

**Московский государственный технический  
университет им. Н.Э. Баумана.**

Факультет «Информатика и системы управления»

Кафедра ИУ5. Курс «Базовые компоненты интернет-технологий»  
Отчёт по лабораторной работе № 3-4.

Выполнил:

студент группы ИУ5-316  
Койбаев Тамерлан Николаевич

Проверил:

преподаватель каф. ИУ5  
Гапанюк Ю.Е.

Подпись и дата:

Подпись и дата:

Москва, 2022 г.

## Задание

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 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.

Шаблон для реализации генератора:

```
# 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
    # Необходимо реализовать генератор
```

### 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:

```
# Пример:
# gen_random(5, 1, 3) должен выдать 5 случайных чисел
# в диапазоне от 1 до 3, например 2, 2, 3, 2, 1
# Hint: типовая реализация занимает 2 строки
def gen_random(num_count, begin, end):
    pass
    # Необходимо реализовать генератор
```

## 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.

### Пример:

```
data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
```

Unique(data) будет последовательно возвращать только 1 и 2.

```
data = gen_random(10, 1, 3)
```

Unique(data) будет последовательно возвращать только 1, 2 и 3.

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

Unique(data) будет последовательно возвращать только a, A, b, B.

Unique(data, ignore\_case=True) будет последовательно возвращать только a, b.

## Шаблон для реализации класса-итератора:

```
# Итератор для удаления дубликатов
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 одинаковыми
строки в разном регистре
        # Например: ignore_case = True, Абв и АБВ - разные строки
        # ignore_case = False, Абв и АБВ - одинаковые строки, одна
из которых удалится
        # По-умолчанию ignore_case = False
        pass

    def __next__(self):
        # Нужно реализовать __next__
        pass

    def __iter__(self):
        return self

```

#### 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.

Шаблон реализации:

```

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

if __name__ == '__main__':
    result = ...
    print(result)

    result_with_lambda = ...
    print(result_with_lambda)

```

#### 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 выводиться в столбик через знак равенства.

Шаблон реализации:

```
# Здесь должна быть реализация декоратора
```

```
@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__':
    print('!!!!!!!')
    test_1()
    test_2()
    test_3()
    test_4()
```

Результат выполнения:

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

## 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.

Шаблон реализации:

```
import json
import sys
# Сделаем другие необходимые импорты

path = None

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

```

with open(path) 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):
    raise NotImplemented

@print_result
def f2(arg):
    raise NotImplemented

@print_result
def f3(arg):
    raise NotImplemented

@print_result
def f4(arg):
    raise NotImplemented

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

```

## Текст программы

### Field.py

```

def field(items, *args):
    assert len(args) > 0
    if len(args) == 1:
        arr = []
        for i in items:
            temp_key = i.get(args[0], "None")
            if temp_key != "None":
                arr.append(temp_key)
        return arr
    else:
        arr = []
        for i in items:
            temp = {}
            for key in args:
                temp_key = i.get(key, "None")
                if temp_key != "None":
                    temp.update({key: temp_key})
            arr.append(temp)
        print(*arr, sep = ", ")

```

## gen\_random.py

```
import random

def gen_random(num_count, begin, end):
    assert num_count > 1
    arr = []
    for i in range(num_count):
        arr.append(random.randrange(begin, end + 1))
    print(*arr, sep = ", ")
    return arr
```

## unique.py

```
class Unique(object):
    def __init__(self, items, **kwargs):
        self.__r = []
        for key, value in kwargs.items():
            if key == "ignore_case" and value == True:
                try:
                    self.__r = sorted(set([i.lower() for i in items]))
                finally:
                    break

    def unique(self):
        return self.__r

    def __next__(self):
        try:
            temp = self.__r[self.begin]
            self.begin += 1
            return temp
        except:
            raise StopIteration

    def __str__(self):
        return str(*self.__r)

    def __iter__(self):
        return self
```

## sort.py

```
def sort_arr(data):
    res = sorted(data, key = abs, reverse = True)
    print(res)

    lambda_res = sorted(data, key = lambda a : -abs(a))
    print(lambda_res)
```

## print\_result.py

```
def print_result(func):
    def wrapper():
        print(func.__name__)
        if isinstance(func(), list):
            print(*func(), sep = "\n")

        elif isinstance(func(), dict):
            temp = func()
            for i in temp:
```



```

        print(i, temp.get(i), sep = " = ")

    else:
        print(func())
    return wrapper

@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]

```

## process\_data.py

