|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ (ИУ5)\_\_\_\_

**О Т Ч Е Т**

**по лабораторной работе №2**

по дисциплине: Разработка интернет-приложений\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

на тему:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Python. Функциональные возможности.\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент \_\_\_\_\_ИУ5-53Б\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_Труфанов В.А.\_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_Гапанюк Ю.Е.\_\_\_**

(Подпись, дата) (И.О.Фамилия)

*2019 г.*

1. **Содержание задания**

С 1 по 5 задачу формируется модуль librip, с помощью которого будет выполняться задание 6.

**Задача 1**

Необходимо реализовать генераторы field и gen\_random

1. В качестве первого аргумента генератор принимает list, дальше через \*args генератор принимает неограниченное кол-во аргументов.
2. Если передан один аргумент, генератор последовательно выдает только значения полей, если поле равно None, то элемент пропускается
3. Если передано несколько аргументов, то последовательно выдаются словари, если поле равно None, то оно пропускается, если все поля None, то пропускается целиком весь элемент

Генератор gen\_random последовательно выдает заданное количество случайных чисел в заданном диапазоне

В ex\_1.py нужно вывести на экран то, что они выдают, с помощью кода в *одну строку*

Генераторы должны располагаться в librip/gen.py

### Задача 2

Необходимо реализовать итератор, который принимает на вход массив или генератор и итерируется по элементам, пропуская дубликаты. Конструктор итератора также принимает на вход именной bool-параметр ignore\_case, в зависимости от значения которого будут считаться одинаковыми строки в разном регистре. По умолчанию этот параметр равен False. Итератор **не должен модифицировать** возвращаемые значения.

### Задача 3

Дан массив с положительными и отрицательными числами. Необходимо одной строкой вывести на экран массив, отсортированный по модулю. Сортировку осуществлять с помощью функции sorted

### Задача 4

Необходимо реализовать декоратор print\_result, который выводит на экран результат выполнения функции. Файл ex\_4.py **не нужно** изменять.

Декоратор должен принимать на вход функцию, вызывать её, печатать в консоль имя функции, печатать результат и возвращать значение.

Если функция вернула список (list), то значения должны выводиться в столбик.

### Задача 5

Необходимо написать контекстный менеджер, который считает время работы блока и выводит его на экран

**Задача 6**

В репозитории находится файл data\_light.json. Он содержит облегченный список вакансий в России в формате json.

Структура данных представляет собой массив словарей с множеством полей: название работы, место, уровень зарплаты и т.д.

В ex\_6.py дано 4 функции. В конце каждая функция вызывается, принимая на вход результат работы предыдущей. За счет декоратора @print\_result печатается результат, а контекстный менеджер timer выводит время работы цепочки функций.

Задача реализовать все 4 функции по заданию, ничего не изменяя в файле-шаблоне. Функции f1-f3 должны быть реализованы в 1 строку, функция f4 может состоять максимум из 3 строк.

Что функции должны делать:

1. Функция f1 должна вывести отсортированный список профессий без повторений (строки в разном регистре считать равными). Сортировка должна **игнорировать регистр**. Используйте наработки из предыдущих заданий.
2. Функция f2 должна фильтровать входной массив и возвращать только те элементы, которые начинаются со слова “программист”. Иными словами нужно получить все специальности, связанные с программированием. Для фильтрации используйте функцию filter.
3. 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.

