Московский государственный технический университет им. Н.Э. Баумана

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

Курс «Базовые компоненты интернет-технологий»

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

Выполнил:
студент группы ИУ5-34Б:
Стукалов Иван Дмитриевич
Подпись и дата:

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

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:
// файл field.py

```
def field(dictionary, *args):
    assert len(args) > 0
    new_list = []
    new_dict = {}
    for i in dictionary:
        for j in args:
            if i.get(j) != None:
                new_dict[j] = i.get(j)
        new_list.append(dict(new_dict))
```

```
        new_dict = {}
    return new_list
```

//файл gen_random.py

```
import random

# генератор случайных чисел

def gen_random(count, min, max):
    rand_list = list()
    for i in range (count):
        rand_list.append(random.randint(min, max))
    return rand_list
```

// файл unique.py

```
# Итератор для удаления дубликатов

class Unique(object):
    def __init__(self, items, **kwargs):
        # Нужно реализовать конструктор
        # В качестве ключевого аргумента, конструктор должен принимать bool-
параметр ignore_case,
        # в зависимости от значения которого будут считаться одинаковыми
строки в разном регистре
        # Например: ignore_case = True, Абв и АБВ - разные строки
        #           ignore_case = False, Абв и АБВ - одинаковые строки, одна
из которых удалится
        # По-умолчанию ignore_case = False

        self.ignore_case = kwargs.get("ignore_case")
        self.list = items


    def __next__(self):
        # Нужно реализовать __next__
        if not self.ignore_case:
            if self.current < len(self.list):
                if self.current == 0:
                    self.current += 1
                    return self.list[self.current - 1]

                is_last = False
                i = 0
                while i < self.current:
                    if self.list[i] == self.list[self.current] and
len(self.list) > self.current + 1:
                        self.current += 1
                        i = -1
                    elif self.list[i] == self.list[self.current] and
len(self.list) == self.current + 1:
                        is_last = True
                    i += 1

                if not is_last:
                    result = self.list[self.current]
                    self.current += 1
                    return result
        else:
            if self.current < len(self.list):
                if self.current == 0:
                    self.current += 1
                    return self.list[self.current - 1]
```

```
                is_last = False
                i = 0
                while i < self.current:
                    if self.list[i].upper() ==
self.list[self.current].upper() and len(self.list) > self.current + 1:
                        self.current += 1
                        i = -1
                    elif self.list[i].upper() ==
self.list[self.current].upper() and len(self.list) == self.current + 1:
                        is_last = True
                    i += 1

                if not is_last:
                    result = self.list[self.current]
                    self.current += 1
                    return result


    def __iter__(self):
        self.current = 0
        return self
```

// файл sort.py

```
# сортировка

def sortedList(list, parameter=lambda x: x):
    for i in range(len(list)):
        for j in range(len(list) - 1):
            if parameter(list[j]) < parameter(list[j + 1]):
                list[j], list[j + 1] = list[j + 1], list[j]

    return list
```

// файл print_result.py

```
# выводит на экран результат выполнения функции

def print_result(function):
    def wrapper(*args, **kwargs):
        print(function.__name__)
        result = function(*args, **kwargs)

        if isinstance(result, list):
            for i in result:
                print(i)
        elif isinstance(result, dict):
            for i in result:
                print("{} = {}".format(i, result[i]))
        else:
            print(result)
    return wrapper
```

//файл cm_timer.py

```
#   время работы кода

import time
from contextlib import contextmanager

class TimerError(Exception):  #5
    """Пользовательское исключение, используемое для сообщения об ошибках при
использовании класса Timer"""

class cm_timer_1:
    def __init__(self):
```

```
        self._start_time = None

    def __enter__(self):
        """Запуск нового таймера"""

        if self._start_time is not None:
            raise TimerError(f"Таймер уже работает. Используйте .stop() чтобы
его остановить")

        self._start_time = time.perf_counter()

    def __exit__(self, exc_type, exc_value, exc_tb):
        """Отстановить таймер и сообщить о времени вычисления"""

        if self._start_time is None:
            raise TimerError(f"Таймер не работает. Используйте .start() для
его запуска")

        elapsed_time = time.perf_counter() - self._start_time
        self._start_time = None
        print(f"Вычисление заняло {elapsed_time:0.4f} секунд")


@contextmanager
def cm_timer_2(*args, **kwargs):
    """Запуск нового таймера"""
    start_time = time.perf_counter()
    try:
        yield start_time

    finally:
        """Отстановить таймер и сообщить о времени вычисления"""
        elapsed_time = time.perf_counter() - start_time
        print(f"Вычисление заняло {elapsed_time:0.4f} секунд")
```

//файл process_data.py

```
import json
import sys

# Сделаем другие необходимые импорты
from lab_python_fp.cm_timer import cm_timer_1
from lab_python_fp.field import field
from lab_python_fp.gen_random import gen_random
from lab_python_fp.print_result import print_result
from lab_python_fp.sort import sorted
from lab_python_fp.unique import Unique

path = "C:\work\Бомонка\3 сем\ВКИД\lab_3\data_light.json"

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

with open(path) 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):
    unique = Unique(arg)
```

```
    unique_it = Unique.__iter__(unique)
    print(Unique.__next__(unique_it))


@print_result
def f2(arg):
    raise NotImplemented


@print_result
def f3(arg):
    raise NotImplemented


@print_result
def f4(arg):
    raise NotImplemented

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

//файл main.py

```
from lab_python_fp.field import field
from lab_python_fp.gen_random import gen_random
from lab_python_fp.unique import Unique
from lab_python_fp.sort import sortedList
from lab_python_fp.print_result import print_result
from lab_python_fp.cm_timer import cm_timer_1
from lab_python_fp.cm_timer import cm_timer_2
# from lab_python_fp.process_data import f1

@print_result
def fun1():
    return 1

@print_result
def fun2():
    return 'iu5'

@print_result
def fun3():
    return [1, 2, 3]

@print_result
def fun4():
    return {'a': 1, 'b': 2}

def main():
    goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'color': 'black'}
    ]
    print(field(goods, 'title', 'price'))

    print(gen_random(3, 1, 10))

    count_list = ["aaa", "b", "c", "b", "A", "b", "AAA", "Aaa", "a"]
    unique = Unique(count_list)
    unique_it = Unique.__iter__(unique)

    print(Unique.__next__(unique_it))
    print(Unique.__next__(unique_it))
```

```
    print(Unique.__next__(unique_it))
    print(Unique.__next__(unique_it))
    print(Unique.__next__(unique_it))
    print(Unique.__next__(unique_it))
    print(Unique.__next__(unique_it))

    list = [-3, 6, -2, 5, -4, 3, -10, -53, 8]
    print(sortedList(list))
    print(sortedList(list, lambda x: abs(x)))

    with cm_timer_1():
        fun1()
        fun2()
        fun3()
        fun4()

    with cm_timer_2():
        fun1()
        fun2()
        fun3()
        fun4()

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

Пример выполнения программы:

```
[{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха'}]
[1, 6, 9]
aaa
b
c
A
AAA
Aaa
a
[8, 6, 5, 3, -2, -3, -4, -10, -53]
[-53, -10, 8, 6, 5, -4, 3, -3, -2]
```

```
fun1
1
fun2
iu5
fun3
1
2
3
fun4
a = 1
b = 2
Вычисление заняло 0.0001 секунд
fun1
1
fun2
iu5
fun3
1
2
3
fun4
a = 1
b = 2
Вычисление заняло 0.0001 секунд
```