МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Н. Э. Баумана

ЛАБОРАТОРНАЯ РАБОТА №3

Тема:

«Функциональные возможности Python»

по учебной дисциплине

«Разработка интернет-приложений»

Группа: *ИУ5-52Б*

Студент: *Кобяк А.В.*

Преподаватель: *Гапанюк Ю. Е.*

Москва, 2020

Задание работы

Цель: 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:

```
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:

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

Задача 1

```
goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},
    {'title': 'Диван НЕ для отдыха', 'price': 500, 'color': 'white'},
    {'title': 'Шкаф', 'price': 10000}
]


def field(items, *args):
    assert len(args) > 0
    if len(args) == 1:
        for elem in items:
            if args[0] in elem and elem[args[0]] is not None:
                yield elem[args[0]]
    else:
        for elem in items:
            result = {}
            for key in args:
                if key in elem and elem[key] is not None:
                    result[key] = elem[key]
            yield result


test = field(goods, 'title', 'price')
while True:
    try:
        i = test.__next__()
        print(i)
    except StopIteration:
        break
```

Задача 2

```
import random


def Get_Rand(count, begin, end):
    for counter in range(count):
        yield random.randint(begin, end)


if __name__ == '__main__':
    random_str = Get_Rand(6, 1, 6)
    print(list(random_str))
```

Задача 3

```python
from get_random import Get_Rand


# Итератор для удаления дубликатов
class Unique:
    def __init__(self, items, **kwargs):
        self.used_elements = set()
        self.data = list(items)
        self.index = 0
        if 'ignore_case' in kwargs.keys():
            self.ignore_case = kwargs['ignore_case']
        else:
            self.ignore_case = False

    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.upper() not in self.used_elements:
                        self.used_elements.add(current.upper())
                        return current
                else:
                    if current not in self.used_elements:
                        self.used_elements.add(current)
                        return current

    def __iter__(self):
        return self

def UnSo(some):
    mas = []
    for i in Unique(some, ignore_case = True):
        mas.append(i)
    return sorted(mas)


if __name__ == '__main__':
    res = []
    for i in Unique([1, 3, 2, 1, 3, 2, 1, 3, 1, 2, 6, 6, 6]):
        res.append(i)
    print(sorted(res), "\n")
    res.clear()
```

```
    for i in Unique(Get_Rand(10, 1, 3)):
        res.append(i)
    print(sorted(res), "\n")
    res.clear()
    for i in Unique(['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']):
        res.append(i)
    print(sorted(res), "\n")
    res.clear()
    for i in Unique(['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B'], ignore_case = True)
:
        res.append(i)
    print(sorted(res), "\n")
    res.clear()
```

Задача 4

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

if __name__ == '__main__':
    result = sorted(data, key = abs, reverse = True)
    print(result, '\n')
    result_lam = sorted(data, key = lambda i: abs(i), reverse = True)
    print(result_lam)
```

Задача 5

```
def print_result(func_t_d):

    def decorated(*args, **kwargs):
        func_t_d(*args, **kwargs)
        print(str(func_t_d.__name__))
        element = func_t_d(*args, **kwargs)
        if isinstance(element, str):
            print(element)
        elif isinstance(element, int):
            print(element)
        elif isinstance(element, dict):
            for key, value in element.items():
                print(key, '=', value)
        elif isinstance(element, list):
            for i in element:
                print(i)
        return element
    return decorated

@print_result
def test_1():
    return 1
```

```python
@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__':
    print('!!!!!!!!')
    test_1()
    test_2()
    test_3()
    test_4()
```

Задача 6

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


class cm_timer_1:

    def __init__(self):
        pass

    def __enter__(self):
        self.start = time.time()
        return 333

    def __exit__(self, exp_type, exp_value, traceback):
        print(time.time() - self.start, '\n')

@contextmanager
def cm_timer_2():
    start = datetime.now()
    yield
    print(datetime.now() - start)

if __name__ == '__main__':
```

```
    with cm_timer_1():
        sleep(2.5)

    with cm_timer_2():
        sleep(3)
```

Задача 7

```
import json
import sys
from print_result import print_result
from cm_timer import cm_timer_1
from time import sleep
from unique import UnSo
from get_random import Get_Rand

way = "lr3/data_light.json"


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


@print_result
def f1(source):
    return UnSo([some['job-name'] for some in source])

@print_result
def f2(source):
    return list(filter(lambda n: "программист" in n, source))

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

@print_result
def f4(source):
    return list(map(lambda i: i + ', зарплата ' + str(*Get_Rand(1, 100_000, 200_0
00)) + ' рублей', source))

if __name__ == '__main__':
    with cm_timer_1():
        f4(f3(f2(f1(data))))
        sleep(1.5)
```

Результат

Задача 1

```
{'title': 'Ковер', 'price': 2000}
{'title': 'Диван для отдыха', 'price': 5300}
{'title': 'Диван НЕ для отдыха', 'price': 500}
{'title': 'Шкаф', 'price': 10000}
```

Задача 2

```
[4, 4, 1, 6, 3, 2]
```

Задача 3

```
[1, 2, 3, 6]

[2, 3]

['A', 'B', 'a', 'b']

['a', 'b']
```

Задача 4

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

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

Задача 5

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

Задача 6

```
2.5007340908050537

0:00:03.000669
```

Задача 7

```
f4
1С программистс опытом Python, зарплата 187868 рублей
Web-программистс опытом Python, зарплата 189150 рублей
Веб - программист (PHP, JS) / Web разработчикс опытом Python, зарплата 127791 р
ублей
Веб-программистс опытом Python, зарплата 191105 рублей
Ведущий инженер-программистс опытом Python, зарплата 117130 рублей
Ведущий программистс опытом Python, зарплата 115307 рублей
Инженер - программист АСУ ТПс опытом Python, зарплата 152417 рублей
Инженер-программист (Клинский филиал)с опытом Python, зарплата 112756 рублей
Инженер-программист (Орехово-Зуевский филиал)с опытом Python, зарплата 189945 р
ублей
Инженер-программист 1 категориис опытом Python, зарплата 123358 рублей
Инженер-программист ККТс опытом Python, зарплата 133193 рублей
Инженер-программист ПЛИСс опытом Python, зарплата 126573 рублей
Инженер-программист САПОУ (java)с опытом Python, зарплата 115084 рублей
Инженер-электронщик (программист АСУ ТП)с опытом Python, зарплата 109842 рублей
Помощник веб-программистас опытом Python, зарплата 152743 рублей
Системный программист (C, Linux)с опытом Python, зарплата 111281 рублей
Старший программистс опытом Python, зарплата 189675 рублей
инженер - программистс опытом Python, зарплата 106718 рублей
инженер-программистс опытом Python, зарплата 117764 рублей
педагог программистс опытом Python, зарплата 122132 рублей
1.668896198272705
```