МГТУ им. Н. Э. Баумана
Кафедра «Системы обработки информации и управления»

Лабораторная работа №3
по курсу «Разработка интернет-приложений»
«Функциональные возможности языка Python»

Выполнила:
Ларионова А.П., ИУ5-53Б _____
Преподаватель:
Гапанюк Ю.Е. _____

Москва, 2021 год

Задание:

Задание лабораторной работы состоит из решения нескольких задач.

Файлы, содержащие решения отдельных задач, должны располагаться в пакете lab_python_fp. Решение каждой задачи должно раполагаться в отдельном файле.

При запуске каждого файла выдаются тестовые результаты выполнения соответствующего задания.

Задача 1 (файл field.py)

Необходимо реализовать генератор field. Генератор field последовательно выдает значения ключей словаря. 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'}

- В качестве первого аргумента генератор принимает список словарей, дальше через *args генератор принимает неограниченное количествово аргументов.
- Если передан один аргумент, генератор последовательно выдает только значения полей, если значение поля равно None, то элемент пропускается.
- Если передано несколько аргументов, то последовательно выдаются словари, содержащие данные элементы. Если поле равно None, то оно пропускается. Если все поля содержат значения None, то пропускается элемент целиком.

Задача 2 (файл gen_random.py)

Необходимо реализовать генератор gen_random(количество, минимум, максимум), который последовательно выдает заданное количество случайных чисел в заданном диапазоне от минимума до максимума, включая границы диапазона. Пример:
gen_random(5, 1, 3) должен выдать 5 случайных чисел в диапазоне от 1 до 3, например 2, 2, 3, 2, 1
Шаблон для реализации генератора:

Задача 3 (файл unique.py)

- Необходимо реализовать итератор Unique(данные), который принимает на вход массив или генератор и итерируется по элементам, пропуская дубликаты.
- Конструктор итератора также принимает на вход именованный bool-параметр ignore_case, в зависимости от значения которого будут считаться одинаковыми строки в разном регистре. По умолчанию этот параметр равен False.
- При реализации необходимо использовать конструкцию **kwargs.
- Итератор должен поддерживать работу как со списками, так и с генераторами.
- Итератор не должен модифицировать возвращаемые значения.

Задача 4 (файл sort.py)

Дан массив 1, содержащий положительные и отрицательные числа. Необходимо **одной строкой кода** вывести на экран массив 2, которые содержит значения массива 1, отсортированные по модулю в порядке убывания. Сортировку необходимо осуществлять с помощью функции sorted. Пример:

data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]
Вывод: [123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
Необходимо решить задачу двумя способами:

1. С использованием lambda-функции.
2. Без использования lambda-функции.

Задача 5 (файл print_result.py)

Необходимо реализовать декоратор print_result, который выводит на экран результат выполнения функции.

- Декоратор должен принимать на вход функцию, вызывать её, печатать в консоль имя функции и результат выполнения, после чего возвращать результат выполнения.
- Если функция вернула список (list), то значения элементов списка должны выводиться в столбик.
- Если функция вернула словарь (dict), то ключи и значения должны выводить в столбик через знак равенства.

Задача 6 (файл cm_timer.py)

Необходимо написать контекстные менеджеры cm_timer_1 и cm_timer_2, которые считают время работы блока кода и выводят его на экран. Пример:

```
with cm_timer_1():
    sleep(5.5)
```

