



**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, искусственный интеллект
и системы управления»
Кафедра «Системы обработки информации и
управления»**

Лабораторная работа №3-4 по курсу
«Базовые компоненты интернет-технологий»

Выполнил:
студент группы ИУ5-33Б
Иванченко Максим

Проверил:
Доцент кафедры ИУ5
Гапанюк Юрий Евгеньевич

2022 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

Постановка 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
    for dictionary in items:
        res_dict = {}
        for key in args:
            if dictionary.get(key) is not None:
                res_dict[key] = dictionary[key]
        if len(res_dict) == 0:
            pass
        elif len(res_dict) == 1:
            key = [key for key in res_dict.keys()][0]
            yield "" + str(res_dict[key]) + ""
        else:
            yield res_dict

def main():
    goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'price': 5300, 'color': None}
    ]
    # 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}
    print(*[elem for elem in field(goods, 'title')], sep=', ')
    print(*[elem for elem in field(goods, 'title', 'price')], sep=', ')
    print(*[elem for elem in field(goods, 'title', 'price', 'color')], sep=', ')

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

Пример выполнения

```
Введите номер задания (0 - завершение): 1
'Ковер', 'Диван для отдыха'
{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}
{'title': 'Ковер', 'price': 2000, 'color': 'green'}, {'title': 'Диван для отдыха', 'price': 5300}
Введите номер задания (0 - завершение): 0

Process finished with exit code 0
```

Задача №2

Постановка 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

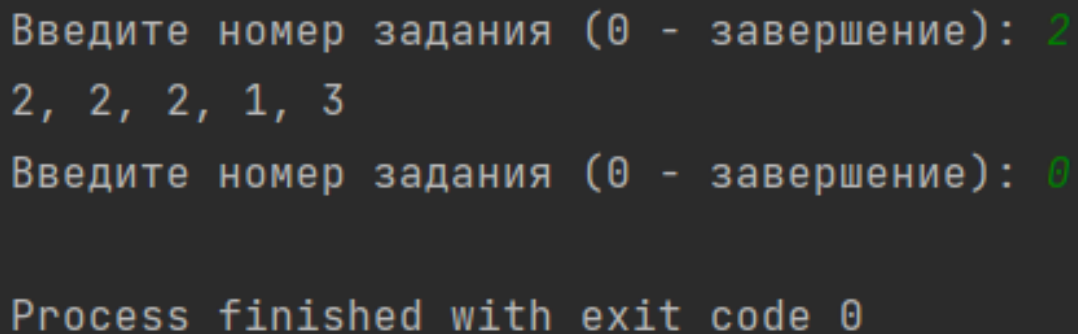
```
from random import randint

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

def main():
    print(*gen_random(5, 1, 3), sep=', ')

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

Пример выполнения



```
Введите номер задания (0 - завершение): 2
2, 2, 2, 1, 3
Введите номер задания (0 - завершение): 0

Process finished with exit code 0
```

Задача №3

Постановка 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 lab_python_fp.Task_2 import gen_random

# Итератор для удаления дубликатов
class Unique(object):
    def __init__(self, items, **kwargs):
        self.used = set()
        self.items = set(items)
        if len(kwargs) != 0:
            for name, value in kwargs.items():
                if name == 'ignore_case':
                    self.ignore_case = value
            else:
                self.ignore_case = False
        if self.ignore_case is True:
            to_change = set()
            for elem in self.items:
                if isinstance(elem, str):
                    if elem.lower() != elem:
                        to_change.add(elem)
            for elem in to_change:
                self.items.remove(elem)
                self.items.add(elem.lower())

    def __next__(self):
        if len(self.used) == len(self.items):
            raise StopIteration
        for elem in self.items:
            if elem not in self.used:
                self.used.add(elem)
                return elem

    def __iter__(self):
        return self

def iter_print(it):
    while True:
```

```

    try:
        print(next(it), end=' ')
    except StopIteration:
        print()
        break

def main():
    data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
    it = Unique(data)
    iter_print(it)

    data = gen_random(10, 1, 3)
    it = Unique(data)
    iter_print(it)

    data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    it = Unique(data)
    iter_print(it)

    it = Unique(data, ignore_case=True)
    iter_print(it)

if __name__ == '__main__':
    main()

```

Пример выполнения

```

Введите номер задания (0 - завершение): 3
1 2
1 2 3
A b B a
b a
Введите номер задания (0 - завершение): 0

Process finished with exit code 0

```

Задача №4

Постановка задачи

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. Необходимо решить задачу двумя способами:

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

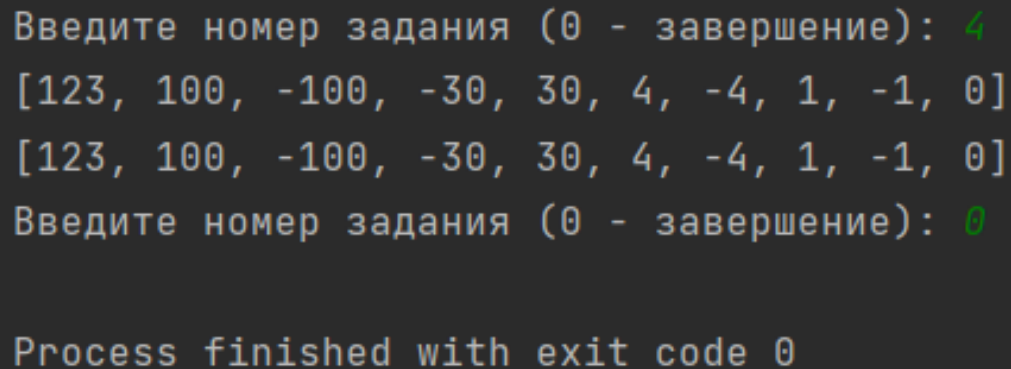
Текст программы

```
def main():
    data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]
    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)

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

Пример выполнения



```
Введите номер задания (0 - завершение): 4
[123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
Введите номер задания (0 - завершение): 0

Process finished with exit code 0
```


Задача №5

Постановка 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(f):
    def _wrapper(*args, **kwargs):
        res = f(*args, **kwargs)
        print(f.__name__)
        if type(res) == list:
            for elem in res:
                print(elem)
        elif type(res) == dict:
            for key, value in res.items():
                print(key, '=', value)
        else:
            print(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]
```

```
def main():
    test_1()
    test_2()
    test_3()
    test_4()
```

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

Пример выполнения

```
Введите номер задания (0 - завершение): 5  
test_1  
1  
test_2  
iu5  
test_3  
a = 1  
b = 2  
test_4  
1  
2  
Введите номер задания (0 - завершение): 0  
  
Process finished with exit code 0
```

Задача №6

Постановка задачи

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`).

Текст программы

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

class cm_timer_1:
    def __init__(self):
        self.start = 0
        self.end = 0

    def __enter__(self):
        self.start = perf_counter()

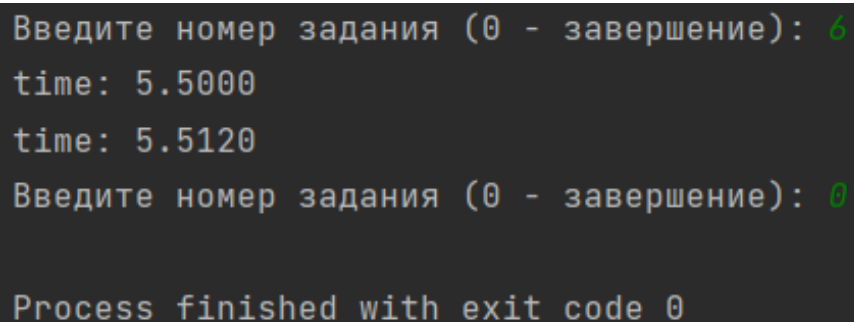
    def __exit__(self, exc_type, exc_val, exc_tb):
        self.end = perf_counter()
        print(f'time: {self.end-self.start:0.4f}')
```

```
@contextmanager
def cm_timer_2():
    start = perf_counter()
    yield
    end = perf_counter()
    print(f'time: {end-start:0.4f}')
```

```
def main():
    with cm_timer_1():
        sleep(5.5)
    with cm_timer_2():
        sleep(5.5)

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

Пример выполнения



```
Введите номер задания (0 - завершение): 6
time: 5.5000
time: 5.5120
Введите номер задания (0 - завершение): 0

Process finished with exit code 0
```

Задача №7

Постановка 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

```
from lab_python_fp.Task_2 import gen_random
from lab_python_fp.Task_3 import Unique
from lab_python_fp.Task_5 import print_result
from lab_python_fp.Task_6 import cm_timer_1
import json
import sys

@print_result
def f1(arg):
    return sorted(Unique([d['job-name'] for d in arg], ignore_case=True))
```

```

@print_result
def f2(arg):
    return list(filter(lambda x: x[:11] == 'программист', arg))

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

@print_result
def f4(arg):
    # return list(zip(arg, list(gen_random(len(arg), 100000, 200000))))
    return [pair[0]+' зарплата '+str(pair[1])+' руб.' for pair in zip(arg,
list(gen_random(len(arg), 100000, 200000)))]

def main():
    path = 'data_light.json'
    with open(path, 'r', encoding='UTF8') as f:
        data = json.load(f)
    with cm_timer_1():
        f4(f3(f2(f1(data))))

if __name__ == '__main__':
    main()

```

Пример выполнения (опущен вывод функций f1 и f2)

```

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
f4
программист с опытом python, зарплата 149058 руб.
программист / senior developer с опытом python, зарплата 176613 руб.
программист 1с с опытом python, зарплата 197014 руб.
программист c# с опытом python, зарплата 111626 руб.
программист c++ с опытом python, зарплата 159426 руб.
программист c++/c#/java с опытом python, зарплата 195503 руб.
программист/ junior developer с опытом python, зарплата 136773 руб.
программист/ технический специалист с опытом python, зарплата 197250 руб.
программист-разработчик информационных систем с опытом python, зарплата 131325 руб.
time: 0.1304
Введите номер задания (0 - завершение): 0

Process finished with exit code 0

```