Московский государственный технический университет им. Н.Э. Баумана

Факультет «Радиотехнический»
Кафедра «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования »

Отчет по лабораторной работе №2
«Функциональные возможности языка Python.»

Выполнил:

студент
группы
РТ5-31Б

Ермаков И.А.

Москва, 2024 г

Описание 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. 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.

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:

field.py

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

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

gen_random.py

```
import random

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

if __name__ == '__main__':
    print(list(gen_random(5, 1, 3)))
```

unique.py

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

    def __iter__(self):
        return self
```

```
    def __next__(self):
        while True:
            item = next(self.items)
            check_item = item.lower() if self.ignore_case and isinstance(item, str) else
item
            if check_item not in self.seen:
                self.seen.add(check_item)
                return item

if __name__ == '__main__':
    data = [1, 1, 1, 2, 2, 3]
    print(list(Unique(data)))

    data = ['a', 'A', 'b', 'B']
    print(list(Unique(data)))
    print(list(Unique(data, ignore_case=True)))
```

sort.py

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

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

    # Без использования lambda-функции
    def abs_key(x):
        return abs(x)

    result = sorted(data, key=abs_key, reverse=True)
    print(result)
```

print_result.py

```
def print_result(func):
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)
        print(func.__name__)
        if isinstance(result, list):
            print(*result, sep="\n")
        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()
```

cm_timer.py

```
import time
from contextlib import contextmanager

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

    def __exit__(self, exc_type, exc_value, traceback):
        print(f"time: {time.time() - self.start_time:.1f}")

@contextmanager
def cm_timer_2():
    start_time = time.time()
    try:
        yield
```

```
    finally:
        print(f"time: {time.time() - start_time:.1f}")

if __name__ == '__main__':
    with cm_timer_1():
        time.sleep(1.5)

    with cm_timer_2():
        time.sleep(2.5)
```

process_data.py

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

# Декоратор для вывода результата
class print_result:
    def __init__(self, func):
        self.func = func

    def __call__(self, *args, **kwargs):
        result = self.func(*args, **kwargs)
        print(result)
        return result

# Контекстный менеджер для измерения времени
@contextmanager
def cm_timer_1():
    start_time = time()
    yield
    print(f"Time elapsed: {time() - start_time:.2f} seconds")

# Определяем путь к файлу
path = 'lab4/data_light.json'

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

# Реализация функций
@print_result
def f1(arg):
```

```
    return sorted(set(job["job-name"].strip().lower() for job in arg),
key=str.casefold)

@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):
    salaries = [random.randint(100000, 200000) for _ in range(len(arg))]
    return [f"{job}, зарплата {salary} руб." for job, salary in zip(arg, salaries)]

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

Пример выполнения программ:

Задача 1 field

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

Задача 2 gen_random

```
[2, 3, 3, 3, 3]
```

Задача 3 unique

```
[1, 2, 3]
['a', 'A', 'b', 'B']
['a', 'b']
```

Задача 4 sort

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

Задача 5 print_result

```
!!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
```

Задача 6 cm_timer

```
time: 1.5
time: 2.5
```

Задача 7 process_data

```
['1с программист', '2-ой механик', '3-ий механик', '4-ый механик', '4-ый электромеханик', '[химик-эксперт', 'asic специалист', 'javasc
ript 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 недвижимос
ти / риэлтор', 'агент по привлечению юридических лиц', 'агент по продажам (интернет, тв, телефония) в пао ростелеком в населенных пунк
тах амурской области: г. благовещенск, г. белогорск, г. свободный, г. шимановск, г. зея, г. тында', 'агент торговый', 'агрегатчик-топл
ивник komatsu', 'агроном', 'агроном по защите растений', 'агроном-полевод', 'агрохимик почвовед', 'администратор', 'администратор (уда
ленно)', 'администратор active directory', 'администратор в парикмахерский салон', 'администратор зала (предприятий общественного пита
ния)', 'администратор кофейни', 'администратор на ресепшен', 'администратор на телефоне', 'администратор по информационной безопасност
и', 'администратор ресторана', 'администратор сайта', 'администратор ярмарок выходного дня', 'администратор-кассир', 'аккомпаниатор на
 0,5 ст.', 'аккумуляторщик 4 разряда', 'акушерка', 'акушерка в родильное отделение', 'акушерка женской консультации', 'акушерка лысого
рская врачебная амбулатория', 'акушерка фап', 'акушерка, ао', 'акушерка, вп', 'альпинист промышленный', 'аналитик', 'анестезиолог - ре
```