Лабораторная работа №3 – 4

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.

def field(items, \*args):  
 assert len(args) > 0  
  
 for i in items:  
 if len(args) == 1 and args[0] in i:  
 print("\'"+ i[args[0]] + "\'", end="")  
  
 else:  
 print("{", end="")  
 for j in args:  
 if j in i:  
 print("\'" + j + "\': " + "\'" + str(i[j]) + "\'", end="")  
 if j != args[len(args) - 1]:  
 print(", ", end="")  
 print("}", end="")  
 if i != items[len(items) - 1]:  
 print(", ", end="")  
  
  
goods = [  
 {'title': 'Ковер', 'price': 2000, 'color': 'green'},  
 {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}  
 ]  
field(goods, 'title')  
print("")  
field(goods, 'title', 'price', 'color')

### 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.

import random  
  
def gen\_random(num\_count, begin, end):  
 pass  
 arr =[]  
 for i in range(num\_count):  
 arr.append(random.randint(begin, end))  
 return arr  
  
print(gen\_random(6, -5, 7))

### 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.
* # Итератор для удаления дубликатов  
  class Unique(object):  
    
   def \_\_init\_\_(self, items, \*\*kwargs):  
   self.unic = set()  
   if kwargs['ignore\_case'] == True:  
   for elem in items:  
   self.unic.add(str(elem).lower())  
   else:  
   self.unic = set(items)  
    
   self.unic = iter(self.unic)  
   pass  
    
   def \_\_next\_\_(self):  
    
   return next(self.unic)  
    
   # Нужно реализовать \_\_next\_\_  
    
    
   def \_\_iter\_\_(self):  
   return self  
    
    
  data = ['A', 'a', 'B', 'b', 'c', 'c']  
  d = Unique(data, ignore\_case=False)  
    
  for elem in d:  
   print(elem)

### 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.

import operator  
from operator import abs  
  
data = [4, -30, 100, -100, 123, 1, 0, -1, -4]  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 result = sorted(data, key=operator.abs, reverse=True)  
 print(result)  
  
 result\_with\_lambda = sorted(data, key=lambda a: abs(a), reverse=True)  
 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.

# Здесь должна быть реализация декоратора  
  
def print\_result(func):  
 def wrapper\_func(\*args, \*\*kwargs):  
 print(func.\_\_name\_\_)  
 res = func(\*args, \*\*kwargs)  
 if type(res) is list:  
 for el in res:  
 print(el)  
 elif type(res) is dict:  
 d\_keys = list(dict(res).keys())  
 for k in d\_keys:  
 print(str(k) + ' = ' + str(res[k]))  
 else:  
 print(res)  
 return res  
  
 return wrapper\_func  
  
  
@print\_result  
def test\_1():  
 return  
  
  
@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.

import time  
from contextlib import contextmanager  
  
  
class cm\_timer\_1:  
 def \_\_enter\_\_(self):  
 self.curr\_time = time.time()  
 return self  
  
 def \_\_exit\_\_(self, exc\_type, exc\_val, exc\_tb):  
 print(time.time() - self.curr\_time)  
  
  
@contextmanager  
def cm\_timer\_2():  
 start\_time = time.time()  
 yield  
 print(time.time() - start\_time)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
  
 with cm\_timer\_1():  
 time.sleep(5.5)  
  
 with cm\_timer\_2():  
 time.sleep(5.5)

### 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.
* import json  
  import sys  
    
  from print\_result import print\_result  
  from unique import Unique  
  from gen\_random import gen\_random  
  from cm\_timer import cm\_timer\_1  
    
  # Сделаем другие необходимые импорты  
    
    
  # 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):  
   return sorted(Unique([d['job-name'] for d in arg], ignore\_case=True))  
    
    
  @print\_result  
  def f2(arg):  
   return list(filter(lambda x: x[:11] == 'программист', arg))  
    
    
  @print\_result  
  def f3(arg):  
   return list(map(lambda x: x + ' с опытом python', arg))  
    
    
  @print\_result  
  def f4(arg):  
   return [pair[0]+', зарплата '+str(pair[1])+' руб.' for pair in zip(arg, list(gen\_random(len(arg), 100000, 200000)))]  
    
    
  if \_\_name\_\_ == '\_\_main\_\_':  
   path = 'data\_light.json'  
    
   # Необходимо в переменную path сохранить путь к файлу, который был передан при запуске сценария  
    
   with open(path, "r", encoding='UTF8') as f:  
   data = json.load(f)  
    
   with cm\_timer\_1():  
   f4(f3(f2(f1(data))))