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 «Радиотехнический»

Кафедра «Системы обработки информации и управления»

Лабораторная работа № 3

по дисциплине «Разработка интернет-приложений».

Выполнил:
студент(ка) группы № РТ5-51Б
А. С. Пакало
подпись, дата

Проверил:
преподаватель
Ю. Е. Гапанюк
подпись, дата

Москва, 2021

Оглавление

Цель работы

Изучение возможностей функционального программирования в языке Python.

Задание

Задача 1 (файл field.py)

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

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.

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.

Задача 4 (файл sort.py)

Дан массив 1, содержащий положительные и отрицательные числа. Необходимо *одной строкой кода* вывести на экран массив 2, который содержит значения массива 1, отсортированные по модулю в порядке убывания. Сортировку необходимо осуществлять с помощью функции sorted.

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 выводить в столбик через знак равенства.

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.

Задача 7 (файл process_data.py)

В предыдущих задачах были написаны все требуемые инструменты для работы с данными. Необходимо применить их на реальном примере.

В файле https://github.com/iu5team/iu5web-fall-2021/tree/main/notebooks/fp/files/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.

Выполнение

main.py

```
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.sort import sort
from lab_python_fp.print_task import print_task
from lab_python_fp.print_result import print_result
from lab_python_fp.cm_timer1 import CmTimer
from lab_python_fp.cm_timer2 import cm_timer2
from lab_python_fp.process_data import process_data

@print_task
def task1() -> None:
    goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'color': 'black'}
    ]

    print('Titles:')
    for good in field(goods, 'title'):
```

```
        print(good)

    print()

    print('Prices:')
    for good in field(goods, 'price'):
        print(good)

    print()

    print('Titles and prices:')
    for good in field(goods, 'title', 'price'):
        print(good)


@print_task
def task2() -> None:
    for r in gen_random(5, 1, 3):
        print(r)


@print_task
def task3() -> None:
    data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    print('case sensitive:')
    for d in Unique(data):
        print(d)

    print('case insensitive:')
    for d in Unique(data, True):
        print(d)


@print_task
def task4() -> None:
    data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
    sort(data)

@print_task
```

```
def task5() -> None:
    @print_result
    def test_primitive_1():
        return 1


    @print_result
    def test_primitive_2():
        return 'iu5'


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


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


    test_primitive_1()
    test_primitive_2()
    test_dictionary()
    test_list()

@print_task
def task6() -> None:
    num_of_iterations = 10_000_000
    def test_function() -> None:
        for i in range(num_of_iterations):
            continue

    print(f'{num_of_iterations:,.0f} iterations were completed in:')
    with CmTimer() as test_timer1:
        test_function()
    print('(class implementation)')
    print()
```

```
    with cm_timer2() as test_timer2:
        test_function()
    print('(contextlib @contextmanager decorator implementation)')

@print_task
def task7() -> None:
    path = '../../notebooks/fp/files/data_light.json'
    process_data(path)


def main() -> None:
    task1()
    task2()
    task3()
    task4()
    task5()
    task6()
    task7()


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

Задание 1 (файл field.py)

```
def field(items, *selected_fields):
    num_of_selected_fields = len(selected_fields)
    assert num_of_selected_fields > 0, 'No fields for selection!
Please pass at least one!'
    if (num_of_selected_fields == 1):
        return (item.get(selected_fields[0]) for item in items
                if item.get(selected_fields[0]) != None)

    return (
        {sf:item.get(sf) for sf in selected_fields
            if item.get(sf) != None}
        for item in items
    )
```

Задание 2 (файл gen_random.py)

```
from random import randrange

def gen_random(num_count, begin, end):
    for i in range(num_count):
         # randrange works in [begin, end) diapason.
        yield randrange(begin, end + 1)
```

Задание 3 (файл unique.py)

```
# Iterating through unique values of a data.
class Unique:
    def __init__(self, data, ignore_case=False):
        self.used_elements = set()
        self.data = list(data)
        self.index = 0
        self.ignore_case = 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):
                    current_lowered = current.lower()
                    if current_lowered not in self.used_elements:
                        self.used_elements.add(current_lowered)
                        return current
                else:
                    if current not in self.used_elements:
                        self.used_elements.add(current)
```

