# Используем "флаг" для изменения статуса исполнения

In [1]:

```
stats = [
    ['2018-01-01', 'google', 25],
    ['2018-01-01', 'yandex', 65],
    ['2018-01-01', 'market', 89],
    ['2018-01-02', 'google', 574],
    ['2018-01-02', 'yandex', 249],
    ['2018-01-02', 'market', 994],
    ['2018-01-03', 'google', 1843],
    ['2018-01-03', 'yandex', 1327],
    ['2018-01-03', 'market', 1764],
]
```

In [2]:

```
date = 'екоеко'
campaign = 'jtrfjrtjrtj'
```

In [3]:

```
flag = 0

for stat in stats:
    if stat[0] == date and stat[1] == campaign:
        print(stat[2])
        flag = 1

if flag == 0:
    print('Нет данных')
```

```
Нет данных
```

# Про f-строки

https://habr.com/ru/post/462179/ (https://habr.com/ru/post/462179/)

# Что будет, если преобразовать во множество список со вложенными списками?

In [4]:

```
income_by_months = [['Jan', 13000], ['Feb', 14000], ['Mar', 14300], ['Apr', 15000], ['May',
month_list = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Aug']
```

In [5]:

```
set(income_by_months)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-5-da86ba08c0d5> in <module>
----> 1 set(income_by_months)

TypeError: unhashable type: 'list'
```

In [6]:

```
income_by_months = [['Jan', 13000], ['Feb', 14000], ['Mar', 14300], ['Apr', 15000], ['May',

res = sum(income_by_months, [])
res
```

Out[6]:

```
['Jan',
 13000,
 'Feb',
 14000,
 'Mar',
 14300,
 'Apr',
 15000,
 'May',
 13800,
 'Jun',
 13000,
 'Jul',
 14900,
 'Aug',
 15200,
 'Sep',
 15300]
```

# Есть ли у множества индексы? можно ли по ним итерироваться?

In [7]:

```
data_scientist_skills = set(['Python', 'R', 'SQL', 'Tableau', 'SAS', 'Git'])

# data_scientist_skills[0]

for skill in data_scientist_skills:
    print(skill)
```

```
Git
Python
SAS
Tableau
R
SQL
```

# Распаковка элементов прямо в цикле

In [8]:

```
company_tuple = ('Orange', 100000000, 20000)
company_name, capitalization, personal = company_tuple
print(company_name)
print(capitalization)
print(personal)
```

```
Orange
100000000
20000
```

In [9]:

```
companies_info = [
    ['Orange', 1.3, 20000, 12, 214],
    ['Maxisoft', 1.5, 40000, 214, 12],
    ['Headbook', 0.8, 15000, 214, 2154],
    ['Nicola', 2.2, 18000, 124, 15]
]

for company in companies_info:
    print(company[1])
    print(company[2])
    print(company[0])
```

```
1.3
20000
Orange
1.5
40000
Maxisoft
0.8
15000
Headbook
2.2
18000
Nicola
```

In [10]:

```
cook_book = {'салат': [ {'ingridient_name': 'сыр', 'quantity': 50, 'measure': 'гр'},
                        {'ingridient_name': 'томаты', 'quantity': 20, 'measure': 'гр'},
                        {'ingridient_name': 'огурцы', 'quantity': 20, 'measure': 'гр'},
                        {'ingridient_name': 'маслины', 'quantity': 10, 'measure': 'гр'},
                        {'ingridient_name': 'оливковое масло', 'quantity': 20, 'measure': 'м
                        {'ingridient_name': 'салат', 'quantity': 10, 'measure': 'гр'},
                        {'ingridient_name': 'перец', 'quantity': 20, 'measure': 'гр'} ],
             'пицца': [ {'ingridient_name': 'сыр', 'quantity': 20, 'measure': 'гр'},
                        {'ingridient_name': 'колбаса', 'quantity': 30, 'measure': 'гр'},
                        {'ingridient_name': 'бекон', 'quantity': 30, 'measure': 'гр'},
                        {'ingridient_name': 'оливки', 'quantity': 10, 'measure': 'гр'},
                        {'ingridient_name': 'томаты', 'quantity': 20, 'measure': 'гр'},
                        {'ingridient_name': 'тесто', 'quantity': 100, 'measure': 'гр'}, ],
             'лимонад': [ {'ingridient_name': 'лимон', 'quantity': 1, 'measure': 'шт'},
                          {'ingridient_name': 'вода', 'quantity': 200, 'measure': 'мл'},
                          {'ingridient_name': 'сахар', 'quantity': 10, 'measure': 'гр'},
                          {'ingridient_name': 'лайм', 'quantity': 20, 'measure': 'гр'}, ]
            }
```

