МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Московский государственный технический университет

им. Н.Э. Баумана

(МГТУ им. Н.Э. Баумана)

Кафедра «Системы обработки информации и управления» (ИУ5)

Отчёт по лабораторной работе № 3-4

По курсу: «Базовые компоненты интернет-технологий»

Выполнил: Никулин Данила Дмитриевич

студент группы ИУ5-31Б.

Проверил: _____

Дата: ___.___.2022г.

Подпись: _____.

г. Москва 2022 г.

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 располагаться 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. В качестве первого аргумента генератор принимает список словарей, дальше через *args генератор принимает неограниченное количество 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:

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. Необходимо решить задачу двумя способами:

- С использованием lambda-функции.
- 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. 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). Функция f4 должна сгенерировать для каждой специальности зарплату от 100 000 до 200 000 рублей и присоединить её к названию специальности.

Приложение 1. Текст программы:

field.py

```
def field(items, *args):
    if len(args) == 1:
        arr = []
        for item in items:
            for key in item:
                if key == args[0] and item[key] is not None:
                    arr.append(item[key])
        return arr
    else:
        for item in items:
            dictionary = dict()
            for key in item:
                for argument in args:
                   if key == argument and item[argument] is not None:
                       dictionary[key] = item[key]
        return(dictionary)

def main():
    goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
    ]
    # должен выдавать 'Ковер', 'Диван для отдыха'.
    res = (field(goods, 'title'))
    for el in res:
        print(el , end = '; ')
    print()
    # должен выдавать {'title': 'Ковер', 'price': 2000}, {'title': 'Диван для
отдыха', 'price': 5300}.
    print(field(goods, 'title', 'price'))

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

gen_random.py

```
from random import randint

def gen_random(count_num, begin, end):
    rand_arr = []
    for i in range(count_num):
        rand_arr.append(randint(begin,end))
    return rand_arr

def main():
    numbers = gen_random(5, 1, 3)
    for i in numbers:
        print(numbers[i], end = ' ')
    print()

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

unique.py

```
from gen_random import gen_random
class Unique(object):
    def __init__(self, items, ignore_case=False, **kwargs):
        self._data = items
        self._ignore_case = ignore_case
        self.__used_data = set()
        self.__index = 0

    def __next__(self):
        # Если игнорируем капс, то пробегаемся по списку и приводим всё к
общему капсу.
        if self._ignore_case:
            for counter, el in enumerate(self._data):
                if type(el) is str:
                    self._data[counter] = el.lower()

        while True:
            if self.__index >= len(self._data):
                raise StopIteration
            else:
                current = self._data[self.__index]
                self.__index += 1
                # если текущего числа ещё не было, добавляем и возвращаем.
                if current not in self.__used_data:
                    self.__used_data.add(current)
                    return current

    def __iter__(self):
        return self

def main():
    data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
```

```
    it = Unique(data, ignore_case=True)
    try:
        while True:
            print(it.__next__())
    except StopIteration:
        print('The error "StopInteration" was caught')
    data = gen_random(10, 1, 3)
    it = Unique(data, ignore_case=True)
    try:
        while True:
            print(it.__next__())
    except StopIteration:
        print('The error "StopInteration" was caught')
    data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    it = Unique(data, ignore_case=True)
    try:
        while True:
            print(it.__next__())
    except StopIteration:
        print('The error "StopInteration" was caught')

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

sort.py

```
def main():
    data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]
    result = sorted(data, key=abs, reverse=True)
    print(result)

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

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

print_rezult.py

```
def print_result(func):
    def wrapper(lst=[], *args, **kwargs):
        print(func.__name__)

        if len(lst) == 0:
            result = func(*args, **kwargs)
        else:
            result = func(lst, *args, **kwargs)

        if type(result) is dict:
            for key, el in result.items():
                print(f'{key} = {el}')
```

```
        elif type(result) is list:
            print('\n'.join(map(str, result)))

        else:
            print(result)

        return result

    return wrapper


@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]


def main():
    print('!!!!!!!!')
    test_1()
    test_2()
    test_3()
    test_4()


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

cm_timer.py

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


class cm_timer_1:
    def __enter__(self):
        self.__time_begin = time()
```

```
    def __exit__(self, type, value, traceback):
        print(f'Time of work: {time()- self.__time_begin}')


@contextmanager
def cm_timer_2():
    time_begin = time()
    yield
    print(f'Time of work: {time() - time_begin}')


def main():
    with cm_timer_1():
       sleep(2.0)

    with cm_timer_2():
        sleep(2.0)

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

process_data.py

```
from print_result import print_result
from cm_timer import cm_timer_1
from gen_random import gen_random
import json
import sys

try:
    path = sys.argv[1]
    print(path)
except:
    path = 'data_light.json'

with open(path) as f:
    data = json.load(f)


@print_result
def f1(arg) -> list:
    return sorted(list(set([el['job-name'] for el in arg])), key=lambda x:
x.lower())


@print_result
def f2(arg) -> list:
    return list(filter(lambda text: (text.split())[0].lower() ==
'программист', arg))


@print_result
def f3(arg) -> list:
```

```
    return list(map(lambda lst: lst + ' с опытом Python', arg))


@print_result
def f4(arg) -> list:
    return list(zip(arg, ['зарплата ' + str(el) + ' руб.' for el in
gen_random(len(arg), 100000, 200000)]))


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

Приложение 2. Результаты тестирования:

```
nikulin_danila@ubuntu:~/github/IU5_BKIT2022/lab3-4/lab_python_fp$ python3 field.py
Ковер; Диван для отдыха;
{'title': 'Диван для отдыха', 'price': 5300}
nikulin_danila@ubuntu:~/github/IU5_BKIT2022/lab3-4/lab_python_fp$ python3 gen_random.py
2 1 2 1 2
nikulin_danila@ubuntu:~/github/IU5_BKIT2022/lab3-4/lab_python_fp$ python3 unique.py
1
2
The error "StopInteration" was caught
3
2
1
The error "StopInteration" was caught
a
b
The error "StopInteration" was caught
nikulin_danila@ubuntu:~/github/IU5_BKIT2022/lab3-4/lab_python_fp$ python3 sort.py
[123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
nikulin_danila@ubuntu:~/github/IU5_BKIT2022/lab3-4/lab_python_fp$ python3 print_result.py
!!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
nikulin_danila@ubuntu:~/github/IU5_BKIT2022/lab3-4/lab_python_fp$ python3 cm_timer.py
Time of work: 2.002420425415039
Time of work: 2.0023550987243652
nikulin_danila@ubuntu:~/github/IU5_BKIT2022/lab3-4/lab_python_fp$ python3 process_data.py
f1
1С программист
2-ой механик
3-ий механик
4-ый механик
4-ый электромеханик
[химик-эксперт
ASIC специалист
JavaScript разработчик
RTL специалист
```