Отчет по лабораторной работе №4 по курсу «Разработка интернет-приложений»

«Функциональные возможности в Python»

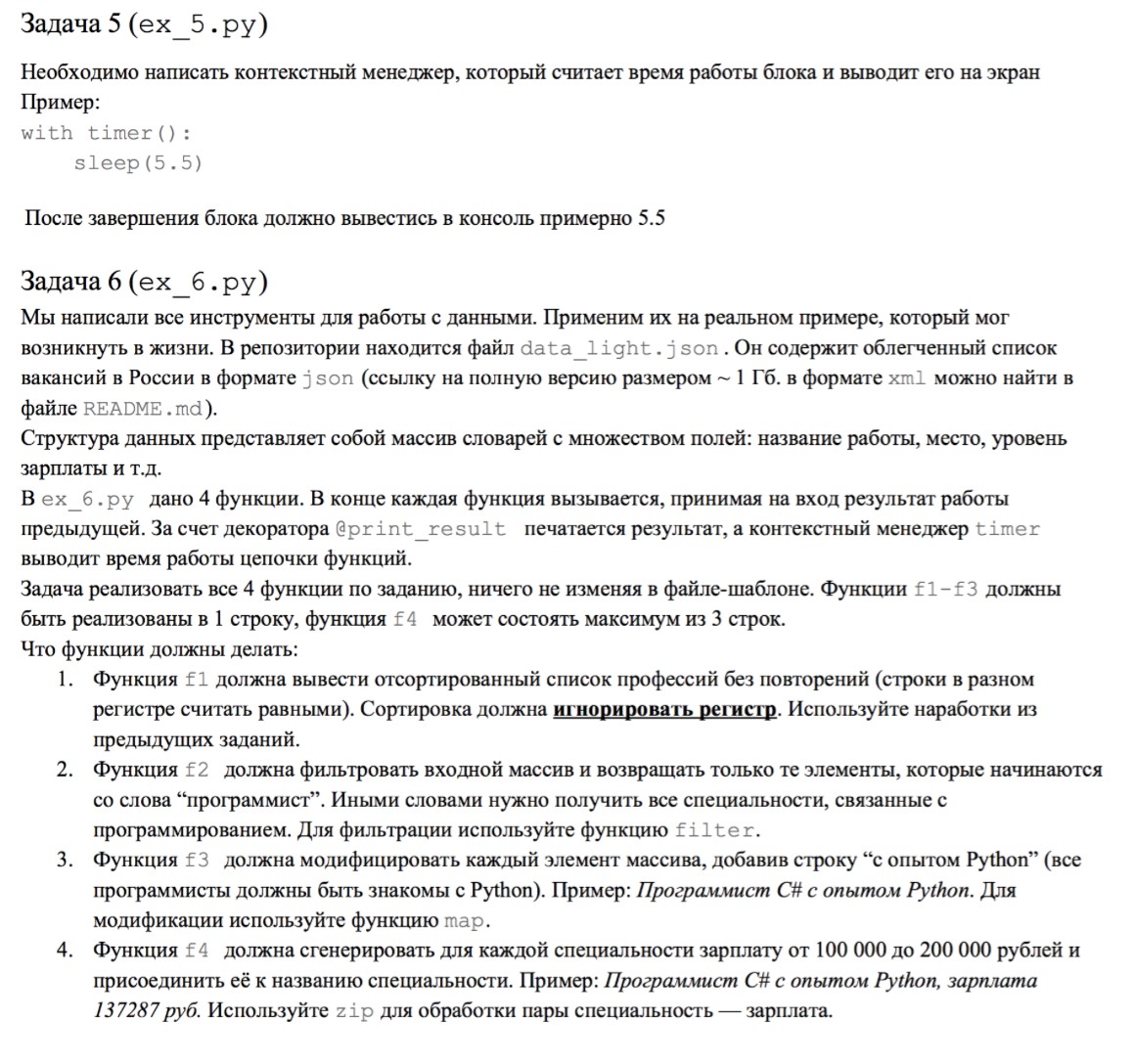
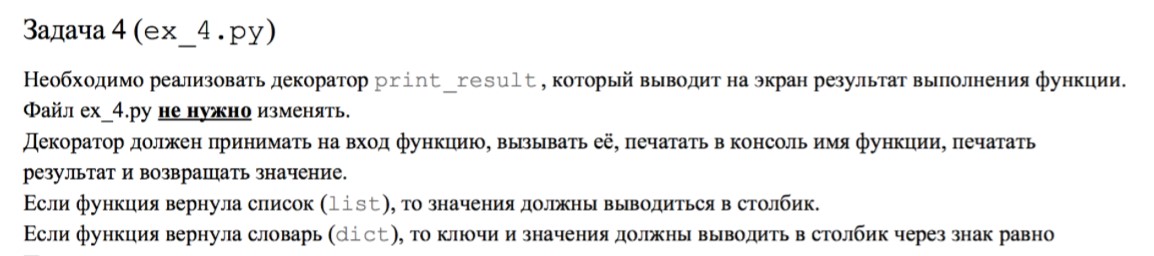
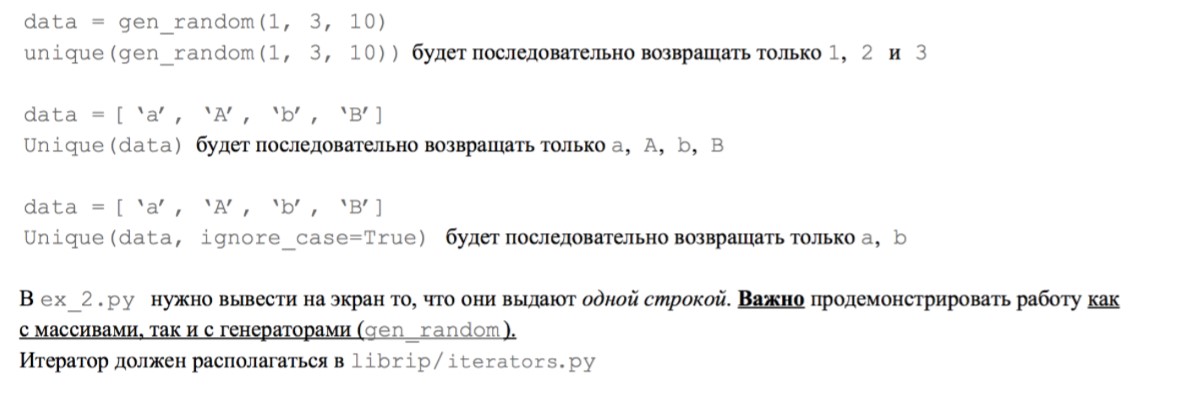