```
                        return current
```

Задание 4 (файл sort.py)

```
# Iterating through unique values of a data.
class Unique:
    def __init__(self, data, ignore_case=False):
        self.used_elements = set()
        self.data = list(data)
        self.index = 0
        self.ignore_case = 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):
                    current_lowered = current.lower()
                    if current_lowered not in self.used_elements:
                        self.used_elements.add(current_lowered)
                        return current
                else:
                    if current not in self.used_elements:
                        self.used_elements.add(current)
                        return current
```

Задание 5 (файл print_result.py)

```
# Iterating through unique values of a data.
class Unique:
    def __init__(self, data, ignore_case=False):
        self.used_elements = set()
```

```
        self.data = list(data)
        self.index = 0
        self.ignore_case = 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):
                    current_lowered = current.lower()
                    if current_lowered not in self.used_elements:
                        self.used_elements.add(current_lowered)
                        return current
                else:
                    if current not in self.used_elements:
                        self.used_elements.add(current)
                        return current
```

Задание 6 (файл cm_timer.py)

```
from time import time
from contextlib import contextmanager


class CmTimer:
    def __init__(self):
        self._start_time = None

    def __enter__(self):
        self._start_time = time()
        return

    def __exit__(self, exp_type, exp_value, traceback):
        if exp_type is not None:
```

```
            print(exp_type, exp_value, traceback)
        else:
            print('--------------------')
            print(time() - self._start_time)
            print('--------------------')


@contextmanager
def cm_timer2():
    start_time = time()
    yield
    print('--------------------')
    print(time() - start_time)
    print('--------------------')
```

Задание 7 (файл process_data.py)

```
from json import load
from lab_python_fp.print_result import print_result
from lab_python_fp.unique import Unique
from lab_python_fp.field import field
from lab_python_fp.cm_timer2 import cm_timer2
from lab_python_fp.gen_random import gen_random


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



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


@print_result
def f3(programmers):
    return list(map(lambda programmer: f'{programmer} с опытом
Python', programmers))
```

```
@print_result
def f4(vacancies):
    salaries = list(gen_random(len(vacancies), 100_000, 200_000))
    return list(map(lambda v: f'{v[0]}, запрлата {v[1]} руб.',
zip(vacancies, salaries)))


def process_data(path) -> None:
    with open(path, encoding='utf8') as f:
        data = load(f)

    with cm_timer2() as process_data_timer:
        f4(f3(f2(f1(data))))
```

Результаты выполнения

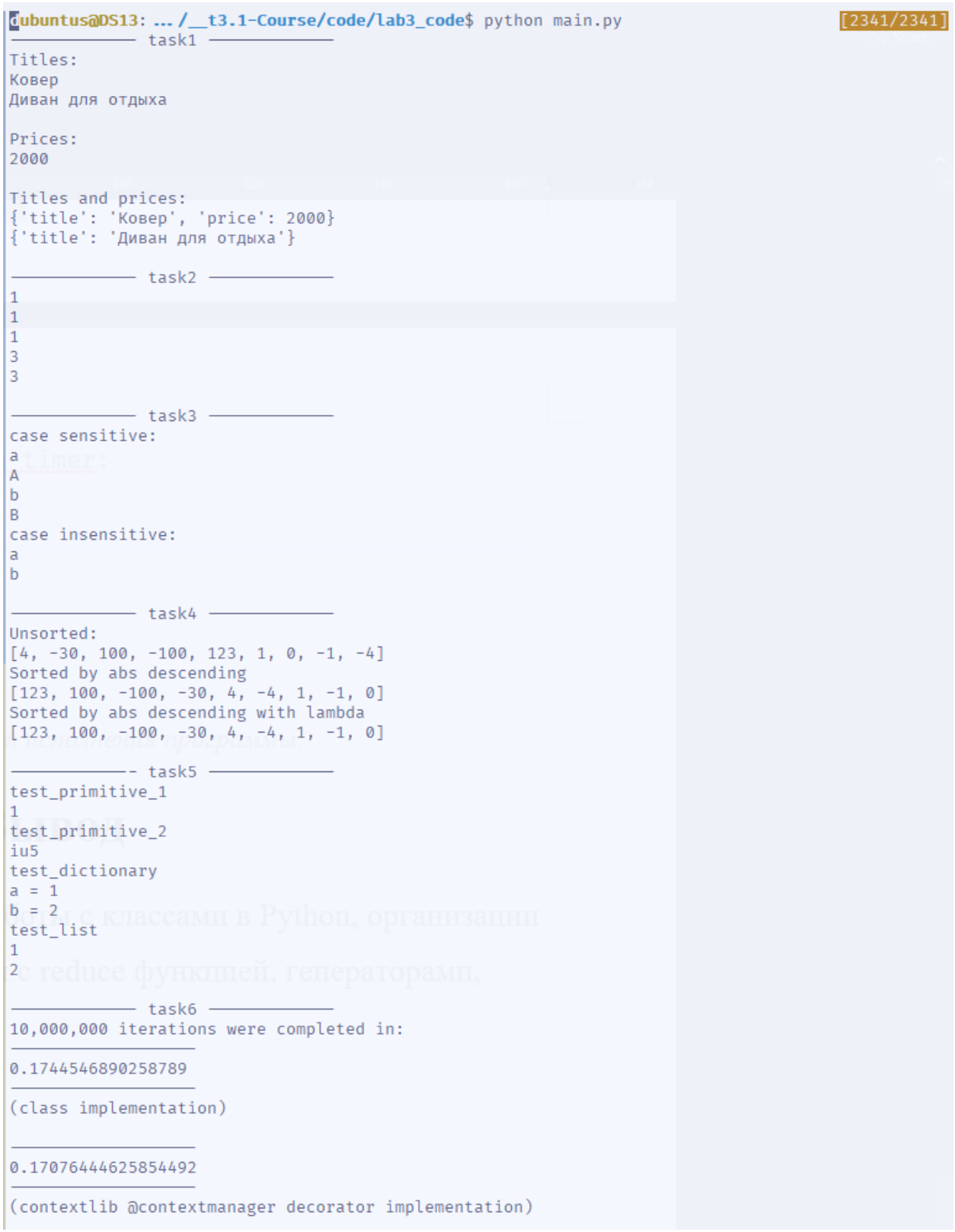

рис. 1 результат выполнения заданий 1-6

```
————————— task7 —————————
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 обла
сти: 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
```

рис. 2 начало результата выполнения функции f1 задания 7

```
Электромонтер по ремонту и обслуживанию электрооборудования 4 разряда-5 разряда        [0/2341]
Электромонтер по ремонту и обслуживанию электрооборудования 5 р.
Электромонтер по ремонту и обслуживанию электрооборудования 5 разряда
Электромонтер по ремонту и обслуживанию электрооборудования 6 разряда-6 разряда
Электромонтер по ремонту оборудования ЗИФ
Электромонтер по ремонту электрооборудования ГПМ
Электромонтер по эксплуатации и ремонту оборудования
электромонтер станционного телевизионного оборудования
Электронщик
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 1С
Программист C#
Программист C++
Программист C++/C#/Java
Программист/ Junior Developer
Программист/ технический специалист
Программистр-разработчик  информационных систем
f3
Программист с опытом Python
Программист / Senior Developer с опытом Python
Программист 1С 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
Программистр-разработчик  информационных систем с опытом Python
f4
Программист с опытом Python, запрлата 115631 руб.
Программист / Senior Developer с опытом Python, запрлата 132962 руб.
Программист 1С с опытом Python, запрлата 198622 руб.
Программист C# с опытом Python, запрлата 151354 руб.
Программист C++ с опытом Python, запрлата 188973 руб.
Программист C++/C#/Java с опытом Python, запрлата 195061 руб.
Программист/ Junior Developer с опытом Python, запрлата 120414 руб.
Программист/ технический специалист с опытом Python, запрлата 183545 руб.
Программистр-разработчик  информационных систем с опытом Python, запрлата 159932 руб.
————————————————————
0.045667409896850586
————————————————————
```

рис. 3 конец результата выполнения функции f1, а также результат выполнения функций f2-f4 задания 7

Вывод

На данной лабораторной работе я изучил возможности функционального программирования в языке Python.