Московский государственный технический университет им. Н.Э. Баумана

Факультет “Радиотехнический”
Кафедра ИУ5 “Системы обработки информации и управления”

Отчет по проделанной лабораторной работе номер 3

по курсу

Базовые компоненты интернет технологий

Вариант 14

Подготовил:

Студент группы РТ5-31Б

Мамаев Т.Э.

Проверил:

Доцент кафедры ИУ5

Гапанюк Ю.Е.

23 Декабря 2021г

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 быть раполагаться в отдельном файле.

При запуске каждого файла выдаются тестовые результаты выполнения соответствующего задания.

Задача 1 (файл field.py)

Необходимо реализовать генераторное поле. Генератор поле последовательностей выдает значения ключей словаря.

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 диапазон

Задача 3 (файл unique.py)

- Необходимо реализовать итератор Unique(данные), который принимает входной массив или генератор итерируется по элементам, пропуская дубликаты.

- Конструктор итератора также принимает на вход именованный bool-параметр ignore_case, в зависимости от значений которого считаются считаться одинаковыми строками в разномре. 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, присущие положительные и отрицательные числа. Необходимо одного **строки кода** вывести на экран массив 2, который содержит значения 1, отсортированные по модулю в порядке убывания. Сортировку необходимо осуществлять с помощью функций сортировки

Задача 5 (файл print_result.py)

Необходимо реализовать декоратор print_result, который выводит на экран результат выполнения функции.

- Декоратор должен принимать на вход функцию, вызвать её, печатать в консоль, функции и результат выполнения, после чего возвращать результат выполнения.
- Если функция вернула список (список), то значения элементов должны быть выведены 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_1и cm_timer_2, как считают, время работы блока кода и выводят его на экран

Задача 7 (файл process_data.py)

- В предыдущих задачах были написаны все требуемые инструменты для работы с данными. Применим их на

реальных примерах.

- 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 предыдущей, 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 одной строке. В реализации функции f4 может быть до 3 строк.

- Функция f1 должна вывести отсортированный список профессий без повторений (строки в разном регистре 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 використовуйте функцию фильтра.

- Функция f3 должна модифицировать каждый элемент данных, добавив «с опытом Python» (все программисты должны быть знакомы с Python). Пример: Программист С

с опытом Python. Для модификации функции 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

```
#field.py
def field(items, *args):
    assert len(args) > 0
    if len(args) == 1:
        for item in items:
            if args[0] in item and item[args[0]] is not None:
                yield item[args[0]]

    else:
        for item in items:
            dictionary = {}
            for value in args:
                if value in item and item[value] is not None:
                    dictionary[value] = item[value]
            if len(dictionary) != 0:
                yield dictionary


def main():
    goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
```

```
        {'title': 'Диван для отдыха', 'price': 5300, 'color':
'black'}
    ]
    a = field(goods, 'title')
    print(next(a))
    print(next(a))


if __name__ == "__main__":
    main()


#cm_timer.py
import time
from contextlib import contextmanager


class cm_timer_1:
    def __init__(self):
        self.start = time.time()

    def __enter__(self):
        return self

    def __exit__(self, *args):
        print('time: {}'.format(time.time() - self.start))


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


if __name__ == '__main__':
    with cm_timer_1():
        time.sleep(5.5)

    with cm_timer_2():
        time.sleep(5.5)



#print_result.py
import json
from print_result import print_result
import unique
from gen_random import gen_random
from cm_timer import cm_timer_1
# Сделаем другие необходимые импорты
```

```
path = "/Users/temirhanmamaev/BKIT/data_light.json"

with open(path, "r", encoding='utf8') as f:
    data = json.load(f)
    args = (job["job-name"] for job in data)

@print_result
def f1(args):
    return sorted(unique.Unique(args, ignore_case=False).data)


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


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


@print_result
def f4(arg):
    return list(zip(arg, list(gen_random(len(arg), 100000,
200000))))


if __name__ == '__main__':
    with cm_timer_1():
        f4(f3(f2(f1(args))))



#sort.py
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)

#unique.py
from gen_random import gen_random

class Unique(object):
    def __init__(self, items, **kwargs):
```

