



**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 «Информатика и системы управления»  
Кафедра ИУ5 «Системы обработки информации и управления»**

Лабораторная работа №3

по дисциплине «Базовые компоненты интернет-технологий»

Выполнил:  
студент группы ИУ5-32Б  
Долинский А.А.

Проверил:  
Канев А.И.  
2021 г.

## Общее описание задания

Задание лабораторной работы состоит из решения нескольких задач.

Файлы, содержащие решения отдельных задач, должны располагаться в пакете lab\_python\_fr. Решение каждой задачи должно располагаться в отдельном файле.

При запуске каждого файла выдаются тестовые результаты выполнения соответствующего задания.

### Задача №1

Описание задачи

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

В качестве первого аргумента генератор принимает список словарей, дальше через \*args генератор принимает неограниченное количество 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.

Текст программы

```
def field(items, *args):    assert
len(args) > 0    if len(args) ==
1:        for string in items:
text = string.get(args[0])
if text is not None:
yield text
else:
```

```

        for j in items:
            string = dict()
for key in args:
    text = j.get(key)
if text is not None:
    string[key] = text        if
    len(string) != 0:
yield string

if __name__ == '__main__':
    goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'price': 5300, 'color': None},
        {'title': 'Стол', 'price': None, 'color': 'white'},
        {'title': None, 'price': 2021, 'color': 'black'}
    ]

    data1 = list()    for i in
field(goods, 'title'):
data1.append(i)
print(data1)    data2 = list()

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

data2.append(i)
print(data2)    data3
= list()

    for i in field(goods, 'title', 'price', 'color'):
        data3.append(i)
print(data3)

```

Экранные формы с примерами выполнения программы

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

field(goods, 'title')
field(goods, 'title', 'price')
```

```
Ковер, Диван для отдыха
{'title': 'Ковер', 'price': '2000'}
{'title': 'Диван для отдыха'}
```

## Задача №2

Описание задачи

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.

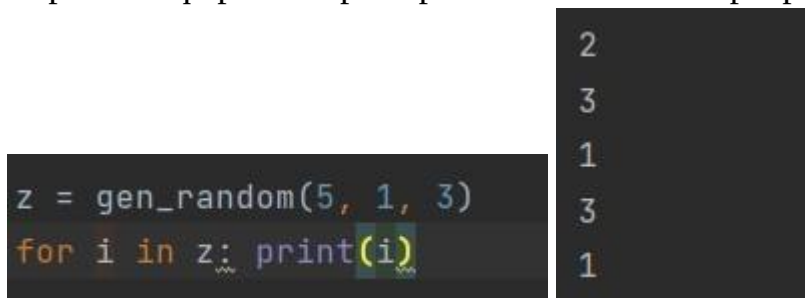
Текст программы

```
import random

def gen_random1(num_count, begin, end):
    for i in range(num_count):
        print(random.randint(begin, end))

def gen_random2(num_count, begin, end):
    for i in range(num_count):
        yield random.randint(begin, end)
if __name__ == '__main__':
    gen_random1(5,1,3)
    print(list(gen_random2(5, 1, 3)))
```

Экранные формы с примерами выполнения программы



```
z = gen_random(5, 1, 3)
for i in z: print(i)
```

```
2
3
1
3
1
```

### Задача №3

Описание задачи

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.

Текст программы

```
from gen_random import gen_random2

class Unique(object):
    def __init__(self, items, **kwargs):
        self.data = iter(items) # метод iter() используется для
        # получения итератора
        self.word = set() # создание пустого
        # множества
        if kwargs:
            self.app =
            kwargs['ignore_case']
```

```

    else:
self.app = False

    def __iter__(self): # возвращаем объект итератора
        return self

    def __next__(self): # вернуть следующий элемент в последовательности
while True:
    x = next(self.data) # перебираются элементы
    if self.app == True and type(x) != int: # если это буквенные
СИМВОЛЫ        x = x.lower() # делает все элементы нижнего регистра
    if x not in self.word:
        self.word.add(x) # добавляет элемент если его ещё не было
    return x

if __name__ == "__main__":
    data1 = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
    data2 = gen_random2(10, 1, 3)
    data3 = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']

    print('Первый пример')
    print(list(Unique(data1)))
    print('Второй пример')
    print(list(Unique(data2)))
    print('Третий пример')
    print(list(Unique(data3, ignore_case=False))) # игнорируем регистр
    print('Четвёртый пример')
    print(list(Unique(data3, ignore_case=True))) # не игнорируем регистр

```

## Экранные формы с примерами выполнения программы

<code>data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]</code>	1
<code>for i in Unique(data):</code>	2
<code>print(i)</code>	
<code>print()</code>	10
<code>data = gen_random(5, 3, 10)</code>	8
<code>for i in Unique(data):</code>	6
<code>print(i)</code>	7
<code>print()</code>	5
<code>data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']</code>	
<code>for i in Unique(data):</code>	a
<code>print(i)</code>	b

## Задача №4

### Описание задачи

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`.

Необходимо решить задачу двумя способами:

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

### Текст программы

