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

Факультет «Системы обработки информации и управления»

Кафедра ИУ5. Курс «РИП»

Отчет по лабораторным работе №4
«Python. Функциональные возможности»

Выполнил:
студент группы ИУ5-53
Белков А.Д.

Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Подпись и дата:

Москва, 2017 г.

Задание и порядок выполнения ЛР №3

Важно выполнять все задачи последовательно . С 1 по 5 задачу формируется модуль `librip` , с помощью которого будет выполняться задание 6 на реальных данных из жизни. Весь вывод на экран (даже в столбик) необходимо реализовывать одной строкой .

Подготовительный этап

1. Зайти на github.com 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

Задача 1 (`ex_1.py`)

Необходимо реализовать генераторы `field` и `gen_random` Генератор `field` последовательно выдает значения ключей словарей массива Пример:

```
goods = [
    {'title': 'Ковер', 'price': 2000, 'color':
'green'},
    {'title': 'Диван для отдыха', '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'}
```

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 Пример: `gen_random(1, 3, 5)` должен выдать 5 чисел от 1 до 3, т.е. примерно `2, 2, 3, 2, 1`

В `ex_1.py` нужно вывести на экран то, что они выдают *о дной строкой* Генераторы должны располагаться в `librip/gen.py`

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`. Итератор **н е должен модифицировать** возвращаемые значения.

Пример:

`data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2] Unique(data)` будет последовательно возвращать только `1` и `2`

`data = gen_random(1, 3, 10) unique(gen_random(1, 3, 10))` будет последовательно возвращать только `1`, `2` и `3`

`data = ['a', 'A', 'b', 'B'] Unique(data)` будет последовательно возвращать только `a`, `A`, `b`, `B`

`data = ['a', 'A', 'b', 'B'] Unique(data, ignore_case=True)` будет последовательно возвращать только `a`, `b`

В `ex_2.py` нужно вывести на экран то, что они выдают *о дной строкой*. **Важно** продемонстрировать работу как с массивами, так и с генераторами (`g en_random`). Итератор должен располагаться в `librip/ iterators .py`

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

Вывод: `[0, 1, -1, 4, -4, -30, 100, -100, 123]`

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 выводить в столбик через знак равно
Пример: `@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()
```

На консоль выведется:

```
test_1 1
```



```
test_2 iu test_3 a= 1 b= 2 test_4 1
```

`2` Декоратор должен располагаться в `librip/ decorators .py`

Задача 5 (`ex_5.py`)

Необходимо написать контекстный менеджер, который считает время работы блока и выводит его на экран Пример: `with timer():`

```
sleep(5.5)
```

После завершения блока должно вывестись в консоль примерно 5.5

Задача 6 (`ex_6.py`)

Мы написали все инструменты для работы с данными. Применим их на реальном примере, который мог возникнуть в жизни. В репозитории

находится файл `data_light.json` . Он содержит облегченный список вакансий в России в формате `json` (ссылку на полную версию размером ~ 1 Гб. в формате `xml` можно найти в файле `README.md`).

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` .

4. 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.

Код программы

ctxmngrs.py

```
from time import time
# Здесь необходимо реализовать
# контекстный менеджер timer
```

```
# Он не принимает аргументов, после выполнения блока он должен вывести время
выполнения в секундах
# Пример использования
# with timer():
#   sleep(5.5)
#
# После завершения блока должно вывестись в консоль примерно 5.5


class timer:
    def __init__(self):
        pass

    def __enter__(self):
        self.time = time()

    def __exit__(self, type, value, traceback):
        print(time() - self.time)
```

decoratos.py

```
def print_result(func, *arg):

    def decorated_function(*arg):
        result = func(*arg)

        print(func.__name__)

        if type(result) is dict:
            for key, value in result.items():
                print("%s=%s" % (str(key), str(value)))
        elif type(result) is list:
            for i in result:
                print(i)
        else:
            print(result)
        return result

    return decorated_function
```

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):
    # Необходимо реализовать генератор
    if len(args) == 1:
        for item in items:
            for arg in args:
```

```
                if arg in item:
                    yield item[arg]
    else:
        for item in items:
            new_item = {}
            for arg in args:
                if arg in item:
                    new_item[arg] = item[arg]
            if len(new_item.keys()) > 0:
                yield new_item

# 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

    if begin > end:
        begin, end = end, begin

    for i in range(1, num_count):
        yield random.randint(begin, end)
```

