Московский государственный технический
университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологиий»

Отчет по лабораторной работе №3-4
“Функциональные возможности языка Python”

Выполнил:
студент группы ИУ5-34Б:
Малютин И.Д.
Подпись и дата:

Проверила:
преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2022

Описание задания:

1) 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 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.

2) Необходимо реализовать генератор, который последовательно выдает заданное количество случайных чисел в заданном диапазоне от минимума до максимума, включая границы диапазона.

3) 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.

4) 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.

   Необходимо решить задачу двумя способами:

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

5) 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.

6) Необходимо написать контекстные менеджеры, которые считают время работы блока кода и выводят его на экран.

7) 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.

Текст программы

Используемые библиотеки:

```
import random
import json
import gen_random
from print_result import print_result
```

```
from cm_timer import cm_timer_1
from contextlib import contextmanager
import time as t
```

field.py

```
goods = [
    {'title': 'Ковер', 'price': int(2000), 'color': 'green'},
    {'title': 'Диван для отдыха', 'color': 'black'}
]
def field(items, *args):
    res = []
    if len(args) == 1:
        for el1 in items:
            for el2 in args:
                if(el1.get(el2) != None):
                    res.append(el1.get(el2))
    else:
        mapp = {}
        for el1 in items:
            for el2 in args:
                if(el1.get(el2) != None):
                    mapp.update({el2 : el1.get(el2)})
            res.append(mapp)
    return res
print(field(goods, 'title'))
print(field(goods, 'title', 'price'))
```

gen_random.py

```
def gen_random(count, bot, top):
    res = []
    for i in range(count):
        res.append(random.randint(bot, top))
    return res

gen_random(5, 1, 3)
```

Unique.py

```
class Unique(object):
    def __init__(self, items, **kwargs):
        self.aboba = []
        for i in items:
            if len(kwargs) > 0 and kwargs['ignore_case']:
                if i.lower() not in self.aboba:
                    self.aboba.append(i.lower())
            elif i not in self.aboba:
                self.aboba.append(i)

    def __next__(self):
        if len(self.aboba) == 0:
            raise StopIteration
        item = self.aboba[0]
        del self.aboba[0]
        return item

    def __iter__(self):
        return self

if __name__ == '__main__':
    data = ['X', 'y', 'y', 'y', 'X', 'X', 'x', 'y']
    t = Unique(data, ignore_case = True)
    print(next(t))
```

```
    print(next(t))

    data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
    t = Unique(data)
    print(next(t))
    print(next(t))

    data = gen_random(10, 1, 3)
    t = Unique(data)
    print(next(t))
    print(next(t))
    print(next(t))
```

sort.py

```
data = gen_random(9, -100, 123)

if __name__ == '__main__':
    result = sorted(data, key = abs, reverse = True)
    print(result)
    res_lambda = sorted(data, key = lambda x:abs(x), reverse = True)
    print(res_lambda)
```

print_result.py

```
def print_result(func):
    def f():
        print(func.__name__)
        output = func()
        if type(output) == list:
            for i in output:
                print(i)
        elif type(output) == dict:
            for key in output.keys():
                print( '{} =  {}'.format(key, output[key]))
        else:
            print(output)
        return output

def test_1():
    return 1



def test_2():
    return 'iu5'



def test_3():
    return {'a': 1, 'b': 2}


def test_4():
    return [1, 2]

if __name__ == '__main__':
    test_1()
    test_2()
    test_3()
    test_4()
```

cm_timer.py

```
@contextmanager
def cm_timer_1():
    start = t.time()
    yield
    print("time: ", t.time()-start)
class cm_timer_2:
    def __init__(self):
        self.start = True
    def __enter__(self):
        self.start = t.time()
        return self.start
    def __exit__(self, exc_type, exc_val, exc_tb):
        print("time: ", t.time() - self.start)


with cm_timer_1():
    t.sleep(5.5)


with cm_timer_2():
    t.sleep(5.5)
```

process_data.py

```
ath ='data_light.json'
with open(path,"rb") 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): return [x for x in arg if x.split()[0].lower() == "программист"]


@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("зарплата " + str(el) + " рублей" for el in
gen_random.gen_random(len(arg),100000,200000))))


with cm_timer_1():
    # f1(data)
    f4(f3(f2(f1(data))))
```

Пример работы:

```
['Ковер', 'Диван для отдыха']
[{'title': 'Диван для отдыха', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 2000}]
x
y
1
2
3
1
2
[114, 111, 111, -96, -92, -31, -28, -7, 7]
[114, 111, 111, -96, -92, -31, -28, -7, 7]
time:  5.502451658248901
time:  5.5135369300842285
```