



**Министерство науки и высшего образования Российской
Федерации Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)**

**Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»**

**Лабораторная работа № 3
по дисциплине «Базовые компоненты интернет-технологий»**

**Выполнила:
студентка группы ИУ5-32Б
Андреева А. А.**

**Проверил:
Канев А.И.**

2021 г.

Содержание

Общее описание задания:.....	3
Задача 1 (файл field.py)	3
Задача 2 (файл gen_random.py).....	4
Задача 3 (файл unique.py).....	5
Задача 4 (файл sort.py)	6
Задача 5 (файл print_result.py)	7
Задача 6 (файл cm_timer.py).....	9
Задача 7 (файл process_data.py)	10

Общее описание задания:

Задание лабораторной работы состоит из решения нескольких задач.

Файлы, содержащие решения отдельных задач, должны располагаться в пакете lab_python_fr. Решение каждой задачи должно располагаться в отдельном файле.

При запуске каждого файла выдаются тестовые результаты выполнения соответствующего задания.

Задача 1 (файл field.py)

Необходимо реализовать генератор field. Генератор field последовательно выдает значения ключей словаря. Пример:

```
goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'color': 'black'}
]
field(goods, 'title') должен выдавать 'Ковер', 'Диван для отдыха'
field(goods, 'title', 'price') должен выдавать {'title': 'Ковер', 'price': 2000},
{'title': 'Диван для отдыха'}
```

- В качестве первого аргумента генератор принимает список словарей, дальше через *args генератор принимает неограниченное количество аргументов.
- Если передан один аргумент, генератор последовательно выдает только значения полей, если значение поля равно None, то элемент пропускается.
- Если передано несколько аргументов, то последовательно выдаются словари, содержащие данные элементы. Если поле равно None, то оно пропускается. Если все поля содержат значения None, то пропускается элемент целиком.

Текст программы:

```
goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
]
def field(items, *args):
    assert len(args) > 0
```

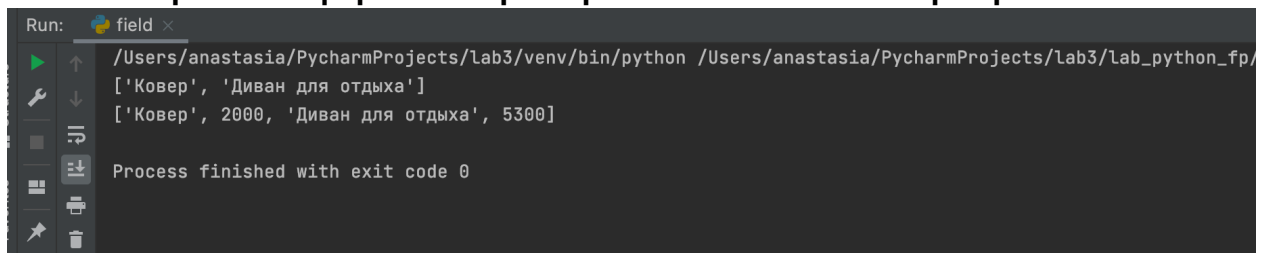
```

dict = []
j=0
for i in items:
    dict.append(i.get(args[j]))
    while j < len(args) - 1:
        j += 1
        dict.append(i.get(args[j]))
    j = 0
return dict

print(field(goods, 'title'))
print(field(goods, 'title', 'price'))

```

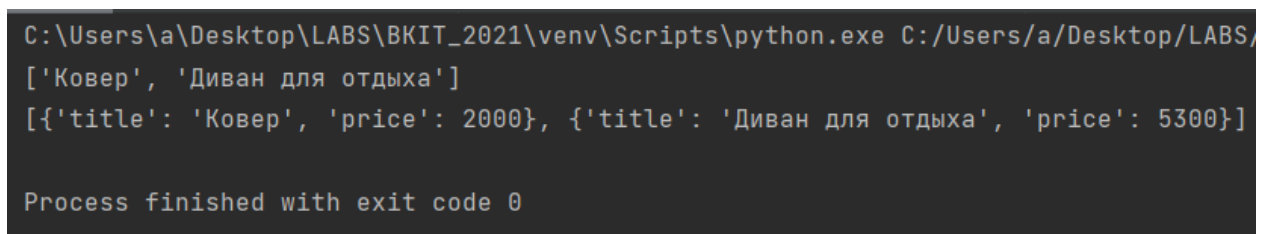
Экранные формы с примерами выполнения программы:



```

Run: field x
/Users/anastasia/PycharmProjects/lab3/venv/bin/python /Users/anastasia/PycharmProjects/lab3/lab_python_fp/
['Ковер', 'Диван для отдыха']
['Ковер', 2000, 'Диван для отдыха', 5300]
Process finished with exit code 0

