

**Московский государственный технический
университет им. Н.Э. Баумана**

Факультет «Радиотехнический»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Разработка интернет-приложений»

Отчет по лабораторной работе №3
«Функциональные возможности языка Python»

Выполнил:
студент группы РТ5-51Б
Борисочкин М.И.
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Балашов А. М.
Подпись и дата:

Москва, 2021 г.

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.

- 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': 'Ковёр', 'price': 2000, 'color': 'green'},
          {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},
          {'title': 'Рабочий стол', 'price': None, 'color': None},
          {'title': None, 'price': 1000, 'color': 'blue'},
          {'price': 1999, 'color': None},
          {'title': None, 'price': None, 'color': None}
        ]

def field(items, *args):
    """
    Генератор для работы со списками словарей

    :param items: Список словарей
    :param args: Ключи словарей
    :return: Значения полей при одном аргументе или словари при нескольких
    """
    assert len(args) > 0
```

```

# Случай одного аргумента
if len(args) == 1:
    return (item[args[0]] for item in items if args[0] in item.keys() and
            item[args[0]] is not None)

    return [{key: value for key, value in item.items() if value is not None}
            for item in items if args <= tuple(item.keys()) and any(v is not
            None for v in item.values())]

def main():
    for i in field(goods, 'title'):
        print(i)

    print()

    for i in field(goods, 'price', 'color'):
        print(i)

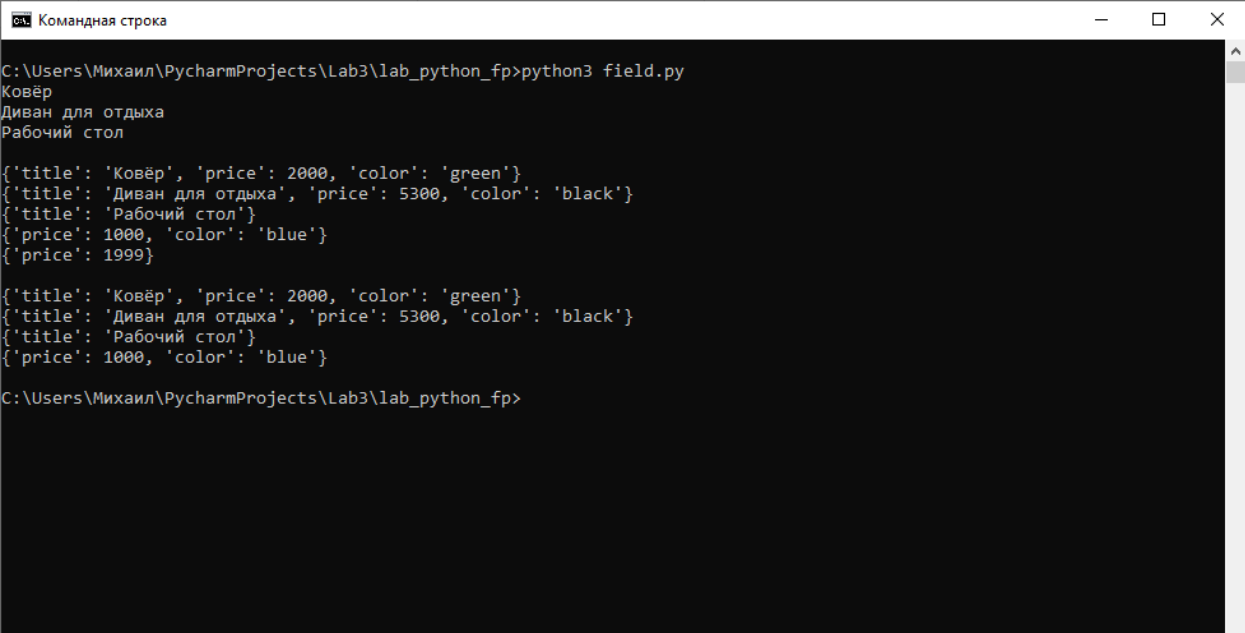
    print()

    for i in field(goods, 'title', 'price', 'color'):
        print(i)

