

Защищено:

Гапанюк Ю.Е.

Демонстрация:

Гапанюк Ю.Е.

"__" _____ 2016 г.

"__" _____ 2016 г.

**Отчет по лабораторной работе № 3 по курсу
Разработка интернет-приложений**

Вариант № 18

ИСПОЛНИТЕЛЬ:

студент группы ИУ5-51

(подпись)

Семенова А.Е.

"__" _____ 2016 г.

Москва, МГТУ - 2016

1) Задание лабораторной работы.

1. Зайти на [github.com](https://github.com/iu5team/ex-lab4) и выполнить fork проекта с заготовленной структурой

<https://github.com/iu5team/ex-lab4>

2. Переименовать репозиторий в lab_4

3. Выполнить git clone проекта из вашего репозитория

4. *Задача 1 (ex_1.py)*

Необходимо реализовать генераторы field и gen_random

Генератор field последовательно выдает значения ключей словарей массива

Генератор gen_random последовательно выдает заданное количество случайных чисел в заданном диапазоне

5. *Задача 2 (ex_2.py)*

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.

6. *Задача 3 (ex_3.py)*

Дан массив с положительными и отрицательными числами. Необходимо одной строкой вывести на экран массив,

отсортированный по модулю. Сортировку осуществлять с помощью функции sorted.

7. *Задача 4 (ex_4.py)*

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. Если функция вернула словарь (dict), то ключи и значения должны

выводиться в столбик через знак равно.

8. *Задача 5 (ex_5.py)*

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

9. *Задача 6 (ex_6.py)*

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:

- Функция f1 должна вывести отсортированный список профессий без повторений (строки в разном регистре считать равными). Сортировка должна игнорировать регистр.
- Функция f2 должна фильтровать входной массив и возвращать только те элементы, которые начинаются со слова “программист”.
- Функция f3 должна модифицировать каждый элемент массива, добавив строку “с опытом Python” (все программисты должны быть знакомы с Python).
- Функция f4 должна сгенерировать для каждой специальности зарплату от 100 000 до 200 000 рублей и присоединить её к названию специальности.

2) Листинг

gens.py

```
from random import randint

def field(items, *args):
    assert len(args) > 0, 'No args'
    # Необходимо реализовать генератор
    if len(args) == 1:
        for el in items:
            if el[args[0]]:
                yield el[args[0]]
    else:
        for el in items:
            dct = {}
            for arg in args:
                if el[arg]:
                    dct[arg] = el[arg]
            if dct:
                yield dct

def gen_random(begin, end, num_count):
    pass
    # Необходимо реализовать генератор
    for i in range(num_count):
        yield randint(begin, end)
```

ex_1.py

```
#!/usr/bin/env python3
from librip.gens import field
from librip.gens import gen_random

goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'price': None, 'color': 'black'},
    {'title': None, 'price': None, 'color': None},
    {'title': 'Вешалка для одежды', 'price': 800, 'color': 'white'}
]

# Реализация задания 1
for i in field(goods, 'title', 'price'):
    print(i, end = " ")

print()

for i in gen_random(1,6,7):
    print(i, end = " ")
```

iterators.py

```
class Unique(object):
    def __init__(self, items, **kwargs):
        if ('ignore_case' in kwargs.keys()) and (kwargs['ignore_case']):
            self.items = [str(i).lower() for i in items]
        else:
            self.items = items
        self.index = 0
```

```

        self.used = []

    def __next__(self):
        # Нужно реализовать __next__
        while self.items[self.index] in self.used:
            if self.index == len(self.items) - 1:
                raise StopIteration
            self.index += 1

        self.used.append(self.items[self.index])
        return self.items[self.index]

    def __iter__(self):
        return self

```

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', 'c', 'B', 'C']

# Реализация задания 2
for i in Unique(data1):
    print(i, end = ' ')

print()

for i in Unique(list(data2)):
    print(i, end = ' ')

print()

for i in Unique(data3):
    print(i, end = ' ')

print()

for i in Unique(data3, ignore_case = True):
    print(i, end = ' ')

print()

```

ex_3.py

```

#!/usr/bin/env python3

data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
# Реализация задания 3

print(sorted(data, key = lambda x: abs(x)), end = " ")

```

decorators.py

```

def print_result(printable_func):
    def decorated(*args):
        print(printable_func.__name__)
        if type(printable_func(*args)) == list:
            for i in printable_func(*args):
                print(i)
    return decorated

```

```

        elif type(printable_func(*args)) == dict:
            for key, val in printable_func(*args).items():
                print('{} = {}'.format(key, val))
        else:
            print(printable_func(*args))
    return decorated

```

ex_4.py

```

#!/usr/bin/env python3
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()

```

ctxmgrs.py

