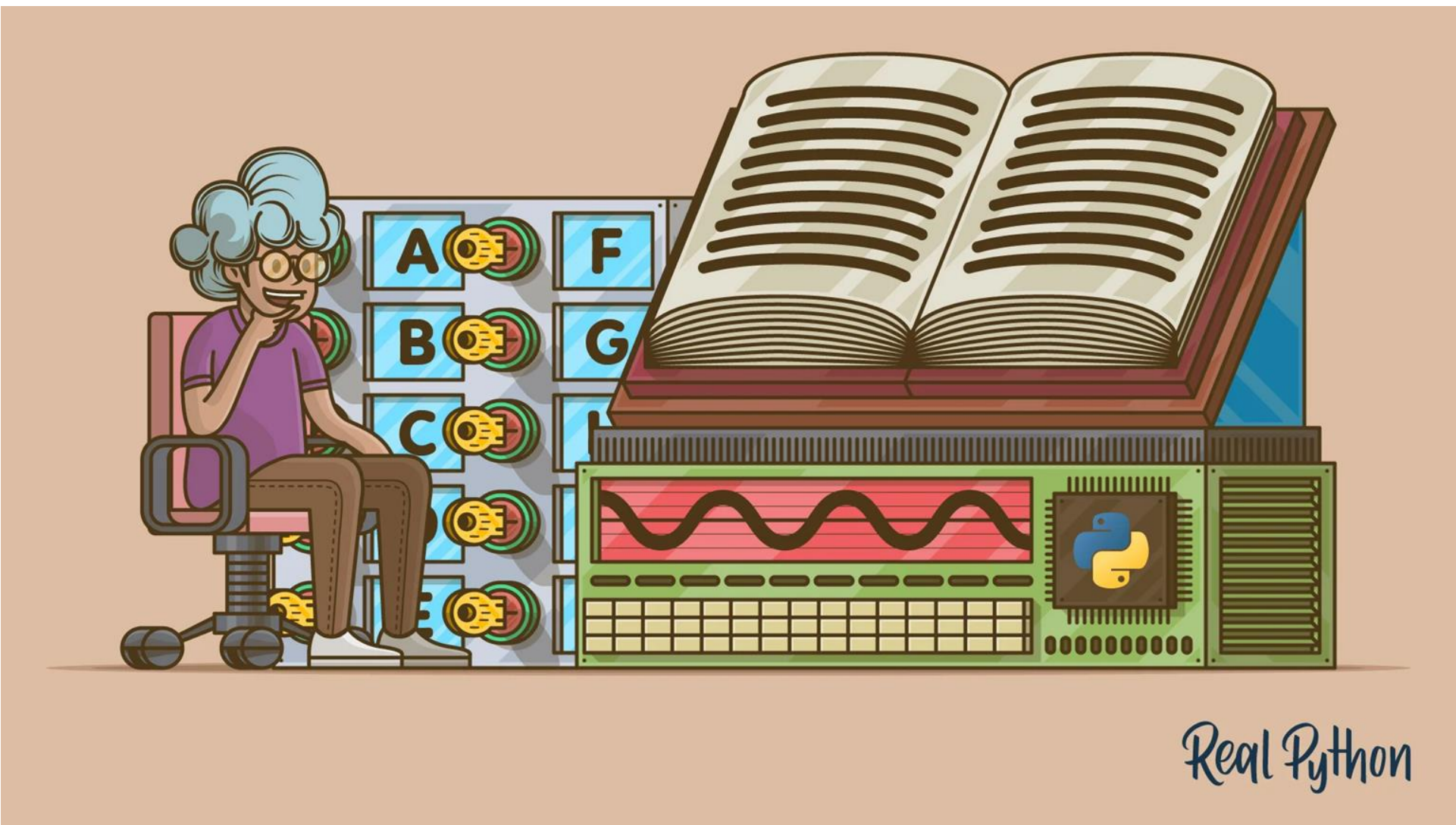
Некоторые из них работают для str и range		
Operation	Result	Результат
x in s	True if an item of s is equal to x, else False	True , если какой-то элемент s равен x, иначе False
x not in s	False if an item of s is equal to x, else True	Наоборот
s + t	the concatenation of s and t	Конкатенация s и t
s * n (or n * s)	equivalent to adding s to itself n times	Эквивалентно сложению с собой n-1 раз
s[i]	ith item of s, origin 0	i – ый элемент, начиная с 0
s[i:j]	slice of s from i to j	Срез от i до j неключительно
s[i:j:k]	slice of s from i to j with step k	Срез с шагом k
len(s)	length of s	Длина
min(s)	smallest item of s	Наименьший элемент
max(s)	largest item of s	Наибольший
s.index(x[, i[, j]])	index of the first occurrence of x in s (at or after index i and before index j)	Индекс первого вхождения x в s, начиная с i до j неключительно
s.count(x)	total number of occurrences of x in s	Количество вхождений x в s