if __name__ == '__main__':
    main()

```

Пример выполнения программы



```

Командная строка
C:\Users\Михаил\PycharmProjects\Lab3\lab_python_fp>python3 field.py
Ковёр
Диван для отдыха
Рабочий стол
{'title': 'Ковёр', 'price': 2000, 'color': 'green'}
{'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
{'title': 'Рабочий стол'}
{'price': 1000, 'color': 'blue'}
{'price': 1999}
{'title': 'Ковёр', 'price': 2000, 'color': 'green'}
{'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
{'title': 'Рабочий стол'}
{'price': 1000, 'color': 'blue'}
C:\Users\Михаил\PycharmProjects\Lab3\lab_python_fp>

```

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

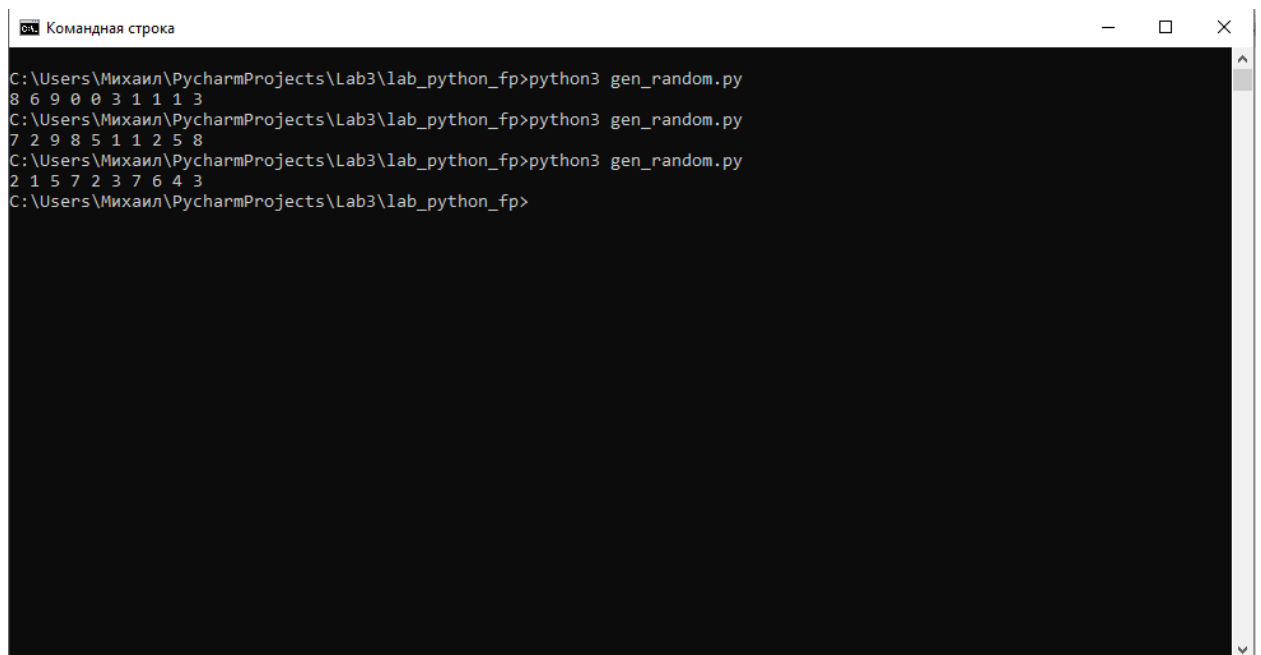
def gen_random(quantity, min_value, max_value):
    """
    Генератор последовательности случайных чисел

    :param quantity: Длина последовательности
    :param min_value: Минимальное значение числа в последовательности
    :param max_value: Максимальное значение числа в последовательности
    :return: Последовательность случайных чисел
    """
    return (randint(min_value, max_value) for _ in range(quantity))

def main():
    for i in gen_random(10, 0, 9):
        print(i, end=" ")

if __name__ == '__main__':
    main()
```

Пример выполнения программы



```
Командная строка
C:\Users\Михаил\PycharmProjects\Lab3\lab_python_fp>python3 gen_random.py
8 6 9 0 0 3 1 1 1 3
C:\Users\Михаил\PycharmProjects\Lab3\lab_python_fp>python3 gen_random.py
7 2 9 8 5 1 1 2 5 8
C:\Users\Михаил\PycharmProjects\Lab3\lab_python_fp>python3 gen_random.py
2 1 5 7 2 3 7 6 4 3
C:\Users\Михаил\PycharmProjects\Lab3\lab_python_fp>
```

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.

