МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра “Системы обработки информации и управления”

Дисциплина “Парадигмы и конструкции языков программирования”

Отчет по лабораторным работам №3-4
“Функциональные возможности языка Python”

Выполнил:
Студент группы ИУ5-35Б
Евдокимов М. С.
Преподаватель:
Гапанюк Ю. Е.

Москва 2025

Цель лабораторной работы

Цель лабораторной работы: изучение возможностей функционального программирования в языке Python.

Задание лабораторной работы

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.

Задача 1 (файл field.py)

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

goods = [

{'title': 'Ковер', 'price': 2000, 'color': 'green'},

{'title': 'Диван для отдыха', 'color': 'black'}

]

- В качестве первого аргумента генератор принимает список словарей, дальше через *args генератор принимает неограниченное количествово 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.

Шаблон для реализации генератора:

```
# Пример:
# goods = [
#    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
#    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
# ]
# 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}
def field(items, *args):
    assert len(args) > 0
    # Необходимо реализовать генератор
```

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.

Шаблон для реализации класса-итератора:

```
# Итератор для удаления дубликатов
class Unique(object):
    def __init__(self, items, **kwargs):
        # Нужно реализовать конструктор
        # В качестве ключевого аргумента, конструктор должен принимать bool-параметр ignore_case,
        # в зависимости от значения которого будут считаться одинаковыми строки в разном регистре
        # Например: ignore_case = True, Абв и АБВ - разные строки
        #           ignore_case = False, Абв и АБВ - одинаковые строки, одна из которых удалится
        # По-умолчанию ignore_case = False
        pass
    def __next__(self):
        # Нужно реализовать __next__
        pass
    def __iter__(self):
        return self
```

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. Пример:

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

Вывод: [123, 100, -100, -30, 30, 4, -4, 1, -1, 0]

Необходимо решить задачу двумя способами:

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

Шаблон реализации:

```
data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
if __name__ == '__main__':
    result = ...
    print(result)
    result_with_lambda = ...
    print(result_with_lambda)
```

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.

Шаблон реализации:

```
# Здесь должна быть реализация декоратора
@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()
```

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.

После завершения блока кода в консоль должно вывестись time: 5.5 (реальное время может несколько отличаться).

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.

Шаблон реализации:

```
import json
import sys
# Сделаем другие необходимые импорты
path = None
# Необходимо в переменную path сохранить путь к файлу, который был передан при запуске сценария
with open(path) as f:
    data = json.load(f)
# Далее необходимо реализовать все функции по заданию, заменив `raise NotImplemented`
# Предполагается, что функции f1, f2, f3 будут реализованы в одну строку
# В реализации функции f4 может быть до 3 строк
@print_result
def f1(arg):
    raise NotImplemented
@print_result
def f2(arg):
    raise NotImplemented
@print_result
def f3(arg):
    raise NotImplemented
@print_result
def f4(arg):
    raise NotImplemented
if __name__ == '__main__':
    with cm_timer_1():
        f4(f3(f2(f1(data))))
```

Листинг программы

Файл field.py

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

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


def task_1():

    goods = [
        {"title": "Ковер", "price": 2000, "color": "green"},
        {"title": "Диван для отдыха", "color": "black"},
        {"title": None, "price": 3500},
    ]

    raw = input("Ключ(и) через запятую: ").strip()
    keys = [k.strip() for k in raw.split(",") if k.strip()]

    if not keys:
        print("Ключи не заданы")
        return

    result = list(field(goods, *keys))
    print("Результат:", result)
```

Файл gen_random.py

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


def task_2():
    num_count = int(input("Сколько чисел сгенерировать: "))
    begin = int(input("Начало диапазона: "))
    end = int(input("Конец диапазона: "))
    result = list(gen_random(num_count, begin, end))
    print([num for num in result])
```

Файл unique.py

```
class Unique:

    def __init__(self, items, **kwargs):
        self._iter = iter(items)
        self._ignore_case = bool(kwargs.get('ignore_case', False))
```

```
        self._seen = set()

    def _key(self, value):
        if isinstance(value, str) and self._ignore_case:
            return value.lower()
        return value

    def __iter__(self):
        return self

    def __next__(self):
        for item in self._iter:
            key = self._key(item)
            try:
                if key in self._seen:
                    continue
                self._seen.add(key)
                return item
            except TypeError:
                key = repr(key)
                if key in self._seen:
                    continue
                self._seen.add(key)
                return item
        raise StopIteration


def task_3():

    numbers = [1, 1, 1, 2, 2, 3, 3, 3]
    print("\nЧисла:", numbers)
    print("Результат:", list(Unique(numbers)))

    words = ["a", "A", "b", "B", "a", "b"]
    print("\nСтроки (без учёта регистра):", words)
    print("Результат:", list(Unique(words)))

    print("\nСтроки (с учётом регистра):", words)
    print("Результат:", list(Unique(words, ignore_case=True)))
```

