

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

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Основы информатики»

Отчет по лабораторной работе №3-4

«Функциональные возможности языка Python»

Выполнил:

студент группы ИУ5-35Б  
Клементьев Артем  
Подпись и дата:

Проверил:

преподаватель каф. ИУ5  
Нардид А. Н.  
Подпись и дата:

Москва, 2022 г.

## 1. 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 (файл `field.py`):*

Необходимо реализовать генератор `field`. Генератор `field` последовательно выдает значения ключей словаря. Пример:

- 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.

### *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`.

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 выводиться 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.

`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.

## 2. Текст программы:

**field.py**

```
def field(items, *args):
    assert len(args) > 0
    if len(args) > 1:
        for el in items:
            dct = {}
            for keys, val in el.items():
                for arg in args:
                    if keys == arg:
                        dct[keys] = val
    yield dct
```

```

else:
    for el in items:
        for keys in el:
            for arg in args:
                if keys == arg:
                    yield el[keys]

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

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

```

## **gen\_random.py**

```

from random import randint

lst = []

def gen_random(num_count, begin, end):
    for i in range(0, num_count):
        lst.append(randint(begin, end))
    numbers = map(str, lst)
    print(', '.join(numbers))
    return lst

if __name__ == '__main__':
    gen_random(5, 0, 10)

```

## **unique.py**

```

# Итератор для удаления дубликатов
class Unique(object):
    def __init__(self, items, **kwargs):
        self.items = [str(i) for i in items]
        self.data = set()
        self.index = 0

```

```

self.lst = []
try:
    self.ignore_case = kwargs["ignore_case"]
except:
    self.ignore_case = False

def __next__(self):
    for i in range(0, len(self.items)):
        self.index = i
        el = self.items[self.index]
        if type(el) is str:
            if not self.ignore_case:
                if el not in self.data:
                    self.data.add(el)
                    return el
            else:
                if (el.lower() not in self.data) and (el.upper() not in self.data):
                    self.data.add(el)
                    return el
        else:
            if el not in self.data:
                self.data.add(el)
                return el

def __iter__(self):
    return self

def filling(self):
    el = next(self)
    while el is not None:
        self.lst.append(el)
        el = next(self)
    return self.lst

if __name__ == '__main__':
    data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    print(Unique(data).filling())

```

## **sort.py**

```
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)

```

## **print\_result.py**

```

def print_result(func):
    def wrapper(*args):
        print(func.__name__)
        res = func(*args)
        if type(res) == dict:
            dct = res
            for i in dct:
                print(i, '=', dct[i])
        elif type(res) == list:
            lst = res
            for i in lst:
                print(i)
        else:
            if func.__name__ != 'print':
                print(res)
            else:
                res
        return res
    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]
```

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

### **cm\_timer.py**

```
from time import time, sleep
from contextlib import contextmanager
```

```
class cm_timer_1:
```

```
    def __init__(self):
        self.start = 0
        self.finish = 0
```

```
    def __enter__(self):
        self.start = time()
```

```
    def __exit__(self, exc_type, exc_val, exc_tb):
        self.finish = time()
        print("Время работы: ", self.finish - self.start)
```

```
@contextmanager
def cm_timer_2():
    start = time()
    yield None
    finish = time()
    print("Время работы: ", finish - start)
```

```
if __name__ == '__main__':
    with cm_timer_1():
```



```
    sleep(2)
with cm_timer_2():
    sleep(2)
```

## **process\_data.py**

```
import json
import sys
from lab_python_fp.print_result import print_result
from lab_python_fp.cm_timer import cm_timer_1
from lab_python_fp.unique import Unique
from lab_python_fp.gen_random import gen_random
from lab_python_fp.field import field
```

```
path = 'data_light.json'
with open(path, "rb") as f:
    data = json.load(f)
```

```
@print_result
def f1(arg):
    return sorted(list(Unique(field(data, "job-name")).filling()))
```

```
@print_result
def f2(arg):
    return list(filter(lambda s: s.startswith("Программист") or
s.startswith("программист"), arg))
```

