Государственное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления
КАФЕДРА Системы обработки информации и управления

Отчёт по лабораторной работе по курсу «Разработка интернет-приложений» Введение в Python

Исполнитель:
студентка группы **РТ5-51**
Карасева А. Д.
Преподаватель: **Гапанюк Ю.Е.**

Москва, 2017

Задача 1 (ex_1.py)

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

Содержание файла «gens.py»:

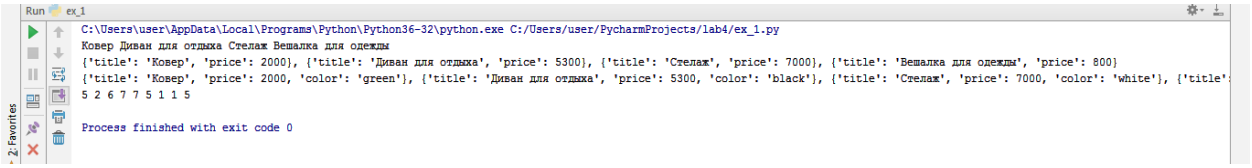
```
import random

# 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
    if len(args) == 1:
        for i in items:
            a = i.get(args[0])
            if a:
                yield a
    else:
        for i in items:
            a = {key: i[key] for key in args if i.get(key)}
            if a:
                yield a


    # Необходимо реализовать генератор


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

Содержание файла “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': 5300, 'color': 'black'},
    {'title': 'Стелаж', 'price': 7000, 'color': 'white'},
    {'title': 'Вешалка для одежды', 'price': 800, 'color': 'white'}
]


# Реализация задания

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

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

print(*gen_random(1, 7, 9))
```

Результат работы программы:



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 **не должен модифицировать** возвращаемые значения.

Содержание файла “iterators.py”:

```
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, Абв и АБВ разные строки
        #           ignore_case = False, Абв и АБВ одинаковые строки, одна из
них удалится
        # По-умолчанию ignore_case = False
        self.ignore_case = kwargs.get('ignore_case', False)
        self.__prev = None
        self.items = list(filter(self.__check_unique, items))
        self.len = len(self.items)
        self.ind = 0

    def __check_unique(self, x):
        if self.ignore_case:
            if type(x) == str and type(self.__prev) == str:
                if x.casefold() == self.__prev.casefold():
                    return False
                if x == self.__prev:
                    return False
            if x == self.__prev:
                return False
        else:
            if x == self.__prev:
                return False

        self.__prev = x
        return True
```

```
    def __next__(self):
        if self.ind == self.len:
            raise StopIteration

        self.ind += 1
        return self.items[self.ind - 1]

    def __iter__(self):
        return self
```

Содержание файла “ex_2.py”:

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

data1 = ['a', 'A', 1, 1, 1, 1, 2, 2, 2, 2, 2]
data2 = gen_random(1, 3, 10)

# Реализация задания 2

kuku = Unique
for i in kuku(data1, ignore_case=True):
   print(i)
```

Результат работы программы:

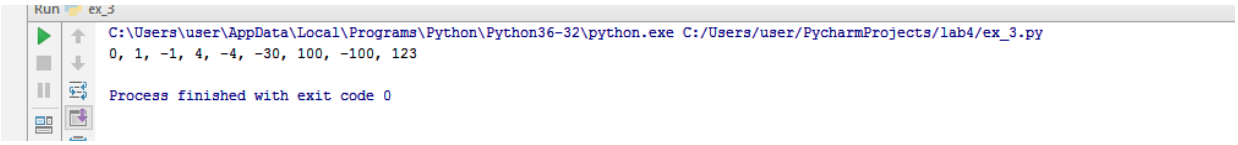
Задача 3 (ex_3.py)

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

Содержание файла “ex_3.py”:

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

data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
# Реализация задания 3
sort = sorted(data, key=lambda x: abs(x))
print(*sort, sep=', ')
```

Результат работы программы:



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

Содержание файла “decorators.py”:

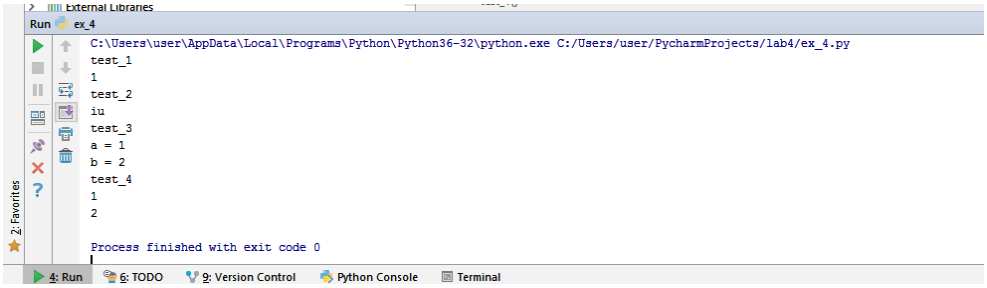
```
def print_result(func):
    def dec_func(*args, **kwargs):
        res = func(*args, **kwargs)
        print(func.__name__)

        if type (res) == list:
            for x in res:
                print(x)

        elif type(res) == dict:
            for key, val in res.items():
                print('{} = {}'.format(key, val))

        else:
            print(res)

        return res

    return dec_func
```

Содержание файла “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()
```

Результат работы программы:



Задача 5 (ex_5.py)

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

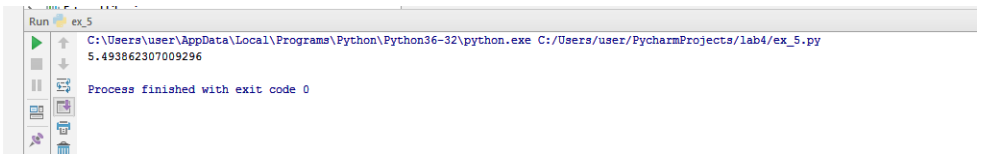
Содержание файла “ctsmngrs.py”:

```
import time

class timer:
    def __init__(self):
        self.time = 0

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

    def __exit__(self, exp_type, exp_value, traceback):
        print(time.clock() - self.time)
```

Содержание файла “ex_5.py”:

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

with timer():
    sleep(5.5)
```

Результат работы программы:



Задача 6 (ex_6.py)

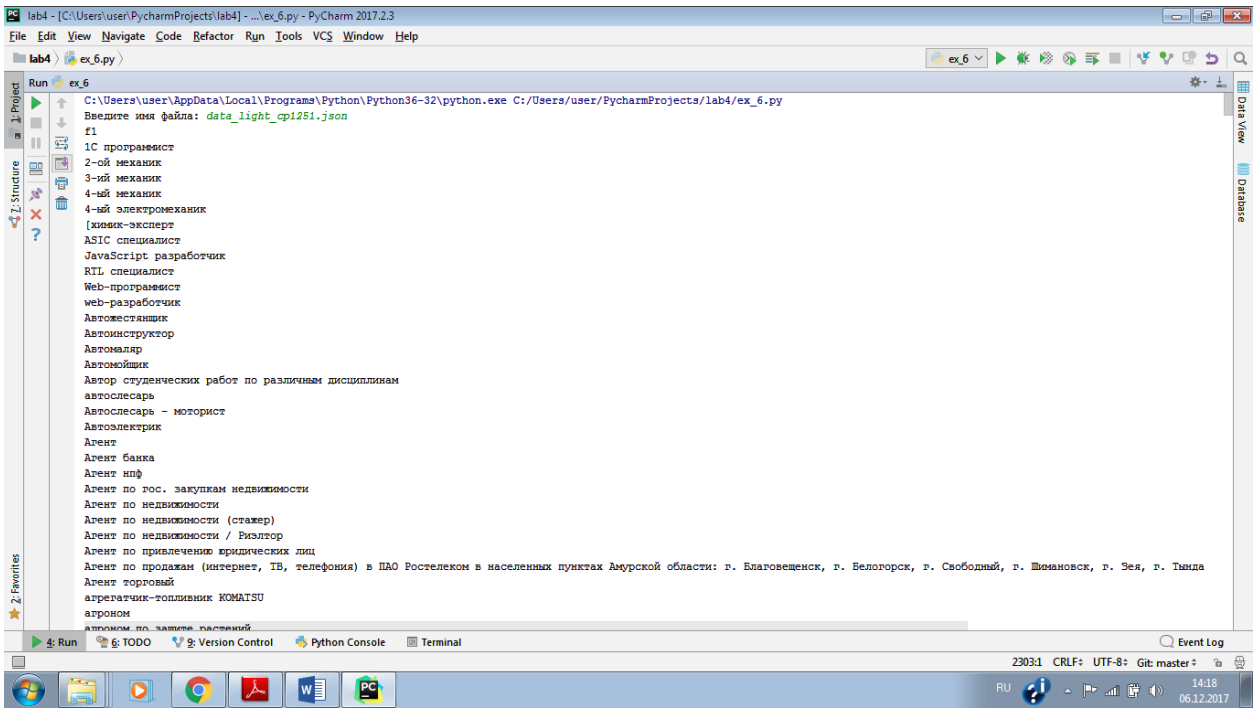
Мы написали все инструменты для работы с данными. Применим их на реальном примере, который мог
возникнуть в жизни. В репозитории находится файл data_light.json . Он содержит облегченный список
вакансий в России в формате json (ссылку на полную версию размером ~ 1 Гб. в формате xml можно найти в
файле 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). П ример: *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.