iterators.py

```
# Итератор для удаления дубликатов
class Unique(object):
    def __init__(self, items, ignore_case=False, **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, Абв и АБВ разные строки
        #           ignore_case = False, Абв и АБВ одинаковые строки, одна из
них удалится
        # По-умолчанию ignore_case = False
        self.unique_items = []
        self.ignore_case = ignore_case
        self.items = iter(items)

    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
```

ex_1.py

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

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 1
print(list(field(goods, 'title')))
print(list(field(goods, 'title', 'price')))
print(list(gen_random(1, 3, 5)))
```

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 = ['A', 'a', 'b', 'B']
data3 = ['A', 'a', 'b', 'B']

# Реализация задания 2
for i in Unique(data1):
    print(i, end=" ")
print(" ")
for i in Unique(data2):
    print(i, end=" ")
print(" ")
for i in Unique(data3, ignore_case=True):
    print(i, end=" ")
print(" ")
```

ex_3.py

```
#!/usr/bin/env python3
import math

data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
# Реализация задания 3
print(sorted(data, key=lambda i: math.fabs(i)))
```

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()
```

ex_5.py

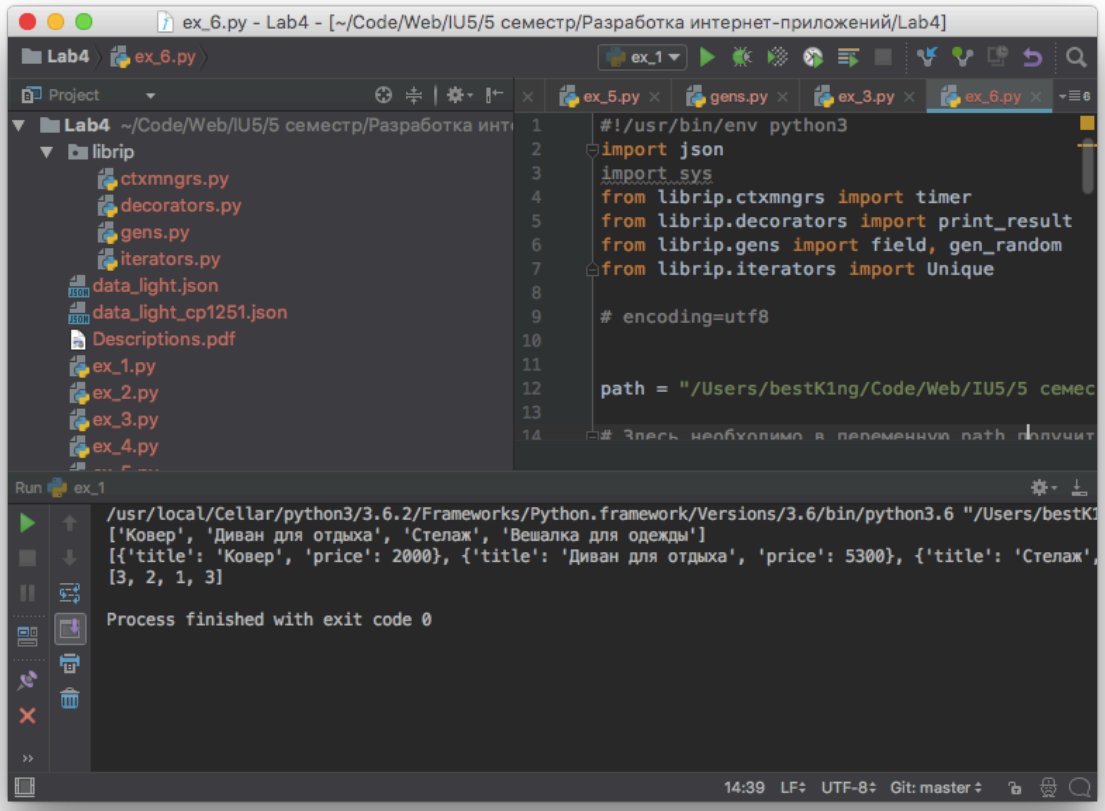
```
from time import sleep
from librip.ctxmngrs import timer

