**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Дисциплина «Разработка интернет приложений»

Лабораторная работа №3

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-52Б |  | преподаватель каф. ИУ5 |
| Алексеев А. С. |  | Гапанюк Ю.Е. |
|  |  |  |

Москва, 2021 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 (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 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**

def field(items, \*args):

assert len(args) > 0

if len(args) == 1:

for num in items:

i = num.get(args[0])

if i != None:

yield i

else:

for num in items:

i = {key: val for key, val in num.items() if key in args and val != None}

if i != {}:

yield i

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

goods = [

{'title': None, 'price': 2000, 'color': 'green'},

{'title': 'Ковер', 'price': None, 'color': 'black'},

{'title': 'Диван для отдыха', 'price': 5300, 'color': None}

]

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

print(i)

for i in field(goods, 'title', 'price'):

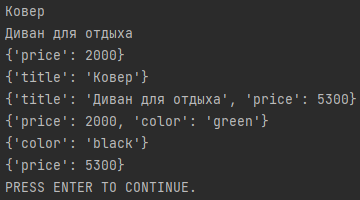
print(i)

for i in field(goods, 'price', 'color'):

print(i)

wait = input("PRESS ENTER TO CONTINUE.")

**Экранная форма с примером выполнения программы**



**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.

**Текст программы**

import random as rand

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

for i in range(num\_count):

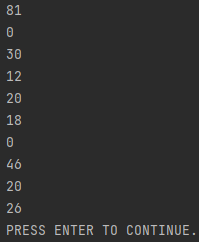
yield rand.randint(begin, end)

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

gen\_random(10, 0, 100)

wait = input("PRESS ENTER TO CONTINUE.")

**Экранная форма с примером выполнения программы**



**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.

**Текст программы**

class Unique(object):

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

self.ignore\_case = kwargs.get('ignore\_case') if len(kwargs) != 0 else False

self.type = type(items)

if self.type == list:

self.items = items[:]

else:

self.items = items

self.index = 0

self.unique\_values = set()

pass

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

if self.type == list:

while self.index < len(self.items):

el = self.items[self.index]

print(el)

self.index += 1

if (el[:].lower() if self.ignore\_case else el) not in self.unique\_values:

self.unique\_values.add(el)

return el

else:

for el in self.items:

print(el)

if (el[:].lower() if self.ignore\_case else el) not in self.unique\_values:

self.unique\_values.add(el)

return el

raise StopIteration

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

return self

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

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

data2 = (i for i in data)

x = 'A'

print(x[:].lower(), x)

for i in Unique(data):

print(i, end=' ')

print()

for i in Unique(data, ignore\_case=True):

print(i, end=' ')

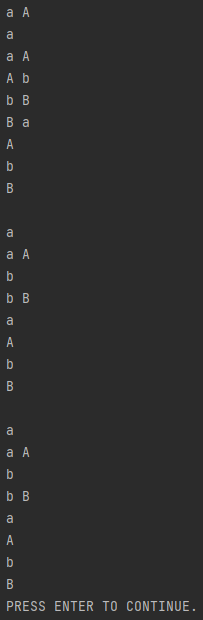
print()

for i in Unique(data2, ignore\_case=True):

print(i, end=' ')

wait = input("PRESS ENTER TO CONTINUE.")

**Экранная форма с примером выполнения программы**



**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.

**Текст программы**

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

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

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

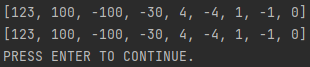
print(result)

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

print(result\_with\_lambda)

wait = input("PRESS ENTER TO CONTINUE.")

**Экранная форма с примером выполнения программы**



**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.

**Текст программы**

def print\_result(func):

def wrapper(\*args, \*\*kwargs):

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

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

if type(result) == list:

for elem in result:

print(elem)

elif type(result) == dict:

for (key, val) in result.items():

print(f"{key} = {val}")

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

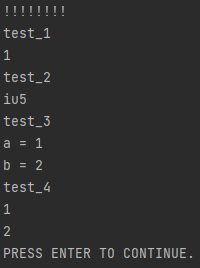
test\_1()

test\_2()

test\_3()

test\_4()

**Экранная форма с примером выполнения программы:**



**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. Пример:

with cm\_timer\_1():

sleep(5.5)

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).

**Текст программы**

import time

from contextlib import contextmanager

class cm\_timer\_1:

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

self.start\_time = time.time()

return True

def \_\_exit\_\_(self, exp\_type, exp\_value, traceback):

print('time: {}'.format(time.time() - self.start\_time))

@contextmanager

def cm\_timer\_2():

start\_time = time.time()

yield True

print('time: {}'.format(time.time() - start\_time))

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

with cm\_timer\_1():

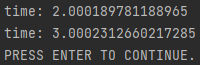
time.sleep(2)

with cm\_timer\_2():

time.sleep(3)

wait = input("PRESS ENTER TO CONTINUE.")

**Экранная форма с примером выполнения программы:**



**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/iu5team/iu5web-fall-2021/tree/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.

**Текст программы**

import json

import sys

# Сделаем другие необходимые импорты

from lab\_python\_fp.field import field

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

from lab\_python\_fp.unique import Unique

from lab\_python\_fp.print\_result import print\_result

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

path = "D:\P1\RIP\Lab№3\lab\_python\_fp\data\_light.json"

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

data = json.load(f)

@print\_result

def f1(arg):

wait = input("PRESS ENTER TO CONTINUE.")

return sorted([unique\_prof for unique\_prof in Unique([prof for prof in field(arg, 'job-name')], ignore\_case=True)])

@print\_result

def f2(arg):

wait = input("PRESS ENTER TO CONTINUE.")

return list(filter(lambda x: x.startswith('Программист'), arg))

@print\_result

def f3(arg):

wait = input("PRESS ENTER TO CONTINUE.")

return list(map(lambda x: x + " c опытом Python", arg))

@print\_result

def f4(arg):

wait = input("PRESS ENTER TO CONTINUE.")

return [f"{prof}, зарплата {salary} руб." for prof, salary in zip(arg, gen\_random(len(arg), 100000, 200000))]

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

with cm\_timer\_1():

f4(f3(f2(f1(data))))

wait = input("PRESS ENTER TO CONTINUE.")

Экранные формы с примером выполнения программы

