|  |  |
| --- | --- |
|  | Министерство науки и высшего образования Российской Федерации  Федеральное государственное бюджетное образовательное учреждение  высшего образования  «Московский государственный технический университет  имени Н.Э. Баумана  (национальный исследовательский университет)»  (МГТУ им. Н.Э. Баумана) |

Факультет «Радиотехнический»

Кафедра «Системы обработки информации и управления»

Лабораторная работа № 3

по дисциплине «Разработка интернет-приложений».

Выполнил:

студент(ка) группы № РТ5-51Б

А. С. Пакало

подпись, дата

Проверил:

преподаватель

Ю. Е. Гапанюк   
подпись, дата

Оглавление

[Цель работы 3](#_Toc86613887)

[Изучение возможностей функционального программирования в языке Python. 3](#_Toc86613888)

[Задание 3](#_Toc86613889)

[Задача 1 (файл field.py) 3](#_Toc86613890)

[Задача 2 (файл gen\_random.py) 3](#_Toc86613891)

[Задача 3 (файл unique.py) 3](#_Toc86613892)

[Задача 4 (файл sort.py) 4](#_Toc86613893)

[Задача 5 (файл print\_result.py) 4](#_Toc86613894)

[Задача 6 (файл cm\_timer.py) 4](#_Toc86613895)

[Задача 7 (файл process\_data.py) 5](#_Toc86613896)

[Выполнение 6](#_Toc86613897)

[main.py 6](#_Toc86613898)

[Задание 1 (файл field.py) 9](#_Toc86613899)

[Задание 2 (файл gen\_random.py) 10](#_Toc86613900)

[Задание 3 (файл unique.py) 10](#_Toc86613901)

[Задание 4 (файл sort.py) 11](#_Toc86613902)

[Задание 5 (файл print\_result.py) 11](#_Toc86613903)

[Задание 6 (файл cm\_timer.py) 12](#_Toc86613904)

[Задание 7 (файл process\_data.py) 13](#_Toc86613905)

[Результаты выполнения 15](#_Toc86613906)

[Вывод 18](#_Toc86613907)

[На данной лабораторной работе я изучил возможности функционального программирования в языке Python. 18](#_Toc86613908)

# Цель работы

# Изучение возможностей функционального программирования в языке Python.

# Задание

## Задача 1 (файл field.py)

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

## 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.

## 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, который содержит значения массива 1, отсортированные по модулю в порядке убывания. Сортировку необходимо осуществлять с помощью функции sorted.

## 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.

## Задача 7 (файл process\_data.py)

В предыдущих задачах были написаны все требуемые инструменты для работы с данными. Необходимо применить их на реальном примере.

В файле https://github.com/iu5team/iu5web-fall-2021/tree/main/notebooks/fp/files/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.

# Выполнение

## main.py

from lab\_python\_fp.field import field

from lab\_python\_fp.gen\_random import gen\_random

from lab\_python\_fp.unique import Unique

from lab\_python\_fp.sort import sort

from lab\_python\_fp.print\_task import print\_task

from lab\_python\_fp.print\_result import print\_result

from lab\_python\_fp.cm\_timer1 import CmTimer

from lab\_python\_fp.cm\_timer2 import cm\_timer2

from lab\_python\_fp.process\_data import process\_data

@print\_task

def task1() -> None:

goods = [

{'title': 'Ковер', 'price': 2000, 'color': 'green'},

{'title': 'Диван для отдыха', 'color': 'black'}

]

print('Titles:')

for good in field(goods, 'title'):

print(good)

print()

print('Prices:')

for good in field(goods, 'price'):

print(good)

print()

print('Titles and prices:')

for good in field(goods, 'title', 'price'):

print(good)

@print\_task

def task2() -> None:

for r in gen\_random(5, 1, 3):

print(r)

@print\_task

def task3() -> None:

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

print('case sensitive:')

for d in Unique(data):

print(d)

print('case insensitive:')

for d in Unique(data, True):

print(d)

@print\_task

def task4() -> None:

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

sort(data)

@print\_task

def task5() -> None:

@print\_result

def test\_primitive\_1():

return 1

@print\_result

def test\_primitive\_2():

return 'iu5'

