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 лабораторной работе №3-4
«Функциональные возможности языка Python»

Выполнила:
Студентка группы ИУ5-33Б
Коренева София
Подпись и дата:

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

2023 г.

Общее описание задания

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.

Задача 1 (файл field.py)

Описание задачи

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

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

Текст программы

```python
def field(items, *args):
    assert len(args) > 0
    res = []
    for item in items:
        tmp = {}
        for arg in args:
            if item[arg] is not None:
                tmp[arg] = item[arg]
        if tmp:
            res.append(tmp)
    return tuple(res)


def task1():
    goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
    ]
    print(*[el for el in field(goods, 'title')], sep=', ')
    print(*field(goods, 'title', 'price'), sep=', ')


if __name__ == '__main__':
    task1()
```

Примеры выполнения программы

```
C:\Users\sofia\AppData\Local\Programs\Python\Python39\python.exe C:\Users\sofia\PycharmProjects\labs_koreneva\lab_python_fp\field.py
{'title': 'Ковер'}, {'title': 'Диван для отдыха'}
{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}

Process finished with exit code 0
```

Задача 2 (файл gen_random.py)

Описание задачи

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

```python
from random import randint


def gen_random(num_count, begin, end):
    for i in range(num_count):
        yield randint(begin, end)


def task2():
    print(*gen_random(5, 1, 3), sep=', ')


if __name__ == '__main__':
    task2()
```

Примеры выполнения программы

```
C:\Users\sofia\AppData\Local\Programs\Python\Python39\python.exe C:\Users\sofia\PycharmProjects\labs_koreneva\lab_python_fp\gen_random.py
3, 1, 1, 1, 3
```

Задача 3 (файл unique.py)

Описание задачи

- 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:

`data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]`

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

`data = gen_random(10, 1, 3)`

`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, b.

Текст программы

```python
from lab_python_fp.gen_random import gen_random


class Unique(object):
    def __init__(self, items, **kwargs):
        self.res = []
        for key, value in kwargs.items():
             if key == 'ignore_case' and value is True:
                items = [i.lower() for i in items]
        for item in items:
            if item not in self.res:
                self.res.append(item)
        pass

    def __next__(self):
        try:
            x = self.res[self.begin]
            self.begin += 1
            return x
        except:
            raise StopIteration

    def __iter__(self):
        self.begin = 0
        return self


def task3():
    print('Проверка работы со списком')
    data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
    for item in Unique(data):
        print(item)
    print('Проверка работы с gen_random')
    data = gen_random(10, 1, 3)
    for item in Unique(data):
        print(item)
    print('Проверка ignore_case')
    data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    print('ignore_case False')
    for item in Unique(data):
        print(item)
    print('ignore_case True')
    for item in Unique(data, ignore_case=True):
        print(item)


if __name__ == '__main__':
    task3()
```

Примеры выполнения программы

```
C:\Users\sofia\AppData\Local\Programs\Python\Python39\python.exe C:\Users\sofia\PycharmProjects\labs_koreneva\lab_python_fp\unique.py
Проверка работы со списком
1
2
Проверка работы с gen_random
3
2
1
Проверка ignore_case
ignore_case False
a
A
b
B
ignore_case True
a
b

Process finished with exit code 0
```

Задача 4 (файл sort.py)

Описание задачи

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:

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

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.

Текст программы

```python
import math

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

result = sorted(list(x for x in data), key=abs, reverse=True)
print(result)

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

Примеры выполнения программы

```
C:\Users\sofia\AppData\Local\Programs\Python\Python39\python.exe C:\Users\sofia\PycharmProjects\labs_koreneva\lab_python_fp\sort.py
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]

Process finished with exit code 0
```

Задача 5 (файл print_result.py)

Описание задачи

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.

Текст программы

```
def print_result(func):
    def wrapper(*args, **kwargs):
        res = func(*args, **kwargs)
        print(f"{func.__name__}")
        if isinstance(res, list):
            for r in res:
                print (r)
        elif isinstance(res, dict):
            for x, y in res.items():
                print (x, " = ", y)
        else:
            print(res)
        return res
    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]


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

Примеры выполнения программы

```
C:\Users\sofia\AppData\Local\Programs\Python\Python39\python.exe C:\Users\sofia\PycharmProjects\labs_koreneva\lab_python_fp\print_result
!!!!!!!!
test_1
1
test_2
iu5
test_3
a  =  1
b  =  2
test_4
1
2

Process finished with exit code 0
```

Задача 6 (файл cm_timer.py)

Описание задачи

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).

Текст программы

```python
import time
from contextlib import contextmanager

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

    def __exit__(self, exc_type, exc_val, exc_tb):
        elapsed_time = time.time() - self.start_time
        print(f"time_1: {elapsed_time}")


@contextmanager
def cm_timer_2():
    start_time = time.time()
    yield
    elapsed_time = time.time() - start_time
    print(f"time_2: {elapsed_time}")

if __name__ == '__main__':
    with cm_timer_1():
        time.sleep(5.5)
    with cm_timer_2():
        time.sleep(5.5)
```

Примеры выполнения программы

```
C:\Users\sofia\AppData\Local\Programs\Python\Python39\python.exe C:\Users\sofia\PycharmProjects\labs_koreneva\lab_python_fp\cm_timer.py
time_1: 5.504914283752441
time_2: 5.507152080535889

Process finished with exit code 0
```

Задача 7 (файл process_data.py)

Описание задачи

- 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

```
import json
from gen_random import gen_random
from cm_timer import cm_timer_1
from print_result import print_result

path =
"C:/Users/sofia/PycharmProjects/labs_koreneva/lab_python_fp/data_light.json"

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

@print_result
def f1(arg):
    return sorted(set(item['job-name'].lower() for item in arg))

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

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

@print_result
def f4(arg):
    salaries = gen_random(len(arg), 100000, 2000000)
    return ['{} зарплата {}'.format(job, salary) for job, salary in zip(arg,
salaries)]

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

Примеры выполнения программы

```
C:\Users\sofia\AppData\Local\Programs\Python\Python39\python.exe C:\Users\sofia\PycharmProjects\labs_koreneva\lab_python_fp\process_data.py
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)
электроэрозионист
эндокринолог
энергетик
энергетик литейного производства
энтомолог
юрисконсульт
юрисконсульт 2 категории
юрисконсульт. контрактный управляющий
юрист
юрист (специалист по сопровождению международных договоров, английский - разговорный)
юрист волонтер
юристконсульт
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 160759
программист / senior developer с опытом Python зарплата 310847
программист 1с с опытом Python зарплата 114940
программист c# с опытом Python зарплата 792268
программист c++ с опытом Python зарплата 725033
программист c++/c#/java с опытом Python зарплата 1625314
программист/ junior developer с опытом Python зарплата 863649
программист/ технический специалист с опытом Python зарплата 1635523
программистр-разработчик  информационных систем с опытом Python зарплата 551629
time_1: 0.035518646240234375
```