

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

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Разработка интернет-приложений»

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

Выполнил:

студент группы ИУ5-51Б
Алехин Сергей

Подпись и дата:

Проверил:

преподаватель каф. ИУ5
Гапанюк Ю.Е.

Подпись и дата:

г. Москва, 2020 г.

Задание лабораторной работы

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.

- 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.

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, которые содержит значения массива 1, отсортированные по модулю в порядке убывания. Сортировку необходимо осуществлять с помощью функции `sorted`. 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 выводиться 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. `cm_timer_1` и `cm_timer_2` реализуют одинаковую функциональность, но должны быть реализованы двумя различными способами.

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

field.py

```
def field(items, *args):
    assert len(args) > 0
    if len(args) == 1:
        for obj in items:
            if args[0] in obj and obj[args[0]] is not None:
                yield obj[args[0]]
    else:
        for obj in items:
            res = {}
            for prop in args:
                if prop in obj and obj[prop] is not None:
                    res[prop] = obj[prop]
            if len(res) > 0:
                yield res
if __name__ == '__main__':
    goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'color': 'black'}
    ]
    for i in field(goods, 'title'):
        print(i)
    for i in field(goods, 'title', 'price'):
        print(i)
```

gen_random.py

```
import random
def gen_random(num_count, begin, end):
    for i in range(num_count):
```

```

        yield random.randrange(begin, end + 1)
if __name__ == '__main__':
    g = gen_random(5, 1, 3)
    for i in gen_random(5, 1, 3):
        print(i)

```

unique.py

```

import types
from lab_python_fp.gen_random import gen_random
class Unique(object):
    def __init__(self, items, **kwargs):
        self.index = 0
        self.data = items
        self.used_elements = set()
        self.ignore_case = False
        for key in kwargs:
            if key == 'ignore_case':
                self.ignore_case = kwargs[key]
        pass
    def __next__(self):
        while True:
            if type(self.data) == types.GeneratorType:
                current = next(self.data)
            elif self.index >= len(self.data):
                raise StopIteration
            else:
                current = self.data[self.index]
                if type(current) == str and self.ignore_case == True:
                    current = current.lower()
                self.index = self.index + 1
            if current not in self.used_elements:
                self.used_elements.add(current)
                return current
        pass
    def __iter__(self):
        return self
if __name__ == '__main__':
    data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    for i in Unique(data, ignore_case=True):
        print(i)
    data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    for i in Unique(data, ignore_case=False):
        print(i)
    data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
    for i in Unique(data, ignore_case=True):
        print(i)
    data = gen_random(10, 1, 3)
    for i in Unique(data, ignore_case=True):
        print(i)

```

sort.py

```

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 n: abs(n), reverse=True)
    print(result_with_lambda)

```

print_result.py

```

def print_result(func_to_decorate):
    def decorated_func(*args):
        print(func_to_decorate.__name__)

```

```

    arg = func_to_decorate(*args)
    if type(arg) == list:
        for i in arg:
            print('{}'.format(i))
    elif type(arg) == dict:
        for key in arg:
            print('{} = {}'.format(key, arg[key]))
    else:
        print(arg)
    return arg
    return decorated_func
@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()

```

cm_timer.py

```

import time
from contextlib import contextmanager
class cm_timer_1:
    def __init__(self):
        self.before_time = 0
        self.after_time = 0
    def __enter__(self):
        self.before_time = time.perf_counter()
        return time.perf_counter()
    def __exit__(self, exp_type, exp_value, traceback):
        if exp_type is not None:
            print(exp_type, exp_value, traceback)
        else:
            self.after_time = time.perf_counter()
            print('time: {}'.format(round(self.after_time - self.before_time, 5)))
@contextmanager
def cm_timer_2():
    before_time = time.perf_counter()
    yield time.perf_counter()
    print('time: {}'.format(round(time.perf_counter() - before_time, 5)))
if __name__ == '__main__':
    with cm_timer_1():
        time.sleep(5.5)
    with cm_timer_2():
        time.sleep(5.5)

```

process_data.py

```

import json
from lab_python_fp.print_result import print_result
from lab_python_fp.cm_timer import cm_timer_1
from lab_python_fp.unique import Unique
from lab_python_fp.field import field

```

```

from lab_python_fp.gen_random import gen_random
path = "../data/data_light.json"
with open(path) as f:
    data = json.load(f)

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

```

Примеры работы программы

Задача 1 (файл field.py)

```

Ковер
Диван для отдыха
{'title': 'Ковер', 'price': 2000}
{'title': 'Диван для отдыха'}