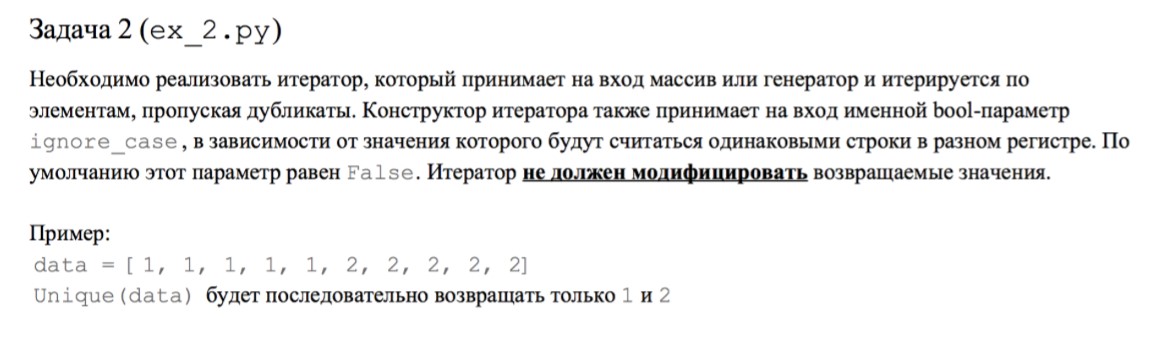
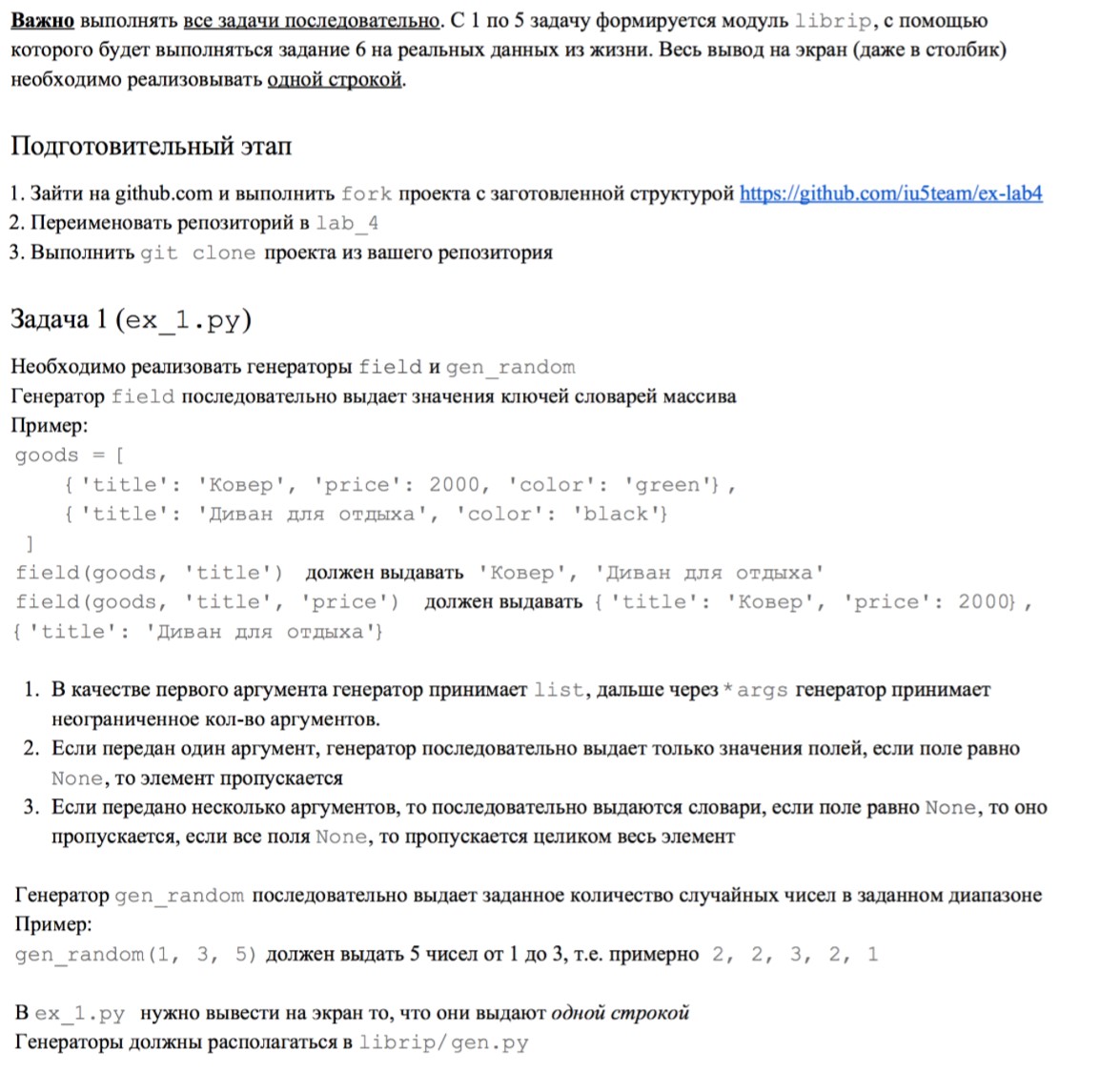
Выполнил: Преподаватель:

Корзин Алексей, ИУ5-51 Гапанюк Ю.Е.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2017 г.

1) Задание лабораторной работы.



2)Код программы librip: gens.py:

**import** random

**def** field(items, \*args):

**assert** len(args) > 0, **'No arguments'** i = 0 **if** len(args) == 1: **while** i < len(items):

**if** items[i].get(args[0]) **is not None**:

**yield** items[i].get(args[0]) i += 1 **else**:

**while** i < len(items):

d = {} **for** el **in** args: **if** items[i].get(el) **is not None**: d[el] = items[i].get(el) **if** len(d) != 0: **yield** d i += 1

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

**yield** random.randint(begin,end)

iterators.py

*# Итератор для удаления дубликатов* **class** Unique(object): **def** \_\_init\_\_(self, items, \*\*kwargs):

self.items = iter(items) **if** isinstance(items, list) **else** items *# проходим по элементам*

self.ignore\_case = kwargs.get(**'ignore\_case'**, **False**) *# По-умолчанию ignore\_case = False* self.lst = set()

**def** \_\_next\_\_(self): **while True**:

el = next(self.items) **if** self.ignore\_case: **if** el.lower() **not in** self.lst: self.lst.add(el.lower()) **return** el **elif** el **not in** self.lst: self.lst.add(el) **return** el

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

decorators.py

**def** print\_result(func): **def** decorated\_func(\*args, \*\*kwargs): *#decorator* print(func.\_\_name\_\_) *#вывод имени функции # test\_1* res=func(\*args, \*\*kwargs) **if** type(res) **is** list: print(**"\n"**.join(map(str,res))) *#преобразование в str* **elif** type(res) **is** dict:

print(**'\n'**.join([str(x)+**"="**+str(res[x]) **for** x **in** res])) **else**:

print(res) **return** res **return** decorated\_func

ctxmngrs.py

**class** timer:

*"""Блок with делает неявный вызов методов \_\_enter\_\_ и \_\_exit\_\_ у объекта. Такой объект называется менеджером контекста."""* **def** \_\_enter\_\_(self):

self.t = time.time() **def** \_\_exit\_\_(self, exp\_type, exp\_value, traceback): print(**'time of block is'**,time.time()-self.t)

lab4: ex\_1.py

**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'**}

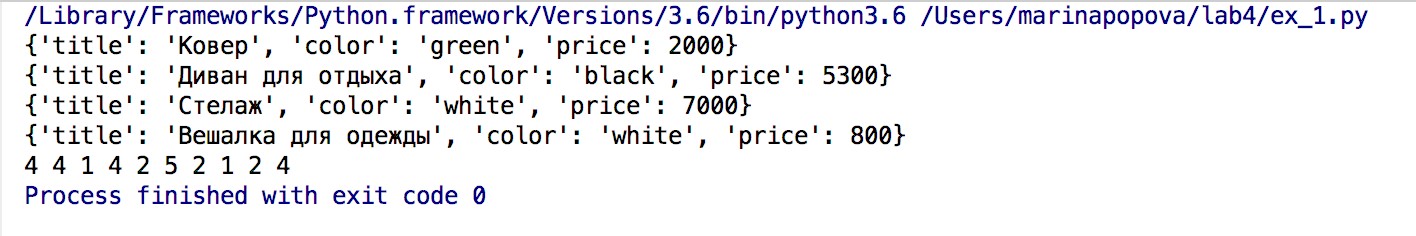
]

*# Реализация задания 1*

g = field(goods, **'title'**, **'color'**, **'price'**) **for** i **in** g:

print(i, end=**"\n"**) num = gen\_random(1,5,10) **for** i **in** num: print(i, end=**" "**)

Результаты:



ex\_2.py

**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'**,**'P'**,**'k'**,**'M'**,**'A'**,**'p'**,**'m'**]

*# Реализация задания 2* print(**'Before Unique'**,data1) u1 = Unique(data1)

print(**'After Unique'**, end = **' '**) **for** i **in** u1:

print(i, end=**" "**)

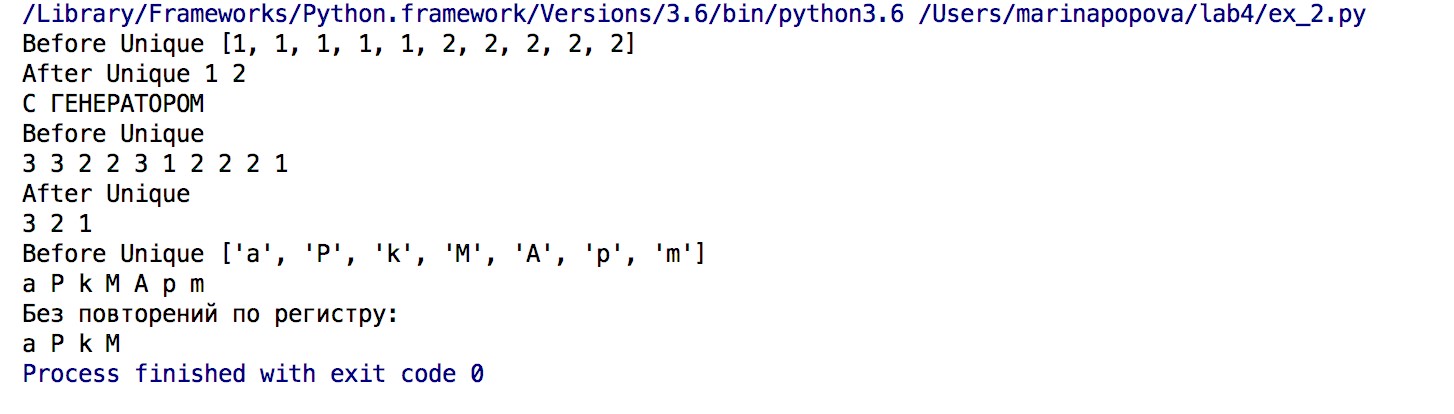
print(**'\n\nC ГЕНЕРАТОРОМ\nBefore Unique'**) d2=[] **for** i **in** data2: d2.append(i) print(i, end=**" "**) print(**'\nAfter Unique'**) u2 = Unique(d2) **for** k **in** u2:

print(k, end=**" "**) print()

print(**'Before Unique'**,data3) u3 = Unique(data3) **for** i **in** u3:

print(i, end=**" "**) print() print(**'Без повторений по регистру:'**) u4 = Unique(data3, ignore\_case = **True**) **for** i **in** u4:

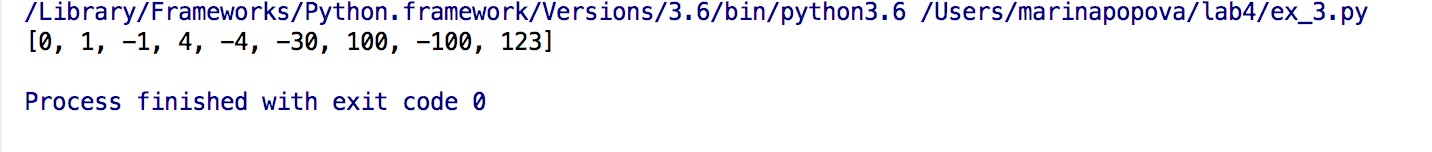
print(i, end=**" "**) Результаты:

 ex\_3.py

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

print(sorted(data, key=abs))

Результаты:

 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()

Результаты:

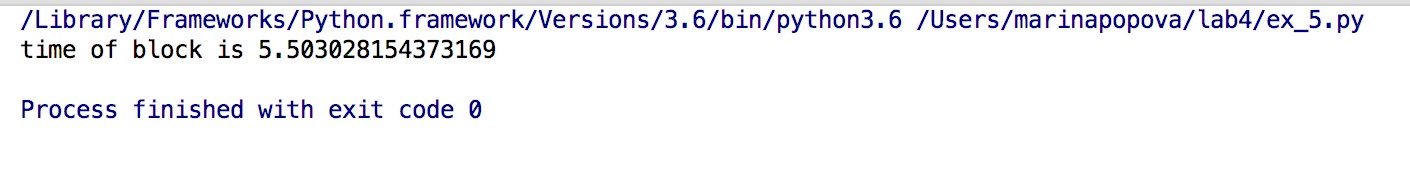
 ex\_5.py

**from** time **import** sleep

**from** librip.ctxmngrs **import** timer

**with** timer(): sleep(5.5)

Результаты:

 ex\_6.py

**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 = sys.argv[1]

*#with open("data\_light\_cp1251.json", encoding="cp1251") as f:*

*# data = json.load(f)*

**with** open(path, encoding=**"cp1251"**) **as** f: data = json.load(f)

@print\_result **def** f1(arg): **return** sorted(unique(field(arg, **'job-name'**), ignore\_case=1), key=**lambda** x: x.lower())

@print\_result **def** f2(arg):

**return** list(filter(**lambda** x: x.startswith(**"Программист"**), arg))

@print\_result **def** f3(arg):

**return** list(map(**lambda** x: x + **" с опытом Python"**, arg)) *# arg уменьшилось по предыдущему фильтру*

@print\_result **def** f4(arg):

s = list(gen\_random(100000, 200000, len(arg)))

**return** list(**'{}, зарплата {} руб.'**.format(arg, s) **for** arg, s **in** zip(arg, s)) **with** timer():

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

Результаты:

