МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
им. Н.Э. Баумана

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

A logo with a gold star and blue ribbon

Description automatically generated

Дисциплина “Парадигмы и конструкции языков программирования”

Отчет по лабораторной работе №2  
“Объектно-ориентированные возможности языка Python”

**Выполнил:**Студент группы ИУ5-35Б  
Ким А. А.  
**Преподаватель:**Нардид А.Н.

Москва 2025

**Задание**

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.

* 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.

**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.

Необходимо решить задачу двумя способами:

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

**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.

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](https://github.com/ugapanyuk/BKIT_2021/blob/main/notebooks/fp/files/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.

**Листинг кода**

**­\_init\_.py:**

def field(items, \*args):  
 assert len(args) > 0  
  
 if len(args) == 1:  
 key = args[0]  
 for item in items:  
 if key in item and item[key] is not None:  
 yield item[key]  
 else:  
 for item in items:  
 result = {}  
 for key in args:  
 if key in item and item[key] is not None:  
 result[key] = item[key]  
 if result:  
 yield result  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 goods = [  
 {'title': 'Ковер', 'price': 2000, 'color': 'green'},  
 {'title': 'Диван для отдыха', 'color': 'black'},  
 {'title': None, 'price': 3000},  
 {'price': 4000}  
 ]  
  
 print("Test 1 - один аргумент:")  
 for item in field(goods, 'title'):  
 print(item)  
  
 print("\nTest 2 - несколько аргументов:")  
 for item in field(goods, 'title', 'price'):  
 print(item)

**circle.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:")  
 for num in gen\_random(5, 1, 3):  
 print(num, end=' ')  
 print()

**color.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):  
 compare\_item = item.lower()  
 else:  
 compare\_item = item  
  
 if compare\_item not in self.seen:  
 self.seen.add(compare\_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()

**figure.py:**

data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 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)

**rectangle.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()

**square.py:**

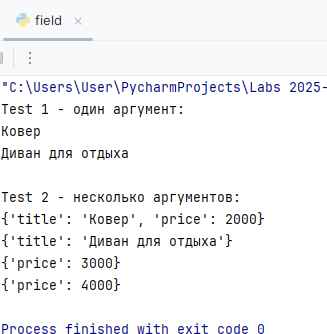
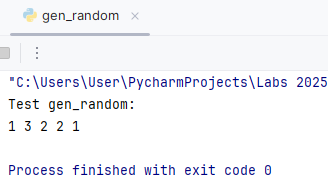
import time  
from contextlib import contextmanager  
  
class cm\_timer\_1:  
 def \_\_enter\_\_(self):  
 self.start\_time = time.time()  
 return self  
  
 def \_\_exit\_\_(self, exc\_type, exc\_val, exc\_tb):  
 elapsed\_time = time.time() - self.start\_time  
 print(f"time: {elapsed\_time:.1f}")  
  
  
@contextmanager  
def cm\_timer\_2():  
 start\_time = time.time()  
 try:  
 yield  
 finally:  
 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(0.5)

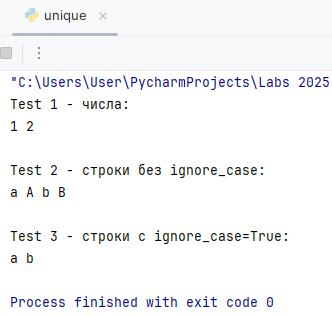
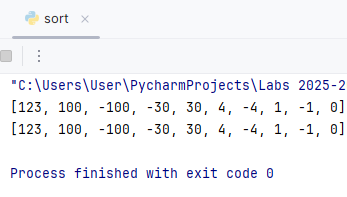
**main.py:**

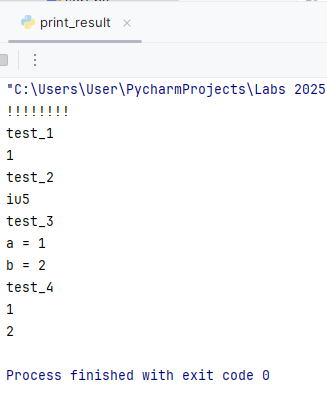
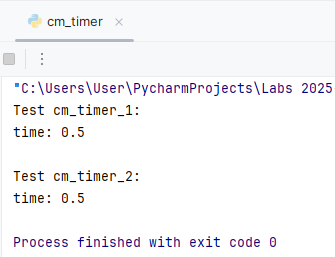
import json  
import sys  
  
from field import field  
from gen\_random import gen\_random  
from unique import Unique  
from print\_result import print\_result  
from cm\_timer import cm\_timer\_1  
  
path = sys.argv[1] if len(sys.argv) > 1 else 'data\_light.json'  
  
with open(path, encoding='utf-8') as f:  
 data = json.load(f)  
  
@print\_result  
def f1(arg):  
 return sorted(Unique([item['job-name'] for item in arg], 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))  
 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))))

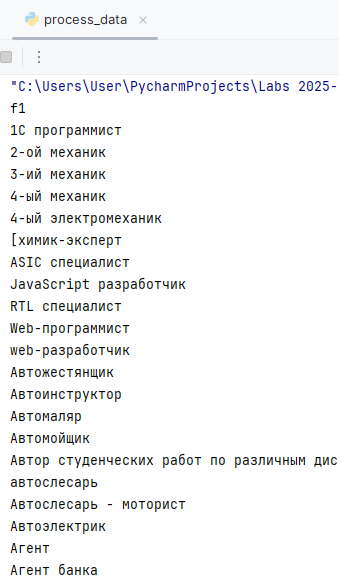
data\_light.json был взят из методички.

**Скриншоты работы программ**



…………………