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

Задача аналогичная домашней, но для символов

- Дано целое число. Сосчитать и напечатать, сколько в его записи нулей, единиц, двоек и т.д.

```
hm = [0] * 10          # формируем массив нулей с индексами от 0 до 9
s = input()
for k in s:
    digit = int(k)      # преобразуем символ цифры в саму цифру, т.е. в индекс массива!
    hm[digit] = hm[digit] + 1
```

- Дана строка символов, например: 'abracadabra'. Сосчитать и напечатать, сколько в строке содержится каждого символа (мы не знаем заранее, какие символы есть в строке)

Было бы здорово перебирать символы в строке и накапливать их количества. Т.е. в качестве индекса использовать символы!

```
hm = {}                # определяем пустой словарь
for k in s:            # перебираем символы в строке
    if k not in hm:    # есть ли уже этот символ в словаре?
        hm[k] = 0     # если нет, то заводим такой индекс
    hm[k] = hm[k] + 1  # увеличиваем на 1 количество символов в строке
print(hm)              # печатаем словарь
```

Решение задачи про количество цифр в числе с помощью словаря и без count()

```
s = input()
```

```
dct = {}
```

```
for k in s:
```

```
    if k not in dct:
```

```
        dct[k] = 0
```

```
    dct[k] += 1
```

```
print(dct)
```

А как сделать, чтобы печатались по порядку?

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': 'mmaasshhaa'  
}  
print(person)
```


Измените Машу на Мишу во всех элементах словаря, где это возможно

Например, person['name'] = 'Миша'

Напечатайте новый person

Доступ к элементу по ключу. Замена значения

dict = { k : v }



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

Пример 2:

#Словарь, где ключи являются целыми числами.

```
dict_sample = {  
    1: 'mango',  
    2: 'coco'  
}
```

Напечатайте словарь в цикле:

```
for k in dict_sample:  
    print(k, dict_sample[k])
```

Список vs словарь

Можно ли определить по оператору `a[0] = 123` является `a` списком или словарем?

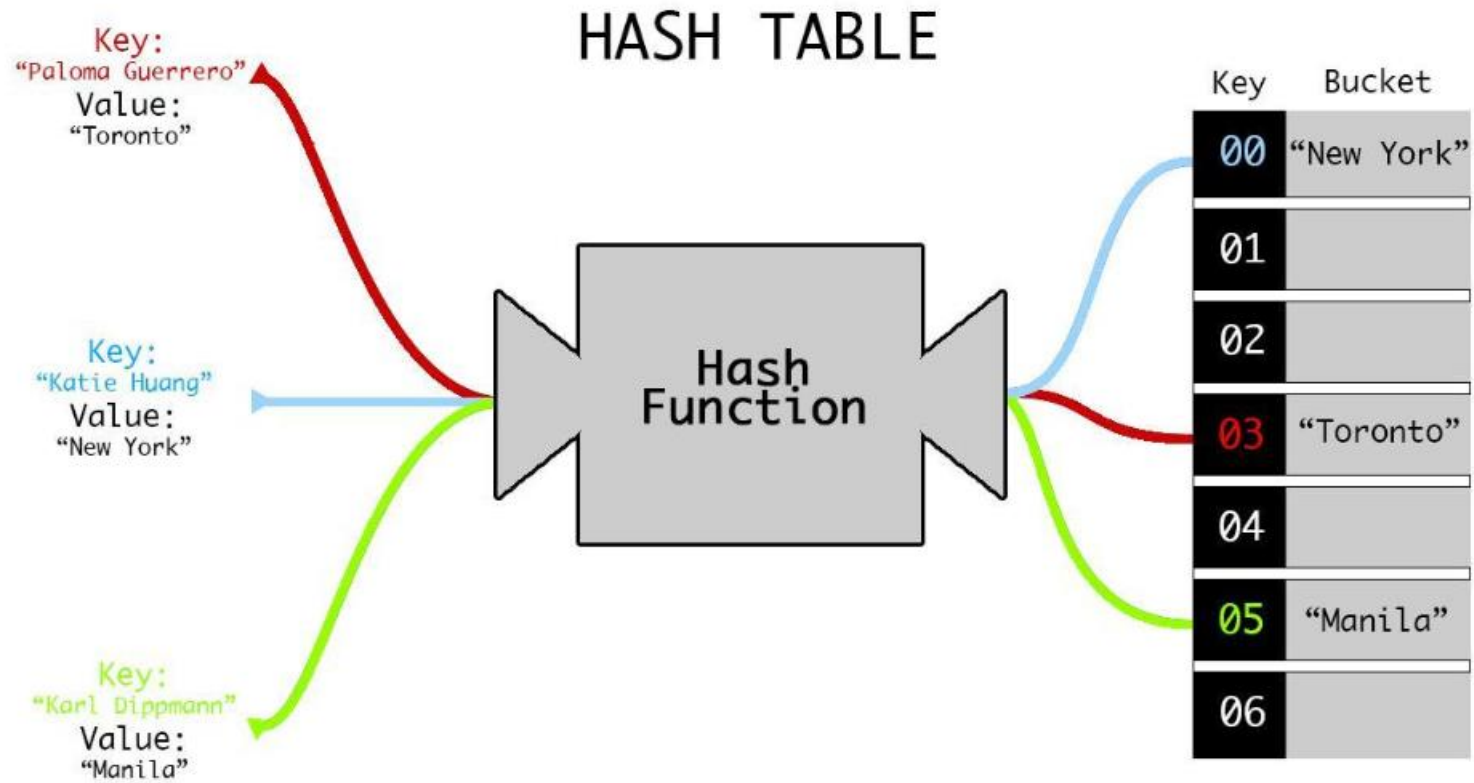
А как отличить?

Можно ли считать, что словарь – это развитие списка в каком-то смысле?

В списке индекс – это порядковый номер: от 0 до какого-то значения.

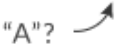
В словаре индекс – это уникальный идентификатор объекта, который преобразуется в некий искусственный «номер».

Хэш-таблицы – быстрый доступ



Проверка на наличие ключа

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