@print\_result

def test\_dictionary():

return {'a': 1, 'b': 2}

@print\_result

def test\_list():

return [1, 2]

test\_primitive\_1()

test\_primitive\_2()

test\_dictionary()

test\_list()

@print\_task

def task6() -> None:

num\_of\_iterations = 10\_000\_000

def test\_function() -> None:

for i in range(num\_of\_iterations):

continue

print(f'{num\_of\_iterations:,.0f} iterations were completed in:')

with CmTimer() as test\_timer1:

test\_function()

print('(class implementation)')

print()

with cm\_timer2() as test\_timer2:

test\_function()

print('(contextlib @contextmanager decorator implementation)')

@print\_task

def task7() -> None:

path = '../../notebooks/fp/files/data\_light.json'

process\_data(path)

def main() -> None:

task1()

task2()

task3()

task4()

task5()

task6()

task7()

if \_\_name\_\_ == "\_\_main\_\_":

main()

## Задание 1 (файл field.py)

def field(items, \*selected\_fields):

num\_of\_selected\_fields = len(selected\_fields)

assert num\_of\_selected\_fields > 0, 'No fields for selection! Please pass at least one!'

if (num\_of\_selected\_fields == 1):

return (item.get(selected\_fields[0]) for item in items

if item.get(selected\_fields[0]) != None)

return (

{sf:item.get(sf) for sf in selected\_fields

if item.get(sf) != None}

for item in items

)

## Задание 2 (файл gen\_random.py)

from random import randrange

def gen\_random(num\_count, begin, end):

for i in range(num\_count):

# randrange works in [begin, end) diapason.

yield randrange(begin, end + 1)

## Задание 3 (файл unique.py)

# Iterating through unique values of a data.

class Unique:

def \_\_init\_\_(self, data, ignore\_case=False):

self.used\_elements = set()

self.data = list(data)

self.index = 0

self.ignore\_case = ignore\_case

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

return self

def \_\_next\_\_(self):

while True:

if self.index >= len(self.data):

raise StopIteration

else:

current = self.data[self.index]

self.index = self.index + 1

if (self.ignore\_case):

current\_lowered = current.lower()

if current\_lowered not in self.used\_elements:

self.used\_elements.add(current\_lowered)

return current

else:

if current not in self.used\_elements:

self.used\_elements.add(current)

return current

## Задание 4 (файл sort.py)

# Iterating through unique values of a data.

class Unique:

def \_\_init\_\_(self, data, ignore\_case=False):

self.used\_elements = set()

self.data = list(data)

self.index = 0

self.ignore\_case = ignore\_case

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

return self

def \_\_next\_\_(self):

while True:

if self.index >= len(self.data):

raise StopIteration

else:

current = self.data[self.index]

self.index = self.index + 1

if (self.ignore\_case):

current\_lowered = current.lower()

if current\_lowered not in self.used\_elements:

self.used\_elements.add(current\_lowered)

return current

else:

if current not in self.used\_elements:

self.used\_elements.add(current)

return current

## Задание 5 (файл print\_result.py)

# Iterating through unique values of a data.

class Unique:

def \_\_init\_\_(self, data, ignore\_case=False):

self.used\_elements = set()

self.data = list(data)

self.index = 0

self.ignore\_case = ignore\_case

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

return self

def \_\_next\_\_(self):

while True:

if self.index >= len(self.data):

raise StopIteration

else:

current = self.data[self.index]

self.index = self.index + 1

if (self.ignore\_case):

current\_lowered = current.lower()

if current\_lowered not in self.used\_elements:

self.used\_elements.add(current\_lowered)

return current

else:

if current not in self.used\_elements:

self.used\_elements.add(current)

return current

## Задание 6 (файл cm\_timer.py)

from time import time  
from contextlib import contextmanager

class CmTimer:

def \_\_init\_\_(self):

self.\_start\_time = None

def \_\_enter\_\_(self):

self.\_start\_time = time()

return

def \_\_exit\_\_(self, exp\_type, exp\_value, traceback):

if exp\_type is not None:

print(exp\_type, exp\_value, traceback)

else:

print('-------------------')

print(time() - self.\_start\_time)

print('-------------------')

@contextmanager

def cm\_timer2():

start\_time = time()

yield

print('-------------------')

print(time() - start\_time)

print('-------------------')

## Задание 7 (файл process\_data.py)

from json import load

from lab\_python\_fp.print\_result import print\_result

from lab\_python\_fp.unique import Unique

from lab\_python\_fp.field import field

from lab\_python\_fp.cm\_timer2 import cm\_timer2

from lab\_python\_fp.gen\_random import gen\_random

@print\_result

def f1(data):

return sorted(Unique(field(data, 'job-name'), True), key=str.lower)

@print\_result

def f2(vacancies):

return list(filter(lambda v: v.lower().startswith('программист'), vacancies))

@print\_result

def f3(programmers):

return list(map(lambda programmer: f'{programmer} с опытом Python', programmers))

@print\_result

def f4(vacancies):

salaries = list(gen\_random(len(vacancies), 100\_000, 200\_000))

return list(map(lambda v: f'{v[0]}, запрлата {v[1]} руб.', zip(vacancies, salaries)))

def process\_data(path) -> None:

with open(path, encoding='utf8') as f:

data = load(f)

with cm\_timer2() as process\_data\_timer:

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

## Результаты выполнения

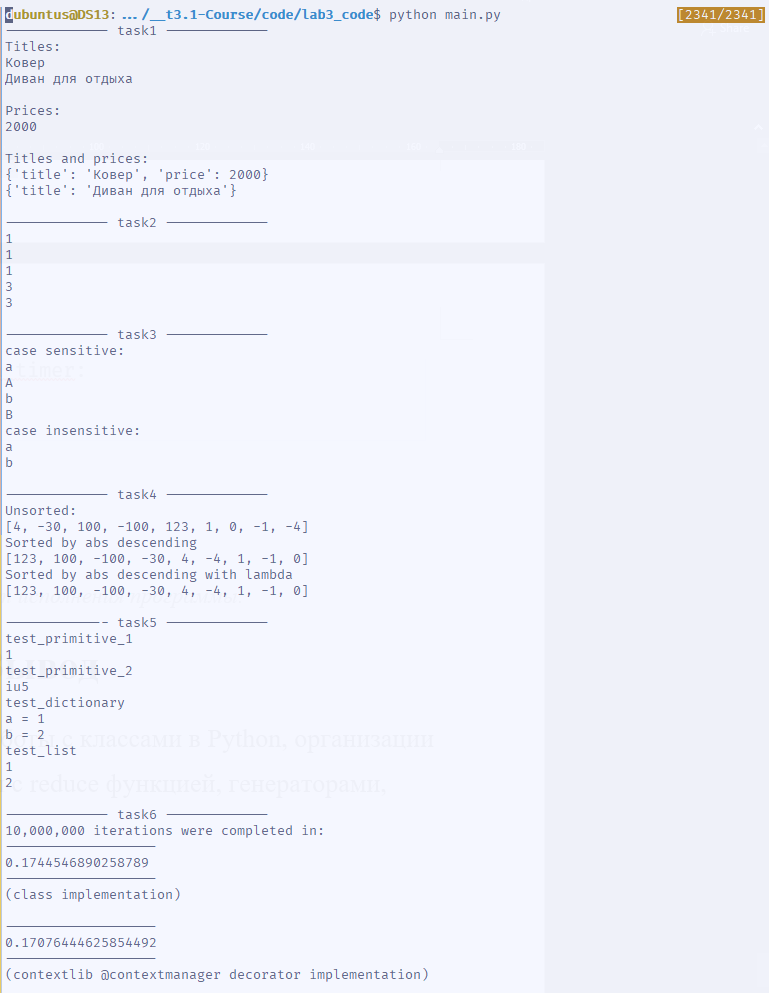


рис. результат выполнения заданий 1-6

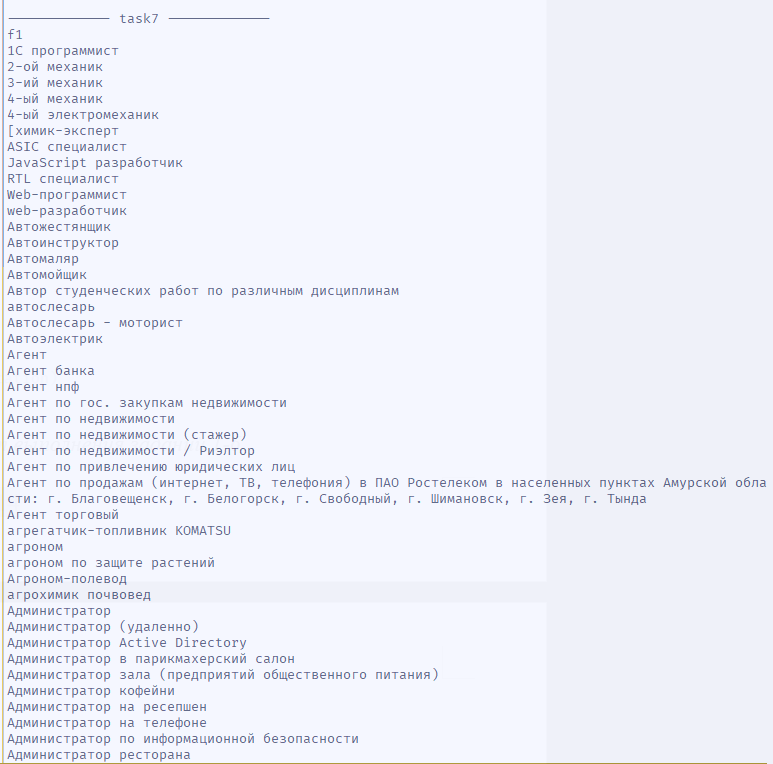


рис. начало результата выполнения функции f1 задания 7

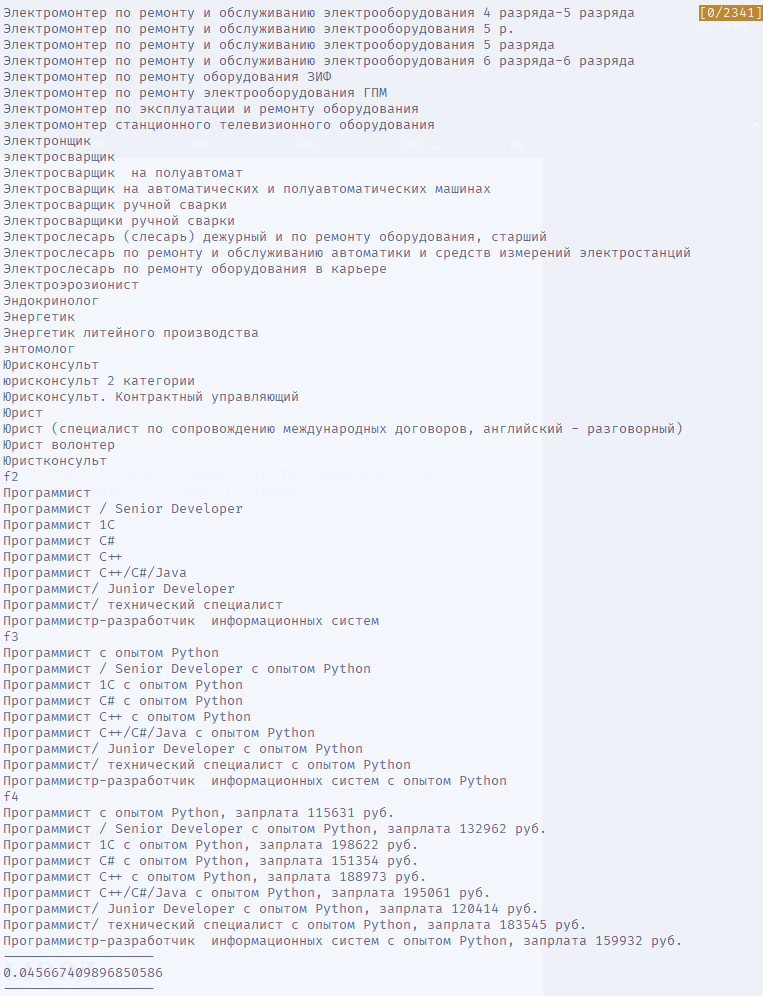


рис. конец результата выполнения функции f1, а также результат выполнения функций f2-f4 задания 7

# Вывод

# На данной лабораторной работе я изучил возможности функционального программирования в языке Python.