**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»

Отчет по лабораторной работе №3-4

«Функциональные возможности языка Python.»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-35Б |  |  |
| Герасимович Н.И. |  |  |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2024 г

**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.

**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](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.

**Текст программы:**

**Sort.py**

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

**Cm\_timer.py**

from contextlib import contextmanager  
import time  
  
class cm\_timer\_1:  
 def \_\_enter\_\_(self):  
 self.start\_time = time.time()  
 return self  
  
 def \_\_exit\_\_(self, exc\_type, exc\_value, traceback):  
 self.end\_time = time.time()  
 elapsed\_time = self.end\_time - self.start\_time  
 print(f"time: {elapsed\_time:.1f}")  
  
  
@contextmanager  
def cm\_timer\_2():  
 start\_time = time.time()  
 yield  
 end\_time = time.time()  
 elapsed\_time = end\_time - start\_time  
 print(f"time: {elapsed\_time:.1f}")  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
  
 with cm\_timer\_1():  
 time.sleep(5.5)  
  
  
 with cm\_timer\_2():  
 time.sleep(5.5)

**Field.py**

def field(items, \*args):  
 assert len(args) > 0  
 if len(args) == 1:  
 for item in items:  
 value = item.get(args[0])  
 for value in items:  
 print(value)  
 if value is not None:  
 yield value  
 else:  
 for item in items:  
 result = {key: item.get(key) for key in args if item.get(key) is not None}  
 print(result)  
 if result:  
 yield result  
  
goods = [  
 {'title': 'Ковер', 'price': 2000, 'color': 'green'},  
 {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},  
 {'title': 'Стол', 'color': None},  
 {'title': None, 'price': None}  
]  
print("/////////////////Первый способ//////////////////////////")  
for title in field(goods, 'title'):  
 print(title)  
print("/////////////////Второй способ//////////////////////////")  
for item in field(goods, 'title', 'price'):  
 print(item)  
print("/////////////////Третий способ//////////////////////////")  
for item in field(goods, 'title', 'color'):  
 print(item)  
print("/////////////////Четвёртый способ//////////////////////////")  
for item in field(goods, 'title', 'price', 'color'):  
 print(item)

**Gen\_random.py**

import random  
  
def gen\_random(num\_count, begin, end):  
 for \_ in range(num\_count):  
 yield random.randint(begin, end)  
  
for number in gen\_random(5,1,3):  
 print(number)

**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 time  
import json  
import sys  
from random import randint  
from contextlib import contextmanager  
  
  
  
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  
  
  
  
class cm\_timer\_1:  
 def \_\_enter\_\_(self):  
 self.start\_time = time.time()  
 return self  
  
 def \_\_exit\_\_(self, exc\_type, exc\_value, traceback):  
 self.end\_time = time.time()  
 elapsed\_time = self.end\_time - self.start\_time  
 print(f"time: {elapsed\_time:.1f}")  
  
  
path = sys.argv[1]  
  
with open(path,encoding = 'utf8') as f:  
 data = json.load(f)  
  
  
  
@print\_result  
def f1(arg):  
 return sorted(set(entry['job-name'] for entry in arg), key=lambda x: x.lower())  
  
  
@print\_result  
def f2(arg):  
 return list(filter(lambda x: x.startswith('Программист'), arg))  
  
  
@print\_result  
def f3(arg):  
 return [f"{jobname} с опытом Python" for jobname in arg]  
  
@print\_result  
def f4(arg):  
 salaries = [randint(100000, 200000) for \_ in arg]  
 return [f"{profession}, зарплата {salary} руб." for profession, salary in zip(arg, salaries)]  
  
  
with cm\_timer\_1():  
 f4(f3(f2(f1(data))))

**Unique.py**

class Unique(object):  
 def \_\_init\_\_(self, items, \*\*kwargs):  
 self.items = iter(items)  
 self.seen = set()  
 self.ignore\_case = kwargs.get('ignore\_case', False)  
  
 def \_\_iter\_\_(self):  
 return self  
  
 def \_\_next\_\_(self):  
 while True:  
 value = next(self.items)  
 if isinstance(value, str) and self.ignore\_case:  
 key = value.lower()  
 else:  
 key = value  
  
 if key not in self.seen:  
 self.seen.add(key)  
 return value  
  
print("/////////////////Первый способ//////////////////////////")  
data1 = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]  
unique\_data1 = Unique(data1)  
for value in unique\_data1:  
 print(value)  
print("/////////////////Второй способ//////////////////////////")  
data2 = (x for x in range(1, 4))  
unique\_data2 = Unique(data2)  
for value in unique\_data2:  
 print(value)  
print("/////////////////Третий способ//////////////////////////")  
data3 = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']  
unique\_data3 = Unique(data3)  
for value in unique\_data3:  
 print(value)  
print("/////////////////Четвёртый способ//////////////////////////")  
data4 = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']  
unique\_data4 = Unique(data4, ignore\_case=True)  
for value in unique\_data4:  
 print(value)