Московский государственный технический университет им. Н.Э. Баумана

Факультет Радиотехнический
Кафедра “Системы обработки информации и управления”

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №3-4
“ Функциональные возможности языка Python.”

Выполнил:
студент группы РТ5-31Б:
Каландаров Алим
Шамильевич
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий
Евгеньевич
Подпись и дата:

Москва, 2025 г.

Постановка задачи

Задание лабораторной работы состоит из решения нескольких задач.

Файлы, содержащие решения отдельных задач, должны располагаться в пакете lab_python_fp. Решение каждой задачи должно раполагаться в отдельном файле.

При запуске каждого файла выдаются тестовые результаты выполнения соответствующего задания.

Задание 1:
Необходимо реализовать генератор field. Генератор field последовательно выдает значения ключей словаря.

Задание 2:
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.

Задание 3:

- 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.

Задание 4:

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:

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

Задание 5:
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.

Задание 6:
Необходимо написать контекстные менеджеры cm_timer_1 и cm_timer_2, которые считают время работы блока кода и выводят его на экран.

Задание 7:

- 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

Задание 1:

```
def field(items, *args):
    assert len(args) > 0

    if len(args) == 1:
        # Если передан один аргумент - возвращаем только значения
        key = args[0]
        for item in items:
            if key in item and item[key] is not None:
                yield item[key]
    else:
        # Если передано несколько аргументов - возвращаем словари
        for item in items:
            result = {}
```

```
            has_values = False
            for key in args:
                if key in item and item[key] is not None:
                    result[key] = item[key]
                    has_values = True
            if has_values:
                yield result


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

    print("Test 1 - один аргумент:")
    for value in field(goods, 'title'):
        print(value)

    print("\nTest 2 - несколько аргументов:")
    for value in field(goods, 'title', 'price'):
        print(value)
```

Задание 2:

```
import random

def gen_random(num_count, begin, end):
    for _ in range(num_count):
        yield random.randint(begin, end)


if __name__ == "__main__":
    print("Test gen_random:")
    for num in gen_random(5, 1, 3):
```

```
        print(num, end=" ")
    print()
```

Задание 3:

```
class Unique(object):
    def __init__(self, items, **kwargs):
        self.ignore_case = kwargs.get('ignore_case', False)
        self.items = iter(items)
        self.seen = set()

    def __next__(self):
        while True:
            item = next(self.items)

            # Для сравнения учитываем регистр, если нужно
            if isinstance(item, str) and self.ignore_case:
                check_item = item.lower()
            else:
                check_item = item

            if check_item not in self.seen:
                self.seen.add(check_item)
                return item

    def __iter__(self):
        return self



if __name__ == "__main__":
    print("Test 1 - числа:")
    data1 = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
    for item in Unique(data1):
        print(item, end=" ")
    print()

    print("\nTest 2 - строки без ignore_case:")
```

```
    data2 = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    for item in Unique(data2):
        print(item, end=" ")
    print()

    print("\nTest 3 - строки с ignore_case=True:")
    for item in Unique(data2, ignore_case=True):
        print(item, end=" ")
    print()
```

Задание 4:

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

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

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

Задание 5:

```
def print_result(func):
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)
        print(func.__name__)

        if isinstance(result, list):
            for item in result:
                print(item)
        elif isinstance(result, dict):
            for key, value in result.items():
                print(f"{key} = {value}")
        else:
            print(result)
```

```
        return result
    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()
```

Задание 6:

```
import time
from contextlib import contextmanager
```

```
class cm_timer_1:
    def __enter__(self):
        self.start_time = time.time()
        return self

    def __exit__(self, exc_type, exc_val, exc_tb):
        self.end_time = time.time()
        print(f"time: {self.end_time - self.start_time:.1f}")


@contextmanager
def cm_timer_2():
    start_time = time.time()
    yield
    end_time = time.time()
    print(f"time: {end_time - start_time:.1f}")


if __name__ == "__main__":
    print("Test cm_timer_1:")
    with cm_timer_1():
        time.sleep(0.5)

    print("\nTest cm_timer_2:")
    with cm_timer_2():
        time.sleep(0.5)
```

Задание 7:

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

```
path = sys.argv[1] if len(sys.argv) > 1 else 'data_light.json'

with open(path, encoding='utf-8') as f:
    data = json.load(f)

@print_result
def f1(arg):
    return sorted(list(Unique(field(arg, 'job-name'), ignore_case=True)),
key=lambda x: x.lower())

@print_result
def f2(arg):
    return list(filter(lambda x: x.lower().startswith('программист'), arg))

@print_result
def f3(arg):
    return list(map(lambda x: f"{x} с опытом Python", arg))

@print_result
def f4(arg):
    salaries = list(gen_random(len(arg), 100000, 200000))
    return list(map(lambda x: f"{x[0]}, зарплата {x[1]} руб.", zip(arg, salaries)))



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

Анализ результатов

Входные данные:

- Отображены в терминале

Выходные данные:

1.

```
Test 1 - один аргумент:
Ковер
Диван для отдыха

Test 2 - несколько аргументов:
{'title': 'Ковер', 'price': 2000}
{'title': 'Диван для отдыха'}
```

2.

```
Test gen_random:
2 1 3 3 3
```

3.

```
Test 1 - числа:
1 2

Test 2 - строки без ignore_case:
a A b B

Test 3 - строки с ignore_case=True:
a b
```

4.

```
E:\3sem\парадигмы\Lab2(3-4)\.venv\Scripts
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]
```

5.

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

6.

```
Test cm_timer_1:
time: 0.5

Test cm_timer_2:
time: 0.5
```

7.

```
f1
Аналитик данных
Менеджер проекта
Программист C#
Программист C++
Программист Java
программист JavaScript
Программист Python
f2
Программист C#
Программист C++
Программист Java
программист JavaScript
Программист Python
f3
Программист C# с опытом Python
Программист C++ с опытом Python
Программист Java с опытом Python
программист JavaScript с опытом Python
Программист Python с опытом Python
f4
Программист C# с опытом Python, зарплата 185429 руб.
Программист C++ с опытом Python, зарплата 182758 руб.
Программист Java с опытом Python, зарплата 128192 руб.
программист JavaScript с опытом Python, зарплата 129229 руб.
Программист Python с опытом Python, зарплата 182467 руб.
time: 0.0
```