Московский государственный технический университет им. Н.Э. Баумана

Факультет ИУ
Кафедра ИУ5

Курс «Основы информатики»

Отчет по лабораторной работе №_3-4_
«Функциональные возможности языка Python»

Выполнил:
студент группы ИУ5-33Б:

Номоконов В.А
Подпись и дата:

Проверил:
преподаватель каф.

____

_____
Подпись и дата:

Москва, 2024 г.

Постановка задачи

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.

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.
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.

Текст программы

\lab3-4\cm_timer.py

```
import time

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

    def __exit__(self, exc_type, exc_val, exc_tb):
        end_time = time.time()
        elapsed_time = end_time - self.start_time
        print(f"time: {elapsed_time:.1f}")

from contextlib import contextmanager

@contextmanager
def cm_timer_2():
    start_time = time.time()
    try:
        yield
    finally:
        end_time = time.time()
        elapsed_time = end_time - start_time
        print(f"time: {elapsed_time:.1f}")


with cm_timer_1():
    time.sleep(5.5)

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

\lab_03-04\data_light.json

```
    {
        "mobile-url":
"https://trudvsem.ru/vacancy/card/1027739174033/6bf457e6-51d8-11e6-853e-
037acc02728d",
        "description": "<p>Умение общаться по телефону и лично,
доброжелательность, ответственность, стрессоустойчивость.</p>",
        "update-date": "2016-10-02 01:33:38 MSK",
        "employment": "Частичная занятость",
        "job-name": "Администратор на телефоне",
        "company": {
            "email": "on.klinik@mail.ru",
            "contact-name": "Светлана",
```

```
                "hr-agency": true,
                "phone": "+7(495)6084488",
                "name": "ООО РОЯЛ КЛИНИК"
            },
            "term": "<p>Присутствуют по результатам работы</p>",
            "addresses": {
                "address": {
                    "location": "г. Москва, Кузнецкий Мост улица, 1",
                    "lat": 55.760808,
```


…

\lab3-4\field.py

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

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

for value in field(goods, 'title'):
    print(value)  # Output: 'Ковер', 'Диван для отдыха'

for result in field(goods, 'title', 'price'):
    print(result)  # Output: {'title': 'Ковер', 'price': 2000}, {'title':
'Диван для отдыха'}
```

\lab3-4\gen_random.py

```
import random

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

for num in gen_random(5, 1, 3):
    print(num)  # Вывод: 5 случайных чисел от 1 до 3, например 2, 2, 3, 2,
1
```

\lab3-4\print_result.py

```
def print_result(func):
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)
        print(f"Function '{func.__name__}' returned:")
        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()
```

\lab3-4\process_data.py

```
import json
import sys
import random
import os
import time

def print_result(func):
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)
        print(f"Function '{func.__name__}' returned:")
```

```
        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_val, exc_tb):
        end_time = time.time()
        elapsed_time = end_time - self.start_time
        print(f"time: {elapsed_time:.1f}")

from contextlib import contextmanager

path = os.path.join(os.path.dirname(__file__), "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 item: item.startswith('программист'), arg))

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

@print_result
def f4(arg):
    salaries = [random.randint(100000, 200000) for _ in arg]
    return list(zip(arg, salaries))

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

\lab3-4\sort.py

```
data = [4, -30, 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), reverse=True)
    print(result_with_lambda)
```

\lab3-4\unique.py

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

    def __next__(self):
        for item in self.items:
            if self.ignore_case and isinstance(item, str):
                item_key = item.lower()
            else:
                item_key = item

            if item_key not in self.seen:
                self.seen.add(item_key)
                return item

        raise StopIteration

    def __iter__(self):
        return self

data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
unique_data = Unique(data)
for item in unique_data:
    print(item)  # prints 1, 2

data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
unique_data = Unique(data)
for item in unique_data:
    print(item)  # prints 'a', 'A', 'b', 'B'

unique_data = Unique(data, ignore_case=True)
for item in unique_data:
    print(item)  # prints 'a', 'b'
```

Анализ результатов

```
PS C:\Users\exxor\PyLabsWW> & c:/Users/exxor/PyLabsWW/.conda/python.exe c:/Users/exxor/PyLabsWW/labs/lab3-4/cm_timer.py
time: 5.5
time: 5.5
PS C:\Users\exxor\PyLabsWW> & c:/Users/exxor/PyLabsWW/.conda/python.exe c:/Users/exxor/PyLabsWW/labs/lab3-4/field.py
Ковер
Диван для отдыха
{'title': 'Ковер', 'price': 2000}
{'title': 'Диван для отдыха'}
PS C:\Users\exxor\PyLabsWW>
```

```
{'title': 'Диван для отдыха'}
PS C:\Users\exxor\PyLabsWW> & c:/Users/exxor/PyLabsWW/.conda/python.exe c:/Users/exxor/PyLabsWW/labs/lab3-4/gen_random.py
1
3
2
2
1
```

```
1
PS C:\Users\exxor\PyLabsWW> & c:/Users/exxor/PyLabsWW/.conda/python.exe c:/Users/exxor/PyLabsWW/labs/lab3-4/print_result.py
!!!!!!!!
Function 'test_1' returned:
1
Function 'test_2' returned:
iu5
Function 'test_3' returned:
a = 1
b = 2
Function 'test_4' returned:
1
2
```

```
Function 'f3' returned:
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
Function 'f4' returned:
```

```
Function 'f4' returned:
('программист с опытом Python', 173576)
('программист / senior developer с опытом Python', 181161)
('программист 1с с опытом Python', 187758)
('программист c# с опытом Python', 125352)
('программист c++ с опытом Python', 189560)
('программист c++/c#/java с опытом Python', 112816)
('программист/ junior developer с опытом Python', 194866)
('программист/ технический специалист с опытом Python', 167192)
('программистр-разработчик  информационных систем с опытом Python', 150880)
time: 0.2
PS C:\Users\exxor\PyLabsWW>
```

```
PS C:\Users\exxor\PyLabsWW> & c:/Users/exxor/PyLabsWW/.conda/python.exe c:/Users/exxor/PyLabsWW/labs/lab3-4/sort.py
[123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
PS C:\Users\exxor\PyLabsWW>
```

```
PS C:\Users\exxor\PyLabsWW> & c:/Users/exxor/PyLabsWW/.conda/python.exe c:/Users/exxor/PyLabsWW/labs/lab3-4/unique.py
1
2
a
A
b
B
a
b
```