Московский государственный технический университет им. Н.Э. Баумана

Факультет “Информатика и системы управления”
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 ИУ5-31Б
Попов А.С.
Подпись и дата:

Проверил:
Преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2025 г.

Задание

Задание лабораторной работы состоит из решения нескольких задач. Цель работы: реализация набора утилит (генераторов, итераторов, декораторов) и их применение для обработки данных о вакансиях. Все файлы должны находиться в пакете lab_python_fp. Список задач:

1. Реализовать генератор, который принимает список словарей и возвращает либо значения конкретного поля, либо под-словари с выбранными полями. Должен обрабатывать пропуски (None).
2. Написать функцию-генератор, выдающую заданное количество случайных чисел в указанном диапазоне.
3. Создать класс-итератор, который принимает список или генератор и возвращает только уникальные элементы. Реализовать параметр ignore_case для игнорирования регистра строк.
4. Отсортировать массив чисел по модулю (abs) в порядке убывания. Реализовать два варианта: с использованием lambda-функции и без неё.
5. Написать декоратор, который перед выполнением функции выводит её имя, а после – результат. Списки и словари должны выводиться в столбик.
6. Реализовать измерение времени выполнения блока кода двумя способами: через класс (методы __enter__, __exit__) и через библиотеку contextlib.
7. Используя реализованные выше инструменты, обработать файл data_light.json (список вакансий) через цепочку функций:

   - f1: получить список уникальных профессий (сортировка без учета регистра).

   - f2: оставить только вакансии, начинающиеся со слова «программист» (использовать filter).

   - f3: добавить к названию «с опытом Python» (использовать map).

   - f4: сгенерировать случайные зарплаты (100к–200к) и объединить их с профессиями (использовать zip).

Текст программы

field.py

```
def field(items, *args):
    assert len(args) > 0, "Должен быть хотя бы один аргумент!"

    if len(args) == 1:
        key = args[0]
        for item in items:
            val = item.get(key)
            if val is not None:
                yield val
    else:
        for item in items:
            new_item = {key: item.get(key) for key in args if
item.get(key) is not None}
            if len(new_item) > 0:
                yield new_item

if __name__ == '__main__':
    goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'price': 5300, 'color':
'black'}
    ]

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

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(list(gen_random(5, 1, 3)))
```

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

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

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

    def __iter__(self):
        return self

if __name__ == '__main__':
    data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']

    print(list(Unique(data)))
    print(list(Unique(data, ignore_case=True)))
```

sort.py

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

if __name__ == '__main__':
    # С использованием lambda-функции
    result_with_lambda = sorted(data, key=lambda x: abs(x),
reverse=True)
    print(result_with_lambda)

    # Без использования lambda-функции
    result = sorted(data, key=abs, reverse=True)
    print(result)
```

print_result.py

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

        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()
```

cm_timer.py

```
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):
        elapsed_time = time.time() - self.start_time
        print(f'time: {elapsed_time}')

@contextmanager
def cm_timer_2():
    start_time = time.time()
    try:
        yield
    finally:
```

```
        elapsed_time = time.time() - start_time
        print(f'time: {elapsed_time}')

if __name__ == '__main__':
    with cm_timer_1():
        time.sleep(5.5)
    with cm_timer_2():
        time.sleep(5.5)
```

process_data.py

```
import json
import sys

from print_result import print_result
from cm_timer import cm_timer_1
from unique import Unique
from gen_random import gen_random

path = 'data_light.json'

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

@print_result
def f1(arg):
    return sorted(Unique([x['job-name'] for x in arg],
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: x + ' с опытом Python', arg))

@print_result
def f4(arg):
    salaries = gen_random(len(arg), 100000, 200000)
    return [f'{job}, зарплата {salary} руб.' for job, salary in
zip(arg, salaries)]

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

Скриншоты работы программы

```
PS D:\proga\lab_python_fp> python field.py
['Ковер', 'Диван для отдыха']
[{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}]
PS D:\proga\lab_python_fp> python gen_random.py
[2, 2, 2, 1, 1]
PS D:\proga\lab_python_fp> python unique.py
['a', 'A', 'b', 'B']
['a', 'b']
PS D:\proga\lab_python_fp> python sort.py
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]
!!!!!!!!
test_1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
PS D:\proga\lab_python_fp> python cm_timer.py
time: 5.500112056732178
time: 5.505842685699463
PS D:\proga\lab_python_fp> python process_data.py
f1
Дизайнер
Менеджер
Программист C#
программист Java
Программист Python
f2
Программист C#
программист Java
Программист Python
f3
Программист C# с опытом Python
программист Java с опытом Python
Программист Python с опытом Python
f4
Программист C# с опытом Python, зарплата 140639 руб.
программист Java с опытом Python, зарплата 125484 руб.
Программист Python с опытом Python, зарплата 185086 руб.
time: 0.002500295639038086
PS D:\proga\lab_python_fp>
```

Рис. 1 – Работа приложения