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
«Функциональные возможности языка Python»

Выполнила:
Студентка группы ИУ5-33Б
Беспалова Виктория
Подпись и дата:

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

2023 год

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
""" Задача №1 """


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


def field(items, *args):
    """

    :param items: список словарей
    :param args: неограниченное количество аргументов

    :returns кортеж или список словарей
    """
    assert len(args) > 0, 'Не передано ни одного аргумента в "args"!'
    if len(args) == 1:
        return (item[item_key] for item in items for item_key in item if
                item[item_key] is not None and item_key == args[0])
    else:
        ans = []
        for item in items:
            tmp = {}
```

```
            for arg in args:
                if not item[arg] is None: tmp[arg] = item[arg]
            if tmp: ans.append(tmp)
        return tuple(ans)


def mainTask1():
    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__':
    mainTask1()
```

Примеры выполнения программы

```
/Users/victoria/PycharmProjects/lab03/venv/bin/python /Users/victoria/PycharmProjects/lab03/lab_python_fp/field.py
Ковер, 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

```
""" Задача 2 """
from random import randint


# Пример:
# gen_random(5, 1, 3) должен выдать выдать 5 случайных чисел
# в диапазоне от 1 до 3, например 2, 2, 3, 2, 1
# Hint: типовая реализация занимает 2 строки

def gen_random(num_count, begin, end):
    """
    Генерация случайных чисел

    :param num_count: количество случайных чисел
    :param begin: минимальное число
    :param end: максимальное число
    :return: кортеж случайных чисео
    """
    return [randint(begin, end) for _ in range(num_count)]


def mainTask2():
    print(*gen_random(5, 1, 3), sep=', ')
```

```
if __name__ == '__main__':
    mainTask2()
```

Примеры выполнения программы

```
/Users/victoria/PycharmProjects/lab03/venv/bin/python /Users/victoria/PycharmProjects/lab03/lab_python_fp/gen_random.py
1, 2, 3, 1, 2

Process finished with exit code 0
```

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

```
Текст программы """ Задача 3 """

# 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 = True, Абв и АБВ - разные строки
#           ignore_case = False, Абв и АБВ - одинаковые строки, одна из
которых удалится
# По-умолчанию ignore_case = False
from lab_python_fp.gen_random import gen_random


# Итератор для удаления дубликатов
class Unique(object):
    def __init__(self, items, **kwargs):
        self._used_data = set()
        self._data = items
        self._index = 0

        if 'ignore_case' not in kwargs:
            self.ignore_case = False
        else:
            self.ignore_case = kwargs['ignore_case']

    def __next__(self):

        # Если bool-параметр ignore_case(игнорирование регистра) истинен,
```

```
        # то все элементы приводятся к нижнему регистру
        if self.ignore_case:
            for counter, el in enumerate(self._data):
                if type(el) is str:
                    self._data[counter] = 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._used_data:
                    self._used_data.add(current)
                    return current

    def __iter__(self):
        return self


def mainTask3():
    print('__________ТЕСТ 1__________')
    data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
    for item in Unique(data):
        print(item)

    print('__________ТЕСТ 2__________')
    data = gen_random(10, 1, 3)
    for item in Unique(data):
        print(item)

    print('__________ТЕСТ 3__________')
    data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    print('Случай 1: без ignore_case')
    for item in Unique(data):
        print(item)

    print('Случай 2: ignore_case=True')
    for item in Unique(data, ignore_case=True):
        print(item)


if __name__ == '__main__':
    mainTask3()
```

Примеры выполнения программы

```
/Users/victoria/PycharmProjects/lab03/venv/bin/python /Users/victoria/PycharmProjects/lab03/lab_python_fp/unique.py
__________ТЕСТ 1__________
1
2
__________ТЕСТ 2__________
3
1
2
__________ТЕСТ 3__________
Случай 1: без ignore_case
a
A
b
B
Случай 2: 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
""" Задача 4"""

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


# Сортировка по модулю в порядке убывания
def mainTask4():
    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__':
    mainTask4()
```

Примеры выполнения программы

```
/Users/victoria/PycharmProjects/lab03/venv/bin/python /Users/victoria/PycharmProjects/lab03/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

```python
""" Задача 5"""


# Реализация декоратора
def print_result(func):
    def wrapper(*args, **kwargs):
        # фактические и формальные парамтеры опциональны в данном случае
        print(func.__name__)
        original_result = func(*args, **kwargs)
```