In [11]:

```
cook_book['пицца']
```

Out[11]:

```
[{'ingridient_name': 'сыр', 'quantity': 20, 'measure': 'гр'},
 {'ingridient_name': 'колбаса', 'quantity': 30, 'measure': 'гр'},
 {'ingridient_name': 'бекон', 'quantity': 30, 'measure': 'гр'},
 {'ingridient_name': 'оливки', 'quantity': 10, 'measure': 'гр'},
 {'ingridient_name': 'томаты', 'quantity': 20, 'measure': 'гр'},
 {'ingridient_name': 'тесто', 'quantity': 100, 'measure': 'гр'}]
```

In [12]:

```
my_res = cook_book['пицца'][3]['ingridient_name']
```

In [13]:

```
for dish, igr in cook_book.items():
    print(dish)
#     print(igr)
    for el in igr:
        print(f'{el["ingridient_name"]} {el["quantity"]} {el["measure"]}')
```

```
салат
сыр 50 гр
томаты 20 гр
огурцы 20 гр
маслины 10 гр
оливковое масло 20 мл
салат 10 гр
перец 20 гр
пицца
сыр 20 гр
колбаса 30 гр
бекон 30 гр
оливки 10 гр
томаты 20 гр
тесто 100 гр
лимонад
лимон 1 шт
вода 200 мл
сахар 10 гр
лайм 20 гр
```

# Практика. Решим задачу с диагональной матрицей при помощи list comprehension

In [14]:

```
data = [
    [13, 25, 23, 34],
    [45, 32, 44, 47],
    [12, 33, 23, 95],
    [13, 53, 34, 35]
]

for i, r in enumerate(data):
    print(i)
    print(r)
```

```
0
[13, 25, 23, 34]
1
[45, 32, 44, 47]
2
[12, 33, 23, 95]
3
[13, 53, 34, 35]
```

In [15]:

```
sum([row[index] for index, row in enumerate(data)])
```

Out[15]:

```
103
```

# ссылки про enumerate

https://habr.com/ru/company/ruvds/blog/485648/ (https://habr.com/ru/company/ruvds/blog/485648/)
https://all-python.ru/osnovy/enumerate.html (https://all-python.ru/osnovy/enumerate.html)

# Практика

Сформируйте словарь, ключами в котором будут числа от 1 до 20, а значениями - квадраты этих чисел

In [16]:

```
my_dict = {}
for num in range(1, 21):
    my_dict[num] = num ** 2

my_dict
```

Out[16]:

```
{1: 1,
 2: 4,
 3: 9,
 4: 16,
 5: 25,
 6: 36,
 7: 49,
 8: 64,
 9: 81,
 10: 100,
 11: 121,
 12: 144,
 13: 169,
 14: 196,
 15: 225,
 16: 256,
 17: 289,
 18: 324,
 19: 361,
 20: 400}
```

In [17]:

```
{num: num**2 for num in range(1, 21)}
```

Out[17]:

```
{1: 1,
 2: 4,
 3: 9,
 4: 16,
 5: 25,
 6: 36,
 7: 49,
 8: 64,
 9: 81,
 10: 100,
 11: 121,
 12: 144,
 13: 169,
 14: 196,
 15: 225,
 16: 256,
 17: 289,
 18: 324,
 19: 361,
 20: 400}
```

# Практика

Удалим из словаря все пустые элементы (со значением None)

In [18]:

```
my_dict = {
    'id1': 123456,
    'id2': 654321,
    'id3': None,
    'id4': 777777
}
```

In [19]:

```
dictionary = {}
for key, value in my_dict.items():
    if value is not None:
        dictionary[key] = value

dictionary
```

Out[19]:

```
{'id1': 123456, 'id2': 654321, 'id4': 777777}
```

In [20]:

```
{key:value for key, value in my_dict.items() if value is not None}
```

Out[20]:

```
{'id1': 123456, 'id2': 654321, 'id4': 777777}
```

# Практика. Посчитайте сколько раз каждое слово встречалось в тексте

Вывести в формате: "слово": "количество вхождений"

За разделитель считаем пробел, все слова приводим к нижнему регистру

In [21]:

```
some_oxy_text = '''
Весь мой рэп, если коротко, про то, что
Уж который год который город под подошвой
В гору, когда прет. Потом под гору, когда тошно
Я не то, что Гулливер, но все же город под подошвой
Город под подошвой, город под подошвой
Светофоры, госпошлины, сборы и таможни
Я не знаю, вброд или на дно эта дорожка
Ты живешь под каблуком, у меня - город под подошвой
'''
```

In [22]:

```
res = {}

some_splitted_oxy_text = some_oxy_text.split(' ')


for word in some_splitted_oxy_text:
    word = word.lower()
    if word not in res:
        res[word] = 1
    else:
        res[word] += 1
for word, count in res.items():
    print(f'{word}: {count}')
```

```
весь: 1
мой: 1
рэп,: 1
если: 1
коротко,: 1
про: 1
то,: 2
что
уж: 1
который: 2
год: 1
город: 4
под: 7
подошвой
в: 1
гору,: 2
когда: 2
прет.: 1
потом: 1
тошно
я: 1
не: 2
что: 1
гулливер,: 1
но: 1
все: 1
же: 1
подошвой
город: 1
подошвой,: 1
подошвой
светофоры,: 1
госпошлины,: 1
сборы: 1
и: 1
таможни
я: 1
знаю,: 1
вброд: 1
или: 1
на: 1
дно: 1
эта: 1
дорожка
```

```
ты: 1
живешь: 1
каблуком,: 1
у: 1
меня: 1
-: 1
подошвой
: 1
```

In [23]:

```
my_dict = {}

my_dict['some_key'] = 'some_value'
my_dict['2'] = 2
```

In [24]:

```
my_dict['2'] = 3
```

In [25]:

```
my_dict
```

Out[25]:

```
{'some_key': 'some_value', '2': 3}
```

# Практика

Дан список с визитами по городам и странам. Напишите код, который возвращает отфильтрованный список geo_logs, содержащий только визиты из России.

In [26]:

```
geo_logs = [
    {'visit1': ['Москва', 'Россия']},
    {'visit2': ['Дели', 'Индия']},
    {'visit3': ['Владимир', 'Россия']},
    {'visit4': ['Лиссабон', 'Португалия']},
    {'visit5': ['Париж', 'Франция']},
    {'visit6': ['Лиссабон', 'Португалия']},
    {'visit8': ['Тула', 'Россия']}
]
```

In [27]:

```
for log in geo_logs:
#     print(log)
    for i in log.values():
#         print(i)
        if i[1] != 'Россия':
            geo_logs.remove(log)
geo_logs
```

Out[27]:

```
[{'visit1': ['Москва', 'Россия']},
 {'visit3': ['Владимир', 'Россия']},
 {'visit5': ['Париж', 'Франция']},
 {'visit8': ['Тула', 'Россия']}]
```

In [28]:

```
res = []
for log in geo_logs:
    for i in log.values():
    #         print(i)
        if i[1] == 'Россия':
            res.append(log)
res
```

Out[28]:

```
[{'visit1': ['Москва', 'Россия']},
 {'visit3': ['Владимир', 'Россия']},
 {'visit8': ['Тула', 'Россия']}]
```

In [29]:

```
res = []
for log in geo_logs:
    if list(log.values())[0][1] == 'Россия':
        res.append(log)
res
```

Out[29]:

```
[{'visit1': ['Москва', 'Россия']},
 {'visit3': ['Владимир', 'Россия']},
 {'visit8': ['Тула', 'Россия']}]
```

In [30]:

```
geo_logs_2 = geo_logs.copy()
# geo_logs_2 = geo_logs[:]

for log in geo_logs_2:
    if list(log.values())[0][1] != 'Россия':
        geo_logs.remove(log)
geo_logs
```

Out[30]:

```
[{'visit1': ['Москва', 'Россия']},
 {'visit3': ['Владимир', 'Россия']},
 {'visit8': ['Тула', 'Россия']}]
```

In [31]:

```
list(filter(lambda log: 'Россия' == list(log.values())[0][1], geo_logs))
```

Out[31]:

```
[{'visit1': ['Москва', 'Россия']},
 {'visit3': ['Владимир', 'Россия']},
 {'visit8': ['Тула', 'Россия']}]
```

In [ ]:

```

```