Московский государственный технический

университет им. Н.Э. Баумана

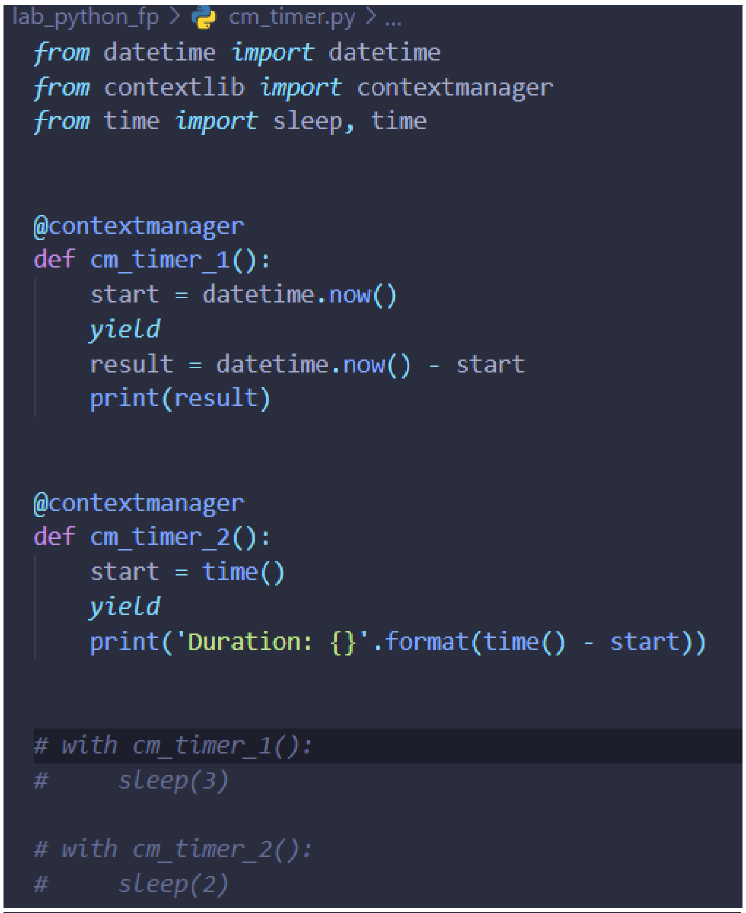
Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