```
        if isinstance(original_result, list):
            for _ in original_result:
                print(_)
            return original_result
        elif isinstance(original_result, dict):
            for key, value in original_result.items():
                print('{} = {}'.format(key, value))
            return original_result
        else:
            print(original_result)
            return original_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]


def mainTask5():
    print('!!!!!!!!!')
    test_1()
    test_2()
    test_3()
    test_4()


if __name__ == '__main__':
    mainTask5()
```

Примеры выполнения программы

```
/Users/victoria/PycharmProjects/lab03/venv/bin/python /Users/victoria/PycharmProjects/lab03/lab_python_fp/print_result.py
!!!!!!!!
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

```
""" Задача 6"""
import time
from contextlib import contextmanager


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

    def __exit__(self, exc_type, exc_value, exc_tb):
        print(f'time: {time.time() - self.start_time}')


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


def mainTask6():
    with cm_timer_1():
        time.sleep(5.5)

    with cm_timer_2():
        time.sleep(1.5)


if __name__ == '__main__':
    mainTask6()
```

Примеры выполнения программы

```
/Users/victoria/PycharmProjects/lab03/venv/bin/python /Users/victoria/PycharmProjects/lab03/lab_python_fp/cm_timer.py
time: 5.505154848098755
time: 1.5051982402801514

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

# Сделаем другие необходимые импорты
from lab_python_fp.print_result import print_result
from lab_python_fp.cm_timer import cm_timer_1
from lab_python_fp.gen_random import gen_random

# Необходимо в переменную path сохранить путь к файлу, который был передан
при запуске сценария
path = "/Users/victoria/PycharmProjects/lab03/data_light.json"

with open(path) 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):
    """
    Выводит отсортированный список профессий без повторений.
    Регистр игнорируется при сортировке.
    :param arg: json-файл
    :return: отсортированный список
    """
    return sorted(list(set([item['job-name'].capitalize() for item in arg])))


@print_result
def f2(arg):
    """
    Фильтрует входной массив и возвращает только те элементы, которые
начинаются со слова "программист"
```

```
    :param arg: список
    :return: отфильтрованный список
    """
    return list(filter(lambda text: (text.split())[0] == 'Программист', arg))


@print_result
def f3(arg):
    """
    Модифицирует каждый элемент списка посредством добавления к нему строки
"с опытом Python"
    :param arg: список
    :return: модифицированный список
    """
    return list(map(lambda text: text + ' с опытом Python', arg))


@print_result
def f4(arg):
    """
    Генерирует для каждой специальности зарплату от 100 000 до 200 000 рублей
и присоединяет её к названию специальности.
    Пример: "Программист C# с опытом Python, зарплата 137287 руб"
    :param arg: список
    :return:список
    """
    tmp = list(zip(arg, [', зарплата ' + str(salary) + ' руб.' for salary in
gen_random(len(arg), 100000, 200000)]))
    return [item[0] + item[1] for item in tmp]


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

Примеры выполнения программы

```
/Users/victoria/PycharmProjects/lab03/venv/bin/python /Users/victoria/PycharmProjects/lab03/lab_python_fp/process_data.py
f1
1с программист
2-ой механик
3-ий механик
4-ый механик
4-ый электромеханик
Asic специалист
Javascript разработчик
Rtl специалист
Web-программист
Web-разработчик
[химик-эксперт
Автожестянщик
Автоинструктор
Автомаляр
Автомойщик
Автор студенческих работ по различным дисциплинам
Автослесарь
Автослесарь - моторист
Автоэлектрик
Агент
Агент банка
```

```
Электронщик
Электросварщик
Электросварщик  на полуавтомат
Электросварщик на автоматических и полуавтоматических машинах
Электросварщик ручной сварки
Электросварщики ручной сварки
Электрослесарь (слесарь) дежурный и по ремонту оборудования, старший
Электрослесарь по ремонту и обслуживанию автоматики и средств измерений электростанций
Электрослесарь по ремонту оборудования в карьере
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
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
f4
Программист с опытом Python, зарплата 175734 руб.
Программист / senior developer с опытом Python, зарплата 103894 руб.
Программист 1с с опытом Python, зарплата 109549 руб.
Программист c# с опытом Python, зарплата 180761 руб.
Программист c++ с опытом Python, зарплата 161350 руб.
Программист c++/c#/java с опытом Python, зарплата 180358 руб.
time: 0.005974769592285156

Process finished with exit code 0
```