1. **Текст программы**
2. **Модуль gens.py**

import random

# Генератор вычленения полей из массива словарей

# Пример:

# goods = [

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

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

# ]

# field(goods, 'title') должен выдавать 'Ковер', 'Диван для отдыха'

# field(goods, 'title', 'price') должен выдавать {'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}

def field(items, \*args):

assert len(args) > 0

# Необходимо реализовать генератор

fields = []

if len(args) == 1 :

for dictionary in items:

for key in dictionary.keys():

if key in args:

yield dictionary[key]

return

for dictionary in items:

yield {key: dictionary[key] for key in args}

# Генератор списка случайных чисел

# Пример:

# gen\_random(1, 3, 5) должен выдать примерно 2, 2, 3, 2, 1

# Hint: реализация занимает 2 строки

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

# Необходимо реализовать генератор

for x in range(num\_count):

yield random.randint(begin, end)

**Работа модуля (ex\_1.py)**

#!/usr/bin/env python3

from librip.gens import field

goods = [

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

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

{'title': 'Стелаж', 'price': 7000, 'color': 'white'},

{'title': 'Вешалка для одежды', 'price': 800, 'color': 'white'}

]

# Реализация задания

print(\*field(goods, 'title', 'price'), sep=', ')

print(\*field(goods,'title'), sep =', ')

1. **Модуль Iterators.py**

from collections import OrderedDict

# 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 одинаковые строки в разном регистре

# Например: ignore\_case = True, Aбв и АБВ разные строки

# ignore\_case = False, Aбв и АБВ одинаковые строки, одна из них удалится

# По-умолчанию ignore\_case = False

if len(kwargs) > 0:

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

self.items = self.init\_items(items)

self.pos = -1

def init\_items(self, items):

if self.ignore\_case:

od = OrderedDict((item, None) for item in items)

else:

od = OrderedDict()

for item in items:

is\_found = False

for key in od.keys():

if key.lower() == item.lower():

is\_found = True

break

if not is\_found:

od[item] = None

# convert to list for iteration

return list(od.keys())

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

# Нужно реализовать \_\_next\_\_

self.pos += 1

if self.pos==len(self.items):

raise StopIteration

return self.items[self.pos]

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

return self

**Работа модуля (ex\_2.py)**

#!/usr/bin/env python3

from librip.gens import gen\_random

from librip.iterators import Unique

data1 = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]

data2 = gen\_random(1, 3, 10)

data3 = ['a','A','b','B','b',"aC","Ac","ac"]

# Реализация задания 2

print(\*list(data2), sep=', ')

distinct = Unique(data3, ignore\_case=True)

for item in distinct:

print(item, end=' ')

1. **Задание 3 (ex\_3.py)**

#!/usr/bin/env python3

data = [4, -30, 100, -100, 123, 1, 0, -1, -4]

# Реализация задания 3

print(sorted(data, key=abs))

1. **Модуль decorators.py**

def print\_result(func):

def wrap\_func(\*args):

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

output = func(args)

if type(output) is dict:

for key, val in output.items():

print(key,' = ',val)

else:

if type(output) is list:

for x in output:

print(x)

else:

print(output)

return output

return wrap\_func

**Работа модуля (ex\_4.py)**

from librip.decorators import print\_result

# Необходимо верно реализовать print\_result

# и задание будет выполнено

@print\_result

def test\_1():

return 1

@print\_result

def test\_2():

return 'iu'

@print\_result

def test\_3():

return {'a': 1, 'b': 2}

@print\_result

def test\_4():

return [1, 2]

test\_1()

test\_2()

test\_3()

test\_4()

1. **Модуль ctxmngrs.py**

import time

class Timer:

def \_\_init\_\_(self):

pass

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

self.begin = time.time()

def \_\_exit\_\_(self, exc\_type, exc\_val, exc\_tb):

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

**Работа модуля (ex\_5.py)**

from time import sleep

from librip.ctxmngrs import Timer

with Timer():

sleep(2.5)

**Задание 6 (ex\_6.py)**

#!/usr/bin/env python3

import json

import sys

from librip.ctxmngrs import Timer

from librip.decorators import print\_result

from librip.gens import field, gen\_random

from librip.iterators import Unique as unique

path = "data\_light.json"

# Здесь необходимо в переменную path получить

# путь до файла, который был передан при запуске

with open(path) as f:

data = json.load(f)