with timer():
    sleep(3.23)
```

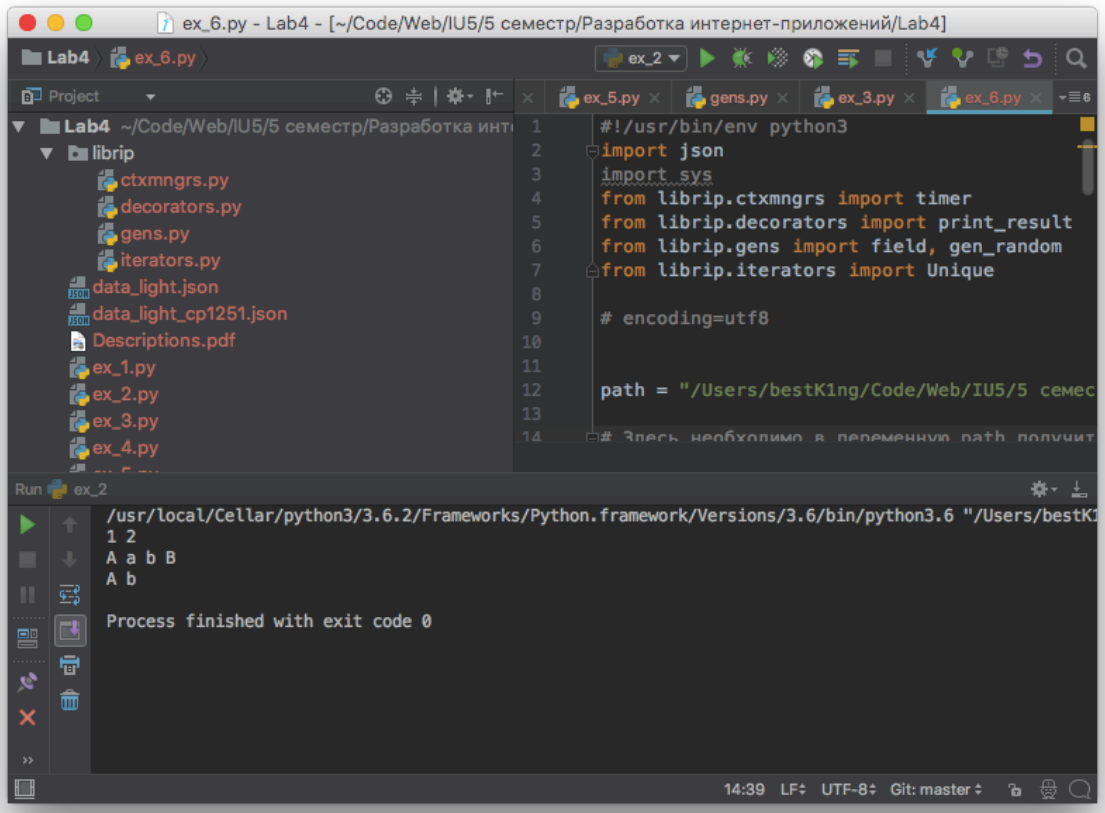
ex_6.py

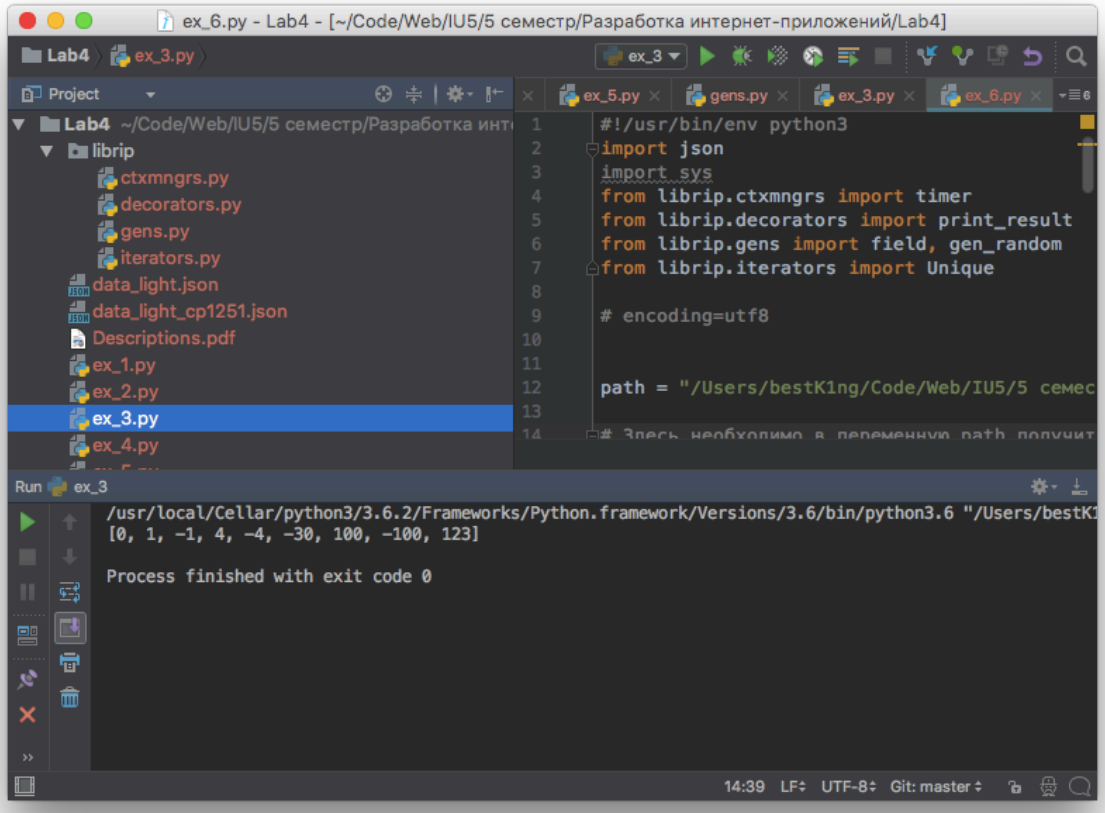
```
#!/usr/bin/env python3
import json
import sys
from librip.ctxmngrs import timer
from librip.decorators import print_result
from librip.gens import field, gen_random
from librip.iterators import Unique

