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

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

Студент Столярова Ольга Денисовна
фамилия, имя, отчество

Группа РТ5-51Б

Название предприятия МГТУ им. Н. Э. Баумана

Студент Столярова О.Д.

Преподаватель Гапанюк Ю.Е.

2021 г.

Цель работы

Изучение возможностей функционального программирования в языке Python.

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. Пример:
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.

Код программы

```python
goods = [
        {'Название': 'Вишневый сад', 'Жанр': 'Пьеса', 'Стоимость': 350},
        {'Название': 'Влюбиться в искусство', 'Жанр': None, 'Стоимость': 680},
        {'Название': 'Шелкопряд', 'Жанр': None, 'Стоимость': 430}
    ]


def field(items, *args):
    assert len(args) > 0, 'Не переданы аргументы полей словаря'
    if len(args) == 1:
        for i in range(len(items)):
            if args[0] in items[i] and items[i].get(args[0]) is not None:
                yield items[i].get(args[0])
    else:
        for i in range(len(items)):
            s = {}
            for j in range(len(args)):
                if args[j] in items[i] and items[i].get(args[j]) is not None:
                    s.update({args[j]: items[i].get(args[j])})
            yield s


def main():
    f = field(goods, 'Название')
    for i in f:
        print(i, end=', ')
    print('\n', end='')
    f = field(goods, 'Название', 'Цена')
    for i in f:
        print(i, end=', ')


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

Пример выполнения программы

```
PS C:\Users\olyas\python\ЛР3\lab_python_fp>  & 'C:\Program Files\WindowsApps\PythonSoftwareFoundation.Python.3.9_3.9.2544.0_x64__qbz5n2kfra8p0\python3.9.exe' 'c:\Users\olyas\.vscode\extensions\ms-python.python-2021.11.1422169775\pythonFiles\lib\python\debugpy\launcher' '55806' '--' 'c:\Users\olyas\python\ЛР3\lab_python_fp\field.py'

Вишневый сад, Влюбиться в искусство, Шелкопряд,
{'Название': 'Вишневый сад'}, {'Название': 'Влюбиться в искусство'}, {'Название': 'Шелкопряд'},
PS C:\Users\olyas\python\ЛР3\lab_python_fp>
```

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.

Код программы

```python
import random


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


def main():
    gen = gen_random(9, 1, 10)
    for i in gen:
        print(i, end =' ')


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

Пример выполнения программы

```
------------------------------------------------------------------------
PS C:\Users\olyas\python\ЛР3\lab_python_fp>  c:; cd 'c:\Users\olyas\python\ЛР3\lab_python_fp'; & 'C:\Program Fi
s\.vscode\extensions\ms-python.python-2021.11.1422169775\pythonFiles\lib\python\debugpy\launcher' '55823' '--'
'c:\Users\olyas\python\ЛР3\lab_python_fp\gen_random.py'
8 2 2 4 5 9 5 8 8
PS C:\Users\olyas\python\ЛР3\lab_python_fp>
```

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.

Код программы

```python
from gen_random import gen_random

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

        if len(kwargs) != 0:
            self.ignore_case = kwargs
        else:
            self.ignore_case = False

    def __next__(self):
        while True:
            for item in self.items:
                temp_item = item
                self.counter += 1
                if (temp_item not in self.used_elements) \
                        and not(self.ignore_case and temp_item.swapcase() in self.used_elements):
                    self.used_elements.add(temp_item)
                    return temp_item
            else:
                raise StopIteration

    def __iter__(self):
        return self


def main():
    data1 = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
    print(data1)
    itr1 = Unique(data1)
    for i1 in itr1:
        print(i1, end=' ')
    print('\n', end='')

    data2 = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    print(data2)
    itr2 = Unique(data2)
    for i2 in itr2:
        print(i2, end=' ')
    print('\n', end='')

    print(data2)
    itr3 = Unique(data2, ignor_case=True)
    for i3 in itr3:
        print(i3, end=' ')
    print('\n', end='')

    data3 = gen_random(5, 1, 3)
    itr4 = Unique(data3)
    for i4 in itr4:
        print(i4, end=' ')


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

Пример выполнения

```
PS C:\Users\olyas\python\ЛР3\lab_python_fp>  c:; cd 'c:\Users\olyas\python\ЛР3\lab_python_fp'; & 'C:\Program Files\WindowsApps\P
ythonSoftwareFoundation.Python.3.9_3.9.2544.0_x64__qbz5n2kfra8p0\python3.9.exe' 'c:\Users\olyas\.vscode\extensions\ms-python.pyt
hon-2021.11.1422169775\pythonFiles\lib\python\debugpy\launcher' '55845' '--' 'c:\Users\olyas\python\ЛР3\lab_python_fp\unique.py'

