|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | Инженерный бизнес и менеджмент |
| КАФЕДРА | Промышленная логистика |

**Лабораторная работа №2-3  
«Парадигмы и конструкции языков программирования»**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент |  |  |  |  |  |
|  | (группа) |  | (подпись, дата) |  | (И.О. Фамилия) |
| Преподаватель |  |  |  |  |  |
|  |  |  | (подпись, дата) |  | (И.О. Фамилия) |
|  |  |  |  |  |  |

Москва, 2024 г

**Функциональные возможности языка Python.**

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

**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.

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)**

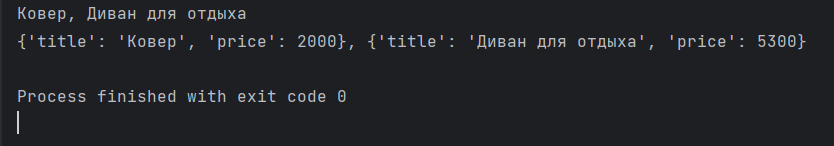
Необходимо написать контекстные менеджеры cm\_timer\_1 и cm\_timer\_2, которые считают время работы блока кода и выводят его на экран.

**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.

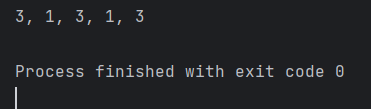
Задание 1

goods = [  
 {'title': 'Ковер', 'price': 2000, 'color': 'green'},  
 {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},  
 {'price': None}]  
  
  
def field(items, \*args):  
 assert len(args) > 0  
 if len(args) == 1:  
 return ', '.join(i.get(args[0]) for i in items if i.get(args[0]))  
 result = [{i: j.get(i) for i in args if j.get(i)} for j in items]  
 result = [str(i) for i in result if i]  
 return ', '.join(result)  
  
  
print(field(goods, 'title'))  
print(field(goods, 'title', 'price'))



Задание 2

from random import randint  
  
  
def get\_random(num\_count, begin, end):  
 return ', '.join(str(randint(begin, end)) for i in range(num\_count))  
  
  
print(get\_random(5, 1, 3))

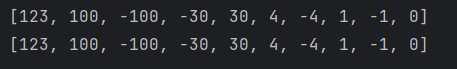


Задание 3

class Unique(object):  
 def \_\_init\_\_(self, items, \*\*kwargs):  
 kwargs['ignore\_case'] = kwargs.setdefault('ignore\_case', False)  
 self.kwargs = kwargs  
 self.items = items  
 self.s = []  
 for i in self.items:  
 if isinstance(i, str) and self.kwargs['ignore\_case']:  
 if i.upper() not in self.s and i.lower() not in self.s:  
 self.s.append(i)  
 elif i not in self.s:  
 self.s.append(i)  
  
 def \_\_next\_\_(self):  
 assert len(self.s) > 0, 'итератор пуст'  
 return self.s.pop(0)  
  
 def \_\_iter\_\_(self):  
 return self  
  
  
data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']  
a = Unique(data)  
print(next(a))  
print(next(a))  
print(next(a))  
print(next(a))  
print(next(a))

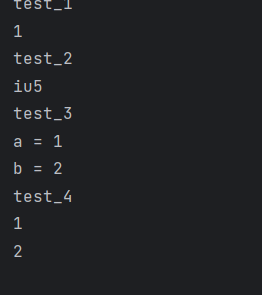
Задание 4

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)



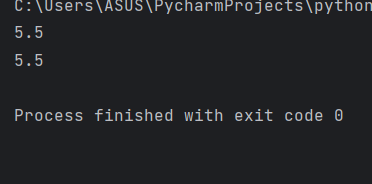
Задание 5

def print\_result(f):  
 def decorated(\*args, \*\*kwargs):  
 result\_f = f(\*args, \*\*kwargs)  
 print(f.\_\_name\_\_)  
 if isinstance(result\_f, list):  
 print('\n'.join(map(str, result\_f)))  
 elif isinstance(result\_f, dict):  
 print('\n'.join(f'{i} = {result\_f[i]}' for i in result\_f))  
 else:  
 print(result\_f)  
 return f(\*args, \*\*kwargs)  
 return decorated  
  
  
@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\_\_':  
 test\_1()  
 test\_2()  
 test\_3()  
 test\_4()



Задание 6

* from contextlib import contextmanager  
  from time import time, sleep  
    
    
  @contextmanager  
  def cm\_timer\_1():  
   start = time()  
   try:  
   yield  
   finally:  
   print(round(time() - start, 2))  
    
    
  class cm\_timer\_2(object):  
   def \_\_enter\_\_(self):  
   self.t = time()  
   return self  
    
   def \_\_exit\_\_(self, type, value, traceback):  
   self.e = time()  
   print(round(self.e - self.t, 2))  
    
    
  if \_\_name\_\_ == '\_\_main\_\_':  
   with cm\_timer\_1():  
   sleep(5.5)  
    
   with cm\_timer\_2():  
   sleep(5.5)



Задание 7

import json  
from print\_result import print\_result  
from cm\_timer import cm\_timer\_1  
from random import randint  
  
  
with open('data\_light.json', encoding='utf-8') as f:  
 data = json.load(f)  
  
  
@print\_result  
def f1(arg):  
 return sorted({i['job-name'] for i in arg})  
  
  
@print\_result  
def f2(arg):  
 return list(filter(lambda x: x.split()[0] == 'программист', arg))  
  
  
@print\_result  
def f3(arg):  
 return list(map(lambda x: x + ' с опытом Python', arg))  
  
  
@print\_result  
def f4(arg):  
 return list(map(lambda x: x + f', зарплата {randint(100000, 200000)} руб.', arg))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 with cm\_timer\_1():  
 print(f1(data))  
 print(f2(f1(data)))  
 print(f3(f2(f1(data))))  
 print(f4(f3(f2(f1(data)))))