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

Факультет «Информатика и системы управления»

Отчёт по лабораторной работе №3
по курсу «Разработка интернет-приложений»
Функциональные возможности языка Python

Выполнил: студент
группы ИУ5-51Б
Бондаренко И. Г.

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю. Е.

Подпись и дата: 24.12.2021

Подпись и дата:
24.12.2021

Москва, 2021 г.

**Описание задания.**

Задание лабораторной работы 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.

- 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.

**Текст программы.**

```
def field(items, *args):
    assert len(args) > 0
    result = []
    if len(args) == 1:
        for item in items:
            if args[0] in item.keys():
                result.append(item[args[0]])
    else:
        for item in items:
            res = dict()
            for key in args:
                if key in item.keys():
                    res[key] = item[key]
            result.append(res)
    return result

goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
]
```

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

### 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.

**Текст программы.**

```
import random

def gen_random(num_count, begin, end):
    for number in range(num_count):
        yield random.randint(begin, end)

for i in gen_random(5, 1, 3):
        print(i)
```

---

### 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.**

```
class Unique(object):
    def __init__(self, items, **kwargs):
        self.used_elements = set()
        self.data = items
        self.index = 0

        if 'ignore_case' not in kwargs:
            self.ignore_case = False
```

```
        else:
            self.ignore_case = kwargs['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:
                    if current.lower() not in self.used_elements:
                        self.used_elements.add(current.lower())
                        return current
                else:
                    if current not in self.used_elements:
                        self.used_elements.add(current)
                        return current

print('test 1: ')
data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
for val in Unique(data):
    print(val)
print('test 2: ')
data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
for val in Unique(data):
    print(val)
print('test 3: ')
data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
for val in Unique(data, ignore_case=True):
    print(val)
```

**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.

**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 i: abs(i), reverse=True)
    print(result_with_lambda)
```

### 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.

### Текст программы.

```
def print_result(func):
    def wrapper(*args):

        out = func(*args)
        print(func.__name__)
        if isinstance(out, list):
            for val in out:
                print(val)
            return out
        elif isinstance(out, dict):
            for key, val in out.items():
                print('{} = {}'.format(key, val))
            return out
        else:
            print(out)
            return out

    return wrapper


@print_result
def test_1():
    return 1


@print_result
def test_2():
    return 'iu5'
```

```python
@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()
```

**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.

**Текст программы.**

```python
import time
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.time()

    def __exit__(self, exp_type, exp_value, traceback):
        self.end_time = time.time()
        print('time: {}'.format(self.end_time - self.start_time))


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


if __name__ == '__main__':
    with Cm_timer_1():
        time.sleep(1.0)
```

```
    with cm_timer_2():
        time.sleep(1.0)
```

**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
import cm_timer
from print_result import print_result
from gen_random import gen_random
# Сделаем другие необходимые импорты

path = '.\data_light.json'
```

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

with open(path, encoding='utf8') as f:
    data = json.load(f)

# Далее необходимо реализовать все функции по заданию, заменив `raise NotImplemented`
# Предполагается, что функции f1, f2, f3 будут реализованы в одну строку
# В реализации функции f4 может быть до 3 строк


@print_result
def f1(arg):
    return sorted(set([val.lower() for val in arg]), key=str.lower)


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


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


@print_result
def f4(arg):
    temp = list(zip(arg, [(', зарплата '+str(el) + ' руб.') for el in list(gen_random(len(arg), 100000, 200000))]))
    return [(el[0]+el[1]) for el in temp]


if __name__ == '__main__':
    with cm_timer.Cm_timer_1():
        f4(f3(f2(f1([el['job-name'] for el in data]))))
```

## Экранные формы с примерами выполнения программы.

```
frinom@Ivan:/mnt/d/files/rip$ python3 lab3/lab_python_fp/field.py
['Ковер', 'Диван для отдыха']
[{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}]
frinom@Ivan:/mnt/d/files/rip$ python3 lab3/lab_python_fp/gen_random.py
2
3
2
2
2
frinom@Ivan:/mnt/d/files/rip$ python3 lab3/lab_python_fp/sort.py
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]
frinom@Ivan:/mnt/d/files/rip$ python3 lab3/lab_python_fp/unique.py
test 1:
1
2
test 2:
a
A
b
B
test 3:
a
b
frinom@Ivan:/mnt/d/files/rip$ python3 lab3/lab_python_fp/print_result.py
!!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
frinom@Ivan:/mnt/d/files/rip$ python3 lab3/lab_python_fp/cm_timer.py
time: 1.0006144046783447
time: 1.0007402896881104
```

```
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, зарплата 143174 руб.
программист / senior developer с опытом Python, зарплата 155225 руб.
программист 1с с опытом Python, зарплата 126932 руб.
программист c# с опытом Python, зарплата 119984 руб.
программист c++ с опытом Python, зарплата 103309 руб.
программист c++/c#/java с опытом Python, зарплата 116375 руб.
программист/ junior developer с опытом Python, зарплата 145485 руб.
программист/ технический специалист с опытом Python, зарплата 115705 руб.
программистр-разработчик  информационных систем с опытом Python, зарплата 134318 руб.
time: 0.676262617111206
```