# encoding=utf8


path = "/Users/bestK1ng/Code/Web/IU5/5 семестр/Разработка интернет-
приложений/Lab4/data_light.json"

# Здесь необходимо в переменную path получить
# путь до файла, который был передан при запуске

with open(path, "r", encoding='utf8') 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):
    jobs = list(field(arg, "job-name"))
```

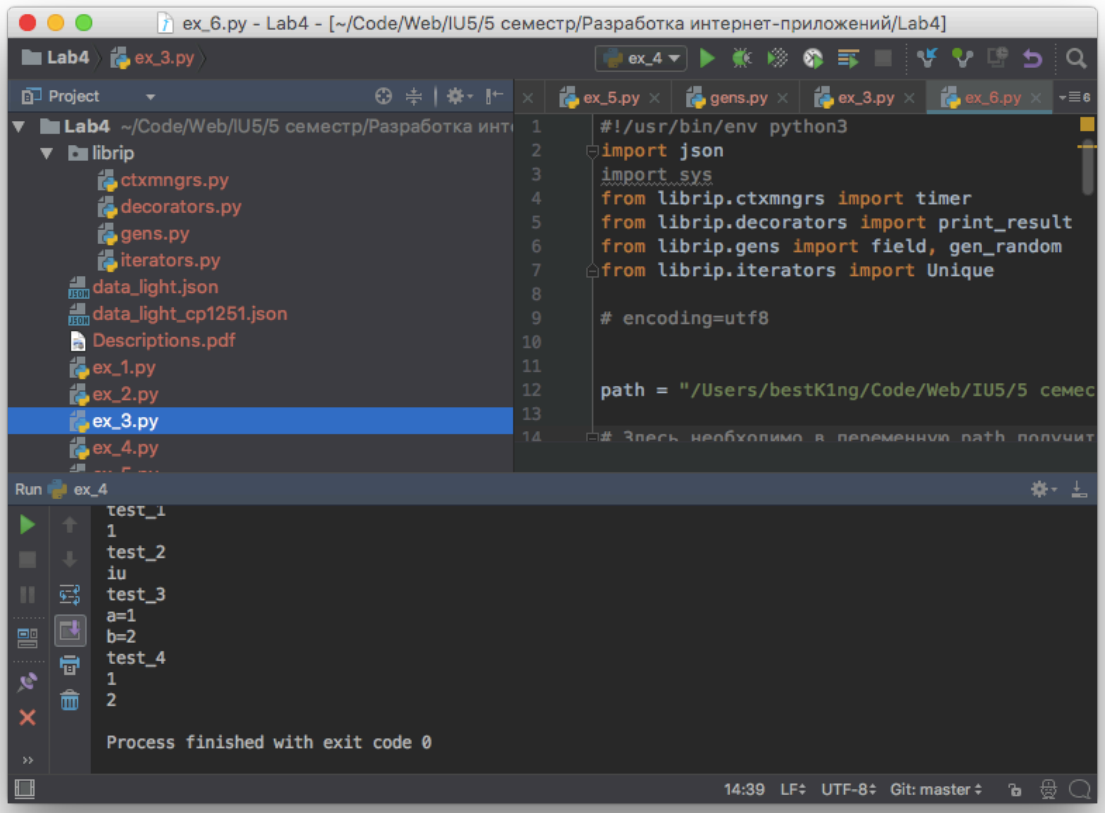
```
    jobs = Unique(jobs, ignore_case=True)
    jobs = sorted(jobs)

