Московский государственный технический
университет им. Н.Э. Баумана

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

Курс «Парадигмы и конструкции языков программирования»
Отчет по лабораторным работам №3-4
«**Функциональные возможности языка Python.**»

Выполнил:
студент группы ИУ5-31Б
Баринов А. А.
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю. Е.
Подпись и дата:

Москва, 2023 г.

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 раполагаться 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.

Текст программы

```
def field(items, *args):
    assert len(args) > 0

    if len(args) == 1:
        for item in items:
            value = item.get(args[0])
            if value is not None:
                yield value
    else:
        for item in items:
            filtered_item = {key: item[key] for key in args if item.get(key) is not None}
            if filtered_item:
                yield filtered_item

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

for value in field(goods, 'title'):
    print(value)

for item in field(goods, 'title', 'price'):
                                                print(item)
```

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.

Текст программы

```
import random

def gen_random1(num_count, begin, end):
```

```
    for x in range(num_count):
        yield random.randint(begin, end)

# Пример использования:
random_numbers = list(gen_random1(5, 1, 3))
                                                print(random_numbers)
```

Задача 3 (файл unique.py)

Необходимо реализовать итератор Unique(данные), который принимает на вход массив или генератор и итерируется по элементам, пропуская дубликаты.

Текст программы

```
from gen_random import *

class Unique(object):
    def __init__(self, items, **kwargs):
        self.ignore_case = kwargs.get('ignore_case', False)
        self.seen = set()

        if self.ignore_case:
            self.items = (str(item).lower() for item in items)
        else:
            self.items = iter(items)

    def __next__(self):
        while True:
            item = next(self.items)
            key = item.lower() if self.ignore_case else item
            if key not in self.seen:
                self.seen.add(key)
                return item

    def __iter__(self):
        return self


# Примеры использования:

data1 = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
unique1 = Unique(data1)
print(list(unique1))

data2 = gen_random1(10, 1, 3)
unique2 = Unique(data2)
print(list(unique2))

data3 = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
unique3 = Unique(data3)
print(list(unique3))

unique4 = Unique(data3, ignore_case=True)
                                                print(list(unique4))
```

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.

Текст программы

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

if __name__ == '__main__':
    # С использованием lambda-функции
    result = sorted(data, key=lambda x: abs(x), reverse=True)
    print(result)

    # Без использования lambda-функции
    result_without_lambda = sorted(data, key=abs, reverse=True)
    print(result_without_lambda)
```

Задача 5 (файл print_result.py)

Необходимо реализовать декоратор print_result, который выводит на экран результат выполнения функции.

Текст программы

```
def print_result(func):
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)

        print(func.__name__)
        if isinstance(result, list):
            for item in result:
                print(item)
        elif isinstance(result, dict):
            for key, value in result.items():
                print(f"{key} = {value}")
        else:
            print(result)

        return result

    return wrapper

# Пример использования декоратора

@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()
```

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.

Текст программы

```
from contextlib import contextmanager
import time

class cm_timer_1:
    def __enter__(self):
        self.start_time = time.time()
        return self

    def __exit__(self, exc_type, exc_value, traceback):
        self.end_time = time.time()
        elapsed_time = self.end_time - self.start_time
        print(f'time: {elapsed_time}')

@contextmanager
def cm_timer_2():
    start_time = time.time()
    yield
    end_time = time.time()
    elapsed_time = end_time - start_time
    print(f'time: {elapsed_time}')

# Пример использования
with cm_timer_1():
    time.sleep(5.5)

with cm_timer_2():
    time.sleep(5.5)
```

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 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

```
from field import *
from gen_random import *
from unique import *
from print_result import *
from sort import *
from cm_timer import *
import json

with open('data_light.json', 'r', encoding='utf-8') as dat:
    data = json.load(dat)

@print_result
def f1(arg):
    return sorted(list(Unique([j['job-name'] for j in arg], ignore_case = True)))

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

@print_result
def f3(arg):
    return list(map(lambda x: f"{x}, с опытом Python", arg))

@print_result
def f4(arg):
    salary = gen_random1(len(arg), 100000, 200000)
    return [f"{job}, зарплата {salary} руб." for job, salary in zip(arg, salary)]

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

Результаты выполнения

```
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
автожестянщик
автоинструктор
автомаляр
автомойщик
автор студенческих работ по различным дисциплинам
автослесарь
автослесарь - моторист
автоэлектрик
агент
агент банка
агент нпф
агент по гос. закупкам недвижимости
агент по недвижимости
агент по недвижимости (стажер)
агент по недвижимости / риэлтор
агент по привлечению юридических лиц
```

И т. д.

```
f2
программист
программист / senior developer
программист 1с
программист c#
программист c++
программист c++/c#/java
программист/ junior developer
программист/ технический специалист
программистр-разработчик  информационных систем
f3
программист, с опытом Python
программист / senior developer, с опытом Python
программист 1с, 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
программистр-разработчик  информационных систем, с опытом Python
f4
программист, с опытом Python, зарплата 165780 руб.
программист / senior developer, с опытом Python, зарплата 136519 руб.
программист 1с, с опытом Python, зарплата 139089 руб.
программист c#, с опытом Python, зарплата 192164 руб.
программист c++, с опытом Python, зарплата 188394 руб.
программист c++/c#/java, с опытом Python, зарплата 166489 руб.
программист/ junior developer, с опытом Python, зарплата 103706 руб.
программист/ технический специалист, с опытом Python, зарплата 127794 руб.
программистр-разработчик  информационных систем, с опытом Python, зарплата 185098 руб.
time: 0.21625447273254395
```