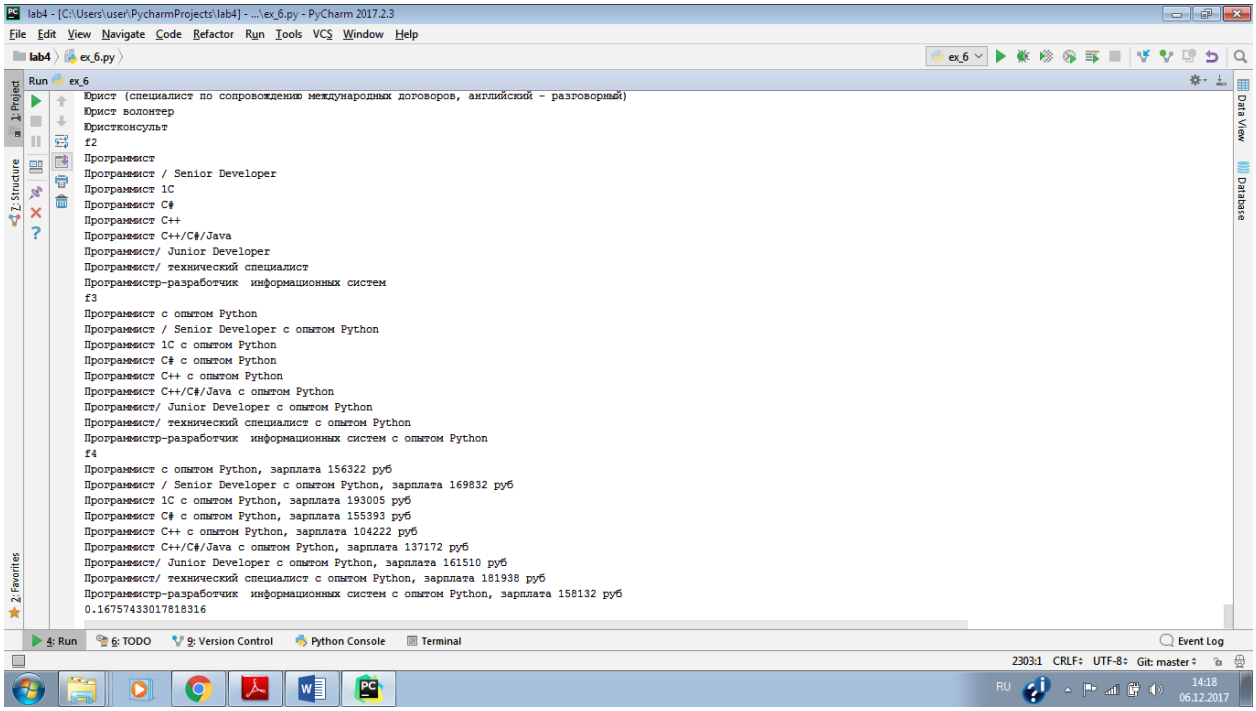
Содержание файла “ex_6.py”:

```
#!/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 = None

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

if len(sys.argv) < 2:
```

```
    path = input('Введите имя файла: ')
else:
    path = sys.argv[1]

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 list(unique(sorted(field(arg, 'job-name'), key=str.casefold),
ignore_case=True))


@print_result
def f2(arg):
    return list(filter(lambda x: x.casefold().startswith('программист'),
arg))


@print_result
def f3(arg):
    return list(map(lambda x: x + ' с опытом Python', arg))


@print_result
def f4(arg):
    salaries = list(gen_random(100000, 200001, len(arg)))
    return list(map(lambda x: x[0] + ', зарплата ' + str(x[1]) + ' руб',
zip(arg, salaries)))


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

Результат работы программы:


lab4 - [C:\Users\user\PycharmProjects\lab4] - ...\ex_6.py - PyCharm 2017.2.3
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Run ex_6
C:\Users\user\AppData\Local\Programs\Python\Python36-32\python.exe C:/Users/user/PycharmProjects/lab4/ex_6.py
Введите имя файла: data_light_cp1251.json
f1
1С программист
2-ой механик
3-ий механик
4-ый механик
4-ый электромеханик
[химик-эксперт
ASIC специалист
JavaScript разработчик
RTL специалист
Web-программист
web-разработчик
Автожестянщик
Автоинструктор
Автомаляр
Автомойщик
Автор студенческих работ по различным дисциплинам
автослесарь
Автослесарь - моторист
Автоэлектрик
Агент
Агент банка
Агент нпф
Агент по рос. закупкам недвижимости
Агент по недвижимости
Агент по недвижимости (стажер)
Агент по недвижимости / Риэлтор
Агент по привлечению юридических лиц
Агент по продажам (интернет, ТВ, телефония) в ПАО Ростелеком в населенных пунктах Амурской области: г. Благовещенск, г. Белогорск, г. Свободный, г. Шимановск, г. Зея, г. Тында