    return jobs


@print_result
def f2(arg):
    jobs = list(filter(lambda x: "программист" in x.lower(), arg))

    return jobs
```

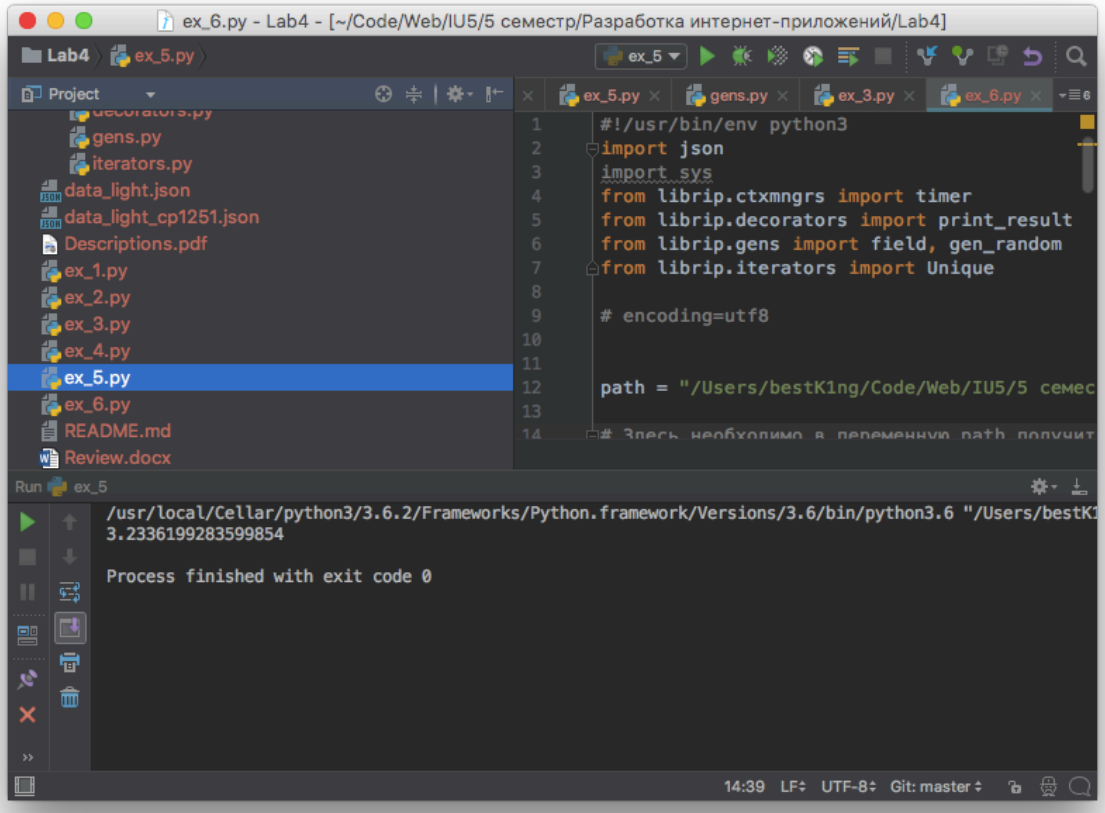
Скришоты выполнения

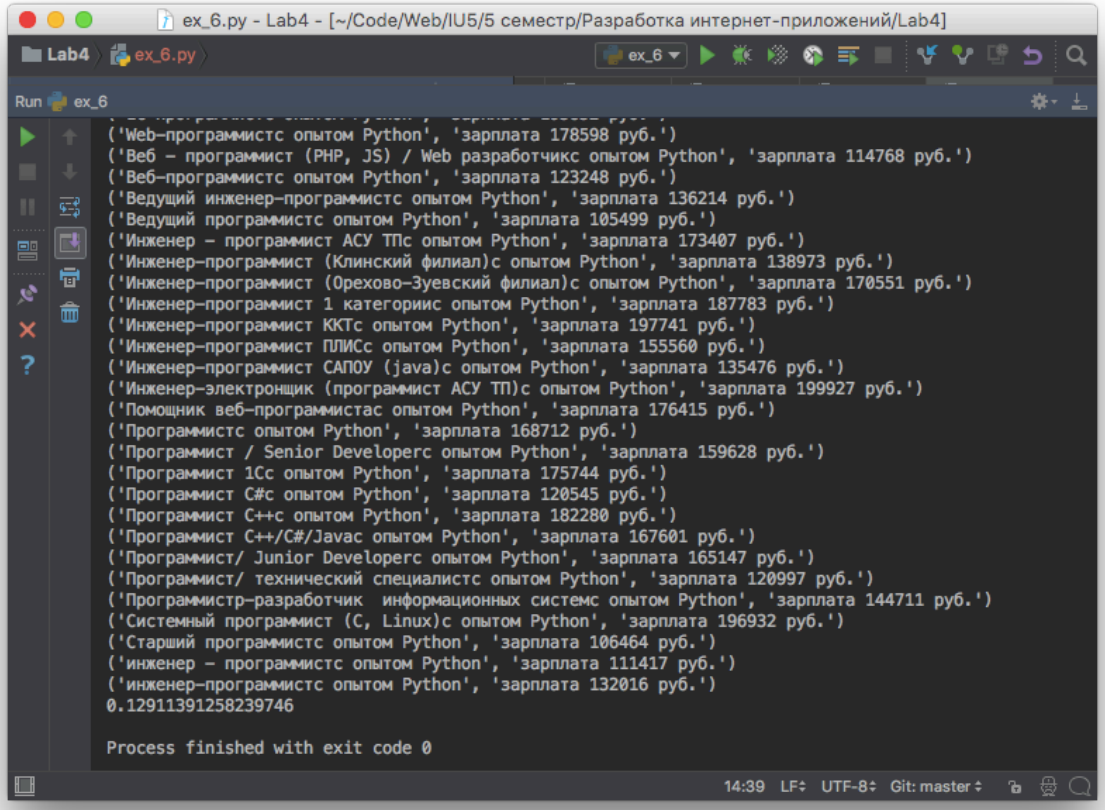



ex_6.py - Lab4 - [~/Code/Web/IU5/5 семестр/Разработка интернет-приложений/Lab4]
Lab4
ex_6.py
ex_2
Project
Lab4 ~/Code/Web/IU5/5 семестр/Разработка инте
librip
ctxmngrs.py
decorators.py
gens.py
iterators.py
data_light.json
data_light_cp1251.json
Descriptions.pdf
ex_1.py
ex_2.py
ex_3.py
ex_4.py
ex_5.py
gens.py
ex_3.py
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
# encoding=utf8
path = "/Users/bestK1ng/Code/Web/IU5/5 семес
Run ex_2
/usr/local/Cellar/python3/3.6.2/Frameworks/Python.framework/Versions/3.6/bin/python3.6 "/Users/bestK1
1 2
A a b B
A b
Process finished with exit code 0
14:39 LF UTF-8 Git: master


ex_6.py - Lab4 - [~/Code/Web/IU5/5 семестр/Разработка интернет-приложений/Lab4]
Lab4
ex_3.py
ex_3
Project
Lab4 ~/Code/Web/IU5/5 семестр/Разработка инте
librip
ctxmngrs.py
decorators.py
gens.py
iterators.py
data_light.json
data_light_cp1251.json
Descriptions.pdf
ex_1.py
ex_2.py
ex_3.py
ex_4.py
ex_5.py
gens.py
ex_3.py
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
# encoding=utf8
path = "/Users/bestK1ng/Code/Web/IU5/5 семес
Run ex_3
/usr/local/Cellar/python3/3.6.2/Frameworks/Python.framework/Versions/3.6/bin/python3.6 "/Users/bestK1
[0, 1, -1, 4, -4, -30, 100, -100, 123]
Process finished with exit code 0
14:39 LF UTF-8 Git: master


ex_6.py - Lab4 - [~/Code/Web/IU5/5 семестр/Разработка интернет-приложений/Lab4]