Файл sort.py

```
def task_4():
    data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
    print("Исходные данные:", data)

    result = sorted(data, key=abs, reverse=True)
    print("Сортировка без lambda:", result)

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

Файл print_result.py

```
from functools import wraps

def print_result(func):

    @wraps(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


def task_5():

    @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]

    print("Результаты вызовов функций:\n")

    test_1()
    test_2()
    test_3()
    test_4()
```

Файл 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):
        print(f"time: {time.time() - self.start_time:.3f} sec")


@contextmanager
def cm_timer_2():
    start = time.time()
    try:
        yield
    finally:
        print(f"time: {time.time() - start:.3f} sec")


def task_6():

    from time import sleep
```

```
print("Демонстрация cm_timer_1 (ожидание 5 с):")
with cm_timer_1():
    sleep(5)

print("\nДемонстрация cm_timer_2 (ожидание 5 с):")
with cm_timer_2():
    sleep(5)
```

Файл process_data.py

```
import json
import os
from .print_result import print_result
from .cm_timer import cm_timer_1
from .gen_random import gen_random
from .unique import Unique

_JOB_KEYS = ('name', 'job-name', 'job', 'profession', 'title')

def Get_title(d):
    for k in _JOB_KEYS:
        if k in d and d[k]:
            return d[k]
    return None


@print_result
def f1(data):
    titles = filter(None, map(Get_title, data))
    return sorted(Unique(titles, ignore_case=True), key=lambda s: s.casefold())


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


@print_result
def f3(professions):
    return [f"{s} с опытом Python" for s in professions]


@print_result
def f4(professions):
    salaries = list(gen_random(len(professions), 100_000, 200_000))
    return [f"{title}, зарплата {salary} руб." for title, salary in zip(professions, salaries)]


def task_7():
    print("Задача 7: обработка данных (f1 → f2 → f3 → f4)\n")

    path = input("Запустить (Enter): ").strip()
    if not path:
        path = os.path.join(os.path.dirname(__file__), "data_light.json")

    try:
        with open(path, "r", encoding="utf-8-sig") as f:
            data = json.load(f)
    except FileNotFoundError:
        print("Файл не найден")
        return
    except json.JSONDecodeError as e:
```

```
        print("Ошибка декодирования JSON:", e)
        return

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

Файл menu.py

```
from lab_python_fp.field import task_1

from lab_python_fp.gen_random import task_2

from lab_python_fp.unique import task_3

from lab_python_fp.sort import task_4

from lab_python_fp.print_result import task_5

from lab_python_fp.cm_timer import task_6

from lab_python_fp.process_data import task_7


def run_menu():

    MENU = """

    ================ Меню =================

    1) Задача 1

    2) Задача 2

    3) Задача 3

    4) Задача 4

    5) Задача 5

    6) Задача 6

    7) Задача 7

    0) Выход

    ----------------------------------------

    Выберите задачу:

    """


    actions = {

        "1": task_1, "2": task_2, "3": task_3, "4": task_4,

        "5": task_5, "6": task_6, "7": task_7,

    }


    while True:

        choice = input(MENU).strip()

        if choice == "0":

            print("Выход.")

            break
```

```
    action = actions.get(choice)
    if action:
        print("\n--- Старт задачи ---")
        action()
        print("--- Конец задачи ---\n")
    else:
        print("Неизвестный пункт. Попробуй ещё раз.\n")
```

Файл main.py

```
from menu import run_menu


if __name__ == "__main__":
    run_menu()
```

Скриншоты работы программы

```
~/Desktop/PyCharmMiscProject/ПиКЯП/Лабы/lab_3 git:(main)  2 files changed, 0 insertions(+), 0 deletions(-)
python3 main.py

    ================ Меню =================
    1) Задача 1
    2) Задача 2
    3) Задача 3
    4) Задача 4
    5) Задача 5
    6) Задача 6
    7) Задача 7
    0) Выход
    --------------------------------------
    Выберите задачу:
    1

--- Старт задачи ---
Ключ(и) через запятую: title, price
Результат: [{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха'}, {'price': 3500}]
--- Конец задачи ---
```