Агент торговый
агрегатчик-топливник KOMATSU
агроном
4: Run 6: TODO 9: Version Control Python Console Terminal Event Log
2303:1 CRLF UTF-8 Git: master
14:18
06.12.2017


lab4 - [C:\Users\user\PycharmProjects\lab4] - ...\ex_6.py - PyCharm 2017.2.3
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Run ex_6
Юрист (специалист по сопровождению международных договоров, английский - разговорный)
Юрист волонтер
Юристконсульт
f2
Программист
Программист / Senior Developer
Программист 1С
Программист C#
Программист C++
Программист C++/C#/Java
Программист/ Junior Developer
Программист/ технический специалист
Программистр-разработчик информационных систем
f3
Программист с опытом Python
Программист / Senior Developer с опытом Python
Программист 1С с опытом Python
Программист C# с опытом Python
Программист C++ с опытом Python
Программист C++/C#/Java с опытом Python
Программист/ Junior Developer с опытом Python
Программист/ технический специалист с опытом Python
Программистр-разработчик информационных систем с опытом Python
f4
Программист с опытом Python, зарплата 156322 руб
Программист / Senior Developer с опытом Python, зарплата 169832 руб
Программист 1С с опытом Python, зарплата 193005 руб
Программист C# с опытом Python, зарплата 155393 руб
Программист C++ с опытом Python, зарплата 104222 руб
Программист C++/C#/Java с опытом Python, зарплата 137172 руб
Программист/ Junior Developer с опытом Python, зарплата 161510 руб
Программист/ технический специалист с опытом Python, зарплата 181938 руб
Программистр-разработчик информационных систем с опытом Python, зарплата 158132 руб
0.16757433017818316
4: Run 6: TODO 9: Version Control Python Console Terminal Event Log
2303:1 CRLF UTF-8 Git: master
14:18
06.12.2017