# Московский государственный технический университет им. Н.Э. Баумана.

Факультет «Информатика и управление» Кафедра «Системы обработки информации и управления»

Курс «Разработка интернет-приложений» Отчет по лабораторной работе №3

|  |  |
| --- | --- |
| Выполнил: | Проверил: |
| студент группы ИУ5-55 | преподаватель каф. ИУ5 |
| Шкарин Егор | Гапанюк Ю.Е. |
| Подпись и дата: | Подпись и дата: |

Москва, 2021 г.

# 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 раполагаться в отдельном файле.

1. Необходимо реализовать генератор field. Генератор field последовательно выдает значения ключей словаря.
2. 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.

1. 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.

1. 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.
2. 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.

1. Необходимо написать контекстные менеджеры cm\_timer\_1 и cm\_timer\_2, которые считают время работы блока кода и выводят его на экран.
2. 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-2021/tree/main/notebooks/fp/files/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

Lab3.py

import json

from print\_result import print\_result from cm\_timer import cm\_timer\_1

from field import field from unique import Unique

from gen\_random import gen\_random path = "data\_light.json"

with open(path, encoding='utf-8') 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 v: v.lower())

@print\_result def f2(arg):

return list(filter(lambda x: x.startswith('Программист') or x.startswith('программист'), arg))

@print\_result def f3(arg):

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

@print\_result def f4(arg):

zrplt = list(gen\_random(len(arg), 100000, 200000)) res = zip(arg, zrplt)

return list(map(lambda x: x[0] + ', зарплата ' + str(x[1]) + ' руб.', res))

if name == ' main ': with cm\_timer\_1():

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

gen\_random.py

import random

def gen\_random(num\_count, begin, end): for i in range(num\_count):

yield random.randint(begin, end)

if name == ' main ':

for i in gen\_random(5, 1, 3): print(i)

field.py

goods = [

{'title': 'Ковер', 'price': 2000, 'color': 'green'},

{'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},

{'color': 'red'}

]

def field(items, \*args): assert len(args) > 0

if len(args) == 1: for i in items:

if i.get(args[0], None): yield i[args[0]]

else:

for i in items: dict = {}

for k in args:

if i.get(k, None):

dict[k] = i[k]

if dict != {}: yield dict

if name == ' main ':

for i in field(goods, 'title', 'price'): print(i)

cm\_timer.py

import time

from contextlib import contextmanager

class cm\_timer\_1:

def init (self): self.delta = 0

def enter (self):

self.delta = time.perf\_counter() return self.delta

def exit (self, exc\_type, exc\_val, exc\_tb): self.delta = time.perf\_counter() - self.delta print("time: ", self.delta)

return self.delta

@contextmanager def cm\_timer\_2():

delta = 0 try:

delta = time.perf\_counter() yield delta

except:

print("We had an error!") finally:

delta = time.perf\_counter() - delta print("time: ", delta)

if name == ' main ':

with cm\_timer\_1(): time.sleep(5.5)

with cm\_timer\_2(): time.sleep(3.2)

unique.py

import gen\_random

# Итератор для удаления дубликатов class Unique(object):

def init (self, items, \*\*kwargs): self.next\_item\_idx = -1 self.items = list(items) self.unique = [] self.ignore\_case = False

if kwargs.get('ignore\_case'): self.ignore\_case = kwargs['ignore\_case']

for i in self.items: try: # для строк

if i not in self.unique and i.lower() not in self.unique and i[0].upper() + i[1:] \

not in self.unique and self.ignore\_case: self.unique.append(i)

elif i not in self.unique and not self.ignore\_case: self.unique.append(i)

except: # для символов

if i not in self.unique: self.unique.append(i)

def next (self): self.next\_item\_idx += 1

if self.next\_item\_idx >= len(self.unique): raise StopIteration()

else:

elem = self.unique[self.next\_item\_idx] return elem

def iter (self): return self

if name == ' main ':

data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']

data2 = gen\_random.gen\_random(7, 3, 10)

data3 = [1,2,3,4,2,1,4,5,6]

u = Unique(data2, ignore\_case=True) while True:

try:

print(next(u)) except:

break

sort.py

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 x: x\*(-1) if x > 0 else x) print(result\_with\_lambda)

print\_result.py

def print\_result(function\_to\_decorate): def wrapper(\*args):

print(function\_to\_decorate. name ) r = function\_to\_decorate(args[0])

if isinstance(r, list): for i in r:

print(i)

elif isinstance(r, dict): for i in list(r.items()):

print(i[0], '=', i[1])

else:

print(r) return r

return wrapper

if name == ' main ': @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]

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

test\_1() test\_2() test\_3() test\_4()

# Анализ результатов