```

# -*- coding: utf-8 -*-
import json
from xml.etree.ElementTree import tostring
from field import field
from gen_random import gen_random
from unique import Unique
from sort import sort_arr
from print_result import print_result
from cm_timer import cm_timer_1, cm_timer_2

from operator import concat

def f1(data):
    return Unique(field(data, 'job-name'), ignore_case = True).unique()

def f2(temp):
    return filter(lambda a: a.startswith('программист'), temp)

def f3(temp):
    return list(map(lambda x: concat(x, ' с опытом Python'), temp))

def f4(temp):
    return zip(temp, gen_random(len(temp), 100000, 200000))

if __name__ == '__main__':
    with open('data_light.json', encoding = "UTF-8-sig") as f:
        data = json.load(f)
    with cm_timer_1():
        for i in f4(f3(f2(f1(data)))):
            print(i)

```

## 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_arr
from lab_python_fp.print_result import *

```

```

from lab_python_fp.cm_timer import *
import time
#from lab_python_fp.process_data import *

def main():
    goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
    ]

    # _____Задача 1

    print(field(goods, 'title', 'price'))
    print(field(goods, 'title', 'color'))
    print(field(goods, 'color'))

    # _____Задача 2

    #gen_random(5, 1, 3)

    # _____Задача 3
    # data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
    # print(Unique(data))
    # data = gen_random(5, 1, 3)
    # print(Unique(data, ignore_case=True))
    # data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    # print(Unique(data, ignore_case=True))

    #data = [1, 1, 1, 1, 1, 2, 4, 7, 2, 2]
    #for i in Unique(data):
    #    print(i)
    #print('-----')
    #data = gen_random(5, 1, 3)
    #for i in Unique(data):
    #    print(i)
    #print('-----')
    #data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    #for i in Unique(data):
    #    print(i)

    # _____Задача 4

    #data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
    #sort_arr(data)

    # _____Задача 5
    #test_1()
    #test_2()
    #test_3()
    #test_4()

    # _____Задача 6

    #with cm_timer_1():

```

```
#     time.sleep(5.5)

#with cm_timer_2():
#     time.sleep(5.5)

if __name__ == '__main__':
    main()
```

## Анализ результатов

### Field

```
C:\Users\Тамерлан\AppData\Local\Programs\Python\Python311\python.exe "D:\Users\Тамерлан\
{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}
None
{'title': 'Ковер', 'color': 'green'}, {'title': 'Диван для отдыха', 'color': 'black'}
None
['green', 'black']
```

### gen\_random

	27	gen_random(5, 1, 3)
	28	
	main()	

main (2) ×

```
C:\Users\Тамерлан\AppData\Local\Programs\Python\Python311\python.exe "D:
3, 2, 1, 1, 3
```

### Sort

55	data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
56	sort_arr(data)
57	

main()

main (2) ×

```
C:\Users\Тамерлан\AppData\Local\Programs\Python\Python311
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]
```

### Print\_result

```
C:\Users\Тамерлан\AppData\Local\Pr
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
```

#### Process\_data

```
172521, 183948, 109736, 175687, 173809, 100008, 175919, 150891, 110970
('программист с опытом Python', 172521)
('программист / senior developer с опытом Python', 183948)
('программист 1с с опытом Python', 109736)
('программист с# с опытом Python', 175687)
('программист с++ с опытом Python', 173809)
('программист с++/с#/java с опытом Python', 100008)
('программист/ junior developer с опытом Python', 175919)
('программист/ технический специалист с опытом Python', 150891)
('программист-разработчик информационных систем с опытом Python', 110970)
0.006002664566040039
```

# Экранные формы с примерами выполнения программы

## Field.py

```
PS D:\BKIT\BKIT\lab_3\lab_python_fp> python .\field.py
['Ковер', 'Диван для отдыха']
[{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха'}]
PS D:\BKIT\BKIT\lab_3\lab_python_fp> █
```

## Get\_random.py

```
PS D:\BKIT\BKIT\lab_3\lab_python_fp> python .\gen_random.py
2 3 3 1 3
-
```

## Unique.py

```
PS D:\BKIT\BKIT\lab_3\lab_python_fp> python .\unique.py
До A a B b
A a B b
PS D:\BKIT\BKIT\lab_3\lab_python_fp> █
PS D:\BKIT\BKIT\lab_3\lab_python_fp> python .\unique.py
До A a B b
a b
PS D:\BKIT\BKIT\lab_3\lab_python_fp> █
```

## Sort.py

```
PS D:\BKIT\BKIT\lab_3\lab_python_fp> python .\sort.py
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]
```

## Print\_result.py

```
!!!!!!!  
test_1  
5  
test_2  
ui  
test_3  
a=1  
b=2  
test_4  
1  
2
```

## Process\_data.py

```
f4  
( 'программист с опытом Python', 103862)  
( 'программист с++/с#/java с опытом Python', 167937)  
( 'программист 1с с опытом Python', 129493)  
( 'программист-разработчик информационных систем с опытом Python', 100452)  
( 'программист с++ с опытом Python', 135397)  
( 'программист/ junior developer с опытом Python', 101938)  
( 'программист / senior developer с опытом Python', 168148)  
( 'программист/ технический специалист с опытом Python', 113188)  
( 'программист с# с опытом Python', 112001)  
0.48728036880493164
```