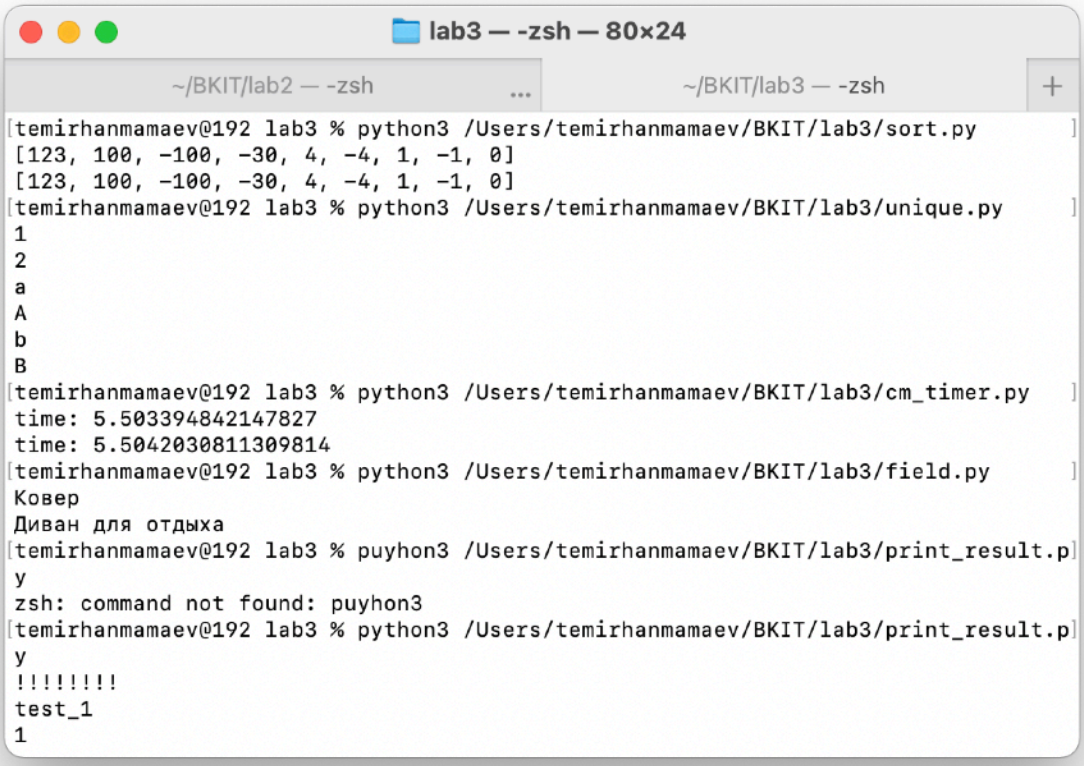
```
        self.used_elements = set()
        self.data = items
        self.ignore_case = False
        if len(kwargs) > 0:
            self.ignore_case = kwargs['ignore_case']

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

    def __iter__(self):
        return self


def main():
    data = gen_random(3, 1, 2)
    iter = Unique(data, ignore_case=True)
    for i in iter:
        print(i)
    data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    iter = Unique(data, ignore_case=False)
    for i in iter:
        print(i)


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

#process_data.py

```
import json
from print_result import print_result
import unique
from gen_random import gen_random
from cm_timer import cm_timer_1
# Сделаем другие необходимые импорты
```

```
path = "/Users/temirhanmamaev/BKIT/data_light.json"

with open(path, "r", encoding='utf8') as f:
    data = json.load(f)
    args = (job["job-name"] for job in data)

@print_result
def f1(args):
    return sorted(unique.Unique(args, ignore_case=False).data)


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


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


@print_result
def f4(arg):
    return list(zip(arg, list(gen_random(len(arg), 100000,
200000))))


if __name__ == '__main__':
    with cm_timer_1():
        f4(f3(f2(f1(args))))

#gen_random.py
import random


def gen_random(num_count, begin, end):
    cur = 0 #текущее значение
    while cur < num_count:
        cur += 1
        yield random.randint(begin, end)
```

Результат выполнения программы


lab3 — -zsh — 80×24
~/BKIT/lab2 — -zsh
~/BKIT/lab3 — -zsh
[temirhanmamaev@192 lab3 % python3 /Users/temirhanmamaev/BKIT/lab3/sort.py
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[temirhanmamaev@192 lab3 % python3 /Users/temirhanmamaev/BKIT/lab3/unique.py
1
2
a
A
b
B
[temirhanmamaev@192 lab3 % python3 /Users/temirhanmamaev/BKIT/lab3/cm_timer.py
time: 5.503394842147827
time: 5.5042030811309814
[temirhanmamaev@192 lab3 % python3 /Users/temirhanmamaev/BKIT/lab3/field.py
Ковер
Диван для отдыха
[temirhanmamaev@192 lab3 % puyhon3 /Users/temirhanmamaev/BKIT/lab3/print_result.p
y
zsh: command not found: puyhon3
[temirhanmamaev@192 lab3 % python3 /Users/temirhanmamaev/BKIT/lab3/print_result.p
y
!!!!!!!!
test_1
1