**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Разработка Интернет-Приложений»

Отчет по лабораторной работе №3

«Функциональные возможности языка Python»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-53 |  | Юрий Евгеньевич Гапанюк |
| Миронова Александра |  |  |
| Подпись и дата:  18.01.22 |  | Подпись и дата: |

Москва, 2022 г.

**Цель лабораторной работы:** изучение возможностей функционального программирования в языке Python.

**Общее задание:**

1. Задание 1

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.

1. Задание 2

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.

1. Задание 3

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.

1. Задание 4

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.
3. Задание 5

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.

1. Задание 6

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).

1. Задание 7

* 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.

**Выполнение заданий**

**Задание 1**

Текст программы:

**field.py**

def field(items, \*args):

assert len(args) > 0

result = []

if len(args) == 1:

key = args[0]

for item in items:

if key in item:

yield item[key]

else:

for item in items:

yield {key: value for key, value in item.items() if key in args}

def main():

goods = [

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

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

]

field\_gen1 = field(goods, 'title')

for i in field\_gen1:

print(i)

field\_gen2 = field(goods, 'title', 'price')

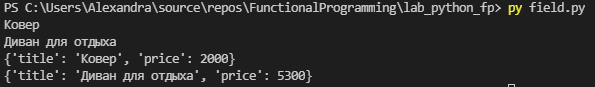
for i in field\_gen2:

print(i)

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

main()

Экранные формы с примерами выполнения программы:



**Задание 2**

Текст программы:

**gen\_random.py**

import random

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

for i in range(num\_count):

yield random.randrange(begin, end + 1, 1)

def main():

random = gen\_random(15, 1, 5)

for i in random:

print(i, end = " ")

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

main()

Экранные формы с примерами выполнения программы:



**Задание 3**

Текст программы:

**unique.py**

class Unique(object):

"""Итератор, оставляющий только уникальные значения."""

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

self.used\_elements = set()

self.data = list(data)

self.index = 0

if 'ignore\_case' in kwargs:

self.ignore\_case = kwargs['ignore\_case']

else:

self.ignore\_case = False

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

return self

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

while True:

if self.index >= len(self.data):

raise StopIteration

else:

if self.ignore\_case:

current = self.data[self.index].lower()

else:

current = self.data[self.index]

self.index = self.index + 1

if current not in self.used\_elements:

self.used\_elements.add(current)

return current

def main():

data1 = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]

data2 = ['a', 'A', 'b', 'b', 'a', 'B', 'A', 'a']

unique\_gen1 = Unique(data1)

for i in unique\_gen1:

print(i, end = " ")

print(" ")

unique\_gen2 = Unique(data2)

for i in unique\_gen2:

print(i, end = " ")

print(" ")

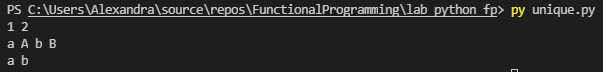
unique\_gen3 = Unique(data2, ignore\_case=True)

for i in unique\_gen3:

print(i, end = " ")

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

main()

Экранные формы с примерами выполнения программы: 

**Задание 4**

Текст программы:

**sort.py**

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

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

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

print(result)

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

print(result\_with\_lambda)

Экранные формы с примерами выполнения программы:



**Задание 5**

Текст программы:

**print\_result.py**

def print\_result(fn):

def fn\_name(\*args, \*\*kwargs):

print(fn.\_\_name\_\_)

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

if isinstance(result, dict):

for key, val in result.items():

print (key, '=', val)

elif isinstance(result, list):

for val in result:

print (val)

else:

print (result)

return result

return fn\_name

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

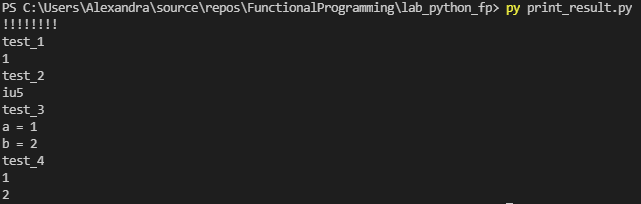
test\_1()

test\_2()

test\_3()

test\_4()

Экранные формы с примерами выполнения программы:



**Задание 6**

Текст программы:

**cm\_timer.py**

import time

from contextlib import contextmanager

from time import sleep

class cm\_timer\_1:

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

pass

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

self.start\_time = time.time()

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

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

@contextmanager

def cm\_timer\_2():

try:

start\_time = time.time()

yield start\_time

except OSError:

print("We had an error!")

finally:

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

def main():

with cm\_timer\_1():

sleep(5.5)

with cm\_timer\_2():

sleep(2.8)

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

main()

Экранные формы с примерами выполнения программы:



**Задание 7**

Текст программы:

**process\_data.py**

import json

from field import field

from gen\_random import gen\_random

from print\_result import print\_result

from cm\_timer import cm\_timer\_1

from unique import Unique

path = 'data\_light.json'

@print\_result

def f1(arg):

return sorted(Unique(field(arg, 'job-name'), ignore\_case=True))

@print\_result

def f2(arg):

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

@print\_result

def f3(arg):

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

@print\_result

def f4(arg):

return list(zip(arg, [' зарплата ' + str(x) + ' руб.' for x in gen\_random(len(arg), 100000, 200000)]))

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

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

data = json.load(f)

with cm\_timer\_1():

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

Экранные формы с примерами выполнения программы:

Тут ооочень большой список профессий. Вставила парочку скринов

