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 №2
“Функциональные возможности языка Python”

Выполнил:
студент группы ИУ5-35Б:
Купцов С.Р.
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.В.
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.

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.

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

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.

Необходимо решить задачу двумя способами:

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

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. Пример:

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).

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.

Код программы

field.py

```
def field(items: list[dict], *args: list): return [dict([[arg, elem[arg]] for
arg in args]) for elem in items]


if __name__ == "__main__":
    goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}]
    print(field(goods, 'title'))
    print(field(goods, 'title', 'price'))
```

gen_random.py

```
from random import randint


def gen_random(count: int, a: int, b: int): return [randint(a, b) for _ in
```

```
range(count)]


if __name__ == '__main__':
    print(gen_random(10, 2, 5))
```

unique.py

```
class Unique(object):
    def __init__(self, items, **kwargs):
        self.i = 0
        ignore_case = False
        if 'ignore_case' in kwargs:
            ignore_case = kwargs['ignore_case']
        self.data = []
        for elem in items:
            if ignore_case:
                if elem not in self.data:
                    self.data.append(elem)
            else:
                if (isinstance(elem, str) and elem.lower() not in
[str(i).lower() for i in self.data]) or (not isinstance(elem, str) and elem
not in self.data):
                    self.data.append(elem)

    def __next__(self):
        self.i += 1

    def __iter__(self):
        return self.data[self.i]


if __name__ == "__main__":
    a = Unique([1, 2, 2, 2, 1, 0, 4, 2, 'bb', 'bd', 'Bd', 'DD'],
ignore_case=True)
    print(a.data)

    from gen_random import gen_random
    b = Unique(gen_random(10, 1, 3))
    print(b.data)
```

sort.py

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

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

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

print_result.py

```
def print_result(func):
    def wrapper(*args, **kwargs):
        original_result = func(*args, **kwargs)
        if isinstance(original_result, list): modified_result =
'\n'.join(map(str, original_result))
        elif isinstance(original_result, dict): modified_result =
'\n'.join([f"{k} = {v}"for k, v in original_result.items()])
        else: modified_result = original_result
        print(f'Function {func.__name__} have returned:\n{modified_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]


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

cm_timer.py

```
import contextlib
from time import time, sleep


class cm_timer_1:
    def __init__(self):
        self.time = time()

    def __enter__(self):
        pass

    def __exit__(self, type, value, traceback):
        print(f"Time: {time() - self.time}")


@contextlib.contextmanager
def cm_timer_2():
    t = time()
    try:
        yield {}
    finally:
        print(f"Time: {time() - t}")


if __name__ == "__main__":
    with cm_timer_2():
        sleep(5.5)

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

process_data.py

```python
from gen_random import gen_random
from cm_timer import cm_timer_1
from print_result import print_result
from unique import Unique
from field import field


import json
import sys

path = "data_light.json"

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


@print_result
def f1(arg):
    return sorted(Unique([item['job-name'] for item in field(arg, "job-name")]).data, key=lambda x: x.lower())


@print_result
def f2(arg):
    return list(filter((lambda x: x.lower().find("программист") == 0), arg))


@print_result
def f3(arg):
    return [item + " с опытом Python" for item in arg]


@print_result
def f4(arg):
    return [item + f", зарплата {gen_random(1, 100000, 200000)[0]} руб." for item in arg]


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

Экранные формы с примерами выполнения программы

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

[2, 2, 2, 3, 3, 5, 4, 5, 4, 2]

[1, 2, 0, 4, 'bb', 'bd', 'Bd', 'DD']
[1, 3, 2]
```

```
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]
!!!!!!!!
Function test_1 have returned:
1
Function test_2 have returned:
iu5
Function test_3 have returned:
a = 1
b = 2
Function test_4 have returned:
1
2
Time: 5.504802703857422
Time: 5.5118725299835205
Function f4 have returned:
Программист с опытом Python, зарплата 124574 руб.
Программист / Senior Developer с опытом Python, зарплата 187334 руб.
Программист 1С с опытом Python, зарплата 138366 руб.
Программист C# с опытом Python, зарплата 114844 руб.
Программист C++ с опытом Python, зарплата 188009 руб.
Программист C++/C#/Java с опытом Python, зарплата 166500 руб.
Программист/ Junior Developer с опытом Python, зарплата 164754 руб.
Программист/ технический специалист с опытом Python, зарплата 177574 руб.
Программистр-разработчик  информационных систем с опытом Python, зарплата 190355 руб.
Time: 1.9828193187713623
```