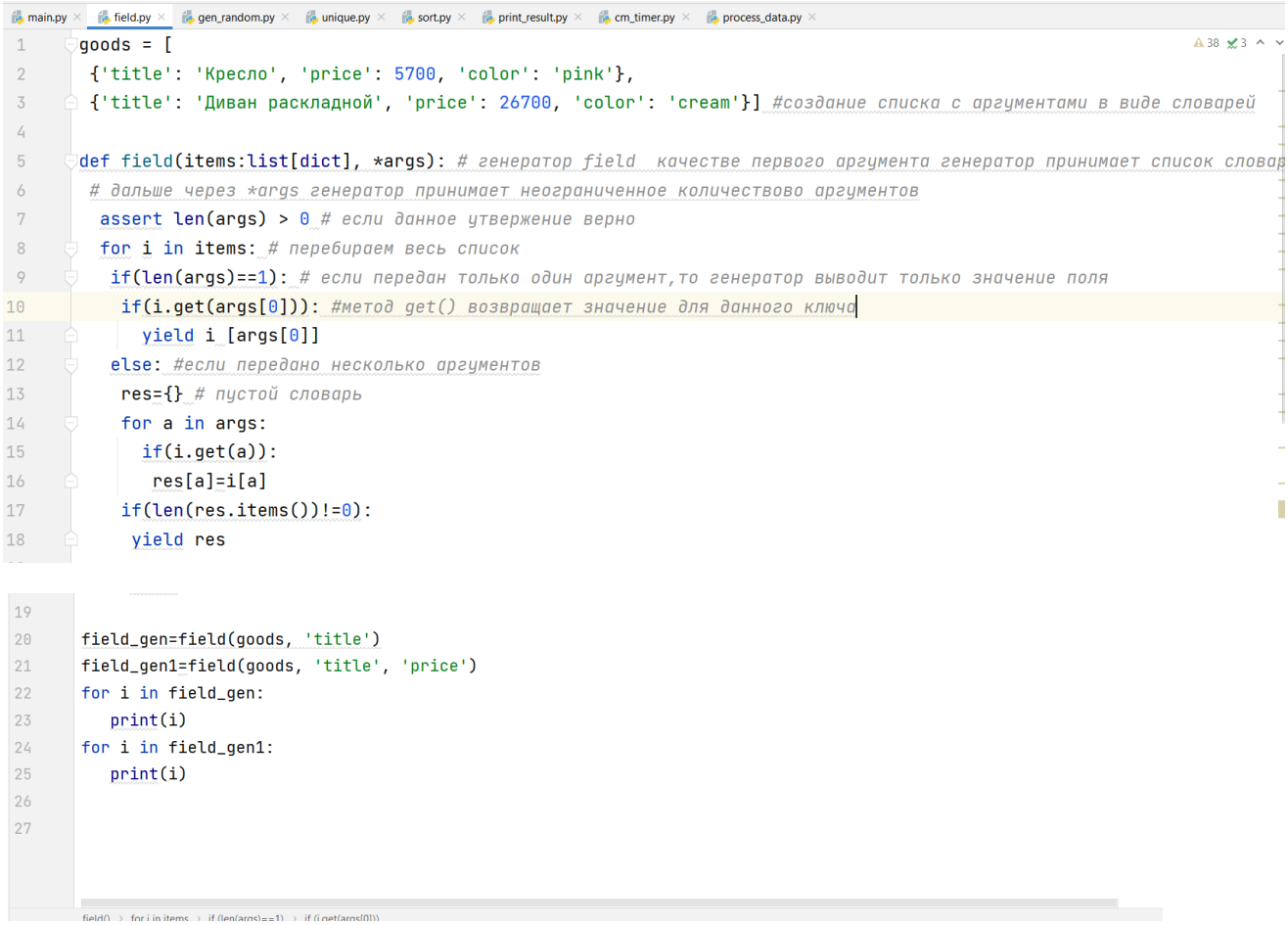
После завершения блока кода в консоль должно вывестись time: 5.5 (реальное время может несколько отличаться).
cm_timer_1 и cm_timer_2 реализуют одинаковую функциональность, но должны быть реализованы двумя различными способами (на основе класса и с использованием библиотеки contextlib).