```



```

C:\Users\anastasia\Desktop\LABS\BKIT_2021\venv\Scripts\python.exe C:/Users/a/Desktop/LABS/
['Ковер', 'Диван для отдыха']
[{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}]
Process finished with exit code 0

```

Задача 2 (файл gen_random.py)

Необходимо реализовать генератор gen_random(количество, минимум, максимум), который последовательно выдает заданное количество случайных чисел в заданном диапазоне от минимума до максимума, включая границы диапазона.

Пример:

gen_random(5, 1, 3) должен выдать 5 случайных чисел в диапазоне от 1 до 3, например 2, 2, 3, 2, 1

Текст программы:

```

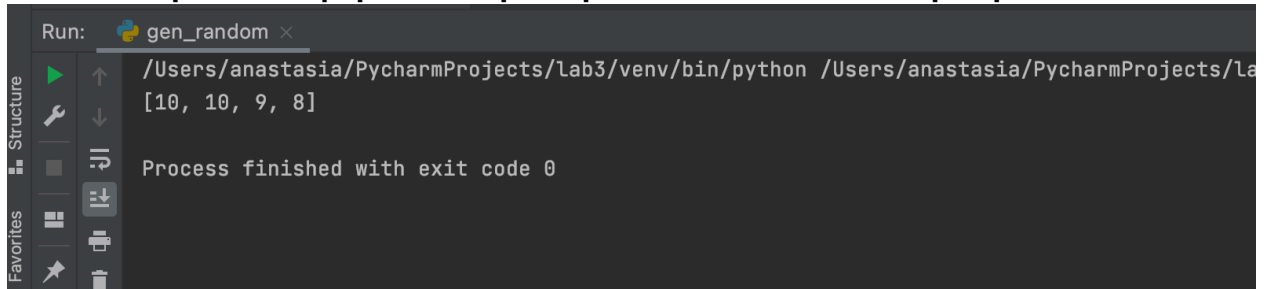
from random import randint

def gen_random(num_count, begin, end):
    for i in range(num_count):
        yield randint(begin, end)

a = gen_random(4, 8, 12)
if __name__ == '__main__':
    print(list(a))

```

Экранные формы с примерами выполнения программы:



```
Run: gen_random x
/Users/anastasia/PycharmProjects/lab3/venv/bin/python /Users/anastasia/PycharmProjects/La
[10, 10, 9, 8]

Process finished with exit code 0
```

Задача 3 (файл unique.py)

- Необходимо реализовать итератор `Unique(данные)`, который принимает на вход массив или генератор и итерируется по элементам, пропуская дубликаты.
- Конструктор итератора также принимает на вход именованный bool-параметр `ignore_case`, в зависимости от значения которого будут считаться одинаковыми строки в разном регистре. По умолчанию этот параметр равен `False`.
- При реализации необходимо использовать конструкцию `**kwargs`.
- Итератор должен поддерживать работу как со списками, так и с генераторами.
- Итератор не должен модифицировать возвращаемые значения.

Пример:

```
data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
```

`Unique(data)` будет последовательно возвращать только 1 и 2.

```
data = gen_random(1, 3, 10)
```

`Unique(data)` будет последовательно возвращать только 1, 2 и 3.

```
data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
```

`Unique(data)` будет последовательно возвращать только a, A, b, B.

`Unique(data, ignore_case=True)` будет последовательно возвращать только a, b.

