Московский государственный технический университет имени Н. Э. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5

Отчёт по

Лабораторной работе 3

Выполнил:

Бахрамов Н.А

Группа ИУ5-53Б

Москва

2021

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.

Пример:

```
data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
```

`Unique(data)` будет последовательно возвращать только 1 и 2.

```
data = gen_random(1, 3, 10)
```

`Unique(data)` будет последовательно возвращать только 1, 2 и 3.

```
data = [‘a’, ‘A’, ‘b’, ‘B’, ‘a’, ‘A’, ‘b’, ‘B’]
```

`Unique(data)` будет последовательно возвращать только a, A, b, B.

`Unique(data, ignore_case=True)` будет последовательно возвращать только a, b.

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:

```
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).

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.

Main

```python
def field(dict, *args):
    assert len(args) > 0
    if len(args) > 1:
        dict1 = [{j: i[j] for j in args if j in i.keys()} for i in dict]
    else:
        dict1 = [i[j] for i in dict for j in args if j in i.keys()]
    return dict1


def main():
    goods = [
        {'title': 'Диван', 'price': 2500, 'color': 'green'},
        {'title': 'Ковер', 'price': 1000}
    ]
    field(goods, 'title')
    field(goods, 'title', 'price')
    field(goods, 'title', 'price', 'color')


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

Cm_timer

```python
import datetime
import time


class cm_timer:
    def __init__(self):
        self.start_time = datetime.datetime.now()

    def __enter__(self):
        self.start_time = datetime.datetime.now()

    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: {}'.format((datetime.datetime.now() - self.start_time).seconds))


def main():
    with cm_timer():
        time.sleep(7)


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

Finish

```python
from lab3 import field
from cm_timer import cm_timer
from unique import Unique
from gen_random import gen_random
# Сделаем другие необходимые импорты

path = 'C:/Users/nikik/web-fall_IU5/lab3/data_light.json'

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

with open(path, 'r', 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(Unique(field(arg, 'job-name'), ignore_case=True))


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


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


@print_result
def f4(arg):
    salary_dict = gen_random(len(arg), 100000, 200000)
    return list(map(lambda x, y: x + ' с запрлатой ' + str(y), arg, salary_dict))


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

Gen_random

```python
import random


def gen_random(num, start, stop):
    arr = []
    for i in range(num):
        arr.append(random.randint(start, stop))
    return arr


def main():
    arr = gen_random(5, 1, 30)
    [print(i) for i in arr]


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

Print result

```python
def print_result(func):
    def wrapper(*args, **kwargs):
        res = func(*args, **kwargs)
        print(func.__name__)
        if type(res) is dict:
            for key in res:
                print('{} = {}'.format(key, res[key]))
        if type(res) is list:
            [print(i) for i in res]
        if (type(res) is not list) & (type(res) is not dict):
            print(res)
        return res
    return wrapper
```

Sort

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

def 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)


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

Unique

```python
# Итератор для удаления дубликатов
class Unique:
    """Итератор, оставляющий только уникальные значения."""
    def __init__(self, data, **kwargs):
        self.used_elements = set()
        self.data = data
        self.index = 0
        self.ignore_case = False
        for v in kwargs.values():
            self.ignore_case = v

    def __iter__(self):
        return self

    def __next__(self):
        while True:
            if self.index >= len(self.data):
                raise StopIteration
            else:
                current = self.data[self.index]
                if self.ignore_case:
                    current = str(current).lower()
                self.index = self.index + 1
                if current not in self.used_elements:
                    # Добавление в множество производится
                    # с помощью метода add
                    self.used_elements.add(current)
                    return current


def main():
    abs = ['A', 'b', 'S', 'A', 'a', 'A', 'B', 'b', 's']
    for i in Unique(abs, ignore_case=True):
        print(i)


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

Результат

```
f4
программист с опытом Python с запрлатой 124269
программист / senior developer с опытом Python с запрлатой 171921
программист 1с с опытом Python с запрлатой 118200
программист c# с опытом Python с запрлатой 191002
программист c++ с опытом Python с запрлатой 102221
программист c++/c#/java с опытом Python с запрлатой 128842
программист/ junior developer с опытом Python с запрлатой 195345
программист/ технический специалист с опытом Python с запрлатой 175441
программистр-разработчик  информационных систем с опытом Python с запрлатой 116964
time: 0
```