Задача 7 (файл process_data.py)

- В предыдущих задачах были написаны все требуемые инструменты для работы с данными. Применим их на реальном примере.
- В файле data_light.json содержится фрагмент списка вакансий.
- Структура данных представляет собой список словарей с множеством полей: название работы, место, уровень зарплаты и т.д.
- Необходимо реализовать 4 функции - f1, f2, f3, f4. Каждая функция вызывается, принимая на вход результат работы предыдущей. За счет декоратора @print_result печатается результат, а контекстный менеджер cm_timer_1 выводит время работы цепочки функций.
- Предполагается, что функции f1, f2, f3 будут реализованы в одну строку. В реализации функции f4 может быть до 3 строк.
- Функция f1 должна вывести отсортированный список профессий без повторений (строки в разном регистре считать равными). Сортировка должна игнорировать регистр. Используйте наработки из предыдущих задач.
- Функция f2 должна фильтровать входной массив и возвращать только те элементы, которые начинаются со слова “программист”. Для фильтрации используйте функцию filter.
- 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. Пример: Программист C# с опытом Python, зарплата

137287 руб. Используйте zip для обработки пары специальность — зарплата.

Задача 1 (файл field.py)



```
goods = [
 {'title': 'Кресло', 'price': 5700, 'color': 'pink'},
 {'title': 'Диван раскладной', 'price': 26700, 'color': 'cream'}] #создание списка с аргументами в виде словарей

def field(items:list[dict], *args): # генератор field  качестве первого аргумента генератор принимает список словар
 # дальше через *args генератор принимает неограниченное количествово аргументов
  assert len(args) > 0 # если данное утверждение верно
  for i in items: # перебираем весь список
   if(len(args)==1): # если передан только один аргумент,то генератор выводит только значение поля
    if(i.get(args[0])): #метод get() возвращает значение для данного ключа
      yield i [args[0]]
   else: #если передано несколько аргументов
    res={} # пустой словарь
    for a in args:
      if(i.get(a)):
       res[a]=i[a]
    if(len(res.items())!=0):
     yield res


field_gen=field(goods, 'title')
field_gen1=field(goods, 'title', 'price')
for i in field_gen:
   print(i)
for i in field_gen1:
   print(i)
```

Экранная форма:

```
C:\Users\user\PycharmProjects\pythonProject5\venv\Scripts\python.exe C:/Users/user/PycharmProjects/pythonProject5/lab_python_fp/
Кресло
Диван раскладной
{'title': 'Кресло', 'price': 5700}
{'title': 'Диван раскладной', 'price': 26700}

Process finished with exit code 0
```

Задача 2 (файл gen_random.py)

```
import random
#gen_random(5, 1, 3) должен выдать выдать 5 случайных чисел
# в диапазоне от 1 до 3, например 2, 2, 3, 2, 1

def gen_random(num_count, begin, end):
    for i in range(num_count):
            yield random.randint(begin,end)

gen_random1=gen_random(5,1,9)
for i in gen_random1:
    print(i)
```

Экранная форма:

```
C:\Users\user\PycharmProjects\pythonProject5\venv\Scripts\python.exe C:/Users/user/PycharmProjects/pythonProject5/lab_python_fp/g
5
3
4
3
7

Process finished with exit code 0
```

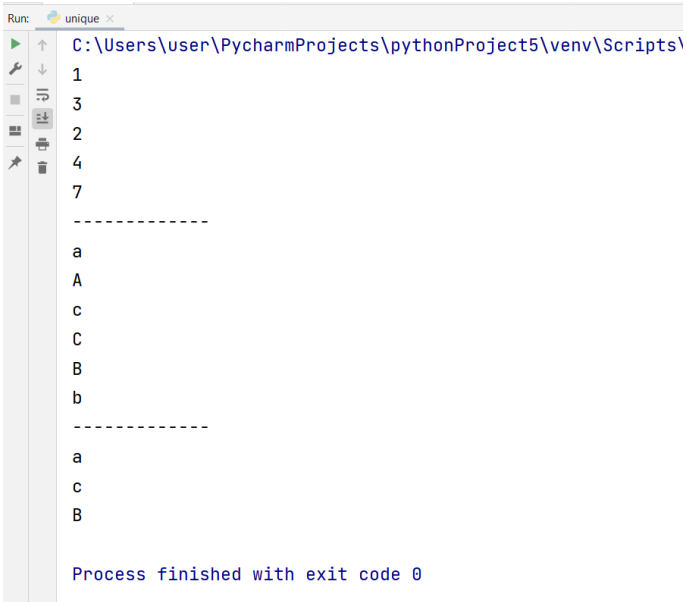
Задача 3 (файл unique.py)

```
# Итератор для удаления дубликатов
class Unique:
    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 одинаковыми строки в разном регистре
        # Например: ignore_case = False, Абв и АБВ - разные строки
        #           ignore_case = True, Абв и АБВ - одинаковые строки, одна из которых удалится
        # По-умолчанию ignore_case = False
        self.used_elements = set() # тут мы будем хранить значения, которые уже занесли для вывода как уникальные
        self.data = items
        self.index = 0
        if 'ignore_case' not in kwargs: # ignore_case-ключевое значение
            self.ignore_case=False
        else:
            self.ignore_case=kwargs['ignore_case']
```

```
17
18      def __next__(self):
19          while True:
20              if self.index >= len(self.data):
21                  raise StopIteration
22              else:
23                  current = self.data[self.index]
24                  self.index = self.index + 1
25                  if self.ignore_case: # если False
26                      if current.lower() not in self.used_elements:
27                          self.used_elements.add(current.lower())
28                          return current
29                  else:
30                      if current not in self.used_elements:
31                      # Добавление в множество производится
32                      # с помощью метода add
33                          self.used_elements.add(current)
34                          return current
35
```

```
36      def __iter__(self):
37          return self
38
39  lst2 = [1,3,2,3,2,1,4,7,3,3]
40  for i in Unique(lst2):
41      print(i)
42  print("-------------")
43  data=['a','A','c','C','C','B','b','b']
44  for a in Unique(data,ignore_case=False):
45      print(a)
46  print("-------------")
47  for a in Unique(data,ignore_case=True):
48      print(a)
49
```

Экранная форма:

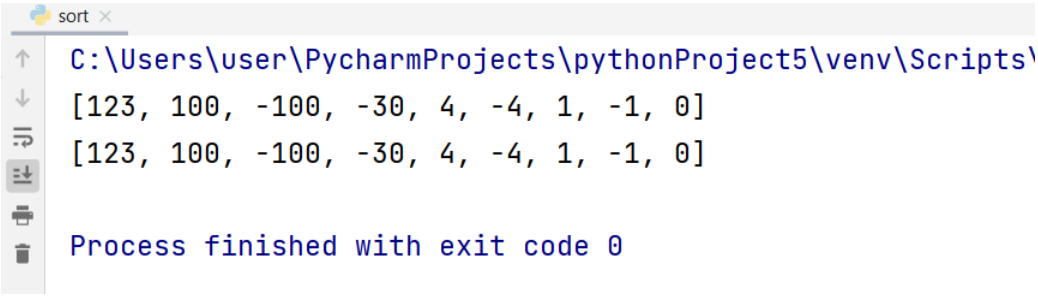


Задача 4 (файл sort.py)

main.py | field.py | gen_random.py | unique.py | sort.py | print_result.py | cm_timer.py | process_data.py

```
1
2   data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
3
4   result=sorted(data,key=abs,reverse=True)
5   print(result)
6
7   result_with_lambda =sorted(data,key=lambda x: abs(x),reverse=True)
8   print(result_with_lambda)
9
```

Экранная форма:



Задача 5 (файл print_result.py)