Текст программы:

```
# Итератор для удаления дубликатов
from lab_python_fp.gen_random import gen_random
class Unique(object):
    def __init__(self, items, ignore_case = False, **kwargs):
        self.seen = set()
        self.it = iter(items)
        self.ignore_case = ignore_case
```

```

def __iter__(self):
    return self

def __next__(self):
    while True:
        current = next(self.it)

        if self.ignore_case:
            if isinstance(current, str) and current.lower() not in self.seen:
                self.seen.add(current.lower())
                return current


        if current not in self.seen:
            self.seen.add(current)
            return current

data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
data_gen = gen_random(5, 3, 10)
data_abc = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']

print(list(Unique(data)))
print(list(Unique(data_gen)))
print(list(Unique(data_abc, ignore_case = True)))

```

Экранные формы с примерами выполнения программы:



```

unique x
/Users/anastasia/PycharmProjects/lab3/venv/bin/python /Users/anastasia/Pychar
[1, 2]
[5, 8, 3, 4, 9]
['a', 'A', 'b', 'B']

Process finished with exit code 0

```

Задача 4 (файл sort.py)

Дан массив 1, содержащий положительные и отрицательные числа.

Необходимо **одной строкой кода** вывести на экран массив 2, которые содержит значения массива 1, отсортированные по модулю в порядке убывания. Сортировку необходимо осуществлять с помощью функции sorted. Пример:

```

data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]
Вывод: [123, 100, -100, -30, 30, 4, -4, 1, -1, 0]

```

Необходимо решить задачу двумя способами:

1. С использованием lambda-функции.
2. Без использования lambda-функции.

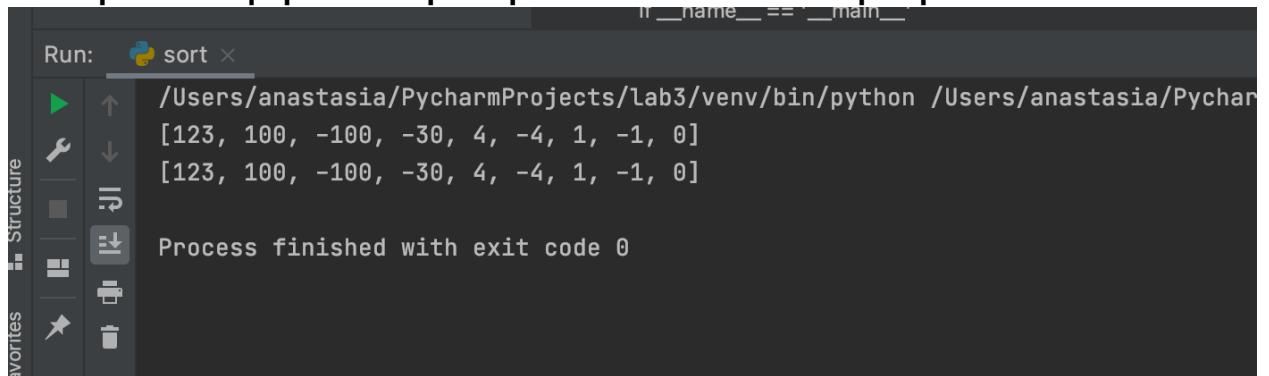
Текст программы:

```
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)
```

Экранные формы с примерами выполнения программы:



Задача 5 (файл print_result.py)

Необходимо реализовать декоратор print_result, который выводит на экран результат выполнения функции.

- Декоратор должен принимать на вход функцию, вызывать её, печатать в консоль имя функции и результат выполнения, после чего возвращать результат выполнения.
- Если функция вернула список (list), то значения элементов списка должны выводиться в столбик.
- Если функция вернула словарь (dict), то ключи и значения должны выводиться в столбик через знак равенства.

Результат выполнения:

```
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
```