Lab4
ex_3.py
ex_4
Project
Lab4 ~/Code/Web/IU5/5 семестр/Разработка инт
librip
ctxmngrs.py
decorators.py
gens.py
iterators.py
data_light.json
data_light_cp1251.json
Descriptions.pdf
ex_1.py
ex_2.py
ex_3.py
ex_4.py
ex_5.py
gens.py
ex_3.py
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
# encoding=utf8
path = "/Users/bestK1ng/Code/Web/IU5/5 семес
Run ex_4
test_1
1
test_2
iu
test_3
a=1
b=2
test_4
1
2
Process finished with exit code 0
14:39 LF UTF-8 Git: master


ex_6.py - Lab4 - [~/Code/Web/IU5/5 семестр/Разработка интернет-приложений/Lab4]
Lab4
ex_5.py
ex_5
Project
gens.py
iterators.py
data_light.json
data_light_cp1251.json
Descriptions.pdf
ex_1.py
ex_2.py
ex_3.py
ex_4.py
ex_5.py
ex_6.py
README.md
Review.docx
ex_5.py
gens.py
ex_3.py
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
# encoding=utf8
path = "/Users/bestK1ng/Code/Web/IU5/5 семес
Run ex_5
/usr/local/Cellar/python3/3.6.2/Frameworks/Python.framework/Versions/3.6/bin/python3.6 "/Users/bestK1
3.2336199283599854
Process finished with exit code 0
14:39 LF UTF-8 Git: master


ex_6.py - Lab4 - [~/Code/Web/IU5/5 семестр/Разработка интернет-приложений/Lab4]
Lab4
ex_6.py
ex_6
Run
ex_6
('Web-программистс опытом Python', 'зарплата 178598 руб.')
('Веб - программист (PHP, JS) / Web разработчикс опытом Python', 'зарплата 114768 руб.')
('Веб-программистс опытом Python', 'зарплата 123248 руб.')
('Ведущий инженер-программистс опытом Python', 'зарплата 136214 руб.')
('Ведущий программистс опытом Python', 'зарплата 105499 руб.')
('Инженер - программист АСУ ТПс опытом Python', 'зарплата 173407 руб.')