```
from functools import reduce
data = [4, -30, 100, -100, 123, 1, 0, -1, -4]

if __name__ == '__main__':
    result = sorted(data, key=abs, reverse=True)
    print(result)

    result_with_lambda = sorted(data, key=lambda x: abs(x), reverse=True)
    print(result_with_lambda)
```

Экранные формы с примерами выполнения программы

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

## Задача №5

Описание задачи

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 выводиться в столбик через знак равенства.

Текст программы

```
def print_result(function): # реализация декоратора print_result    def
decorated(*a): # *a так как неизвестно количество аргументов
    print(function.__name__)    if type(function(*a)) == list:        for
x in function(*a):
        print(x)    elif
type(function(*a)) == dict:
    for x in function(*a):
        print(x, '=', function(*a)[x])
    else:        print(function(*a))
    return decorated

@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]

if __name__ == '__main__':
    test_1()
    test_2()
    test_3()
    test_4()

```

Экранные формы с примерами выполнения программы

```

!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2

```

## Задача №6

### Описание задачи

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

### Текст программы

```
import time from contextlib import
contextmanager

class cm_timer_1:      #Первая реализация через класс def
    __enter__(self):    self.time = time.time() def __exit__(self,
exc_type, exc_val, exc_tb):    print("Время выполнения
программы: ", time.time() - self.time)

@contextmanager
def cm_timer_2():
    start_time = time.time()
    yield
    end_time = time.time() - start_time
    print("Время выполнения программы - {}".format(end_time))
```

### Экранные формы с примерами выполнения программы

```
Время выполнения программы:  2.510998487472534
Время выполнения программы - 2.510741949081421
```

## Задача №7

### Описание задачи

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.

Текст программы

```
import json import sys from print_result
import print_result from cm_timer import
cm_timer_1 from unique import Unique
from field import field from gen_random import gen_random path =
"C:\Users\LENOVO\PycharmProjects\4\lab_python_fp\data_light.json"

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

with open(path, 'r', encoding='utf8') as f:    data =
json.load(f)
```

```

# 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
list(Unique(field(arg, 'job-name'))))

@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))

@print_result def f4(arg):    salary =
gen_random(len(arg), 100000, 200000)    return
list(zip(arg, salary))

if __name__ == '__main__':
with cm_timer_1():
f4(f3(f2(f1(data))))

```

## Экранные формы с примерами выполнения программы