Текст программы:

```
def print_result(func):
    def wrapper(*args, **kwargs):
        res = func(*args, **kwargs)
        if type(res) == list: print(*res, sep='\n')
        elif type(res) == dict: [print(key, '=', value) for key, value in
res.items()]
        else: print(res)

        return res

    return wrapper

@print_result
def test_1():
    return 1

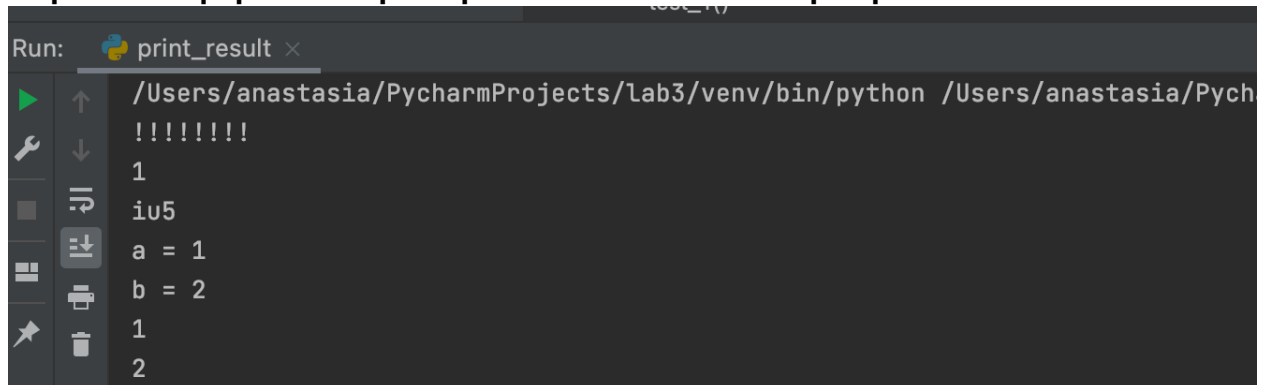
@print_result
def test_2():
    return 'iu5'

@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()
```

Экранные формы с примерами выполнения программы:



```
Run: print_result x
/Users/anastasia/PycharmProjects/Lab3/venv/bin/python /Users/anastasia/Pych
!!!!!!!
1
iu5
a = 1
b = 2
1
2
```


Задача 6 (файл cm_timer.py)

Необходимо написать контекстные менеджеры cm_timer_1 и cm_timer_2, которые считают время работы блока кода и выводят его на экран. Пример:

```
with cm_timer_1():  
    sleep(5.5)
```

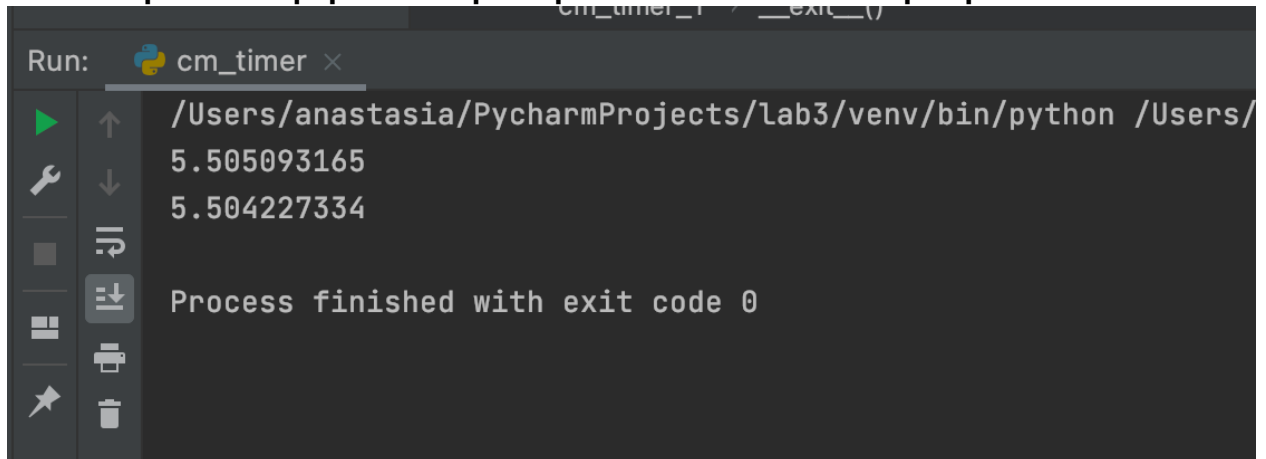
После завершения блока кода в консоль должно вывестись time: 5.5 (реальное время может несколько отличаться).

cm_timer_1 и cm_timer_2 реализуют одинаковую функциональность, но должны быть реализованы двумя различными способами (на основе класса и с использованием библиотеки contextlib).

