

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

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «БКИТ»

Отчет по лабораторной работе №3-4

Функциональные возможности языка Python»

Выполнил:

студент группы ИУ5-35Б

Нгуен Зуи Лам

Подпись и дата:

Проверил:

преподаватель каф. ИУ5

Нардид А.Н.

Подпись и дата:

Москва, 2022 г.

Описание задания

Задание лабораторной работы состоит из решения нескольких задач.

Файлы, содержащие решения отдельных задач, должны располагаться в пакете lab_python_fr. Решение каждой задачи должно располагаться 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 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.

2) 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. Пример:

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

3) 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.

Пример:

Unique(data) будет последовательно возвращать только 1 и 2.

Unique(data) будет последовательно возвращать только 1, 2 и 3.

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

Unique(data) будет последовательно возвращать только a, A, b, B.

Unique(data, ignore_case=True) будет последовательно возвращать только a,

4) 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.

Пример:

Вывод: [123, 100, -100, -30, 30, 4, -4, 1, -1, 0]

Необходимо решить задачу двумя способами:

1. С использованием lambda-функции.
2. Без использования lambda-функции.

5) 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 выводиться в столбик через знак равенства.

6) 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).

7) 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 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

Файлы пакета lab_python_fp:

Задача 1.

Field.py

```
def field(items, *args):
    assert len(args) > 0
    if len(args) == 1:
        b = [item[i] for item in items for i in item if i == args[0] and
item[i] is not None]
        return b
    else:
        b = [{i: item[i]} for item in items for i in item for arg in args if
arg == i and item[i] is not None]
        return b
```

Задача 2.

Gen_random.py

```
from random import randint

def gen_random(num_count, begin, end):
    a = [randint(begin, end) for i in range(num_count)]
    return a
```

Задача 3.

Unique.py

```
class Unique(object):
    def __init__(self, items, ignore_case, **kwargs):
        self.data = items
        self.ignore_case = ignore_case
        self.index = 0
        self.data2 = set()

    def __next__(self):
        if not self.ignore_case:
            for count, el in enumerate(self.data):
                if type(el) is str:
                    self.data[count] = el.lower()

        while True:
            if self.index >= len(self.data):
                raise StopIteration
            else:
                current = self.data[self.index]
                self.index += 1
                if current not in self.data2:
                    self.data2.add(current)
                    return current

    def __iter__(self):
        return
```

Задача 4.

Sort.py

```
def sort(data):
    result = sorted(data, key=abs, reverse=True)
    print(result)

    result_with_lambda = sorted(data, key=lambda a: abs(a), reverse=True)
    print(result_with_lambda)
```

Задача 5.

Print_result.py

```
def print_result(func):
    def wrapper(lst=[], *args, **kwargs):
        print(func.__name__)

        if len(lst) == 0:
            result = func(*args, **kwargs)
        else:
            result = func(lst, *args, **kwargs)

        if type(result) is dict:
            for key, el in result.items():
```

```

        print(f'{key} = {el}')

    elif type(result) is list:
        print('\n'.join(map(str, result)))

    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]

```

Задача 6.

Cm_timer.py

```

from time import time
from contextlib import contextmanager

@contextmanager
def cm_timer_1():
    start = time()
    yield None
    finish = time()
    print("Время работы: {}".format(finish - start))

class cm_timer_2:
    def __init__(self):
        self.start = 0
        self.finish = 0

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

    def __exit__(self, ex_type, ex_value, ex_traceback):
        self.finish = time()
        print("Время работы: {}".format(self.finish - self.start))

```

Задача 7.

Process_data.py

```

from lab_python_fp.print_result import print_result
from lab_python_fp.gen_random import gen_random

@print_result
def f1(arg) -> list:
    return sorted(list(set([i['job-name'] for i in arg])), key=lambda x:
x.lower())

```

```

@print_result
def f2(arg) -> list:
    return list(filter(lambda s: (s.split())[0].lower() == 'программист',
arg))

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

@print_result
def f4(arg) -> list:
    return list(zip(arg, ['зарплата ' + str(i) + ' руб.' for i in
gen_random(len(arg), 100000, 200000)]))

```

Результат выполнения.

Задача 1.

```

Task 1
['Ковер', 'Диван для отдыха']
[{'title': 'Ковер'}, {'price': 2000}, {'title': 'Диван для отдыха'}, {'price': 5300}]

```

Задача 2.

```

Task 2
[2, 3, 0, 1, 2]

```

Задача 3.

```

Task 3
a
b
StopIteration

```

Задача 4.

```

Task 4
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]

```

Задача 5.

```

Task 5
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2

```

Задача 6.

Task 6

Время работы: 2.013078212738037

Время работы: 2.0141351222991943

Задача 7.

Часть выполнения 7 задачи:

Task 7

f1

1С программист

2-ой механик

3-ий механик

4-ый механик

4-ый электромеханик

[химик-эксперт

ASIC специалист

JavaScript разработчик

RTL специалист

Web-программист

web-разработчик

Web-разработчик

Автожестящик

Автоинструктор

Автомаляр

Автомойщик

автомойщик

Автор студенческих работ по различным дисциплинам

f2

программист

Программист

Программист / Senior Developer

программист 1C

Программист 1C

Программист C#

Программист C++

Программист C++/C#/Java

f3

программист с опытом Python

Программист с опытом Python

Программист / Senior Developer с опытом Python

программист 1C 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

f4

('программист с опытом Python', 'зарплата 176132 руб.')

('Программист с опытом Python', 'зарплата 137536 руб.')

('Программист / Senior Developer с опытом Python', 'зарплата 150971 руб.')

('программист 1C с опытом Python', 'зарплата 183249 руб.')

('Программист 1C с опытом Python', 'зарплата 122146 руб.')

('Программист C# с опытом Python', 'зарплата 110944 руб.')

('Программист C++ с опытом Python', 'зарплата 173651 руб.')

('Программист C++/C#/Java с опытом Python', 'зарплата 104475 руб.')

Время работы: 0.03151750564575195