```

import time

class timer:
    def __enter__(self):
        self.start = time.clock()
    def __exit__(self, exp_type, exp_value, traceback):
        print(time.clock() - self.start)

```

ex_5.py

```

#!/usr/bin/env python3
from time import sleep
from librip.ctxmgrs import timer

with timer():
    sleep(5.5)

```

ex_6.py

```

#!/usr/bin/env python3
import os.path
import json
import sys

```

```

from librip.ctxmgrs import timer
from librip.decorators import print_result
from librip.gens import field, gen_random
from librip.iterators import Unique as unique

path = os.path.abspath(sys.argv[1])

# 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 должны быть реализованы в одну строку
# В реализации функции 4 может быть до 3 строк
# При этом строки должны быть не длиннее 80 символов

def f1(arg):
    return(sorted([i for i in unique([j['job-name'] for j in arg], ignore_case
= True)]))

def f2(arg):
    return([x for x in arg if 'программист' in x])

def f3(arg):
    return(["{} {}".format(x, "с опытом Python") for x in arg])

@print_result
def f4(arg):
    return(["{}", {} {} {}".format(x, "зарплата", y, "руб.") for x, y in zip(arg,
list(gen_random(100000, 200000, len(arg))))])

with timer():
    f4(f3(f2(f1(data))))

```

3) Результаты работы

Задача 1

```

C:\Python34\python.exe C:/Users/Александра/Documents/УЧЕБНАЯ/рип/ex-lab4-
master/ex-lab4-master/ex_1.py
{'title': 'Ковер', 'price': 2000} {'title': 'Диван для отдыха'} {'title':
'Вешалка для одежды', 'price': 800}
3 6 3 3 4 3 1
Process finished with exit code 0

```

Задача 2

```

C:\Python34\python.exe C:/Users/Александра/Documents/УЧЕБНАЯ/рип/ex-lab4-
master/ex-lab4-master/ex_2.py
1 2
2 1 3
a A b c B C
a b c

Process finished with exit code 0

```

Задача 3

```
C:\Python34\python.exe C:/Users/Александра/Documents/УЧЕБНАЯ/рип/ex-lab4-master/ex-lab4-master/ex_3.py
[0, 1, -1, 4, -4, -30, 100, -100, 123]
Process finished with exit code 0
```

Задача 4

```
C:\Python34\python.exe C:/Users/Александра/Documents/УЧЕБНАЯ/рип/ex-lab4-master/ex-lab4-master/ex_4.py
test_1
1
test_2
iu
test_3
b = 2
a = 1
test_4
1
2

Process finished with exit code 0
```

Задача 5

5.498863997576528

Process finished with exit code 0

Задача 6

```
C:\Users\Александра\Documents\УЧЕБНАЯ\рип\ex-lab4-master\ex-lab4-master>python ex_6.py data_light_cp1251.json
f4
1с программист с опытом Python, зарплата 178593 руб.
web-программист с опытом Python, зарплата 163916 руб.
веб - программист (php, js) / web разработчик с опытом Python, зарплата 140216 руб.
веб-программист с опытом Python, зарплата 190194 руб.
ведущий инженер-программист с опытом Python, зарплата 112402 руб.
ведущий программист с опытом Python, зарплата 173799 руб.
инженер - программист с опытом Python, зарплата 193606 руб.
инженер - программист асу тп с опытом Python, зарплата 124547 руб.
инженер-программист с опытом Python, зарплата 190748 руб.
инженер-программист (клинский филиал) с опытом Python, зарплата 181943 руб.
инженер-программист (орехово-зуевский филиал) с опытом Python, зарплата 112281 руб.
инженер-программист 1 категории с опытом Python, зарплата 150893 руб.
инженер-программист ккт с опытом Python, зарплата 126535 руб.
инженер-программист плис с опытом Python, зарплата 148424 руб.
инженер-программист сапоу (java) с опытом Python, зарплата 152436 руб.
инженер-электронщик (программист асу тп) с опытом Python, зарплата 157465 руб.
педагог программист с опытом Python, зарплата 134303 руб.
помощник веб-программиста с опытом Python, зарплата 105524 руб.
программист с опытом Python, зарплата 138874 руб.
программист / senior developer с опытом Python, зарплата 148741 руб.
программист 1с с опытом Python, зарплата 131390 руб.
```

программист с# с опытом Python, зарплата 106319 руб.
программист с++ с опытом Python, зарплата 150436 руб.
программист с++/с#/java с опытом Python, зарплата 143508 руб.
программист/ junior developer с опытом Python, зарплата 183804 руб.
программист/ технический специалист с опытом Python, зарплата 112410 руб.
программист-разработчик информационных систем с опытом Python, зарплата 107341 руб.
системный программист (с, linux) с опытом Python, зарплата 159917 руб.
старший программист с опытом Python, зарплата 179188 руб.
0.12197658021670446