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 «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Лабораторная работа № 3

по дисциплине «Базовые компоненты интернет-технологий»

Выполнил:
студент группы ИУ5-32Б
Кульктна Д.А.

Проверил:
Канев А.И.

2021 г.

Полученное задание:

Задание лабораторной работы состоит из решения нескольких задач.

Файлы, содержащие решения отдельных задач, должны располагаться в пакете lab_python_fp. Решение каждой задачи должно раполагаться 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.

```
def field(items, *args):
    assert len(args) > 0, 'Can`t be zero encounters'
    if len(args) == 1:
        for dictionary in items:
            note = dictionary.get(args[0])
            if note is not None:
                yield note
    else:
        for d in items:
            dictionary = dict()
            for key in args:
                note = d.get(key)
                if note is not None:
                    dictionary[key] = note
            if len(dictionary) != 0:
                yield dictionary


if __name__ == '__main__':
    goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},
        {'title': 'Кресло', 'price': None, 'color': None},
        {'title': None, 'price': 10000, 'color': 'pink'}
    ]
    data1 = list()
    data2 = list()

    for i in field(goods, 'title'):
        data1.append(i)
    print(str(data1))
    print()

    for i in field(goods, 'title', 'price'):
        data2.append(i)
    print(data2)
```

Результат

```
['Ковер', 'Диван для отдыха', 'Кресло']

[{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}, {'title': 'Кресло'}, {'price': 10000}]

Process finished with exit code 0
```

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.

```
import random


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


data = gen_random(7, 1, 3)

if __name__ == '__main__':
    print(list(data))
```

Результат

```
[1, 1, 1, 2, 2, 1, 3]


Process finished with exit code 0
```

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.

```
from get_random import gen_random


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

```
            if len(kwargs) > 0:
                self.ignore_case = kwargs['ignore_case']

        def __next__(self):
            it = iter(self.data)
            while True:
                try:
                    current = next(it)
                except StopIteration:
                    raise StopIteration
                else:
                    if self.ignore_case is True and isinstance(current,
    str):
                        current = current.lower()
                    if current not in self.used_elements:
                        self.used_elements.add(current)
                        return current

        def __iter__(self):
            return self


    if __name__ == '__main__':
        data1 = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
        data2 = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
        data3 = ['a', 'b', 'c', 'd', 'c', "A", "B", "C", 'c', 'b', 1, 2, 2,
    3, 3, 1, 2, 3, 4]
        data4 = gen_random(10, 1, 3)

        print('first')
        print(list(Unique(data1)))
        print('second')
        print(list(Unique(data2, ignore_case=False)))
        print('third')
        print(list(Unique(data4)))
        print('fourth')
        print(list(Unique(data3, ignore_case=True)))
```

Результат

```
first
[1, 2]
second
['a', 'A', 'b', 'B']
third
[1, 3, 2]
fourth
['a', 'b', 'c', 'd', 1, 2, 3, 4]

Process finished with exit code 0
```

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.

```
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 x: abs(x), reverse=True)
    print(result_with_lambda)
```

Результат

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

Process finished with exit code 0
```

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.

```
def print_result(func_to_decorate):
    def decorated_func(*args, **kwargs):
        incoming = func_to_decorate(*args, **kwargs)
        print(func_to_decorate.__name__)
        if isinstance(incoming, list):
            for i in incoming:
                print(i)
        elif isinstance(incoming, dict):
            for i in incoming:
                print(str(i) + ", " + str(incoming[i]))
        else:
            print(incoming)
        return incoming
```

```
        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()
```

Результат

```
test_1
1
test_2
iu5
test_3
a, 1
b, 2
test_4
1
2

Process finished with exit code 0
```

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.

```
import time
from contextlib import contextmanager


class cm_timer1:

    def __init__(self):
        self.startTime = time.time()

    def __enter__(self):
```

```
        self.startTime = time.time()

    def __exit__(self, exp_type, exp_value, traceback):
        if exp_type is not None:
            print(exp_type, exp_value, traceback)
        else:
            print("time: {}".format(time.time() - self.startTime))


@contextmanager
def cm_timer2():
    startTime = time.time()
    yield
    print("time: {}".format(time.time() - startTime))


if __name__ == '__main__':
    with cm_timer1():
        time.sleep(2)
    with cm_timer2():
        time.sleep(2)
```

Результат

```
time: 2.008740186691284
time: 2.010369300842285


Process finished with exit code 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.

```
import json
from pathlib import Path
import sys
from print_result import print_result
from cm_timer import cm_timer1
from unique import Unique
from field import field
from get_random import gen_random

# Сделаем другие необходимые импорты
path = Path(r"C:\Users\eldorado\PycharmProjects\Lab_3\lab_python_fp",
"data_light1.json")
#path =
r'C:\Users\eldorado\PycharmProjects\Lab_3\lab_python_fp\data_light.json
'

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

with open(path, encoding="utf-8") 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(list(Unique(field(arg, 'job-name'),
ignore_case=True)), key=str.lower)


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


@print_result
def f3(arg):
    return list(map(lambda s: s + " с опытом python", arg))
```

```
@print_result
def f4(arg):
    return dict(zip(arg, list('зарплата {} руб.'.format(val) for val in
gen_random(len(arg), 1000000, 2000000))))


if __name__ == '__main__':
    with cm_timer1():
        f4(f3(f2(f1(data))))
```

Частичный результат

```
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
```

```
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 1423512 руб.
программист / senior developer с опытом python, зарплата 1081860 руб.
программист 1с с опытом python, зарплата 1850673 руб.
программист c# с опытом python, зарплата 1024772 руб.
программист c++ с опытом python, зарплата 1148761 руб.
программист c++/c#/java с опытом python, зарплата 1298499 руб.
программист/ junior developer с опытом python, зарплата 1738214 руб.
программист/ технический специалист с опытом python, зарплата 1397883 руб.
программистр-разработчик  информационных систем с опытом python, зарплата 1091363 руб.
time: 0.036942243576049805

Process finished with exit code 0
```