# Далее необходимо реализовать все функции по заданию, заменив `raise NotImplemented`

# Важно!

# Функции с 1 по 3 дожны быть реализованы в одну строку

# В реализации функции 4 может быть до 3 строк

# При этом строки должны быть не длиннее 80 символов

@print\_result

def f1(arg):

return unique(list(field(arg[0], "job-name")), ignore\_case=True).items

@print\_result

def f2(arg):

return list(filter(lambda x: x.lower().startswith("программист"), arg[0]))

@print\_result

def f3(arg):

return list(map(lambda x: str(x)+" с опытом Python", arg[0]))

@print\_result

def f4(arg):

salaries = list(gen\_random(100000,200000,len(arg[0])))

return [f"{job}, зарплата {sal}" for job,sal in zip(arg[0], salaries)]

with Timer():

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

1. **Экранные формы с примерами работы программы**

ex\_1.py

/usr/bin/python3.7 /home/vadim/PycharmProjects/lab\_2/ex\_1.py

{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}, {'title': 'Стелаж', 'price': 7000}, {'title': 'Вешалка для одежды', 'price': 800}

Ковер, Диван для отдыха, Стелаж, Вешалка для одежды

Process finished with exit code 0

ex\_2.py

/usr/bin/python3.7 /home/vadim/PycharmProjects/lab\_2/ex\_2.py

3, 1, 3, 3, 2, 3, 3, 2, 3, 1

a A b B aC Ac ac

Process finished with exit code 0

ex\_3.py

/usr/bin/python3.7 /home/vadim/PycharmProjects/lab\_2/ex\_3.py

[0, 1, -1, 4, -4, -30, 100, -100, 123]

Process finished with exit code 0

ex\_4.py

/usr/bin/python3.7 /home/vadim/PycharmProjects/lab\_2/ex\_4.py

test\_1

1

test\_2

iu

test\_3

a = 1

b = 2

test\_4

1

2

Process finished with exit code 0

ex\_5.py

/usr/bin/python3.7 /home/vadim/PycharmProjects/lab\_2/ex\_5.py

2.502143621444702

Process finished with exit code 0

ex\_6.py

/usr/bin/python3.7 /home/vadim/PycharmProjects/lab\_2/ex\_6.py

f1

Администратор на телефоне

Медицинская сестра

Охранник сутки-день-ночь-вахта

ВРАЧ АНЕСТЕЗИОЛОГ РЕАНИМАТОЛОГ

теплотехник

разнорабочий

Электро-газосварщик

Водитель Gett/Гетт и Yandex/Яндекс такси на личном автомобиле

Монолитные работы

Организатор – тренер

f2

Программист

Программист С++/С#/Java

программист

Программист 1С

Программистр-разработчик информационных систем

Программист С++

Программист/ Junior Developer

Программист / Senior Developer

Программист/ технический специалист

программист 1С

Программист C#

f3

Программист с опытом Python

Программист С++/С#/Java с опытом Python

программист с опытом Python

Программист 1С с опытом Python

Программистр-разработчик информационных систем с опытом Python

Программист С++ с опытом Python

Программист/ Junior Developer с опытом Python

Программист / Senior Developer с опытом Python

Программист/ технический специалист с опытом Python

программист 1С с опытом Python

Программист C# с опытом Python

f4

Программист с опытом Python, зарплата 181831

Программист С++/С#/Java с опытом Python, зарплата 135215

программист с опытом Python, зарплата 133258

Программист 1С с опытом Python, зарплата 137900

Программистр-разработчик информационных систем с опытом Python, зарплата 115823

Программист С++ с опытом Python, зарплата 177271

Программист/ Junior Developer с опытом Python, зарплата 118582

Программист / Senior Developer с опытом Python, зарплата 143684

Программист/ технический специалист с опытом Python, зарплата 162995

программист 1С с опытом Python, зарплата 155717

Программист C# с опытом Python, зарплата 112173

0.03108954429626465