Текст программы:

```
import time  
from contextlib import contextmanager  
from time import sleep  
  
class cm_timer_1():  
    def Begin(self):  
        self.begin = time.perf_counter()  
    def End(self):  
        t = time.perf_counter() - self.begin  
        print(t)  
    def __enter__(self):  
        self.Begin()  
    def __exit__(self, *args):  
        self.End()  
  
@contextmanager  
def cm_timer_2():  
    try:  
        begin = time.perf_counter()  
        yield begin  
    finally:  
        print(time.perf_counter()-begin)  
  
with cm_timer_1():  
    sleep(5.5)  
  
with cm_timer_2():  
    sleep(5.5)
```

Экранные формы с примерами выполнения программы:



```
Run: cm_timer x
/Users/anastasia/PycharmProjects/lab3/venv/bin/python /Users/
5.505093165
5.504227334
Process finished with exit code 0
```

Задача 7 (файл process_data.py)

- В предыдущих задачах были написаны все требуемые инструменты для работы с данными. Применим их на реальном примере.
- В файле [data_light.json](#) содержится фрагмент списка вакансий.
- Структура данных представляет собой список словарей с множеством полей: название работы, место, уровень зарплаты и т.д.
- Необходимо реализовать 4 функции - f1, f2, f3, f4. Каждая функция вызывается, принимая на вход результат работы предыдущей. За счет декоратора @print_result печатается результат, а контекстный менеджер cm_timer_1 выводит время работы цепочки функций.
- Предполагается, что функции f1, f2, f3 будут реализованы в одну строку. В реализации функции f4 может быть до 3 строк.
- Функция f1 должна вывести отсортированный список профессий без повторений (строки в разном регистре считать равными). Сортировка должна игнорировать регистр. Используйте наработки из предыдущих задач.
- Функция f2 должна фильтровать входной массив и возвращать только те элементы, которые начинаются со слова "программист". Для фильтрации используйте функцию filter.
- Функция f3 должна модифицировать каждый элемент массива, добавив строку "с опытом Python" (все программисты должны быть знакомы с Python). Пример: Программист C# с опытом Python. Для модификации используйте функцию map.

- Функция f4 должна сгенерировать для каждой специальности зарплату от 100 000 до 200 000 рублей и присоединить её к названию специальности. Пример: Программист C# с опытом Python, зарплата 137287 руб. Используйте zip для обработки пары специальность — зарплата.

