|  |  |
| --- | --- |
|  | 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» (ИБМ-3)  Лабораторная работа №3  По дисциплине:  «Парадигмы и конструкции языков программирования»  Студент ИБМ3- 34Б П. Я. Головастикова  (Подпись, дата)    Руководитель Ю. Е. Гапанюк  (Подпись, дата)  2025 г. | |

**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.**

Задача 1. Необходимо реализовать генератор field.

def field(items, \*args):  
 assert len(args) > 0  
   
 if len(args) == 1:  
 key = args[0]  
 for i in items:  
 if key in i and i[key] is not None:  
 yield i[key]  
 else:  
 for i in items:  
 res = {}  
 flag = False  
 for key in args:  
 if key in i and i[key] is not None:  
 res[key] = i[key]  
 flag = True  
   
 if flag:  
 yield res  
goods = [  
 {'title': 'Ковер', 'price': 2000, 'color': 'green'},  
 {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}]  
print("Один аргумент:")  
for value in field(goods, 'title'):  
 print(value)  
  
print("\nНесколько аргументов:")  
for item in field(goods, 'title', 'price'):  
 print(item)

Результат выполнения:

Изображение выглядит как текст, Шрифт, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Задача 2. 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):  
 return (random.randint(begin,end) for \_ in range(num\_count))  
print(\*list(gen\_random(5,1,3)))

Результат выполнения:

Изображение выглядит как текст, Шрифт, типография

Содержимое, созданное искусственным интеллектом, может быть неверным.

Задача 3. Необходимо реализовать итератор Unique(данные), который принимает на вход массив или генератор и итерируется по элементам, пропуская дубликаты.

class Unique(object):  
 def \_\_init\_\_(self, items, \*\*kwargs):  
 self.ignore\_case = kwargs.get('ignore\_case', False)  
 self.items = iter(items) #возвращает объект итератора  
 self.seen = set()  
 self.next\_item = None  
 self.\_get\_next()  
  
 def \_get\_next(self):  
 while True:  
 try:  
 item = next(self.items)  
 if self.ignore\_case and isinstance(item, str): #проверка на наличие в классе  
 key = item.lower() #вовзращает копию с маленькой буквы  
 else:  
 key = item  
   
 if key not in self.seen:  
 self.seen.add(key)  
 self.next\_item = item  
 return  
   
 except StopIteration:  
 self.next\_item = None  
 return  
  
 def \_\_next\_\_(self):  
 if self.next\_item is None:  
 raise StopIteration  
   
 result = self.next\_item  
 self.\_get\_next()  
 return result  
  
 def \_\_iter\_\_(self):  
 return self  
print('first test')   
data1 = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]  
for item in Unique(data1):  
 print(item, end=" ")  
  
print('\nsecond test')  
from gen\_random import gen\_random  
data2 = gen\_random(10, 1, 3)  
for item in Unique(data2):  
 print(item, end=" ")  
  
print('\nthird test')  
data3 = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']  
for item in Unique(data3):  
 print(item, end=" ")  
   
print('\nforth test')  
for item in Unique(data3, ignore\_case=True):  
 print(item, end=" ")

Результат выполнения:

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.

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.

data = [4, -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)

Результат выполнения:

Изображение выглядит как текст, Шрифт, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Задача 5. Необходимо реализовать декоратор print\_result, который выводит на экран результат выполнения функции.

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()

Результат выполнения:

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.

Задача 6. 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 time import sleep  
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):  
 end\_time = time.time()  
 execution\_time = end\_time - self.start\_time  
 print(f"time: {execution\_time}")  
   
@contextmanager  
def cm\_timer\_2():  
 start\_time = time.time()  
 try:  
 yield  
 finally:  
 end\_time = time.time()  
 execution\_time = end\_time - start\_time  
 print(f"time: {execution\_time}")  
 from time import sleep  
   
if \_\_name\_\_ == "\_\_main\_\_":  
 print("Тест cm\_timer\_1 (на основе класса):")  
 with cm\_timer\_1():  
 sleep(2.5)  
   
print("\nТест cm\_timer\_2 (с использованием contextlib):")  
with cm\_timer\_2():  
 sleep(1.5)

Задача 7.

import json  
import sys  
from time import time  
from contextlib import contextmanager  
import random  
  
path = r'C:\Users\golov\Pictures\data\_light.json'  
  
with open(path, encoding='utf-8') as f:  
 data = json.load(f)  
  
def print\_result(func):  
 def wrapper(\*args, \*\*kwargs):  
 result = func(\*args, \*\*kwargs)  
 print(result)  
 return result  
 return wrapper  
   
class cm\_timer\_1:  
 def \_\_enter\_\_(self):  
 self.start\_time = time()  
 return self  
   
 def \_\_exit\_\_(self, exc\_type, exc\_val, exc\_tb):  
 print(f"time: {time() - self.start\_time}")  
  
@print\_result  
def f1(arg):  
 return sorted(set(map(str.lower,map(lambda x:x['job-name'],arg))),key=str.lower)  
  
  
@print\_result  
def f2(arg):  
 return list(map(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):  
 salary=[f", зарплата {random.randint(100000,200000)} руб." for \_ in range(len(arg))]  
 return list(map(lambda x: x[0]+x[1],zip(arg,salary)))  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 with cm\_timer\_1():  
 f4(f3(f2(f1(data))))

Результат выполнения программы:

