**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 ИУ5-32Б: |  | преподаватель каф. ИУ5 |
| Семчёнок Владимир Дмитриевич |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2022 г.

**Задание**

**Задание**

**Задача 1 (файл field.py)**

Необходимо реализовать генератор field. Генератор field последовательно выдает значения ключей словаря.

### 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.

### 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.

### 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.

### 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.

### 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.

### 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/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). 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.

Листинг

cm\_timer.py

from contextlib import contextmanager

import time

from unicodedata import name

@contextmanager

def cm\_timer\_1():

start\_time = time.time()

yield

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

class cm\_timer\_2:

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

self.start\_time = 0

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

self.start\_time = time.time()

def \_\_exit\_\_(self, type, value, traceback):

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

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

with cm\_timer\_1():

time.sleep(1.5)

with cm\_timer\_2():

time.sleep(1.5)

field.py

goods = [

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

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

]

def field(items, \*args):

assert len(args) > 0

if len(args) == 1:

for i in items:

yield i[args[0]]

else:

for i in items:

yield {key:i[key] for key in args}

ans = [i for i in field(goods, 'title', 'color')]

print(ans)

#print(field(goods, 'title'))

gen\_random.py

from random import sample

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

return sample(range(begin,end), num\_count)

print(gen\_random(2,1,5))

print\_result.py

def print\_result(function\_to\_decorate):

def the\_wrapper(\*args, \*\*kwargs):

data = function\_to\_decorate(\*args, \*\*kwargs)

if (type(data) == list):

for i in data:

print(i)

elif(type(data) == dict):

for i in data:

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

else:

print(data)

return(data)

return the\_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()

sort.py

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

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)

unique.py

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

# Итератор для удаления дубликатов

class Unique:

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

self.used\_elements = set()

self.index = 0

if len(kwargs) == 0:

self.bool\_ignore\_case = False

else:

self.bool\_ignore\_case = kwargs["bool\_ignore\_case"]

self.tmp\_list = data

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

return self

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

while True:

if self.index >= len(self.tmp\_list):

raise StopIteration

else:

current = self.tmp\_list[self.index]

if (type(current) == str and self.bool\_ignore\_case):

current = current.lower()

self.index = self.index + 1

if current not in self.used\_elements:

self.used\_elements.add(current)

return current

for i in Unique(data, bool\_ignore\_case = False):

print(i, end = " ")

Тесты

