**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

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-31Б |  | преподаватель каф. ИУ5 |
| Прошкин Георгий |  |  |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2021 г.

**Задача №1**

**Описание задания**

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.

**Текст программы**

**field.py**

goods = [

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

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

]

def field(items, \*args):

assert len(args) > 0

if len(args) == 1:

for arg in args:

for i in range(len(items)):

for j in items[i]:

if j == arg and items[i][j] != None:

yield items[i][j]

else:

for i in range(len(items)):

d = {}

for arg in args:

for j in items[i]:

if j == arg and items[i][j] != None:

d[j] = items[i][j]

yield d

x = field(goods, 'title', 'color')

for i in range(len(goods)):

if i != len(goods) - 1:

print(next(x), end=', ')

else:

print(next(x))

**Пример выполнения программы**



**Задача №2**

**Описание задания**

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**

**gen\_random.py**

from random import randint

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

for i in range(num\_count):

print(randint(begin, end))

gen\_random(5, 1, 3)

**Пример выполнения программы**



**Задача №3**

**Описание задания**

* 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**

**gen\_random.py**

data = ['A', 'a', 'b', 'B', 'a', 'A', 'b', 1, 2, 1]

class Unique(object):

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

self.set = set()

self.items = items

self.ic = ignore\_case

self.it = iter(self.items)

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

while True:

try:

current = next(self.it)

if self.ic == True and isinstance(current, str):

temp = current[:]

if temp.lower() not in self.set:

self.set.add(temp.lower())

return current

else:

if current not in self.set:

self.set.add(current)

return current

except StopIteration:

raise

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

return self

x = Unique(data)

for i in Unique(x):

print(i)

**Пример выполнения программы**





При ignore\_case = False При ignore\_case = True

**Задача №4**

**Описание задания**

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**

**sort.py**

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

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

result = sorted(data)[::-1]

print(result)

result\_with\_lambda = sorted(data, key=lambda x: -x)

print(result\_with\_lambda)

**Пример выполнения программы**



**Задача №5**

**Описание задания**

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**

**print\_result.py**

def print\_result(func):

def decor():

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

res = func()

if isinstance(res, dict):

for i in res:

print(f"{i} = {res[i]}")

elif isinstance(res, list):

for i in res:

print(i)

else:

print(res)

return decor

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

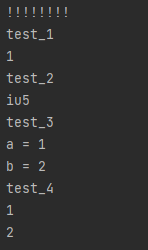
test\_1()

test\_2()

test\_3()

test\_4()

**Пример выполнения программы**



**Задача №6**

**Описание задания**

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).

**Текст программы**

**print\_result.py**

import time

from contextlib import contextmanager

from time import sleep

class cm\_timer\_1:

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

self.start\_time = time.time()

def \_\_exit\_\_(self, exc\_type, exc\_val, exc\_tb):

print('time:', time.time() - self.start\_time)

@contextmanager

def cm\_timer\_2():

start\_time = time.time()

yield

print('time:', time.time() - start\_time)

with cm\_timer\_1():

sleep(5.5)

with cm\_timer\_2():

sleep(5.5)

**Пример выполнения программы**



**Задача №7**

**Описание задания**

* В предыдущих задачах были написаны все требуемые инструменты для работы с данными. Применим их на реальном примере.
* 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.

**Текст программы**

**process\_data.py**

import json

import sys

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

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

from lab\_python\_fp.unique.unique import Unique

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

path = "data\_light.json"

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

data = json.load(f)

def Filter(x):

if x['job-name'][0:11].lower() == 'программист':

return True

else:

return False

@print\_result

def f1(arg):

return sorted([x for x in Unique([x.get('job-name') for x in arg], ignore\_case=True)])

@print\_result

def f2(arg):

return list(filter(lambda x: x[0:11].lower() == 'программист', arg))

@print\_result

def f3(arg):

return list(map(lambda x: x + ' с опытом Python', arg))

@print\_result

def f4(arg):

zipped = list(zip(arg, gen\_random(len(arg), 100000, 200000)))

return [x + str(y) for x, y in zipped]

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

with cm\_timer\_1():

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

**Пример выполнения программы**