```
~/Desktop/PyCharmMiscProject/ПиКЯП/Лабы/lab_3 git:(main)  3 files changed, 0 insertions(+), 0 deletions(-)
python3 main.py

    ================ Меню =================
    1) Задача 1
    2) Задача 2
    3) Задача 3
    4) Задача 4
    5) Задача 5
    6) Задача 6
    7) Задача 7
    0) Выход
    --------------------------------------
    Выберите задачу:
    2

--- Старт задачи ---
Сколько чисел сгенерировать: 10
Начало диапазона: 10
Конец диапазона: 100
[27, 65, 29, 54, 85, 90, 30, 26, 22, 72]
--- Конец задачи ---
```

```
    ================ Меню =================
    1) Задача 1
    2) Задача 2
    3) Задача 3
    4) Задача 4
    5) Задача 5
    6) Задача 6
    7) Задача 7
    0) Выход
    --------------------------------------
    Выберите задачу:
    3

--- Старт задачи ---

Числа: [1, 1, 1, 2, 2, 3, 3, 3]
Результат: [1, 2, 3]

Строки (без учёта регистра): ['a', 'A', 'b', 'B', 'a', 'b']
Результат: ['a', 'A', 'b', 'B']

Строки (с учётом регистра): ['a', 'A', 'b', 'B', 'a', 'b']
Результат: ['a', 'b']
--- Конец задачи ---
```

```
    ================ Меню =================
    1) Задача 1
    2) Задача 2
    3) Задача 3
    4) Задача 4
    5) Задача 5
    6) Задача 6
    7) Задача 7
    0) Выход
    --------------------------------------
    Выберите задачу:
    4

--- Старт задачи ---
Исходные данные: [4, -30, 100, -100, 123, 1, 0, -1, -4]
Сортировка без lambda: [123, 100, -100, -30, 4, -4, 1, -1, 0]
Сортировка с lambda: [123, 100, -100, -30, 4, -4, 1, -1, 0]
--- Конец задачи ---
```

```
    ================ Меню =================
    1) Задача 1
    2) Задача 2
    3) Задача 3
    4) Задача 4
    5) Задача 5
    6) Задача 6
    7) Задача 7
    0) Выход
    ---------------------------------------
    Выберите задачу:
    5

--- Старт задачи ---
Результаты вызовов функций:

test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
--- Конец задачи ---
```

```
    ================ Меню =================
    1) Задача 1
    2) Задача 2
    3) Задача 3
    4) Задача 4
    5) Задача 5
    6) Задача 6
    7) Задача 7
    0) Выход
    ---------------------------------------
    Выберите задачу:
    6

--- Старт задачи ---
Демонстрация cm_timer_1 (ожидание 5 с):
time: 5.001 sec

Демонстрация cm_timer_2 (ожидание 5 с):
time: 5.005 sec
--- Конец задачи ---
```

Для файла process_data.py вывод слишком длинный, потому предоставлен только скриншот работы функций f2,f3,f4.

```
~/Desktop/PyCharmMiscProject/ПиКЯП/Лабы/lab_3 git:(main)  4 files changed, 1 deletion(-)
python3 main.py

    ================ Меню =================
    1) Задача 1
    2) Задача 2
    3) Задача 3
    4) Задача 4
    5) Задача 5
    6) Задача 6
    7) Задача 7
    0) Выход
    ---------------------------------------
    Выберите задачу:
    7

--- Старт задачи ---
Задача 7: обработка данных (f1 → f2 → f3 → f4)

Запустить (Enter):
f2
Программист
Программист / Senior Developer
Программист 1С
Программист C#
Программист C++
Программист C++/C#/Java
Программист/ Junior Developer
Программист/ технический специалист
Программистр-разработчик  информационных систем
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
Программистр-разработчик  информационных систем с опытом Python
f4
Программист с опытом Python, зарплата 146186 руб.
Программист / Senior Developer с опытом Python, зарплата 129751 руб.
Программист 1С с опытом Python, зарплата 178928 руб.
Программист C# с опытом Python, зарплата 110405 руб.
Программист C++ с опытом Python, зарплата 122364 руб.
Программист C++/C#/Java с опытом Python, зарплата 158698 руб.
Программист/ Junior Developer с опытом Python, зарплата 103060 руб.
Программист/ технический специалист с опытом Python, зарплата 122420 руб.
Программистр-разработчик  информационных систем с опытом Python, зарплата 125796 руб.
time: 0.006 sec
--- Конец задачи ---
```