[1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
1 2
['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
a A b B
['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
a b
3 2
```

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:
С использованием lambda-функции.
Без использования lambda-функции.

Код программы

```python
def sort(x):
    return abs(x)


def main():
    data = [4, -30, 100, -100, 123, 1, 0, -1, -4]

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

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


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

Пример выполнения

```
PS C:\Users\olyas\python\ЛР3\lab_python_fp>  c:; cd 'c:\Users\olyas\python\ЛР3\lab_python_fp'; & 'C:\Program Files\WindowsApps\P
hon-2021.11.1422169775\pythonFiles\lib\python\debugpy\launcher' '55858' '--' 'c:\Users\olyas\python\ЛР3\lab_python_fp\sort.py'
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]
PS C:\Users\olyas\python\ЛР3\lab_python_fp>
```

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.

Код программы

```python
def print_result(func_to_decorate):

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

        result = func_to_decorate(*args)

        if type(result) is list:
            for i in result:
                print(i)
        elif type(result) is dict:
            for i in result:
                print(i, result.get(i), sep=' = ')
        else:
            print(result)

        return result

    return decorated_func


@print_result
def f1():
    return 1


@print_result
def f2():
    return 'olya'


@print_result
def f3():
    return {'a': 1, 'b': 2}


@print_result
def f4():
    return [1, 2]


def main():
    f1()
    f2()
    f3()
    f4()


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

Пример выполнения

```
PS C:\Users\olyas\python\ЛР3\lab_python_fp>  c:; cd 'c:\Users\olyas\python\ЛР3\lab_python_fp'; & 'C:\Program Files\WindowsApps\P
ythonSoftwareFoundation.Python.3.9_3.9.2544.0_x64__qbz5n2kfra8p0\python3.9.exe' 'c:\Users\olyas\.vscode\extensions\ms-python.pyt
hon-2021.11.1422169775\pythonFiles\lib\python\debugpy\launcher' '55876' '--' 'c:\Users\olyas\python\ЛР3\lab_python_fp\print_resu
lt.py'
f1
1
f2
olya
f3
a = 1
b = 2
f4
1
2
```

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).

Текст программы

```python
import time
from contextlib import contextmanager


class cm_timer_1:

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

    def __enter__(self):
        pass

    def __exit__(self, exc_type, exc_val, exc_tb):
        if exc_type is not None:
            print(exc_type, exc_val, exc_tb)
        else:
            print('time: ', time.time() - self.begin_time)


@contextmanager
def cm_timer_2():
    begin_time = time.time()
    yield 1
    print('time: ', time.time() - begin_time)


```

```python
def main():
    with cm_timer_1():
        time.sleep(5.5)

    with cm_timer_2():
        time.sleep(2.5)