```
f1
Администратор на телефоне
Медицинская сестра
Охранник сутки-день-ночь-вахта
ВРАЧ АНЕСТЕЗИОЛОГ РЕАНИМАТОЛОГ
теплотехник
разнорабочий
Электро-газосварщик
Водитель Gett/Гетт и Yandex/Яндекс такси на личном автомобиле
Монолитные работы
Организатор – тренер
Помощник руководителя
Автоэлектрик
Врач ультразвуковой диагностики в детскую поликлинику
Менеджер по продажам ИТ услуг (B2B)
Менеджер по персоналу
Аналитик
Воспитатель группы продленного дня
Инженер по качеству
Инженер по качеству 2 категории (класса)
Водитель автомобиля
Пекарь
Переводчик
Терапевт
врач-анестезиолог-реаниматолог
Инженер-конструктор в наружной рекламе
Монтажник-сборщик рекламных конструкций
Оператор фрезерно-гравировального станка
Зоотехник
Сварщик
Рабочий-строитель
врач-трансфузиолог
```

врач-трансфузиолог  
Юрисконсульт  
Специалист отдела автоматизации  
Растворщик реагентов  
Бармен  
Официант  
Технолог  
Фельдшер-лаборант  
Медицинская сестра по физиотерапии  
врач функциональной диагностики  
Рентгенолаборант  
диспетчер по навигации  
водитель погрузчика, штабелер  
Машинист автогрейдера  
наладчик ЧПУ  
УПАКОВЩИК-ГРУЗЧИК  
Слесарь по ремонту обогатительного оборудования  
Слесарь тепловодоснабжения и вентиляции  
главный специалист Отдела ЖКХ  
Механик по ремонту спецтехники и тракторов  
Мастер леса Сосновского участкового лесничества  
Врач общей практики  
Мастер леса Кытлымского участкового лесничества  
Врач-педиатр  
Водитель автомобиля Волчанского участкового лесничества  
Водитель автомобиля Кытлымского участкового лесничества  
Врач-психиатр участковый  
Санитар  
Врач-хирург  
врач-педиатр участковый

менеджер по работе с клиентами  
Уборщик производственных и служебных помещений  
Дворник  
Заведующий ФАП - фельдшер д. Киселево  
Ведущий специалист отдела отчетности  
Бухгалтер по расчету заработной платы  
Ведущий специалист отдела учета дебиторской и кредиторской задолженности  
Врач-гастроэнтеролог  
Дробильщик ЗИФ  
Слесарь по ремонту горного оборудования  
Электромонтер по ремонту оборудования ЗИФ  
Водитель автобуса ПАЗ  
Штукатур  
Облицовщик-плиточник  
Монтажник технологического оборудования и связанных с ним конструкций  
Маляр  
Монтажник технологических трубопроводов  
Полицейский и полицейский-водитель группы задержания, полицейский по охране объектов.  
Слесарь  
Инженер-конструктор 1 категории  
Кочегар-слесарь  
Токарь 5 разряда  
Комплектовщик окон ПВХ  
Мастер смены  
весовщик  
водитель погрузчика  
Специалист отдела информатизации и информационной безопасности, место работы г. Новый Уренгой  
Врач ультразвуковой диагностики  
Врач общей практики (семейный)  
Врач-невролог  
Врач-терапевт



Менеджер по подбору персонала  
 Швея-портной в ателье  
 Специалист по кредитным услугам г. Новый Оскол  
 Риэлтор  
 Агент по недвижимости  
 Ассистент главы отделения  
 Операционная медицинская сестра хирургического отделения  
 Операционная медицинская сестра стоматологического отделения  
 Фельдшер-лаборант биохимической лаборатории  
 Водитель  
 специалист по административному производству  
 медицинская сестра - анестезист  
 медицинская сестра палатная  
 слесарь КИПиА  
 Инженер-технолог  
 Начальник участка (в прочих отраслях)  
 Токарь 4 разряда-6 разряда  
 Водитель категории В,С,Е  
 Рабочий  
 Бухгалтер  
 Обрубщик, занятый на обработке литья наждаком или вручную  
 Ведущий экономист  
 врач детский хирург  
 токарь  
 Слесарь-сантехник  
 Врач-кардиолог  
 Врач-терапевт участковый  
 Врач-офтальмолог  
 Врач терапевт участковый  
 Аппаратчик обработки зерна 5 разряда

Аппаратчик обработки зерна 3 разряда  
 Главный инженер  
 Сменный мастер  
 Стикеровщик, Маркировщик, Рабочий  
 Уборщик производственных помещений - мойщик посуды  
 Электромонтер по ремонту и обслуживанию оборудования подстанций  
 Секретарь, с ведением кадрового документооборота  
 Системный программист (C, Linux)  
 помощник воспитателя  
 повар  
 Электромонтер по ремонту и обслуживанию электрооборудования  
 Слесарь по контрольно-измерительным приборам и автоматике  
 специалист по снабжению  
 Слесарь аварийно-восстановительных работ  
 Машинист крана (крановщик)  
 Врач-рентгенолог  
 Продавец продовольственных товаров  
 младший приемщик товаров  
 контролер торгового зала  
 Приемщик товаров  
 Продавец  
 Слесарь механосборочных работ  
 плиточник-разнорабочий  
 Слесарь по сборке металлоконструкций  
 Слесарь-электромонтажник  
 Экономист  
 Токарь-расточник  
 Менеджер по продажам деревянных домов  
 Автомойщик  
 прораб  
 Врач-дерматовенеролог поликлиники

f2

Программист

Программист C++/C#/Java

Программист 1C

Программист-разработчик информационных систем

Программист C++

Программист/ Junior Developer

Программист / Senior Developer

Программист/ технический специалист

Программист C#

f3

Программист с опытом Python

Программист C++/C#/Java с опытом Python

Программист 1C с опытом Python

Программист-разработчик информационных систем с опытом Python

Программист C++ с опытом Python

Программист/ Junior Developer с опытом Python

Программист / Senior Developer с опытом Python

Программист/ технический специалист с опытом Python

Программист C# с опытом Python

f4

('Программист с опытом Python', 153497)

('Программист C++/C#/Java с опытом Python', 117741)

('Программист 1C с опытом Python', 172624)

('Программист-разработчик информационных систем с опытом Python', 118916)

('Программист C++ с опытом Python', 184107)

('Программист/ Junior Developer с опытом Python', 152944)

('Программист / Senior Developer с опытом Python', 129561)

('Программист/ технический специалист с опытом Python', 153792)

('Программист C# с опытом Python', 185301)