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

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

# Курс «Парадигмы и конструкции языков программирования» Отчёт по лабораторной работе №3-4

## Функциональные возможности языка Python

Выполнил:
студент группы ИУ5-35Б
Александров Георгий

Проверил: преподаватель каф. ИУ5
Нардид Анатолий Николаевич

Москва, 2025 г.

# Описание задания

**Задача 1 (файл field.py)**

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

**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, которые содержит значения массива 1, отсортированные по модулю в порядке убывания. Сортировку необходимо осуществлять с помощью функции sorted.

**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 выводить 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.

**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.

# Текст программы

## cm_timer.py

```
import time
import contextlib

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

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

@contextlib.contextmanager
def cm_timer_2():
    start_time = time.time()
    yield
    elapsed_time = time.time() - start_time
    print(f"time: {elapsed_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(0.3)
```

## field.py

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

    if len(args) == 1:
        field_name = args[0]
        for item in items:
            if field_name in item and item[field_name] is not None:
                yield item[field_name]
    else:
        for item in items:
            result = {}
            has_valid_fields = False

            for field_name in args:
                if field_name in item and item[field_name] is not None:
```

```
                    result[field_name] = item[field_name]
                    has_valid_fields = True

            if has_valid_fields:
                yield result
if __name__ == "__main__":
    goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},
        {'title': None, 'price': 3000, 'color': 'white'},
    ]

    print("Test 1:")
    for item in field(goods, 'title'):
        print(item)

    print("\nTest 2:")
    for item in field(goods, 'title', 'price'):
        print(item)
```

## gen_random.py

```
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(5, 1, 3):")
    result = list(gen_random(5, 1, 3))
    print(result)
```

## print_result.py

```
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()
```

## process_data.py

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

path = None

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

with open(path, 'r', 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=str.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))

    result = []
    for profession, salary in zip(arg, salaries):
        result.append(f"{profession}, зарплата {salary} руб.")

    return result

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

## sort.py

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

if __name__ == "__main__":
    def abs_key(x):
        return abs(x)

    result = sorted(data, key=abs_key, reverse=True)
    print(result)

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

## unique.py

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

    def __iter__(self):
```

```
        return self

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

                if self.ignore_case and isinstance(item, str):
                    key = item.lower()
                else:
                    key = item

                if key not in self.seen:
                    self.seen.add(key)
                    return item
            except StopIteration:
                raise StopIteration

if __name__ == "__main__":
    print("Test Unique with numbers:")
    data1 = [1, 1, 2, 2, 3, 3]
    result1 = list(Unique(data1))
    print(result1)

    print("\nTest Unique with strings (ignore_case=False):")
    data2 = ['a', 'A', 'b', 'B', 'a']
    result2 = list(Unique(data2))
    print(result2)
```

## init.py

```
__all__ = ['field', 'gen_random', 'unique', 'sort', 'print_result', 'cm_timer', 'process_data']
```

## data_light.json


```
[
    {"job-name": "Программист C#", "company": "Яндекс", "salary": 150000},
    {"job-name": "Программист Python", "company": "Google", "salary": 180000},
    {"job-name": "Аналитик данных", "company": "МТС", "salary": 120000},
    {"job-name": "программист Java", "company": "Тинькофф", "salary": 160000},
    {"job-name": "Дизайнер UI/UX", "company": "ВКонтакте", "salary": 110000},
    {"job-name": "Программист C++", "company": "Яндекс", "salary": 170000},
    {"job-name": "Менеджер проектов", "company": "Сбер", "salary": 130000},
    {"job-name": "Тестировщик", "company": "Озон", "salary": 90000},
    {"job-name": "Программист JavaScript", "company": "Яндекс", "salary": 155000},
    {"job-name": "Data Scientist", "company": "Сбер", "salary": 190000}
]
```

# Листинг кода

## cm_timer.py

```
(venv) PS C:\Users\MSI-Crosshair16\Documents\лабы\lab_python_fp\lab_python_fp> python cm_timer.py
Test cm_timer_1:
time: 0.5

Test cm_timer_2:
time: 0.3
```

## field.py

```
(venv) PS C:\Users\MSI-Crosshair16\Documents\лабы\lab_python_fp\lab_python_fp> python field.py
Test 1:
Ковер
Диван для отдыха

Test 2:
{'title': 'Ковер', 'price': 2000}
{'title': 'Диван для отдыха', 'price': 5300}
{'price': 3000}
```

## gen_random.py

```
(venv) PS C:\Users\MSI-Crosshair16\Documents\лабы\lab_python_fp\lab_python_fp> python gen_random.py
Test gen_random(5, 1, 3):
[1, 1, 2, 1, 2]
```

## process_data.py

```
(venv) PS C:\Users\MSI-Crosshair16\Documents\лабы\lab_python_fp\lab_python_fp> python process_data.py
f1
Data Scientist
Аналитик данных
Дизайнер UI/UX
Менеджер проектов
Программист C#
Программист C++
программист Java
Программист JavaScript
Программист Python
Тестировщик
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 159406 руб.
Программист C++ с опытом Python, зарплата 121500 руб.
программист Java с опытом Python, зарплата 100199 руб.
Программист JavaScript с опытом Python, зарплата 147938 руб.
Программист Python с опытом Python, зарплата 162610 руб.
time: 0.0
```

## sort.py

```
(venv) PS C:\Users\MSI-Crosshair16\Documents\лабы\lab_python_fp\lab_python_fp> python sort.py
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]
```

## unique.py

```
(venv) PS C:\Users\MSI-Crosshair16\Documents\лабы\lab_python_fp\lab_python_fp> python unique.py
Test Unique with numbers:
[1, 2, 3]

Test Unique with strings (ignore_case=False):
['a', 'A', 'b', 'B']
```

## print_result.py

```
(venv) PS C:\Users\MSI-Crosshair16\Documents\лабы\lab_python_fp\lab_python_fp> python print_result.py
!!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
```