Текст программы:

```
import json
import sys
from lab_python_fp.field import field
from lab_python_fp.gen_random import gen_random
from lab_python_fp.unique import Unique
from lab_python_fp.print_result import print_result
from lab_python_fp.cm_timer import cm_timer_1

from re import search

path = '../data_light.json'

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

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

@print_result
def f1(arg):
    p1 = Unique(list(sorted(field(arg, 'job-name'), key=str.casefold)))
    return p1

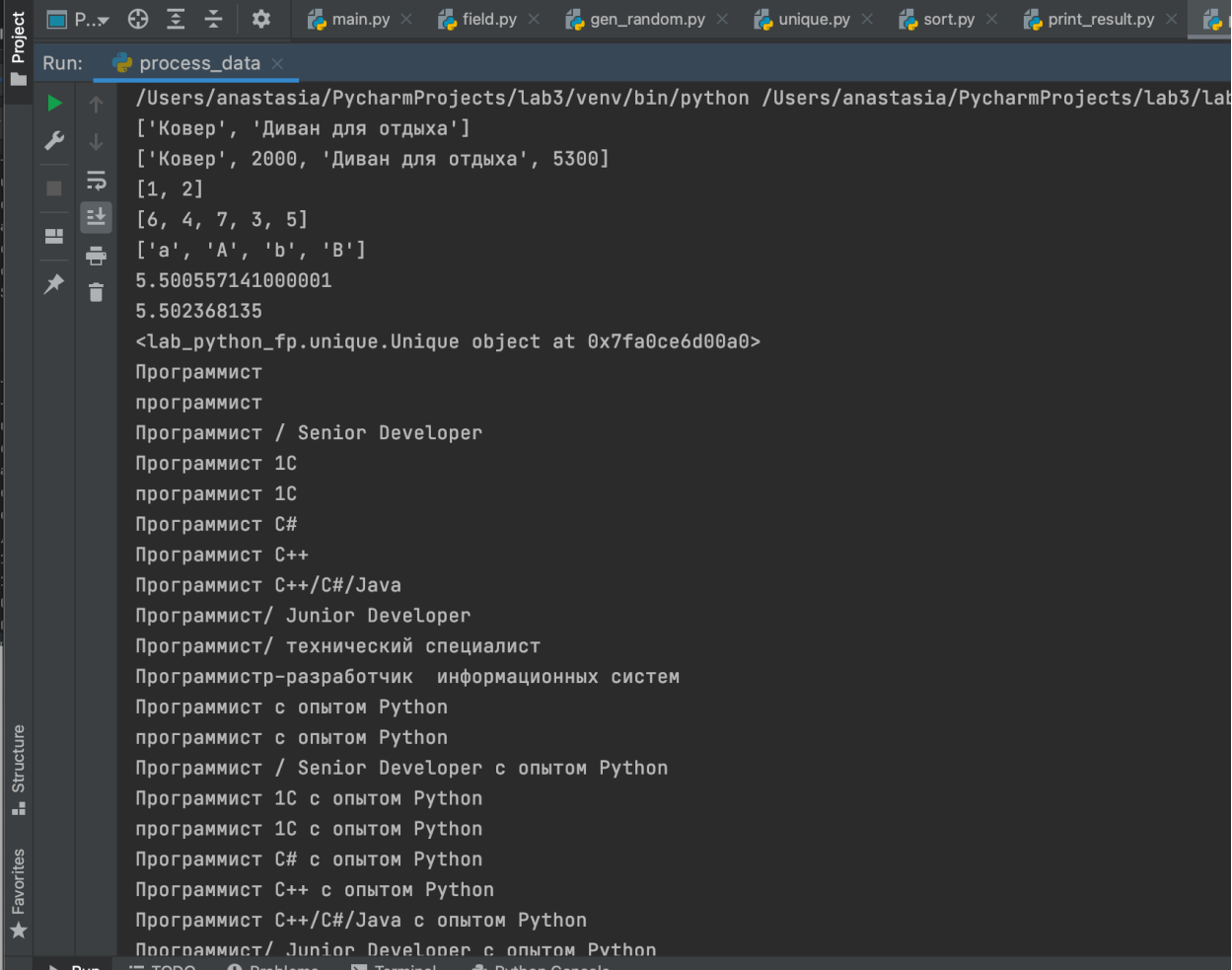
@print_result
def f2(arg):
    p2 = list(filter(lambda k: search('^программист', k) != None or
search('^Программист', k) != None, arg))
    return p2

@print_result
def f3(arg):
    p3 = list(map(lambda k: k + ' с опытом Python', arg))
    return p3

@print_result
def f4(arg):
    p = len(list(filter(lambda k: search('^программист', k) != None or
search('^Программист', k) != None, arg)))
    p4 = list(zip(arg, gen_random(p, 100000, 200000)))
    return p4

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

Экранные формы с примерами выполнения программы:



```
Run: process_data x
/Users/anastasia/PycharmProjects/Lab3/venv/bin/python /Users/anastasia/PycharmProjects/Lab3/Lab3/process_data.py
['Ковер', 'Диван для отдыха']
['Ковер', 2000, 'Диван для отдыха', 5300]
[1, 2]
[6, 4, 7, 3, 5]
['a', 'A', 'b', 'B']
5.500557141000001
5.502368135
<lab_python_fp.unique.Unique object at 0x7fa0ce6d00a0>
Программист
программист
Программист / Senior Developer
Программист 1C
программист 1C
Программист C#
Программист C++
Программист C++/C#/Java
Программист/ Junior Developer
Программист/ технический специалист
Программист-разработчик информационных систем
Программист с опытом Python
программист с опытом Python
Программист / Senior Developer с опытом Python
Программист 1C с опытом Python
программист 1C с опытом Python
Программист C# с опытом Python
Программист C++ с опытом Python
Программист C++/C#/Java с опытом Python
Программист/ Junior Developer с опытом Python
Программист/ технический специалист с опытом Python
Программист-разработчик информационных систем с опытом Python
('Программист с опытом Python', 136392)
('программист с опытом Python', 162190)
('Программист / Senior Developer с опытом Python', 181058)
('Программист 1C с опытом Python', 164526)
('программист 1C с опытом Python', 120550)
('Программист C# с опытом Python', 151208)
('Программист C++ с опытом Python', 111980)
('Программист C++/C#/Java с опытом Python', 127267)
('Программист/ Junior Developer с опытом Python', 110287)
('Программист/ технический специалист с опытом Python', 105037)
('Программист-разработчик информационных систем с опытом Python', 197294)
0.011532121000000117

Process finished with exit code 0
```