**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 ИУ5 «Системы обработки информации и управления»**

Лабораторная работа №3

по дисциплине «Разработка интернет-приложений»

Выполнил:
студент группы РТ5-51Б
М.А. Ходосов

2021 г.

## Задание лабораторной работы:

Задание лабораторной состоит из решения нескольких задач.

Файлы, содержащие решения отдельных задач, должны располагаться в пакете lab_python_fp. 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 количествово 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.

## 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`

## 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, 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.

## 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. 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).

## 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:

Файл field.py (в пакете lab_python_fp):

```python
goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'color': 'black'},
    {'title': None, 'color': 'white', 'price': None},
    {'title': None, 'price': 1234}
]


def field(items, *args):
    assert len(args) > 0
    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: val for key, val in item.items() if val is not None}
        for item in items if item.keys() & args and any(i is not None for i in item.values())
    ]


if __name__ == '__main__':
    for i in field(goods, 'title'):
        print(i)

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

Файл gen_random.py (в пакете lab_python_fp):

```python
from random import randint


def gen_random(num_count, min_value, max_value):
    return (randint(min_value, max_value) for i in range(num_count))


if __name__ == '__main__':
    for i in gen_random(10, 1, 5):
        print(i, end=" ")
    print()

```

Файл unique.py (в пакете lab_python_fp):

```python
class Unique(object):
    def __init__(self, items, **kwargs):
        self.used = set()
        self.data = items
        self.current_index = 0

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

    def __next__(self):
        while True:
            if self.current_index == len(self.data):
                raise StopIteration
            current_item = self.data[self.current_index]
            self.current_index += 1

            if self.ignore_case and current_item.lower() not in self.used:
                self.used.add(current_item.lower())
                return current_item

            if not self.ignore_case and current_item not in self.used:
                self.used.add(current_item)
                return current_item

    def __iter__(self):
        return self


if __name__ == '__main__':
    data = [1, 1, 2, 2, 1, 2, 1, 2, 2, 1]
    for val in Unique(data):
        print(val, end=" ")
    print()

    data = ['a', 'A', 'b', 'B', 'ab', 'aB', 'Ab', 'AB']

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

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

Файл sort.py (в пакете lab_python_fp):

```python
if __name__ == '__main__':
    data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]

    result = sorted(data, key=abs, reverse=True)
    print(result)

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

```

Файл print_result.py (в пакете lab_python_fp):

```python
def print_result(func):
    def wrapper(*args):
        result = func(*args)
        print(func.__name__)

        if type(result) == list:
            for val in result:
                print(val)
        elif type(result) == dict:
            for key, val in result.items():
                print("{} = {}".format(key, val))
        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__':
    test_1()
    test_2()
    test_3()
    test_4()
```

Файл cm_timer.py (в пакете lab_python_fp):

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


class cm_timer_1:
    def __init__(self):
        self.start_time = None
        self.end_time = None

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

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


@contextmanager
def cm_timer_2():
    start_time = time()
    yield
    end_time = time()
    print("time: {}".format(end_time - start_time))


if __name__ == '__main__':
    with cm_timer_1():
        sleep(5.0)
    with cm_timer_2():
        sleep(5.0)
```

Файл process_data.py (в пакете lab_python_fp):

```python
import json
from field import field
from cm_timer import cm_timer_1
from print_result import print_result
from gen_random import gen_random

path = "../data_light.json"

with open(path) as f:
    data = json.load(f)


@print_result
def f1(arg):
    return sorted(set(val.lower() for val in field(arg, 'job-name')), key=str.lower)


@print_result
def f2(arg):
    return list(filter(lambda val: str.startswith(val, "программист"), arg))


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


@print_result
def f4(arg):
    return [t[0] + t[1] for t in list(
        zip(arg, [(', зарплата ' + str(val) + ' руб.') for val in list(gen_random(len(arg), 100000, 200000))]))]


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

```

## Экранные формы с примерами выполнения программы:

```
lonkidely@HomePC:/media/lonkidely/Work/bmstu/FifthSemester/Web/Lab3/lab_python_fp$ py
thon3 field.py
Ковер
Диван для отдыха
{'title': 'Ковер', 'price': 2000, 'color': 'green'}
{'title': 'Диван для отдыха', 'color': 'black'}
{'color': 'white'}
{'price': 1234}
lonkidely@HomePC:/media/lonkidely/Work/bmstu/FifthSemester/Web/Lab3/lab_python_fp$ py
thon3 gen_random.py
2 4 5 2 1 4 3 4 3 3
lonkidely@HomePC:/media/lonkidely/Work/bmstu/FifthSemester/Web/Lab3/lab_python_fp$ py
thon3 unique.py
1 2
a A b B ab aB Ab AB
a b ab
lonkidely@HomePC:/media/lonkidely/Work/bmstu/FifthSemester/Web/Lab3/lab_python_fp$ py
thon3 sort.py
[123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
lonkidely@HomePC:/media/lonkidely/Work/bmstu/FifthSemester/Web/Lab3/lab_python_fp$
```

```
lonkidely@HomePC:/media/lonkidely/Work/bmstu/FifthSemester/Web/Lab3/lab_python_fp$ py
thon3 print_result.py
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
lonkidely@HomePC:/media/lonkidely/Work/bmstu/FifthSemester/Web/Lab3/lab_python_fp$ py
thon3 cm_timer.py
time: 5.005097389221191
time: 5.005115985870361
lonkidely@HomePC:/media/lonkidely/Work/bmstu/FifthSemester/Web/Lab3/lab_python_fp$
```

```
lonkidely@HomePC:/media/lonkidely/Work/bmstu/FifthSemester/Web/Lab3/lab_python_fp$ py
thon3 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
```

```
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, зарплата 123864 руб.
программист / senior developer с опытом Python, зарплата 133240 руб.
программист 1с с опытом Python, зарплата 164594 руб.
программист c# с опытом Python, зарплата 127791 руб.
программист c++ с опытом Python, зарплата 178163 руб.
программист c++/c#/java с опытом Python, зарплата 153120 руб.
программист/ junior developer с опытом Python, зарплата 189006 руб.
программист/ технический специалист с опытом Python, зарплата 186476 руб.
программистр-разработчик  информационных систем с опытом Python, зарплата 147950 руб.
time: 0.020502805709838867
lonkidely@HomePC:/media/lonkidely/Work/bmstu/FifthSemester/Web/Lab3/lab_python_fp$
```