|  |  |
| --- | --- |
|  | **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 «ГУИМЦ»**

**Кафедра ИУ5 «Системы обработки информации и управления»**

Дисциплина «Базовые компоненты ИТ»

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №3-4

«Функциональные возможности языка Python»

Студент: Смирнов Д.И., группа ИУ5Ц-51Б

Преподаватель: Гапанюк Ю.Е.

2022г.

**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 располагаться 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 количество 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.

def field(items, \*args):

assert len(args) > 0

if len(args) == 1:

for i in items:

yield i.get(args[0])

else:

for i in items:

d = {}

for a in args:

if i.get(a) is not None:

d[a] = i.get(a)

yield d

goods = [

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

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

]

**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.

**9 lines (4 sloc) 130 Bytes**

**import random**

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

**for i in range(num\_count):**

**yield random.randint(begin, end)**

**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.

**class Unique(object):**

**def \_\_init\_\_(self, items, ignore\_case=False, \*\*kwargs):**

**self.items = items**

**self.case = ignore\_case**

**self.kwargs = kwargs**

**self.MySet = set()**

**pass**

**def \_\_next\_\_(self):**

**iterator = iter(self.items)**

**while(True):**

**try:**

**obj = next(iterator)**

**except StopIteration:**

**raise**

**# else:**

**# if obj not in self.MySet and self.case==False:**

**# self.MySet.add(obj)**

**# return obj**

**# elif self.case==True:**

**# a = str(obj)**

**# if a.lower() not in self.MySet:**

**# self.MySet.add(a)**

**# return a**

**else:**

**if self.case == True and isinstance(obj, str):**

**a = str(obj)**

**if a.lower() not in self.MySet:**

**self.MySet.add(a.lower())**

**return obj**

**elif obj not in self.MySet:**

**self.MySet.add(obj)**

**return obj**

**def \_\_iter\_\_(self):**

**return self**

**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.

**def print\_result(func):**

**def dec\_func(\*args):**

**print(func.\_\_name\_\_)**

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

**print(result)**

**return result**

**return dec\_func**

**@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]**

#

**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.

После завершения блока кода в консоль должно вывестись time: 5.5 (реальное время может несколько отличаться).

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).

**import time**

**from contextlib import contextmanager**

**class cm\_timer\_1:**

**def \_\_init\_\_(self):**

**self.start\_t = None**

**def \_\_enter\_\_(self):**

**self.start\_t = time.time()**

**def \_\_exit\_\_(self,x,y,z):**

**print(f'time: {time.time() - self.start\_t}')**

**@contextmanager**

**def cm\_timer\_2():**

**pass**

**Задача 7 (файл process\_data.py)**

В файле 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.

**`**

. . .

from lab\_python\_fp.gen\_random import gen\_random

from lab\_python\_fp.unique import Unique

from lab\_python\_fp.print\_result import test\_1,test\_2,test\_3,test\_4

from lab\_python\_fp.cm\_timer import cm\_timer\_1

from lab\_python\_fp.fieldd import field

from lab\_python\_fp.process\_data import f1,f2,f3,f4

from contextlib import contextmanager

import json

import time

def main():

print("1.Field:")

goods = [

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

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

print(\*field(goods, 'title'))

print(\*field(goods, 'title', 'price'))

print("2.Gen\_Random:", \*gen\_random(10, 2, 100))

print("3.Unique:")

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

data1 = ["a", "A", "b", "B", "a", "A", "b", "B"]

data3 = gen\_random(10, 1, 4)

print(\*Unique(data))

print(\*Unique(data1))

print(\*Unique(data1, True))

print(\*Unique(data3))

print("4.Sort:")

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

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

result\_with\_lambda = sorted(data,key = lambda a: a if a > 0 else -a, reverse=True)

print(result)

print(result\_with\_lambda)

print("5.Print\_result:")

print('!!!!!!!!')

test\_1()

test\_2()

test\_3()

test\_4()

#print("6.Cm\_timer:")

#with cm\_timer\_1():

# time.sleep(5.5)

#print("7.process\_data:")

path = 'data\_light2.json'

path2 = 'data\_light.json'

with open(path2, encoding="utf-8") as f:

data\_j = json.load(f)

#F1=f1(data\_j)

#f3(f2(f1(data\_j)))

with cm\_timer\_1():

f4(f3(f2(f1(data\_j))))

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

main()

Результат:

1.Field:

Ковер Диван для отдыха

{'title': 'Ковер', 'price': 2000} {'title': 'Диван для отдыха', 'price': 5300}

2.Gen\_Random: 38 52 33 65 58 57 51 85 58 33

3.Unique:

1 2

a A b B

a b

2 3 4

4.Sort:

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

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

5.Print\_result:

!!!!!!!!