```
>>> 'name' in person  
True
```

```
if ('name' in person):  
    print('Ключ есть')  
else:  
    print('Ключа нет')
```

А что будет, если все-таки попытаться обратиться к ключу, которого нет?

```
d = {1:123}  
print(d[2])
```

Задание

Вводим по очереди товары, которые были проданы в интернет-магазине.

Окончание ввода – 0.

Надо сосчитать и напечатать, сколько и какого товара продано.

Например:

Бананы

Яблоки

Бананы

0

Вывод:

Бананы – 2

Яблоки – 1

Пример 3:

```
dict_sample1 = {  
    True: 'mango',  
    False: 'coco'  
}
```

```
# Напечатайте этот словарь.
```

```
# Поменяйте порядок элементов:
```

```
dict_sample2 = {  
    False: 'coco'  
    True: 'mango',  
}
```

```
# Сравните их.
```

```
print(dict_sample1 == dict_sample2)
```

Пример 4:

```
# .. пойдём дальше
```

```
dict_sample = {  
    None: 'mango',  
    None: 'coco'  
}
```

```
# Напечатайте, что получилось. Почему?
```

Задание

Вводится число, например: 1231

Вывести строчку, например: “один два три один”

Подсказка: предварительно создайте словарь, где ключами являются цифры, а значениями являются слова их обозначающие.

Способы создания

```
{'name':'Маша','age': 16} # литеральным выражением
```

```
person = {} # пустой словарь
```

```
person['name'] = 'Маша' # динамическое присваивание по ключам
```

```
person['age'] = 16
```

```
dict(name='Маша', age=16) # через конструктор класса dict (позже узнаем детали)
```

```
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'}
```

```
# Проверьте, что все эти способы работают
```

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 (в феврале будет 29 дней).

Выход из цикла: ввод двух нулей.

Добавление нового элемента

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



```
>>> person['surname'] = 'Медведева'
```

```
{  
    'name': 'Даша',  
    'login': 'masha',  
    'age': 25, 'email': 'masha@yandex.ru',  
    'password': 'fhei23jj~',  
    'surname': 'Медведева'  
}
```


Удаление элемента

dict = { k : v ,  }

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

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
-

Задание

Введите длинный текст с пробелами

Напечатайте наиболее редко встречающееся слово

Если таких слов несколько, то создайте список из этих слов и напечатайте его.

`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}
```

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

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

```
print(x)
```

```
# Выполните этот код
```

get не меняет словарь!!!

Используем get для расчета количества цифр

```
s = input()
```

```
dct = {}
```

```
for k in s:
```

```
    dct[k] = dct.get(k, 0) + 1 # Тут и проверка, и присвоение 0.
```

```
print(dct)
```

А что может стоять вместо 0 в операторе get? Что угодно – любое начальное значение для какого-то алгоритма.

Иногда пустой список [], тогда вместо + можно использовать .append

Иногда пустая строка "", если мы накапливаем символы.

Иногда может быть и пустой словарь {})).

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 печатаем `True`

Если нет, то печатаем `False`

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

Задание (по аналогии, без использования count())

Вводится текст.

Составьте словарь, который для каждого слова хранит количество его вхождений в введенный текст.

Напечатайте словарь.

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