

**Московский государственный технический
университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Казовые компоненты интернет технологий»

Отчет по лабораторной работе №3
«Функциональные возможности языка Python»

Выполнил:
студент группы ИУ5-33Б:

Лебедева С.К.

Руководитель:
преподаватель каф.
ИУ5

Гапанюк Ю.Е.

Москва, 2022 г.

Цель лабораторной работы:

Изучение возможностей функционального программирования в языке Python.

Задание:

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.

Задание №1:

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

field(goods, 'title') должен выдавать 'Ковер', 'Диван для отдыха'

field(goods, 'title', 'price') должен выдавать {'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха'}

- 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:

lab_python_fp/field.py

```
import sys
import math

goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
]

def field(items, *args):
    assert len(args) > 0
    s = ''
    if len(args)==1:
        return(items[j].get(args[0]) for j in range(0,len(items)))
    else:
        my_list = list()
        for i in range(0,len(items)):
            s = '{'
```

```

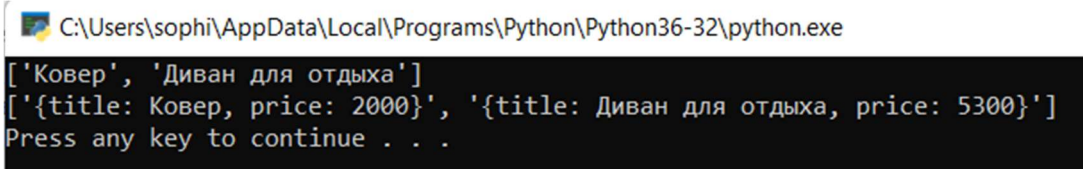
        j = 0
        for arg in args:
            if (j == 0):
                s = s + str(arg) + ': ' + str(items[i].get(arg))
            else:
                s = s + ', ' + str(arg) + ': ' + str(items[i].get(arg))
            j+=1
        s = s + '}'
        #print(s)
        my_list.append(s)
    return my_list

def main_f():
    l1 = list(field(goods, 'title')) #должен выдавать 'Ковер', 'Диван для отдыха'
    l2 = field(goods, 'title', 'price') #должен выдавать {'title': 'Ковер', 'price':
2000}, {'title': 'Диван для отдыха', 'price': 5300}
    print(l1)
    print(l2)

if __name__ == "__main__":
    main_f()

```

Примеры выполнения:



```

C:\Users\sophi\AppData\Local\Programs\Python\Python36-32\python.exe
['Ковер', 'Диван для отдыха']
['{title: Ковер, price: 2000}', '{title: Диван для отдыха, price: 5300}']
Press any key to continue . . .

```

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.

Пример:

`gen_random(5, 1, 3)` должен выдать 5 случайных чисел в диапазоне от 1 до 3, например 2, 2, 3, 2, 1.

Текст программы:

lab_python_fp/gen_random.py

```

import sys
import random

# Hint: типовая реализация занимает 2 строки
def gen_random(num_count, begin, end):
    #return(random.randint(begin, end) for j in range(0, num_count))
    for i in range(0, num_count):
        yield random.randint(begin, end)

def main_rand():
    l = list(gen_random(5, 1, 3))
    print(l)

if __name__ == "__main__":
    main_rand()

```

Примеры выполнения:

```
C:\Users\sophi\AppData\Local\Programs\Python\Python36-32\python.exe
[2, 2, 2, 1, 3]
Press any key to continue . . .
```

Задание №3:

- 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:

lab_python_fp/unique.py

```
from lab_python_fp.gen_random import gen_random

class Unique(object):
    def __init__(self, items, **kwargs):
        # Нужно реализовать конструктор
        # В качестве ключевого аргумента, конструктор должен принимать bool-параметр
        ignore_case,
        # в зависимости от значения которого будут считаться одинаковыми строки в разном
        регистре
        # Например: ignore_case = False, Абв и АБВ - разные строки
        # ignore_case = True, Абв и АБВ - одинаковые строки, одна из которых
        удалится
        # По-умолчанию ignore_case = False
        self.used_elements = set()
        self.data = items
        self.iterator = iter(items)
        self.ignore_case = False
        if 'ignore_case' in kwargs:
            self.ignore_case = kwargs['ignore_case']

    def __next__(self):
        # Нужно реализовать __next__
        while True:
            try:
                current = next(self.iterator)
            except StopIteration:
                raise StopIteration
            else:
                cur_line = str(current)
                if self.ignore_case == True:
                    if cur_line.lower() not in self.used_elements:
```

```

        self.used_elements.add(cur_line.lower())
        return current
    else:
        if cur_line not in self.used_elements:
            self.used_elements.add(cur_line)
            return current

def __iter__(self):
    return self

def main_u():
    # будет последовательно возвращать только 1 и 2
    data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
    for i in Unique(data):
        print(i, end = " ")
    print()
    # будет последовательно возвращать только 1, 2 и 3
    data = gen_random(10, 1, 3)
    for i in Unique(data):
        print(i, end = " ")
    print()
    # будет последовательно возвращать только a, A, b, B
    data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    for i in Unique(data):
        print(i, end = " ")
    print()
    # будет последовательно возвращать только a, b
    for i in Unique(data, ignore_case = True):
        print(i, end = " ")
    print()

if __name__ == "__main__":
    main_u()

```

Примеры выполнения:

```

C:\Users\sophi\AppData\Local\Programs\Python\Python36-32\python.exe
1 2
3 1 2
a A b B
a b
Press any key to continue . . .

```

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.

Текст программы:

lab_python_fp/sort.py

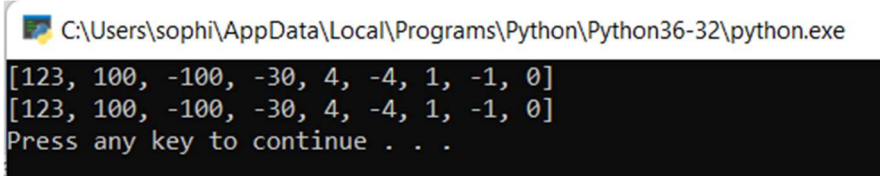
```
data = [4, -30, 100, -100, 123, 1, 0, -1, -4]

def main_s():
    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)

if __name__ == "__main__":
    main_s()
```

Примеры выполнения:



```
C:\Users\sophi\AppData\Local\Programs\Python\Python36-32\python.exe
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]
Press any key to continue . . .
```

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 выводиться в столбик через знак равенства.

Текст программы:

lab_python_fp/print_result.py

```
def print_result(func):
    def decorated_func(*args, **kwargs):
        res = func(*args, **kwargs)
        print(func.__name__)
        if isinstance(res, list):
            for i in res:
                print(i)
        elif isinstance(res, dict):
            for key, value in res.items():
                print('{key} = {value}'.format(key, value))
        else:
            print(res)
        return res
    return decorated_func

@print_result
def test_1():
    return 1
```

```

@print_result
def test_2():
    return 'iu5'

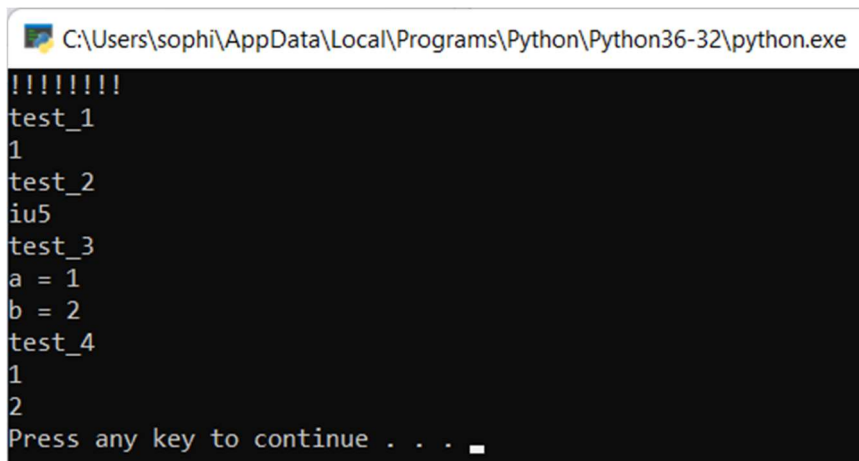
@print_result
def test_3():
    return {'a': 1, 'b': 2}

@print_result
def test_4():
    return [1, 2]

if __name__ == '__main__':
    print('!!!!!!!')
    test_1()
    test_2()
    test_3()
    test_4()

