Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Автоматизированные системы обработки информации и управления»



**Отчет**

**По курсу «Разработка интернет приложений»**

**«Лабораторная работа №3»**

**ИСПОЛНИТЕЛЬ:**

Группа ИУ5-53Б

Жидков Е.И

“25" декабря 2020 г.

**ПРЕПОДАВАТЕЛЬ:**

Гапанюк Ю.Е

Москва 2020

**Задание:**

### 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 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.

### 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:

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.

### 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](https://github.com/iu5team/iu5web-fall-2020/tree/master/labs/lab3/data/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.

**Файл field.py**

# field

goods = [{'title': 'Ковер', 'price': 2000, 'color': 'green'},

{'title': 'Диван', 'color': 'black'}]

#field(goods, 'title')

def field(items, \*args):

assert len(args) > 0

for i in range (len(items)):

for j in range (len(args)):

n = args[j]

print((items[i]).get(n,'Не обозначено'), end=' ')

print('')

field(goods, 'title')

print(20\*'-')

field(goods, 'title', 'price')

print(20\*'-')

field(goods, 'title', 'price', 'color')

print(20\*'-')

**Файл gen\_random.py**

import random

def gen\_random(num\_count, begin, end):

A = [0]\*num\_count

for i in range(num\_count):

n = random.randint(begin, end)

A[i] = str(n)

print(n, end = ',')

print('')

print(A)

return A

gen\_random(5, 1, 3)

**Файл unique.py**

class Unique(object):

def \_\_init\_\_(self, items, \*\*kwargs):

self.unique\_items = []

self.items = iter(items)

if 'ignore\_case' not in kwargs:

self.ignore\_case = False

else:

self.ignore\_case = kwargs['ignore\_case']

def \_\_next\_\_(self):

# Нужно реализовать \_\_next\_\_

while True:

item = self.items.\_\_next\_\_()

compare\_item = None

if self.ignore\_case and type(item) is str:

compare\_item = item.lower()

else:

compare\_item = item

if compare\_item not in self.unique\_items:

self.unique\_items.append(compare\_item)

return item

def \_\_iter\_\_(self):

return self

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

print(list(Unique(data)))

**Файл sort.py**

data = [2, -123, -34, 232, 0, 23, 5, -4, 34]

if \_\_name\_\_=='\_\_main\_\_':

result = sorted(data)

print(result)

result\_with\_lambda = lambda data: sorted(data)

print(result\_with\_lambda(data))

**Файл print\_result.py**

def print\_result(func, \*arg):

def wrapper(\*arg):

n = func(\*arg)

if type(n) == list:

print(func.\_\_name\_\_)

for i in range(len(n)):

print(n[i])

elif type(n) == dict:

print(func.\_\_name\_\_)

for key in n:

print(key, ' = ', n[key])

elif type(n) == int:

print(func.\_\_name\_\_)

print(n)

else:

print(func.\_\_name\_\_)

print(n)

return n

print('')

return wrapper

print(a)

@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]

if \_\_name\_\_ == '\_\_main\_\_':

print('!!!!!!!!')

test\_1()

test\_2()

test\_3()

test\_4()

**Файл cm\_timer.py**

import time

from contextlib import contextmanager

class cm\_time\_1:

def \_\_enter\_\_(self):

self.time = time.time()

def \_\_exit\_\_(self, exp\_type, exp\_value, traceback):

if exp\_type is not None:

print(exp\_type, exp\_value, traceback)

else:

print(time.time() - self.time)

@contextmanager

def cm\_time\_2():

timer = time.time()

yield

print(time.time() - timer)

if \_\_name\_\_ == '\_\_main\_\_':

with cm\_time\_1():

time.sleep(1)

with cm\_time\_2():

time.sleep(2)

**Файл process\_data.py**

import json

from lab\_python\_fp.unique import Unique

from lab\_python\_fp.print\_result import print\_result

from lab\_python\_fp.cm\_timer import cm\_time\_1

from lab\_python\_fp.field import field

from lab\_python\_fp.gen\_random import gen\_random

import sys

# Сделаем другие необходимые импорт

path = "/Users/alexzudin/Downloads/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):

return sorted(list(Unique(field(arg, 'job-name'), ignore\_case=True)), key=lambda x: str.casefold(x))

@print\_result

def f2(arg):

return list(filter(lambda x: "программист" in x.lower(), arg))

@print\_result

def f3(arg):

return list(map(lambda x: x + " с опытом Python", arg))

@print\_result

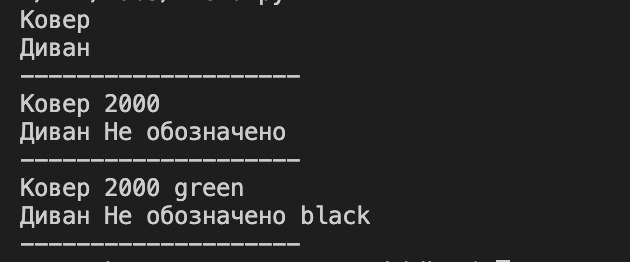
def f4(arg):

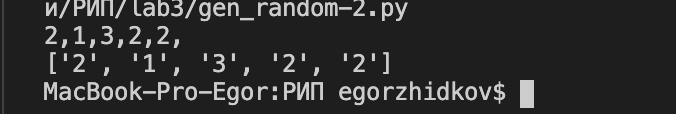
return dict((zip(arg, gen\_random(len(arg), 100000, 200000))))

if \_\_name\_\_ == '\_\_main\_\_':

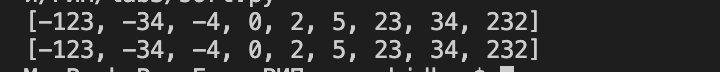
with cm\_time\_1():

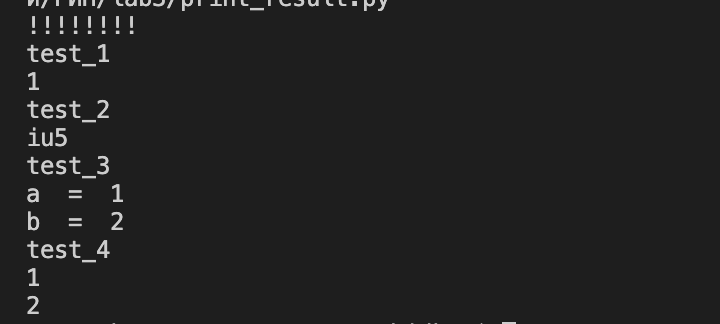
f4(f3(f2(f1(data))))

**Вывод файлов в той же последовательности: **

****

****

****

****

****