Текст программы

```
from gen_random import gen_random

class Unique(object):
    """ Итератор для удаления дубликатов """
    def __init__(self, items, **kwargs):
        self.used_elements = set()
        self.data = items
        self.index = 0

        # Установка флага ignore_case
        if 'ignore_case' not in kwargs:
            self.ignore_case = False
        else:
            self.ignore_case = kwargs['ignore_case']

    def __next__(self):
        while True:
            if self.index >= len(self.data):
                raise StopIteration
            else:
                current = self.data[self.index]
                self.index = self.index + 1

                # Проверка флага ignore_case
                if self.ignore_case and current.lower() not in self.used_elements:
                    self.used_elements.add(current.lower())
                    return current
                elif not self.ignore_case and current not in self.used_elements:
                    self.used_elements.add(current)
                    return current

    def __iter__(self):
        return self

def main():
    data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]

    for value in Unique(data):
        print(value, end=" ")
    print()

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

```

for value in data:
    print(value, end=" ")
print()

for value in Unique(data):
    print(value, end=" ")
print()

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

for value in Unique(data):
    print(value, end=" ")
print()

for value in Unique(data, ignore_case=True):
    print(value, end=" ")
print()

if __name__ == '__main__':
    main()

```

Пример выполнения программы

```

C:\Users\Михаил\PycharmProjects\Lab3\lab_python_fp>python3 unique.py
1 2
3 1 2 1 2 2 3 3 3
3 1 2
a A b B
a b
C:\Users\Михаил\PycharmProjects\Lab3\lab_python_fp>_

```

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`.

Необходимо решить задачу двумя способами:

- С использованием `lambda`-функции.
- 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 value: abs(value),
reverse=True)
    print(result_with_lambda)
```

Пример выполнения программы

```
Командная строка
C:\Users\Михаил\PycharmProjects\Lab3\lab_python_fp>python3 sort.py
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]
C:\Users\Михаил\PycharmProjects\Lab3\lab_python_fp>
```

Задача 5 (файл print_result.py)

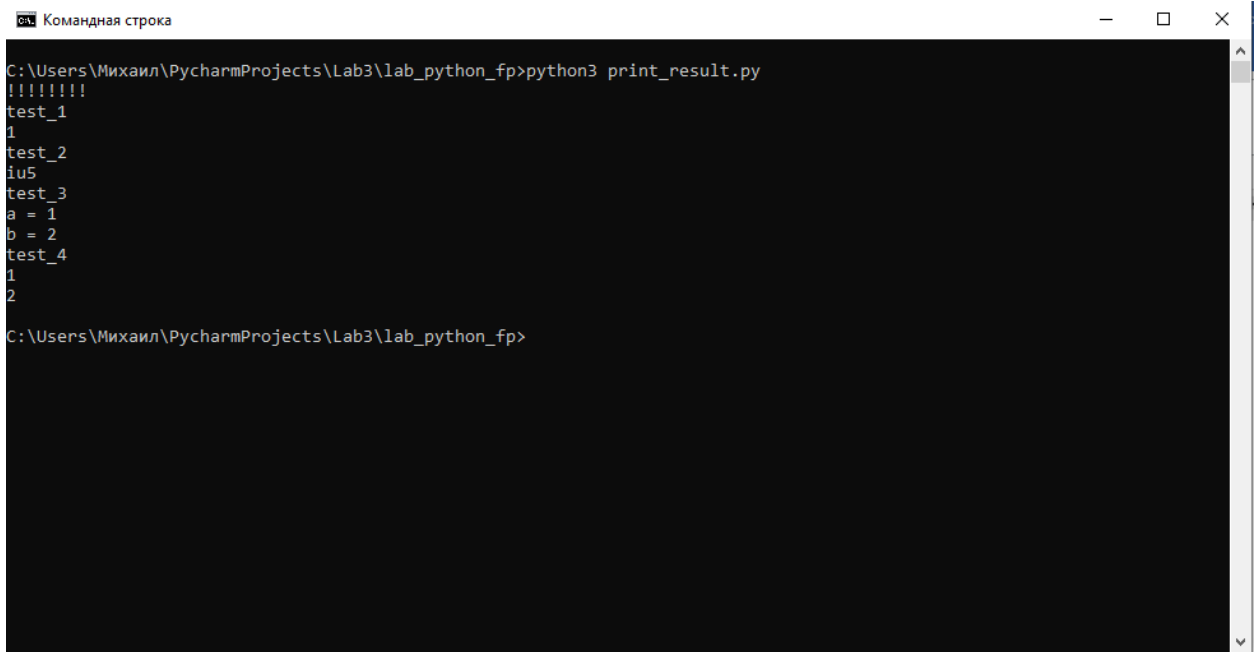
Необходимо реализовать декоратор `print_result`, который выводит на экран результат выполнения функции.