('Инженер-программист (Клинский филиал)с опытом Python', 'зарплата 138973 руб.')
('Инженер-программист (Орехово-Зуевский филиал)с опытом Python', 'зарплата 170551 руб.')
('Инженер-программист 1 категориис опытом Python', 'зарплата 187783 руб.')
('Инженер-программист ККТс опытом Python', 'зарплата 197741 руб.')
('Инженер-программист ПЛИСс опытом Python', 'зарплата 155560 руб.')
('Инженер-программист САПОУ (java)с опытом Python', 'зарплата 135476 руб.')
('Инженер-электронщик (программист АСУ ТП)с опытом Python', 'зарплата 199927 руб.')
('Помощник веб-программистас опытом Python', 'зарплата 176415 руб.')
('Программистс опытом Python', 'зарплата 168712 руб.')
('Программист / Senior Developerс опытом Python', 'зарплата 159628 руб.')
('Программист 1Сс опытом Python', 'зарплата 175744 руб.')
('Программист C#с опытом Python', 'зарплата 120545 руб.')
('Программист C++с опытом Python', 'зарплата 182280 руб.')
('Программист C++/C#/Javac опытом Python', 'зарплата 167601 руб.')
('Программист/ Junior Developerс опытом Python', 'зарплата 165147 руб.')
('Программист/ технический специалистс опытом Python', 'зарплата 120997 руб.')
('Программистр-разработчик информационных системс опытом Python', 'зарплата 144711 руб.')
('Системный программист (C, Linux)с опытом Python', 'зарплата 196932 руб.')
('Старший программистс опытом Python', 'зарплата 106464 руб.')
('инженер - программистс опытом Python', 'зарплата 111417 руб.')
('инженер-программистс опытом Python', 'зарплата 132016 руб.')
0.12911391258239746
Process finished with exit code 0
14:39
LF
UTF-8
Git: master