**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и системы управления»

Кафедра ИУ5. Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №4

«Функциональные возможности языка Python ч.2»

|  |  |  |
| --- | --- | --- |
| Выполнил:  Макеева Е. А.  ИУ5-31Б |  | Проверил:  Нардид А. Н. |
| Подпись и дата: |  | Подпись и дата: |

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.

**Задача 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. Пример:

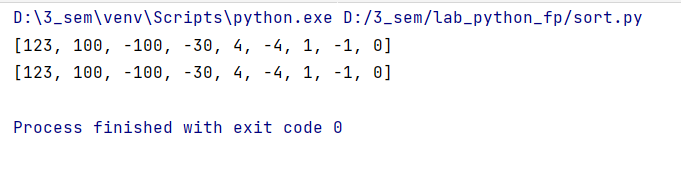
data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]

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.

data = [4, -30, 100, -100, 123, 1, 0, -1, -4]  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 print(sorted(data, key=abs, reverse=True))  
  
 print(sorted(data, key=lambda x: abs(x), reverse=True))

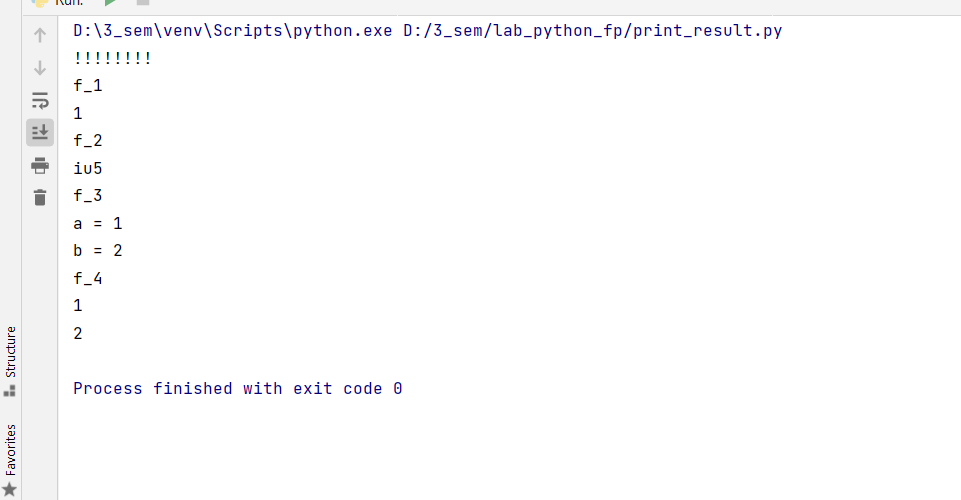


**Задача 5**

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.

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 f\_1():  
 return 1  
  
  
@print\_result  
def f\_2():  
 return 'iu5'  
  
  
@print\_result  
def f\_3():  
 return {'a': 1, 'b': 2}  
  
  
@print\_result  
def f\_4():  
 return [1, 2]  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 print('!!!!!!!!')  
 f\_1()  
 f\_2()  
 f\_3()  
 f\_4()

****

**Задача 6**

Необходимо написать контекстные менеджеры cm\_timer\_1 и cm\_timer\_2, которые считают время работы блока кода и выводят его на экран. Пример:

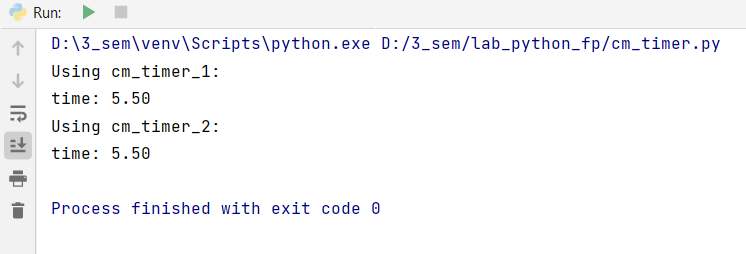
with cm\_timer\_1():

sleep(5.5)

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).

from time import time, sleep  
from contextlib import contextmanager  
  
  
class cm\_timer\_1:  
 def \_\_enter\_\_(self):  
 self.start\_time = time()  
 return self  
  
 def \_\_exit\_\_(self, exception\_type, exception\_value, traceback):  
 self.end\_time = time()  
 elapsed\_time = self.end\_time - self.start\_time  
 print(f'time: {elapsed\_time:.2f}')  
  
  
@contextmanager  
def cm\_timer\_2():  
 start\_time = time()  
 try:  
 yield  
 finally:  
 end\_time = time()  
 elapsed\_time = end\_time - start\_time  
 print(f'time: {elapsed\_time:.2f}')  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 print("Using cm\_timer\_1:")  
 with cm\_timer\_1():  
 sleep(5.5)  
  
 print("Using cm\_timer\_2:")  
 with cm\_timer\_2():  
 sleep(5.5)



**Задача 7**

* 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.

import json  
import random  
from print\_result import print\_result  
from cm\_timer import cm\_timer\_1  
  
path = 'data\_light.json'  
with open(path, encoding="utf-8") as f:  
 data = json.load(f)  
  
@print\_result  
def f1(arg):  
 return sorted(set(job['job-name'].strip().lower() for job in arg))  
  
  
@print\_result  
def f2(arg):  
 return list(filter(lambda x: x.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 = random.sample(range(100000, 200001), len(arg))  
 return [f"{profession}, зарплата {salary} руб." for profession, salary in zip(arg, salaries)]  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 with cm\_timer\_1():  
 f4(f3(f2(f1(data))))

и т. д.