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

```
@print_result
def f4(arg):
    return list(zip(arg, ['зарплата ' + str(i) + ' руб.' for i in gen_random(len(arg),
100000, 200000)]))
```

```
if __name__ == '__main__':
```

```
with cm_timer_1():  
    f4(f3(f2(f1(data))))
```

**main.py**

### 3. Экранные формы с примерами выполнения программы:

```
C:\Users\jrsyt\PycharmProjects\Lab3-4\venv\Scripts\python.exe C:/Users/jrsyt/PycharmProjects/Lab3-4/lab_python_fp/field.py  
['Ковер', 'Диван для отдыха']  
[{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха'}]  
  
Process finished with exit code 0
```

```
C:\Users\jrsyt\PycharmProjects\Lab3-4\venv\Scripts\python.exe C:/Users/jrsyt/PycharmProjects/Lab3-4/lab_python_fp/gen_random.py  
4, 2, 1, 4, 6  
  
Process finished with exit code 0
```

```
C:\Users\jrsyt\PycharmProjects\Lab3-4\venv\Scripts\python.exe C:/Users/jrsyt/PycharmProjects/Lab3-4/lab_python_fp/unique.py  
['a', 'A', 'b', 'B']  
  
Process finished with exit code 0
```

```
C:\Users\jrsyt\PycharmProjects\Lab3-4\venv\Scripts\python.exe C:/Users/jrsyt/PycharmProjects/Lab3-4/lab_python_fp/sort.py  
[123, 100, -100, -30, 4, -4, 1, -1, 0]  
[123, 100, -100, -30, 4, -4, 1, -1, 0]  
  
Process finished with exit code 0
```

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

```
C:\Users\jrsyt\PycharmProjects\Lab3-4\venv\Scripts\python.exe C:/Users/jrsyt/PycharmProjects/Lab3-4/lab_python_fp/cm_timer.py  
Время работы: 2.0019474029541016  
Время работы: 2.0028648376464844  
  
Process finished with exit code 0
```

```
C:\Users\jrsyt\PycharmProjects\Lab3-4\venv\Scripts\python.exe C:/Users/jrsyt/PycharmProjects/Lab3-4/lab_python_fp/process_data.py
f1
1С программист
2-ой механик
3-ий механик
4-ый механик
4-ый электромеханик
ASIC специалист
JavaScript разработчик
RTL специалист
```

И так далее

```
f2
Программист
Программист / Senior Developer
Программист 1С
Программист C#
Программист C++
Программист C++/C#/Java
Программист/ Junior Developer
```

```
Программист/ технический специалист
Программист-разработчик информационных систем
программист
программист 1С
f3
Программист с опытом Python
Программист / Senior Developer с опытом Python
Программист 1С с опытом Python
Программист C# с опытом Python
```

```
Программист C++ с опытом Python
Программист C++/C#/Java с опытом Python
Программист/ Junior Developer с опытом Python
Программист/ технический специалист с опытом Python
Программист-разработчик информационных систем с опытом Python
программист с опытом Python
программист 1С с опытом Python
f4
120642, 102644, 197844, 184546, 159800, 186232, 139168, 127064, 188808, 185980, 161400
```

```
('Программист с опытом Python', 'зарплата 120642 руб.')
('Программист / Senior Developer с опытом Python', 'зарплата 102644 руб.')
('Программист 1С с опытом Python', 'зарплата 197844 руб.')
('Программист C# с опытом Python', 'зарплата 184546 руб.')
('Программист C++ с опытом Python', 'зарплата 159800 руб.')
('Программист C++/C#/Java с опытом Python', 'зарплата 186232 руб.')
('Программист/ Junior Developer с опытом Python', 'зарплата 139168 руб.')
('Программист/ технический специалист с опытом Python', 'зарплата 127064 руб.')
('Программист-разработчик информационных систем с опытом Python', 'зарплата 188808 руб.')
```

```
('программист с опытом Python', 'зарплата 185980 руб.')
('программист 1С с опытом Python', 'зарплата 161400 руб.')
Время работы: 1.3456602096557617
```

```
Process finished with exit code 0
```