**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №3-4

«Функциональные возможности языка Python»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-32 |  | преподаватель каф. ИУ5 |
| Щепетов Дмитрий |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2022 г.

**Описание задания**

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 (field.py)**

**Описание задачи**

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.

**Текст программы**

goods = [

    {'title': 'Ковер', 'price': 2000, 'color': 'green'},

    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}

 ]

def field(goods, \*args):

    if len(args) == 0:

        yield None

    for good in goods:

        res = {}

        for a in good:

            if a in args:

                res[a] = good[a]

        yield res

if \_\_name\_\_ == "\_\_main\_\_":

    f1 = field(goods, 'title')

    print(list(f1))

    f2 = field(goods, 'title', 'price')

    print(list(f2))

**Анализ результатов**



**Задача 2 (gen\_random.py)**

**Описание задачи**

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.

**Текст программы**

import random

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

    for i in range(num\_count):

        yield random.randint(begin, end)

    # Необходимо реализовать генератор

if \_\_name\_\_ == "\_\_main\_\_":

    print(\*gen\_random(5,1,3))

**Анализ результатов**



**Задача 3 (unique.py)**

**Описание задачи**

* 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.

**Текст программы**

# Итератор для удаления дубликатов

from gen\_random import gen\_random

class Unique(object):

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

        self.\_data = items

        self.\_ignore\_case = kwargs["ignore\_case"] if "ignore\_case" in kwargs.keys() else False

    def \_\_next\_\_(self):

        res = []

        for temp in self.\_data:

            temp = temp.lower() if type(temp) == str and self.\_ignore\_case else temp

            if temp not in res:

                res.append(temp)

        return res

    def \_\_iter\_\_(self):

        return self

if \_\_name\_\_ == "\_\_main\_\_":

    data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]

    print(Unique(data).\_\_next\_\_())

    data = gen\_random(10, 1, 3)

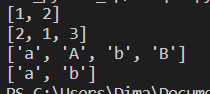
    print(Unique(data).\_\_next\_\_())

    data = ["a", "A", "b", "B", "a", "A", "b", "B"]

    print(Unique(data).\_\_next\_\_())

    print(Unique(data, ignore\_case=True).\_\_next\_\_())

**Анализ результатов**



**Задача 4 (sort.py)**

**Описание задачи**

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.

Необходимо решить задачу двумя способами:

1. С использованием lambda-функции.
2. Без использования lambda-функции.

**Текст программы**

data1 = [4, -30, 100, -100, 123, 1, 0, -1, -4]

def res(data):

    res = sorted(data, key=abs, reverse = False)

    print(res)

def res\_with\_lambda(data):

    result\_with\_lambda = sorted(data, key=lambda n: n\*n, reverse = True)

    print(result\_with\_lambda)

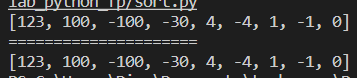
if \_\_name\_\_ == '\_\_main\_\_':

    res(data1)

    print('=====================')

    res\_with\_lambda(data1)

**Анализ результатов**



**Задача 5 (print\_result.py)**

**Описание задачи**

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.

**Текст программы**

def print\_result(func):

    def wrapper():

        print(func.\_\_name\_\_)

        f = func()

        if type(f) == dict:

            for i in f:

                print(i, '=', f.get(i))

            return 0

        if type(f) == list:

            print(\*f, sep="\n")

            return 0

        else:

            print(f)

    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\_\_':

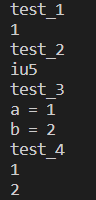
    test\_1()

    test\_2()

    test\_3()

    test\_4()

**Анализ результатов**

****

**Задача 6 (cm\_timer.py)**

**Описание задачи**

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

cm\_timer\_1 и cm\_timer\_2 реализуют одинаковую функциональность, но должны быть реализованы двумя различными способами (на основе класса и с использованием библиотеки contextlib).

**Текст программы**

**Анализ результатов**

**Задача 7 (process\_data.py)**

**Описание задачи**

* 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).
* Функция f4 должна сгенерировать для каждой специальности зарплату от 100 000 до 200 000 рублей и присоединить её к названию специальности.

**Текст программы**

import json

from print\_result import print\_result1

from cm\_timer import cm\_timer\_1

from gen\_random import gen\_random

from unique import Unique

path = "C:/Users/Dima/Documents/tasks\_c++/Python/Laba3/lab\_python\_fp/data\_light.json"

with open(path, encoding="utf8") as f:

    data = json.load(f)

def f1(arg):

    return sorted(Unique([i["job-name"] for i in arg], ignore\_case=True).\_\_next\_\_())

def f2(arg):

    return list(filter(lambda n: "программист" == n.split()[0].strip(), arg))

def f3(arg):

    return list(map(lambda n: n + " с опытом Python", arg))

@print\_result1

def f4(arg):

    randres = gen\_random(len(arg), 100000, 200000)

    res = list(zip(arg, randres))

    return [x[0] + ", зарплата " + str(x[1]) + " руб." for x in res]

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

    with cm\_timer\_1():

        f4(f3(f2(f1(data))))

**Анализ результатов**