- 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.

Текст программы

```
def print_result(func):  
    """  
    Декоратор для вывода результата функций  
  
    :param func: Функция  
    :return: Результат функции  
    """  
    def wrapper(*args):  
        result = func(*args)  
        print(func.__name__)  
  
        if type(result) == list:  
            for value in result:  
                print(value)  
        elif type(result) == dict:  
            for key, value in result.items():  
                print("{} = {}".format(key, value))  
        else:  
            print(result)  
  
        return result  
  
    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()
```


Пример выполнения программы



```
Командная строка
C:\Users\Михаил\PycharmProjects\Lab3\lab_python_fp>python3 print_result.py
!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
C:\Users\Михаил\PycharmProjects\Lab3\lab_python_fp>
```

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

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

class cm_timer_1:
    """Контекстный менеджер на основе класса"""
    def __init__(self):
        self.start_time = None
        self.finish_time = None

    def __enter__(self):
        self.start_time = time()

    def __exit__(self, exc_type, exc_val, exc_tb):
        self.finish_time = time()
        print("Время: {}".format(self.finish_time - self.start_time))
```

```

@contextmanager
def cm_timer_2():
    """Контекстный менеджер на основе библиотеки contextlib"""
    start_time = time()
    yield
    finish_time = time()
    print("Время: {}".format(finish_time - start_time))

if __name__ == '__main__':
    with cm_timer_1():
        sleep(5.5)

    with cm_timer_2():
        sleep(5.5)

```

Пример выполнения программы

```

C:\Users\Михаил\PycharmProjects\Lab3\lab_python_fp>python3 cm_timer.py
Время: 5.5075743198394775
Время: 5.511022329330444
C:\Users\Михаил\PycharmProjects\Lab3\lab_python_fp>

```

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
from field import field
from gen_random import gen_random
from unique import Unique
from print_result import print_result
from cm_timer import cm_timer_1

path = '../data_light.json'

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

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

@print_result
def f2(arg):
    return list(filter(lambda value: value.lower().startswith('программист'),
arg))

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

@print_result
def f4(arg):
    salary = list(gen_random(len(arg), 100000, 200000))
    return list(map(lambda value: value[0] + ', зарплата: ' + str(value[1]) +
```

```
' pyб', list(zip(arg, salary)))

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

Пример выполнения программы

```
Командная строка
C:\Users\Михаил\PycharmProjects\Lab3\lab_python_fp>python3 process_data.py
f1
1С программист
2-ой механик
3-ий механик
4-ый механик
4-ый электромеханик
[химик-эксперт
ASIC специалист
JavaScript разработчик
RTL специалист
Web-программист
web-разработчик
Автожестянщик
Автоинструктор
Автомаляр
Автомойщик
Автор студенческих работ по различным дисциплинам
автослесарь
Автослесарь - моторист
Автоэлектрик
Агент
Агент банка
Агент нпф
Агент по гос. закупкам недвижимости
Агент по недвижимости
Агент по недвижимости (стажер)
Агент по недвижимости / Риэлтор
Агент по привлечению юридических лиц
Агент по продажам (интернет, ТВ, телефония) в ПАО Ростелеком в населенных пунктах Амурской области: г. Благовещенск, г.
Белогорск, г. Свободный, г. Шимановск, г. Зея, г. Тында
Агент торговый
агрегатчик-топливник KOMATSU
агроном
агроном по защите растений
Агроном-полевод
агрохимик почвовед
Администратор
Администратор (удаленно)
Администратор Active Directory
Администратор в парикмахерский салон
Администратор зала (предприятий общественного питания)
Администратор кофейни
Администратор на ресепшен
Администратор на телефоне
Администратор по информационной безопасности
Администратор ресторана
Администратор сайта
Администратор ярмарок выходного дня
Администратор-кассир
Аккомпаниатор на 0,5 ст.
аккумуляторщик 4 разряда
Акушерка
акушерка в родильное отделение
Акушерка женской консультации
Акушерка Лысогорская врачебная амбулатория
Акушерка ФАП
Акушерка, АО
Акушерка, ВП
```

Командная строка

Альпинист промышленный
Аналитик
Анестезиолог - реаниматолог
анестезиолог-реаниматолог
анестезиолог-реаниматолог детский
аниматор
антенщик-мачтовик 4 разряда
аппаратчик обработки зерна
Аппаратчик обработки зерна 5 разряда
Аппаратчик пастеризации
Аппаратчик установки опытного производства
Аппаратчик химодоочистки
Арматурщик
арматурщик кузовного цеха
арматурщики
Артист (кукловод) театра кукол
Артист оркестра
Артист отдела социально - культурной деятельности Районного ЦНК
Артист хора
артист(кукловод) театра кукол
Артист-вокалист (солист)
артист-кукловод
архивариус
Архивариус (Орехово-Зуевский филиал)
Архитектор, картограф, инженер-проектировщик
Ассистент главы отделения
Ассистент отдела продаж
Ассистент режиссера
балетмейстер-постановщик
Бармен

Командная строка

Бармен-кассир в кафе
Бармен-официант
Бетонщик
Бетонщик (на срубку свай)
Бетонщик - арматурщик
Бетонщик-монолитчик
Библиограф
Библиотекарь
Библиотекарь отдела абонемента
Биолог
Боец скота
Бригадир в животноводстве
бригадир животноводства
бригадир мобильной бригады
Бригадир технического обслуживания газоиспользующего оборудования
Бригадир, производитель работ
Брокер коммерческой недвижимости
Брошюровщик
Бухгалтер
Бухгалтер (по заработной плате)
Бухгалтер 2 категории
Бухгалтер на группу "Обработка первичной документации"
Бухгалтер по ведению первичной документации
Бухгалтер по заработной плате
Бухгалтер по МТП и ГСМ
Бухгалтер по начислению заработной платы
Бухгалтер по расчету заработной платы
Бухгалтер по расчету калькуляции
Бухгалтер, ведущий
Бухгалтер, Ведущий бухгалтер

```
Командная строка
Электросварщик
Электросварщик на полуавтомат
Электросварщик на автоматических и полуавтоматических машинах
Электросварщик ручной сварки
Электросварщики ручной сварки
Электрослесарь (слесарь) дежурный и по ремонту оборудования, старший
Электрослесарь по ремонту и обслуживанию автоматики и средств измерений электростанций
Электрослесарь по ремонту оборудования в карьере
Электроэрозионист
Эндокринолог
Энергетик
Энергетик литейного производства
Энтомолог
Юрисконсульт
Юрисконсульт 2 категории
Юрисконсульт. Контрактный управляющий
Юрист
Юрист (специалист по сопровождению международных договоров, английский - разговорный)
Юрист волонтер
Юристконсульт
f2
Программист
Программист / Senior Developer
Программист 1C
Программист C#
Программист C++
Программист C++/C#/Java
Программист/ Junior Developer
Программист/ технический специалист
Программист-разработчик информационных систем
```

```
Командная строка
Программист C#
Программист C++
Программист C++/C#/Java
Программист/ Junior Developer
Программист/ технический специалист
Программист-разработчик информационных систем
f3
Программист с опытом Python
Программист / Senior Developer 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
f4
Программист с опытом Python, зарплата: 117248 руб
Программист / Senior Developer с опытом Python, зарплата: 178975 руб
Программист 1C с опытом Python, зарплата: 193734 руб
Программист C# с опытом Python, зарплата: 119207 руб
Программист C++ с опытом Python, зарплата: 168062 руб
Программист C++/C#/Java с опытом Python, зарплата: 131548 руб
Программист/ Junior Developer с опытом Python, зарплата: 110795 руб
Программист/ технический специалист с опытом Python, зарплата: 142706 руб
Программист-разработчик информационных систем с опытом Python, зарплата: 126133 руб
Время: 0.6514537334442139
C:\Users\Михаил\PycharmProjects\Lab3\lab_python_fp>
```