test\_1

1

test\_2

iu5

test\_3

{'a': 1, 'b': 2}

test\_4

[1, 2]

f1

['1С программист', '2-ой механик', '3-ий механик', '4-ый механик', '4-ый электромеханик', '[химик-эксперт', 'ASIC специалист', 'JavaScript разработчик', 'RTL специалист', 'Web-программист', 'web-разработчик', 'Автожестянщик', 'Автоинструктор', 'Автомаляр', 'Автомойщик', 'Автор студенческих работ по различным дисциплинам', 'автослесарь', 'Автослесарь - моторист', 'Автоэлектрик', 'Агент', 'Агент банка', 'Агент нпф', 'Агент по гос. закупкам недвижимости', 'Агент по недвижимости', 'Агент по недвижимости (стажер)', 'Агент по недвижимости / Риэлтор', 'Агент по привлечению юридических лиц', 'Агент по продажам (интернет, ТВ, телефония) в ПАО Ростелеком в населенных пунктах Амурской области: г. Благовещенск, г. Белогорск, г. Свободный, г. Шимановск, г. Зея, г. Тында', 'Агент торговый', 'агрегатчик-топливник KOMATSU', 'агроном', 'агроном по защите растений', 'Агроном-полевод', 'агрохимик почвовед', 'Администратор', 'Администратор (удаленно)', 'Администратор Active Directory', 'Администратор в парикмахерский салон', 'Администратор зала (предприятий общественного питания)', 'Администратор кофейни', 'Администратор на ресепшен', 'Администратор на телефоне', 'Администратор по информационной безопасности', 'Администратор ресторана', 'Администратор сайта', 'Администратор ярмарок выходного дня', 'Администратор-кассир', 'Аккомпаниатор на 0,5 ст.', 'аккумуляторщик 4 разряда', 'Акушерка', 'акушерка в родильное отделение', 'Акушерка женской консультации', 'Акушерка Лысогорская врачебная амбулатория', 'Акушерка ФАП', 'Акушерка, АО', 'Акушерка, ВП', 'Альпинист промышленный', 'Аналитик', 'Анестезиолог - реаниматолог', 'анестезиолог-реаниматолог', 'анестезиолог-реаниматолог детский', 'аниматор', 'антенщик-мачтовик 4 разряда', 'аппаратчик обработки зерна', 'Аппаратчик обработки зерна 5 разряда', 'Аппаратчик пастеризации', 'Аппаратчик установки опытного производства', 'Аппаратчик химводоочистки', 'Арматурщик', 'арматурщик кузовного цеха', 'арматурщики', 'Артист (кукловод) театра кукол', 'Артист оркестра', 'Артист отдела социально - культурной деятельности Районного ЦНК', 'Артист хора', 'артист(кукловод) театра кукол', 'Артист-вокалист (солист)', 'артист-кукловод', 'архивариус', 'Архивариус (Орехово-Зуевский филиал)', 'Архитектор, картограф, инженер-проектировщик', 'Ассистент главы отделения', 'Ассистент отдела продаж', 'Ассистент режиссера', 'балетмейстер-постановщик', 'Бармен', 'Бармен-кассир в кафе', 'Бармен-официант', 'Бетонщик', 'Бетонщик (на срубку свай)', 'Бетонщик - арматурщик', 'Бетонщик-монолитчик', 'Библиограф', 'Библиотекарь', 'Библиотекарь отдела абонемента', 'Биолог', 'Боец скота', 'Бригадир в животноводстве', 'бригадир животноводства', 'бригадир мобильной бригады', 'Бригадир технического обслуживания газоиспользующего оборудования', 'Бригадир, производитель работ', 'Брокер коммерческой недвижимости', 'Брошюровщик', 'Бухгалтер', 'Бухгалтер (по заработной плате)', 'Бухгалтер 2 категории', 'Бухгалтер на группу "Обработка первичной документации"', 'Бухгалтер по ведению первичной документации', 'Бухгалтер по заработной плате', 'Бухгалтер по МТП и ГСМ', 'Бухгалтер по начислению заработной платы', 'Бухгалтер по расчету заработной платы', 'Бухгалтер по расчету калькуляции', 'Бухгалтер, ведущий', 'Бухгалтер,Ведущий бухгалтер', 'БУХГАЛТЕР-Делопроизводитель', 'бухгалтер-кассир', 'Бухгалтер-кассир 1 категории', 'Бухгалтер-материалист', 'Бухгалтер-ревизор', 'Бухгалтер-экономист', 'Вальцовщик', 'варщик зефира', 'варщик мармеладных изделий', 'Вахта', 'Вахтер', 'Вахтёр', 'Веб - программист (PHP, JS) / Web разработчик', 'веб-дизайнер', 'Веб-программист', 'ведущий агрохимик лаборатории полевых изысканий отдела агроэкологического мониторинга почв', 'Ведущий библиотекарь отдела книгохранения',