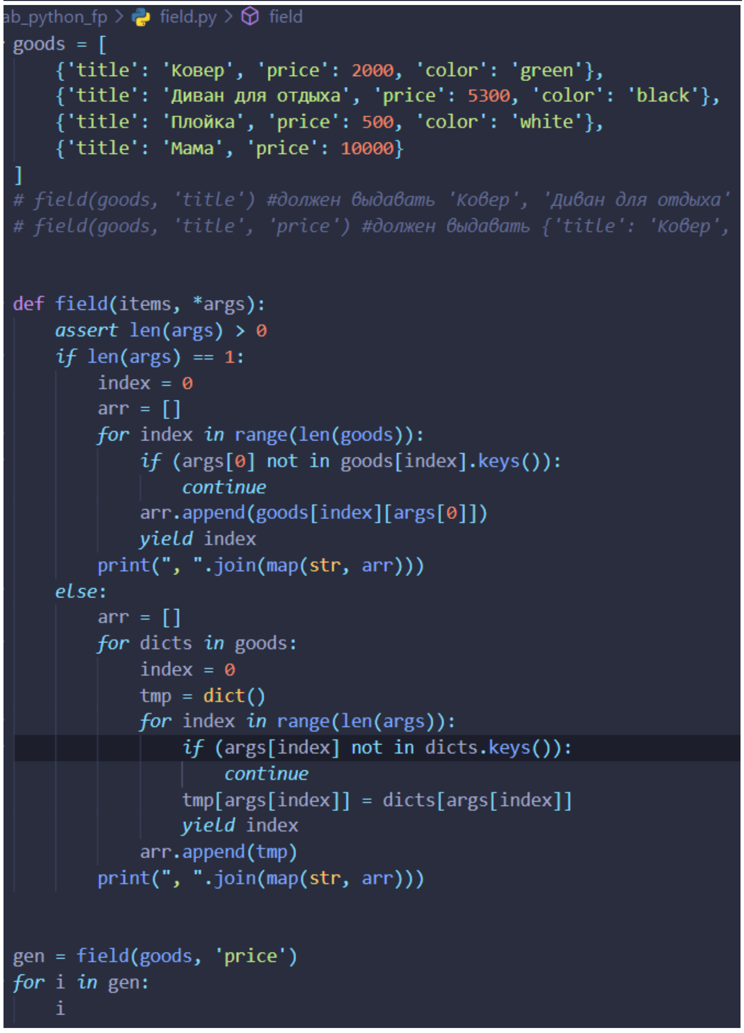
Отчет по лабораторной работе №3

Выполнил:	Проверил:
студент группы ИУ5-52Б	преподаватель каф. ИУ5
Запруднов М.С.	Гапанюк Ю.Е.
Подпись и дата:	Подпись и дата:

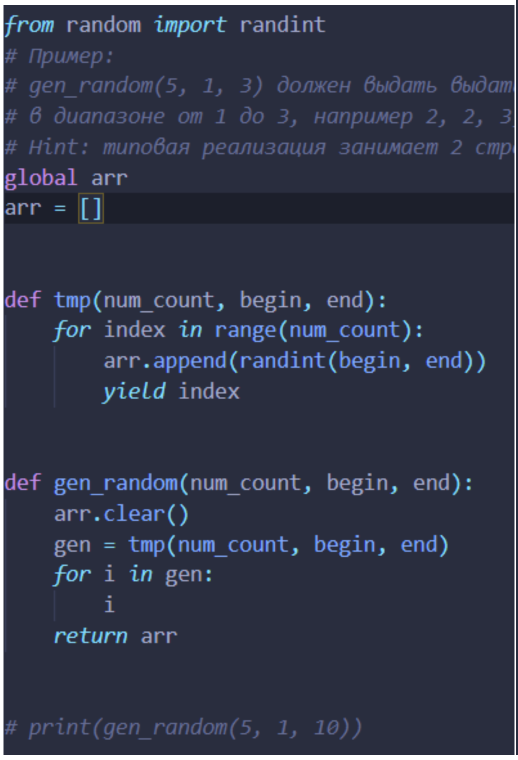
г. Москва, 2020 г.


lab_python_fp > cm_timer.py > ...

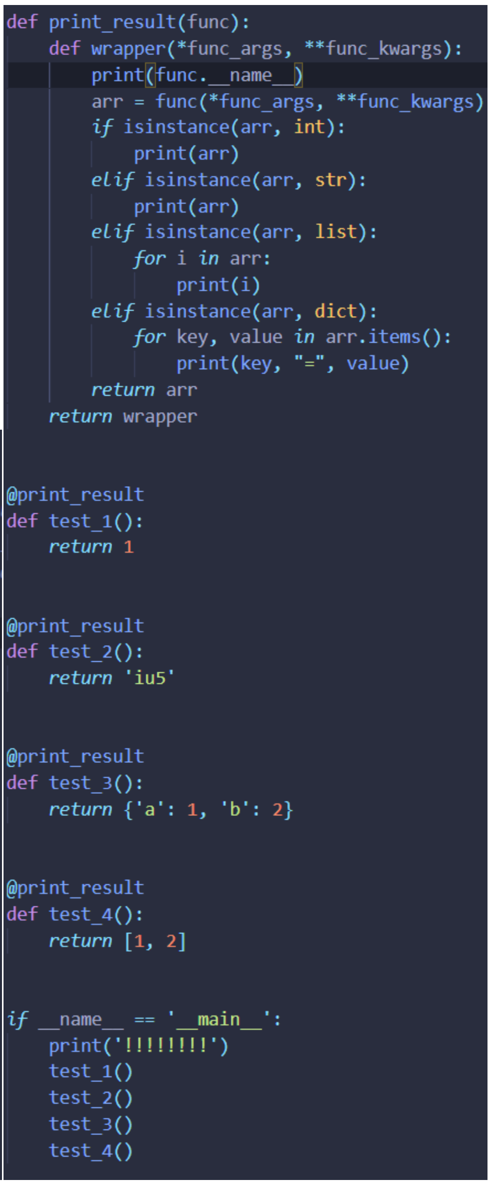
```python
from datetime import datetime
from contextlib import contextmanager
from time import sleep, time


@contextmanager
def cm_timer_1():
    start = datetime.now()
    yield
    result = datetime.now() - start
    print(result)


@contextmanager
def cm_timer_2():
    start = time()
    yield
    print('Duration: {}'.format(time() - start))


# with cm_timer_1():
#     sleep(3)

# with cm_timer_2():
#     sleep(2)
```


ab_python_fp > field.py > field

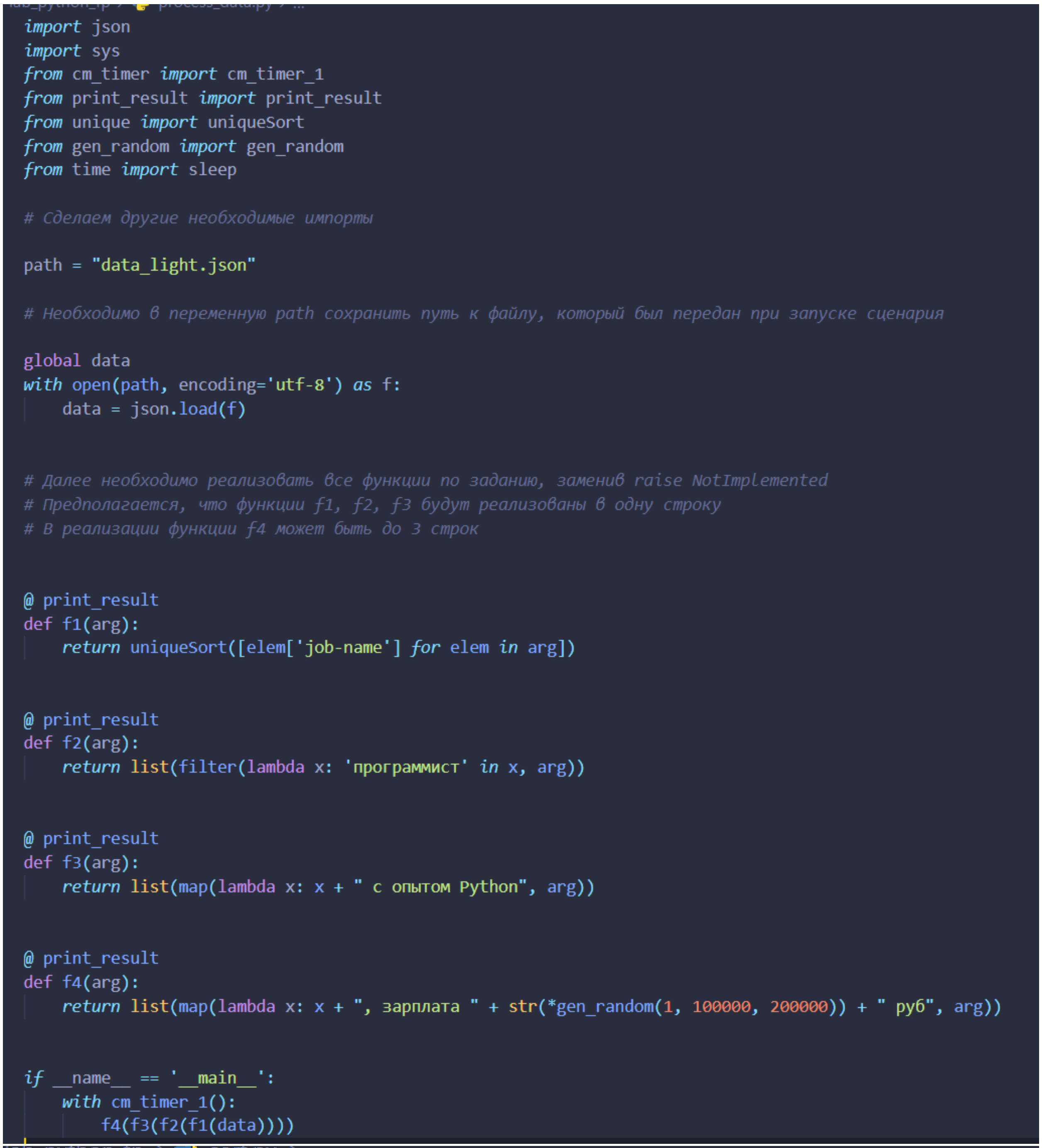
```python
goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},
    {'title': 'Плойка', 'price': 500, 'color': 'white'},
    {'title': 'Мама', 'price': 10000}
]
# field(goods, 'title') #должен выдавать 'Ковер', 'Диван для отдыха'
# field(goods, 'title', 'price') #должен выдавать {'title': 'Ковер',


def field(items, *args):
    assert len(args) > 0
    if len(args) == 1:
        index = 0
        arr = []
        for index in range(len(goods)):
            if (args[0] not in goods[index].keys()):
                continue
            arr.append(goods[index][args[0]])
            yield index
        print(", ".join(map(str, arr)))
    else:
        arr = []
        for dicts in goods:
            index = 0
            tmp = dict()
            for index in range(len(args)):
                if (args[index] not in dicts.keys()):
                    continue
                tmp[args[index]] = dicts[args[index]]
                yield index
            arr.append(tmp)
        print(", ".join(map(str, arr)))


gen = field(goods, 'price')
for i in gen:
    i
```



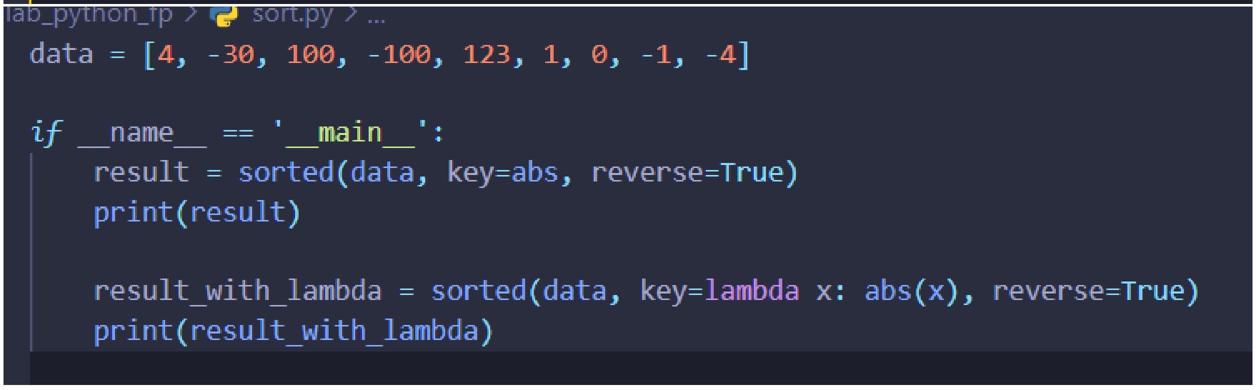
```python
from random import randint
# Пример:
# gen_random(5, 1, 3) должен выдать выдат
# в диапазоне от 1 до 3, например 2, 2, 3
# Hint: типовая реализация занимает 2 стр
global arr
arr = []


def tmp(num_count, begin, end):
    for index in range(num_count):
        arr.append(randint(begin, end))
        yield index


def gen_random(num_count, begin, end):
    arr.clear()
    gen = tmp(num_count, begin, end)
    for i in gen:
        i
    return arr


# print(gen_random(5, 1, 10))
```



```python
def print_result(func):
    def wrapper(*func_args, **func_kwargs):
        print(func.__name__)
        arr = func(*func_args, **func_kwargs)
        if isinstance(arr, int):
            print(arr)
        elif isinstance(arr, str):
            print(arr)
        elif isinstance(arr, list):
            for i in arr:
                print(i)
        elif isinstance(arr, dict):
            for key, value in arr.items():
                print(key, "=", value)
        return arr
    return wrapper


@print_result
def test_1():
    return 1


@print_result
def test_2():
    return 'iu5'


@print_result
def test_3():
    return {'a': 1, 'b': 2}


@print_result
def test_4():
    return [1, 2]


if __name__ == '__main__':
    print('!!!!!!!!')
    test_1()
    test_2()
    test_3()
    test_4()
```