if __name__ == '__main__':
    main()

```

Пример выполнения

```
PS C:\Users\olyas\python\ЛР3\lab_python_fp>  c:; cd 'c:\Users\olyas\python\ЛР3\lab_python_fp'; & 'C:\Program Files\WindowsApps\P
ythonSoftwareFoundation.Python.3.9_3.9.2544.0_x64__qbz5n2kfra8p0\python3.9.exe' 'c:\Users\olyas\.vscode\extensions\ms-python.pyt
y'
time:  5.502758979797363
time:  2.511715888977051
```

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.

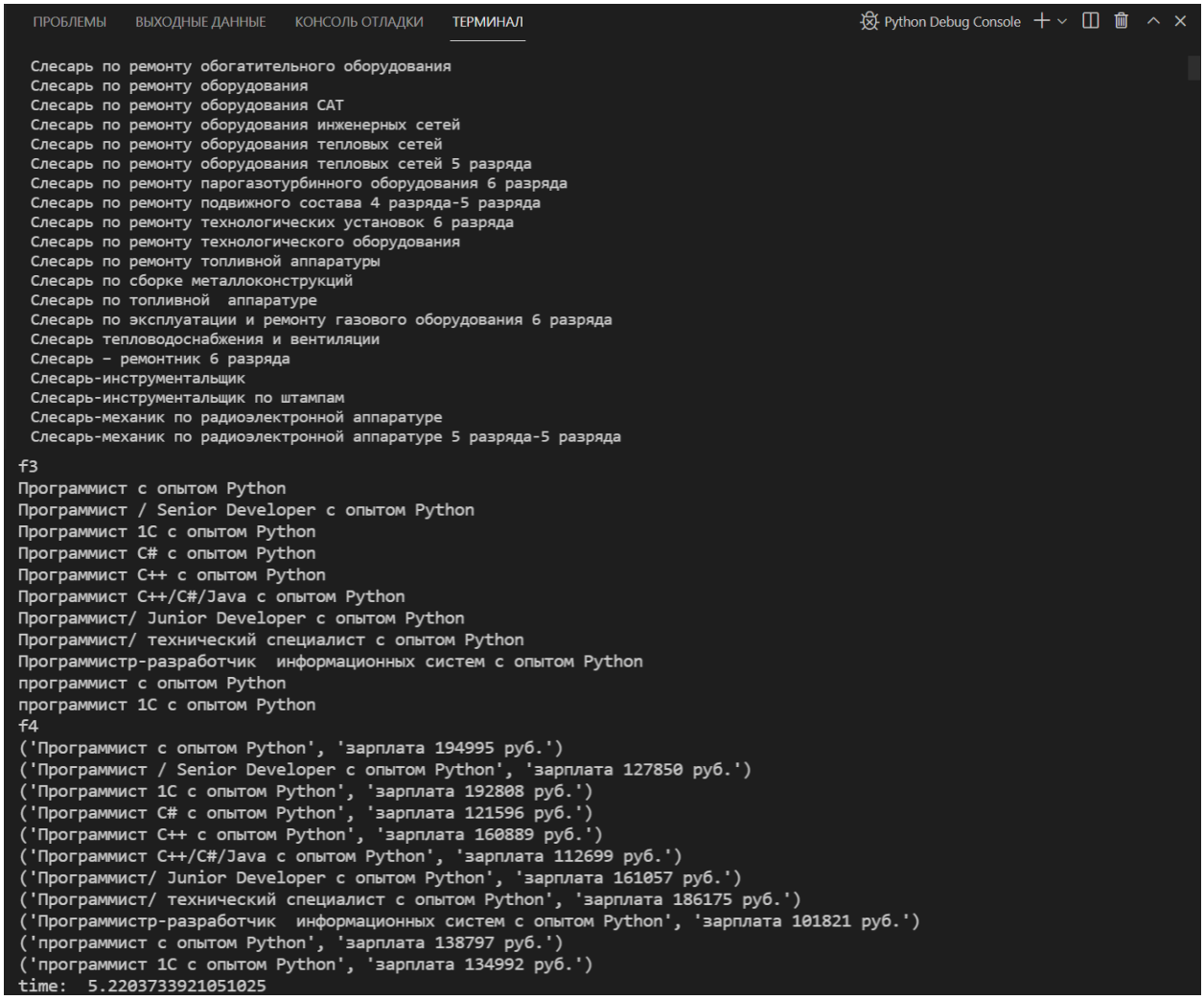
Функция f3 должна модифицировать каждый элемент массива, добавив строку “с опытом Python” (все программисты должны быть знакомы с Python). Пример: Программист C# с опытом Python. Для модификации используйте функцию map.

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.

Код программы

```python
import json
import print_result as print_result
import cm_timer as cm_timer
import unique as unique
import gen_random as gen_random

# Сделаем другие необходимые импорты

path = "../data_light.json"

with open(path, encoding='utf-8') as f:
    data = json.load(f)


@print_result.print_result
def f1(arg):
    professions = list()
    for element in arg:
        professions.append(dict(element).get('job-name'))
    u = unique.Unique(professions, ignore_case=True)
    return sorted(u)


@print_result.print_result
def f2(arg: list):
    def f_func(arg_: str):
        if arg_.startswith('программист') or arg_.startswith('Программист'):
            return True
        else:
            return False

    filtered = filter(f_func, arg)
    return list(filtered)


@print_result.print_result
def f3(arg :list):
    def mod_f(arg_m : str):
        arg_m += ' с опытом Python'
        return arg_m

    return list(map(mod_f, arg))


@print_result.print_result
def f4(arg :list):
    l = len(arg)
    r_m = gen_random.gen_random(l, 100000, 200000)
    zp = list()
    for _ in arg:
        zp.append('зарплата ' + str(int(next(r_m))) + ' руб.')
    ZP = list(zip(arg, zp))
    return ZP


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

Пример выполнения



Ссылка на GitHub
https://github.com/OlyaSto/Olyabmstu