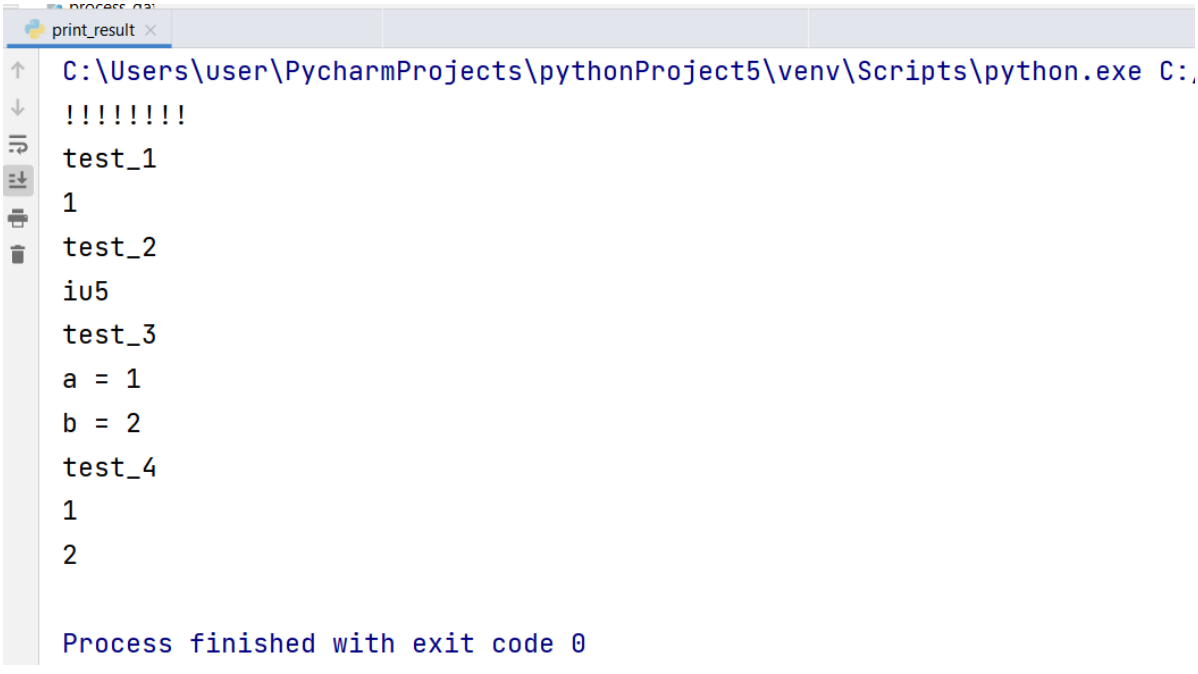
main.py | field.py | gen_random.py | unique.py | sort.py | print_result.py | cm_timer.py | process_data.py

```
1   # Здесь должна быть реализация декоратора
2   def print_result(func_to_decorate):
3       def decorate_func(*args):
4           print(func_to_decorate.__name__)
5           if isinstance(func_to_decorate(*args),list):
6            for v in func_to_decorate(*args):
7               print(v)
8            return func_to_decorate(*args)
9           elif isinstance(func_to_decorate(*args),dict):
10            for key,value in func_to_decorate(*args).items():
11               print ("{} = {}".format(key,value))
12            return func_to_decorate(*args)
13          else:
14              print(func_to_decorate(*args))
15              return func_to_decorate(*args)
16      return decorate_func
17
```

main.py | field.py | gen_random.py | unique.py | sort.py | print_result.py | cm_timer.py | process_data.py

```python
@print_result
def test_1():
    return 1

@print_result
def test_2():
    return 'iu5'

@print_result
def test_3():
    return {'a': 1, 'b': 2}

@print_result
def test_4():
    return [1, 2]

print('!!!!!!!!')
test_1()
test_2()
test_3()
test_4()
```

Экранная форма:



Задача 6 (файл cm_timer.py)

main.py | field.py | gen_random.py | unique.py | sort.py | print_result.py | cm_timer.py | process_data.py

```python
from contextlib import contextmanager
import time

class Cm_timer_1:
    def __init__(self,before_ms,after_ms):
        self.before_ms=before_ms
        self.after_ms=after_ms
    def __enter__(self):
        print(self.before_ms)
        self.time=time.time()
        return self.time
    def __exit__(self, exc_type, exc_val, exc_tb):
        if exc_type is not None:
           print(exc_type,exc_val,exc_tb)
        else:
            print("time1:",time.time()-self.time)
            print(self.after_ms)
before_ms = "Сообщение при входе в контекстный менеджер на основе классса"
after_ms= "Сообщение при выходе из контекстного менеджера на основе классса"
with Cm_timer_1(before_ms,after_ms) as cm_object:
    time.sleep(5.5)


@contextmanager
def cm_timer_2():
    t=time.time() # начальное время
    yield t
    print ("time2:",time.time()-t)# текущее время- начальное
with cm_timer_2():
        time.sleep(5.5)

```

Экранная форма:

cm_timer

```
C:\Users\user\PycharmProjects\pythonProject5\venv\Scripts\python.exe C:/Users/user/
Сообщение при входе в контекстный менеджер на основе классса
time1: 5.514895677566528
Сообщение при выходе из контекстного менеджера на основе классса
time2: 5.510926246643066

Process finished with exit code 0
```

Задача 7 (файл process_data.py)

main.py | field.py | gen_random.py | unique.py | sort.py | print_result.py | cm_timer.py | process_data.py

```python
import json
from gen_random import gen_random
import sys
from lab_python_fp.cm_timer import Cm_timer_1
from lab_python_fp.print_result import print_result
# Сделаем другие необходимые импорты

path ="data_light.json"

# Необходимо в переменную path сохранить путь к файлу, который был передан при запуске сценария

with open(path) as f:
    data = json.load(f) # метод считывает файл в формате JSON и возвращает объекты Python

# Далее необходимо реализовать все функции по заданию, заменив `raise NotImplemented`
# Предполагается, что функции f1, f2, f3 будут реализованы в одну строку
# В реализации функции f4 может быть до 3 строк

@print_result
def f1(arg):
    return sorted(set([p.lower() for p in arg]),key=str.lower)

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


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

@print_result
def f4(arg):
    t=list(zip(arg,[" зарплата " + str(el)+ " руб." for el in list(gen_random(len(arg),100000,200000))]))
    return [e[0]+e[1] for e in t]

if __name__=='__main__':
    before_ms = "Сообщение при входе в контекстный менеджер на основе классса"
    after_ms = "Сообщение при выходе из контекстного менеджера на основе классса"
    with Cm_timer_1(before_ms,after_ms) as cm_object:
        f4(f3(f2(f1([el['job-name'] for el in data]))))

```

Экранная форма:

```
Сообщение при входе в контекстный менеджер на основе классса
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 гос. 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. белогорс
агент торговый
агрегатчик-топливник komatsu
агроном
агроном по защите растений
```

И т.д.

```
юрист (специалист по сопровождению международных договоров, английский - разговорный)
юрист волонтер
юристконсульт
f2
программист
программист / senior developer
программист 1с
программист с#
программист с++
программист с++/с#/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 с# с опытом Python
программист с++ с опытом Python
программист с++/с#/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 161479 руб.
программист / senior developer с опытом Python зарплата 180521 руб.
программист 1с с опытом Python зарплата 132505 руб.
программист с# с опытом Python зарплата 121041 руб.
программист с++ с опытом Python зарплата 182315 руб.
программист с++/с#/java с опытом Python зарплата 171669 руб.
программист/ junior developer с опытом Python зарплата 149993 руб.
программист/ технический специалист с опытом Python зарплата 146235 руб.
программистр-разработчик  информационных систем с опытом Python зарплата 149055 руб.
time1: 0.03988933563232422
Сообщение при выходе из контекстного менеджера на основе классса

Process finished with exit code 0
```