```

Задача 2 (файл gen_random.py)

```

3 2 2 2 2

```

Задача 3 (файл unique.py)

```

A b
a A b B
1 2
3 2 1

```

Задача 4 (файл sort.py)

```

[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]

```

Задача 5 (файл print_result.py)

```

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

```

Задача 6 (файл cm_timer.py)

```

time: 5.50372
time: 5.50351

```

Задача 7 (файл process_data.py)

f1

1с программист

2-ой механик

3-ий механик

4-ый механик

4-ый электромеханик

asic специалист

javascript разработчик

rtl специалист

web-программист

web-разработчик

[химик-эксперт

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. зeya, г. тында

агент торговый

f2

1с программист

web-программист

веб - программист (php, js) / web разработчик

веб-программист

ведущий инженер-программист

ведущий программист

инженер - программист асу тп

инженер-программист

инженер-программист (клинский филиал)

инженер-программист (орехово-зуюевский филиал)

инженер-программист 1 категории

инженер-программист ккт

инженер-программист плис

инженер-программист сапоу (java)

инженер-электронщик (программист асу тп)

помощник веб-программиста

программист

программист / senior developer

программист 1с

программист c#

программист c++

программист c++/c#/java

программист/ junior developer

программист/ технический специалист

программист-разработчик информационных систем

системный программист (с, linux)

старший программист

инженер - программист

педагог программист

f3

1с программист с опытом Python
web-программист с опытом Python
веб - программист (php, js) / web разработчик с опытом Python
веб-программист с опытом Python
ведущий инженер-программист с опытом Python
ведущий программист с опытом Python
инженер - программист асу тп с опытом Python
инженер-программист с опытом Python
инженер-программист (клинский филиал) с опытом Python
инженер-программист (орехово-зюевский филиал) с опытом Python
инженер-программист 1 категории с опытом Python
инженер-программист ккт с опытом Python
инженер-программист плис с опытом Python
инженер-программист сапоу (java) с опытом Python
инженер-электронщик (программист асу тп) с опытом Python
помощник веб-программиста с опытом Python
программист с опытом Python
программист / senior developer с опытом Python
программист 1с с опытом Python
программист с# с опытом Python
программист с++ с опытом Python
программист с++/с#/java с опытом Python
программист/ junior developer с опытом Python
программист/ технический специалист с опытом Python
программист-разработчик информационных систем с опытом Python
системный программист (с, linux) с опытом Python
старший программист с опытом Python
инженер - программист с опытом Python
педагог программист с опытом Python

f4

1с программист с опытом Python, зарплата 134114 руб
web-программист с опытом Python, зарплата 183827 руб
веб - программист (php, js) / web разработчик с опытом Python, зарплата 164479 руб
веб-программист с опытом Python, зарплата 164701 руб
ведущий инженер-программист с опытом Python, зарплата 114821 руб
ведущий программист с опытом Python, зарплата 139759 руб
инженер - программист асу тп с опытом Python, зарплата 114307 руб
инженер-программист с опытом Python, зарплата 197429 руб
инженер-программист (клинский филиал) с опытом Python, зарплата 157595 руб
инженер-программист (орехово-зюевский филиал) с опытом Python, зарплата 193795 руб
инженер-программист 1 категории с опытом Python, зарплата 168494 руб
инженер-программист ккт с опытом Python, зарплата 146810 руб
инженер-программист плис с опытом Python, зарплата 111475 руб
инженер-программист сапоу (java) с опытом Python, зарплата 147535 руб
инженер-электронщик (программист асу тп) с опытом Python, зарплата 137518 руб
помощник веб-программиста с опытом Python, зарплата 172158 руб
программист с опытом Python, зарплата 186122 руб
программист / senior developer с опытом Python, зарплата 171843 руб
программист 1с с опытом Python, зарплата 166244 руб
программист с# с опытом Python, зарплата 173647 руб
программист с++ с опытом Python, зарплата 174990 руб
программист с++/с#/java с опытом Python, зарплата 138202 руб
программист/ junior developer с опытом Python, зарплата 146993 руб
программист/ технический специалист с опытом Python, зарплата 167765 руб
программист-разработчик информационных систем с опытом Python, зарплата 193696 руб
системный программист (с, linux) с опытом Python, зарплата 192625 руб
старший программист с опытом Python, зарплата 149291 руб
инженер - программист с опытом Python, зарплата 145587 руб
педагог программист с опытом Python, зарплата 181654 руб
time: 0.02062