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.»

Выполнил:
студент группы ИУ5-35Б
Хрипков Т.А.
Подпись и дата:

Проверил:
Подпись и дата:

Москва, 2024 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.

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.

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.

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.

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 выводить в столбик через знак равенства.

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.

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:

Sort.py

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

if __name__ == '__main__':
# не лямбда
    result = sorted(data, key=abs, reverse=True)
    print(result)

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

Cm_timer.py

```
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:.1f}")


@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:.1f}")


if __name__ == '__main__':

    with cm_timer_1():
        time.sleep(5.5)
```

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

Field.py

```
def field(items, *args):
    assert len(args) > 0
    if len(args) == 1:
        for item in items:
            value = item.get(args[0])
            for value in items:
                print(value)
            if value is not None:
                yield value
    else:
        for item in items:
            result = {key: item.get(key) for key in args if item.get(key) is
not None}
            print(result)
            if result:
                yield result

goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},
    {'title': 'Стол', 'color': None},
    {'title': None, 'price': None}
]
print("//////////////////Первый способ//////////////////////////")
for title in field(goods, 'title'):
    print(title)
print("//////////////////Второй способ//////////////////////////")
for item in field(goods, 'title', 'price'):
    print(item)
print("//////////////////Третий способ//////////////////////////")
for item in field(goods, 'title', 'color'):
    print(item)
print("//////////////////Четвёртый способ//////////////////////////")
for item in field(goods, 'title', 'price', 'color'):
    print(item)
```

Gen_random.py

```
import random

def gen_random(num_count, begin, end):
    for _ in range(num_count):
        yield random.randint(begin, end)

for number in gen_random(5,1,3):
    print(number)
```

Print_result.py

```
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


@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()
```

Process_data.py

```
import time
import json
import sys
from random import randint
from contextlib import contextmanager



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



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:.1f}")


path = sys.argv[1]

with open(path,encoding = 'utf8') as f:
    data = json.load(f)



@print_result
def f1(arg):
    return sorted(set(entry['job-name'] for entry in arg), key=lambda x:
```

```
x.lower())


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


@print_result
def f3(arg):
    return [f"{jobname} с опытом Python" for jobname in arg]

@print_result
def f4(arg):
    salaries = [randint(100000, 200000) for _ in arg]
    return [f"{profession}, зарплата {salary} руб." for profession, salary in
zip(arg, salaries)]


with cm_timer_1():
    f4(f3(f2(f1(data))))
```

Unique.py

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

    def __iter__(self):
        return self

    def __next__(self):
        while True:
            value = next(self.items)
            if isinstance(value, str) and self.ignore_case:
                key = value.lower()
            else:
                key = value

            if key not in self.seen:
                self.seen.add(key)
                return value

print("////////////////Первый способ//////////////////////////")
data1 = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
unique_data1 = Unique(data1)
for value in unique_data1:
    print(value)
print("////////////////Второй способ//////////////////////////")
```

```
data2 = (x for x in range(1, 4))
unique_data2 = Unique(data2)
for value in unique_data2:
    print(value)
print("/////////////////Третий способ//////////////////////////")
data3 = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
unique_data3 = Unique(data3)
for value in unique_data3:
    print(value)
print("/////////////////Четвёртый способ//////////////////////////")
data4 = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
unique_data4 = Unique(data4, ignore_case=True)
for value in unique_data4:
    print(value)
```

Пример работы(тот же порядок):

```
C:\Users\tima\AppData\Local\Programs\Python\Python311\python.exe: ca
(venv) PS C:\Users\tima\Piton\lab3-4> cd lab_python_fp
(venv) PS C:\Users\tima\Piton\lab3-4\lab_python_fp> python sort.py
[123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
(venv) PS C:\Users\tima\Piton\lab3-4\lab_python_fp>
```

```
(venv) PS C:\Users\tima\Piton\lab3-4\lab_python_fp> python cm_timer.py
time: 5.5
```

```
(venv) PS C:\Users\tima\Piton\lab3-4\lab_python_fp> python field.py
/////////////////Первый способ//////////////////////////
{'title': 'Ковер', 'price': 2000, 'color': 'green'}
{'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
{'title': 'Стол', 'color': None}
{'title': None, 'price': None}
{'title': None, 'price': None}
{'title': 'Ковер', 'price': 2000, 'color': 'green'}
{'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
{'title': 'Стол', 'color': None}
{'title': 'Диван для отдыха', 'color': 'black'}
{'title': 'Диван для отдыха', 'color': 'black'}
{'title': 'Стол'}
{'title': 'Стол'}
{}
/////////////////Четвёртый способ//////////////////////////
{'title': 'Ковер', 'price': 2000, 'color': 'green'}
{'title': 'Ковер', 'price': 2000, 'color': 'green'}
{'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
{'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
{'title': 'Стол'}
{'title': 'Стол'}
{}
(venv) PS C:\Users\tima\Piton\lab3-4\lab_python_fp>
```

```
(venv) PS C:\Users\tima\Piton\lab3-4\lab_python_fp> python gen_random.py
3
3
1
2
2
(venv) PS C:\Users\tima\Piton\lab3-4\lab_python_fp>
```

```
2
(venv) PS C:\Users\tima\Piton\lab3-4\lab_python_fp> python print_result.py
!!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
(venv) PS C:\Users\tima\Piton\lab3-4\lab_python_fp>
```

Только для f4

```
f4
Программист с опытом Python, зарплата 170193 руб.
Программист / Senior Developer с опытом Python, зарплата 143011 руб.
Программист 1С с опытом Python, зарплата 166409 руб.
Программист C# с опытом Python, зарплата 116731 руб.
Программист C++ с опытом Python, зарплата 118748 руб.
Программист C++/C#/Java с опытом Python, зарплата 183580 руб.
Программист/ Junior Developer с опытом Python, зарплата 113510 руб.
Программист/ технический специалист с опытом Python, зарплата 102766 руб.
Программистр-разработчик  информационных систем с опытом Python, зарплата 110294 руб.
time: 0.1
(venv) PS C:\Users\tima\Piton\lab3-4\lab_python_fp>
```

```
(venv) PS C:\Users\tima\Piton\lab3-4\lab_python_fp> python unique.py
//////////////////Первый способ//////////////////////////
1
2
//////////////////Второй способ//////////////////////////
1
2
3
//////////////////Третий способ//////////////////////////
a
A
b
B
//////////////////Четвёртый способ//////////////////////////
a
b
(venv) PS C:\Users\tima\Piton\lab3-4\lab_python_fp>
```