```

Примеры выполнения:



```

C:\Users\sophi\AppData\Local\Programs\Python\Python36-32\python.exe
!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
Press any key to continue . . . _

```

Задание №6:

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: 2.5` (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:

lab_python_fp/cm_timer.py

```

from contextlib import contextmanager
import time
from time import sleep

```

```

class cm_timer_1:

    def __init__(self):
        self.t = None

    def __enter__(self):
        self.t = time.time()

    def __exit__(self, exp_type, exp_value, traceback):
        print('time: {:.3f}'.format(time.time() - self.t))

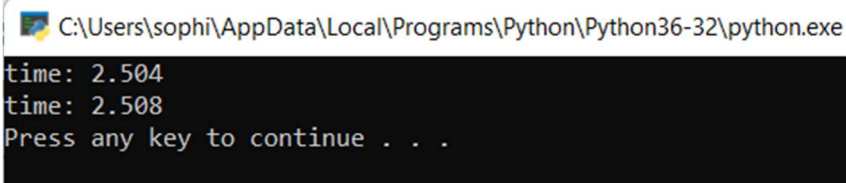
@contextmanager
def cm_timer_2():
    t = time.time()
    yield
    print('time: {:.3f}'.format(time.time() - t))

def main_timer():
    with cm_timer_1():
        sleep(2.5)
    with cm_timer_2():
        sleep(2.5)

if __name__ == "__main__":
    main_timer()

```

Примеры выполнения:



```

C:\Users\sophi\AppData\Local\Programs\Python\Python36-32\python.exe
time: 2.504
time: 2.508
Press any key to continue . . .

```

Задание №7:

- 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 С# 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 С# с опытом Python, зарплата 137287 руб. Используйте zip для обработки пары специальность — зарплата.

Текст программы:

lab_python_fp/process_data.py

```
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 = r'data_light.json'

# Необходимо в переменную path сохранить путь к файлу, который был передан при запуске
сценария

with open(path, encoding='UTF-8') as f:
    data = json.load(f)

# Далее необходимо реализовать все функции по заданию, заменив `raise NotImplemented`
# Предполагается, что функции f1, f2, f3 будут реализованы в одну строку
# В реализации функции f4 может быть до 3 строк

@print_result
def f1(arg):
    return sorted(Unique(field(arg, 'job-name'), ignore_case = True), key = lambda let:
let.lower())

@print_result
def f2(arg):
    return list(filter(lambda prog: (prog.lower()).startswith('программист'), arg))


@print_result
def f3(arg):
    return list(map(lambda prog_py: prog_py + ' с опытом Python', arg))

@print_result
def f4(arg):
```

```
return ['{0}, зарплата {1} руб.'.format(job, money) for job, money in zip(arg,
gen_random(len(arg), 100000, 200000))]
```

```
if __name__ == '__main__':
    with cm_timer_1():
        f4(f3(f2(f1(data))))
```

Примеры выполнения:

 C:\Users\sophi\AppData\Local\Programs\Python\Python36-32\python.exe

```
f2
Программист
Программист / Senior Developer
Программист 1C
Программист C#
Программист C++
Программист C++/C#/Java
Программист/ Junior Developer
Программист/ технический специалист
Программист-разработчик информационных систем
f3
Программист с опытом Python
Программист / Senior Developer с опытом Python
Программист 1C с опытом Python
Программист C# с опытом Python
Программист C++ с опытом Python
Программист C++/C#/Java с опытом Python
Программист/ Junior Developer с опытом Python
Программист/ технический специалист с опытом Python
Программист-разработчик информационных систем с опытом Python
f4
Программист с опытом Python, зарплата 114401 руб.
Программист / Senior Developer с опытом Python, зарплата 100043 руб.
Программист 1C с опытом Python, зарплата 173526 руб.
Программист C# с опытом Python, зарплата 160507 руб.
Программист C++ с опытом Python, зарплата 146992 руб.
Программист C++/C#/Java с опытом Python, зарплата 165244 руб.
Программист/ Junior Developer с опытом Python, зарплата 126474 руб.
Программист/ технический специалист с опытом Python, зарплата 186429 руб.
Программист-разработчик информационных систем с опытом Python, зарплата 161332 руб.
time: 0.299
Press any key to continue . . .
```