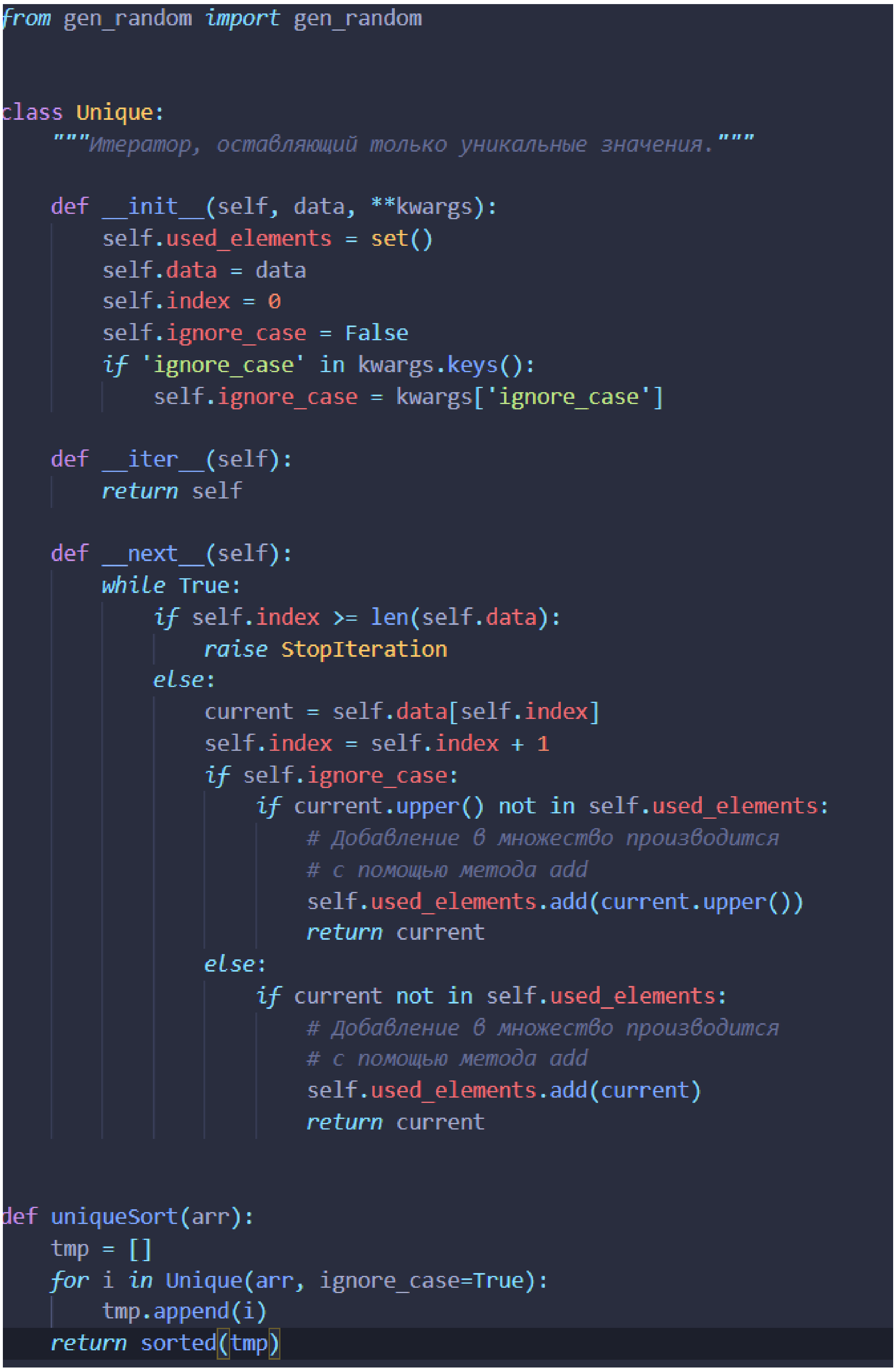


```python
import json
import sys
from cm_timer import cm_timer_1
from print_result import print_result
from unique import uniqueSort
from gen_random import gen_random
from time import sleep

# Сделаем другие необходимые импорты

path = "data_light.json"

# Необходимо в переменную path сохранить путь к файлу, который был передан при запуске сценария

global data
with open(path, encoding='utf-8') as f:
    data = json.load(f)


# Далее необходимо реализовать все функции по заданию, заменив raise NotImplemented
# Предполагается, что функции f1, f2, f3 будут реализованы в одну строку
# В реализации функции f4 может быть до 3 строк


@ print_result
def f1(arg):
    return uniqueSort([elem['job-name'] for elem in arg])


@ print_result
def f2(arg):
    return list(filter(lambda x: 'программист' in x, arg))


@ print_result
def f3(arg):
    return list(map(lambda x: x + " с опытом Python", arg))


@ print_result
def f4(arg):
    return list(map(lambda x: x + ", зарплата " + str(*gen_random(1, 100000, 200000)) + " руб", arg))


if __name__ == '__main__':
    with cm_timer_1():
        f4(f3(f2(f1(data))))
```



