|  |  |
| --- | --- |
|  | 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)  Отчет по выполнению лабораторной работы №3-4 по дисциплине “Парадигмы и конструкции языков программирования”  38.03.05 Бизнес-Информатика (уровень бакалавриата)  Студент ИБМ3-34 Шевченко Г.А.  2024 г. | |

Код программы:  
def field(items, \*args):

    assert len(args) > 0

    if len(args) == 1:

        # Один аргумент

        field\_name = args[0]

        for item in items:

            val = item.get(field\_name)

            if val is not None:

                yield val

    else:

        # Несколько аргументов

        for item in items:

            result = {}

            for field\_name in args:

                val = item.get(field\_name)

                if val is not None:

                    result[field\_name] = val

            if result:

                yield result

goods = [

    {'title': 'Стол', 'price': 4000, 'material': 'Дуб'},

    {'title': 'Окно', 'material': 'Стекло'}

]

# Пример с одним аргументом

for item in field(goods, 'title'):

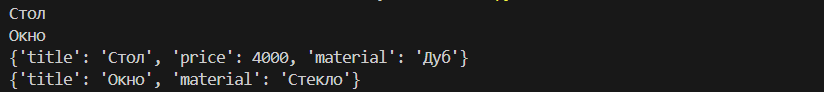
    print(item)

# Пример с несколькими аргументами

for item in field(goods, 'title', 'price','material'):

    print(item)

Результат выполнения программы:



Код программы:

import random

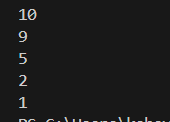
def gen\_random(num\_count, begin, end):

    for \_ in range(num\_count):

        yield random.randint(begin, end)

for number in gen\_random(5, 1, 10):

    print(number)

Результат выполнения программы:  


Код программы:

class Unique(object):

    def \_\_init\_\_(self, items, \*\*kwargs):

        self.ignore\_case = kwargs.get('ignore\_case', False)

        self.seen = set()

        self.iterator = iter(items)

    def \_\_next\_\_(self):

        while True:

            item = next(self.iterator)

            check\_item = item

            if isinstance(item, str) and self.ignore\_case:

                check\_item = item.lower()

            if check\_item not in self.seen:

                self.seen.add(check\_item)

                return item

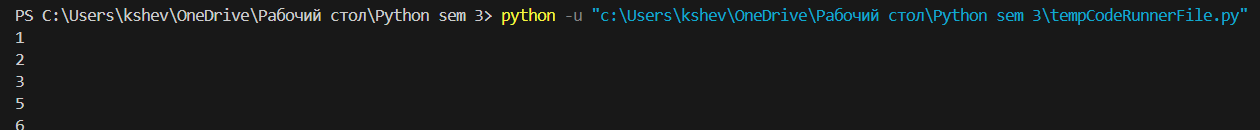
            # else continue looping

    def \_\_iter\_\_(self):

        return self

for i in Unique([1, 2, 1, 2, 3, 5, 6]):

    print(i)

Результат выполнения программы:  


Код программы:

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

if \_\_name\_\_ == '\_\_main\_\_':

    # Без использования lambda-функции

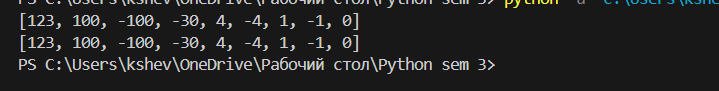
    result = sorted(data, key=abs, reverse=True)

    print(result)

    # С использованием lambda-функции

    result\_with\_lambda = sorted(data, key=lambda x: abs(x), reverse=True)

    print(result\_with\_lambda)

Результат выполнения программы:  


Код программы:  
def print\_result(func):

    def wrapper(\*args, \*\*kwargs):

        # Вызов функции и получение результата

        result = func(\*args, \*\*kwargs)

        # Печать имени функции

        print(func.\_\_name\_\_)

        # Обработка и печать результата выполнения

        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

# Применение декоратора к тестовым функциям

@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('!!!!!!!!')

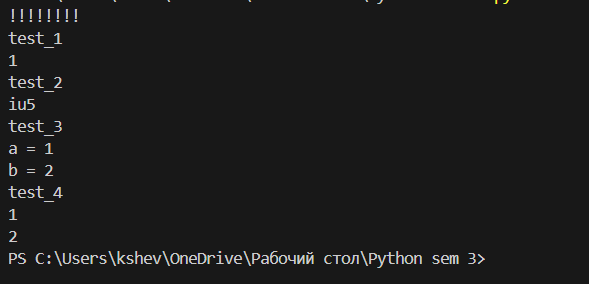
    test\_1()

    test\_2()

    test\_3()

    test\_4()

Результат выполнения программы:



Код программы:  
import json

import random

import tkinter as tk

from tkinter import filedialog

from contextlib import contextmanager

from time import time

# Функция для создания и отображения окна с результатами

def show\_results(results):

    result\_window = tk.Tk()

    result\_window.title('Результаты выполнения функций')

    for i, (func\_name, result) in enumerate(results.items()):

        tk.Label(result\_window, text=f'Результат {func\_name}:').grid(row=i\*2, column=0, sticky='w')

        text\_widget = tk.Text(result\_window, wrap='none', height=10, width=100)  # Увеличили ширину

        text\_widget.grid(row=i\*2+1, column=0, sticky='ew')  # Расширили на всю ширину

        text\_widget.insert('end', "\n".join(result))

        text\_widget.config(state='disabled')  # Сделать текстовое поле только для чтения

    result\_window.geometry("900x600")  # Устанавливаем начальные размеры окна

    result\_window.mainloop()

# Функция для печати результата выполнения функции

def print\_result(func):

    def wrapper(\*args, \*\*kwargs):

        result = func(\*args, \*\*kwargs)

        return result

    return wrapper

# Контекстный менеджер для замера времени выполнения

@contextmanager

def cm\_timer\_1():

    start\_time = time()

    yield

    elapsed\_time = time() - start\_time

    print(f"Elapsed time: {elapsed\_time:.2f} seconds")

# Открыть диалоговое окно для выбора файла

def select\_file():

    root = tk.Tk()

    root.withdraw()  # Скрыть главное окно

    file\_path = filedialog.askopenfilename(

        filetypes=[("JSON files", "\*.json")],

        title="Выберите файл JSON"

    )

    return file\_path

# Выбор файла и чтение данных

path = select\_file()

if path:

    with open(path, encoding='utf-8') as f:  # Добавили параметр encoding='utf-8'

        data = json.load(f)

@print\_result

def f1(arg):

    return sorted({job['job-name'].strip().lower() for job in arg}, key=str.lower)

@print\_result

def f2(arg):

    return list(filter(lambda job\_name: job\_name.lower().startswith("программист"), arg))

@print\_result

def f3(arg):

    return list(map(lambda job\_name: f"{job\_name} с опытом Python", arg))

@print\_result

def f4(arg):

    salaries = (random.randint(100000, 200000) for \_ in arg)

    return [f"{job\_name}, зарплата {salary} руб." for job\_name, salary in zip(arg, salaries)]

if \_\_name\_\_ == '\_\_main\_\_':

    results = {}

    with cm\_timer\_1():

        result\_f1 = f1(data)

        results['f1'] = result\_f1

        result\_f2 = f2(result\_f1)

        results['f2'] = result\_f2

        result\_f3 = f3(result\_f2)

        results['f3'] = result\_f3

        result\_f4 = f4(result\_f3)

        results['f4'] = result\_f4

    show\_results(results)

Результат выполнения программы:  