lab_python_fp > sort.py > ...

```python
data = [4, -30, 100, -100, 123, 1, 0, -1, -4]

if __name__ == '__main__':
    result = sorted(data, key=abs, reverse=True)
    print(result)

    result_with_lambda = sorted(data, key=lambda x: abs(x), reverse=True)
    print(result_with_lambda)
```



```python
from gen_random import gen_random


class Unique:
    """Итератор, оставляющий только уникальные значения."""

    def __init__(self, data, **kwargs):
        self.used_elements = set()
        self.data = data
        self.index = 0
        self.ignore_case = False
        if 'ignore_case' in kwargs.keys():
            self.ignore_case = kwargs['ignore_case']

    def __iter__(self):
        return self

    def __next__(self):
        while True:
            if self.index >= len(self.data):
                raise StopIteration
            else:
                current = self.data[self.index]
                self.index = self.index + 1
                if self.ignore_case:
                    if current.upper() not in self.used_elements:
                        # Добавление в множество производится
                        # с помощью метода add
                        self.used_elements.add(current.upper())
                        return current
                else:
                    if current not in self.used_elements:
                        # Добавление в множество производится
                        # с помощью метода add
                        self.used_elements.add(current)
                        return current


def uniqueSort(arr):
    tmp = []
    for i in Unique(arr, ignore_case=True):
        tmp.append(i)
    return sorted(tmp)
```