Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

Факультет «Радиотехнический»

Кафедра ИУ5 «Системы обработки информации и управления»

Лабораторная работа №3

по дисциплине «Разработка интернет-приложений»

Выполнил:

студент группы РТ5-51Б

Плешаков В. И.

Проверил:

преподаватель

Балашов А. М.

2021 г.

Задача 1

Необходимо реализовать генератор field. Генератор field последовательно выдает значения ключей словаря.

Код программы:

```
def field(items, *args):
    assert len(args) > 0

    if len(args) == 1:
        for item in items:
            if args[0] in item.keys():
                yield item.get(args[0])

    else:
        for item in items:
            res = {key: item.get(key) for key in args if key in item.keys()}
            if len(res) != 0:
                yield res
```

Результат выполнения:

```
PS D:\Stud\5 term\RiP\lab3\lab_python_fp> python field.py
Ковер
Диван для отдыха
Стул
{'title': 'Ковер', 'price': 2000, 'color': 'green'}
{'title': 'Диван для отдыха', 'color': 'black'}
{'title': 'Стул', 'price': 1200}
```

Задача 2

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.

Код программы:

```
from random import randint

def gen_random(num_count, begin, end):
    for i in range(num_count):
        yield randint(begin, end)


if __name__ == "__main__":
    gen = gen_random(5, 1, 3)
    for el in gen:
        print(el)
```

Результат выполнения:

```
PS D:\Stud\5 term\RiP\lab3\lab_python_fp> python .\gen_random.py
1
2
2
2
2
```

Задача 3

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.

Код программы:

```python
from gen_random import gen_random

class Unique(object):
    def __init__(self, items, ignore_case=False):
        self.used_items = set()
        self.items = list(items)
        self.index = 0
        self.ignore_case = ignore_case

    def __next__(self):
        while True:
            if self.index >= len(self.items):
                raise StopIteration
            else:
                current = self.items[self.index]
                self.index += 1
                if self.ignore_case:
                    if current.lower() not in self.used_items:
                        self.used_items.add(current.lower())
                        return current
                elif current not in self.used_items:
                    self.used_items.add(current)
                    return current

    def __iter__(self):
        return self


if __name__ == '__main__':
    data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
    for el in Unique(data):
        print(el)

    data = gen_random(10, 1, 3)
    for el in Unique(data):
        print(el)

    data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    for el in Unique(data):
        print(el)

    for el in Unique(data, ignore_case=True):
        print(el)
```

Результат выполнения:

```
PS D:\Stud\5 term\RiP\lab3\lab_python_fp> python .\unique.py
1
2
1
2
3
a
A
b
B
a
b
```

Задача 4

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

Код программы:

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

sort_with_lambda = lambda lst: sorted(lst, key=abs, reverse=True)

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

    print(sort_with_lambda(data))
```

Результат выполнения:

```
PS D:\Stud\5 term\RiP\lab3\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]
```

Задача 5

Необходимо реализовать декоратор print_result, который выводит на экран результат выполнения функции

```python
def print_result(func_to_decorate):
    def decorated_func(*args):
        print(func_to_decorate.__name__)
        res = func_to_decorate(*args)
        if isinstance(res, list):
            for el in res:
                print(el)
            return res
        elif isinstance(res, dict):
            for el in res:
                print('{} = {}'.format(el, res.get(el)))
            return res
        else:
            print(res)
    return decorated_func

@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()
```

Результат выполнения:

```
PS D:\Stud\5 term\RiP\lab3\lab_python_fp> python .\print_result.py
!!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
```

Задача 6

Необходимо написать контекстные менеджеры cm_timer_1 и cm_timer_2, которые считают время работы блока кода и выводят его на экран

Код программы:

```python
from contextlib import contextmanager
from time import time, sleep

class cm_timer_1:
    def __init__(self):
        pass

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

    def __exit__(self, exp_type, exp_value, traceback):
        if exp_type is not None:
            print(exp_type, exp_value, traceback)
        else:
            print('time: {}'.format(time() - self.start_time))


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


if __name__ == '__main__':
    with cm_timer_1():
        sleep(5.5)

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

Результат выполнения:

```
PS D:\Stud\5 term\RiP\lab3\lab_python_fp> python .\cm_timer.py
time: 5.505347490310669
time: 5.500852823257446
```

Задача 7

- 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:

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

path = 'data_light.json'

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


@print_result
def f1(arg):
    return list(Unique(sorted(field(arg, 'job-name')), ignore_case=True))


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


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


@print_result
def f4(arg):
    salary_list = [x for x in gen_random(len(arg), 100_000, 200_000)]
    return ['{}, зарплата {} руб.'.format(x, y) for x, y in zip(arg, salary_list)]


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

Результат выполнения:

```
f2
Программист
Программист / Senior Developer
Программист 1С
Программист C#
Программист C++
Программист C++/C#/Java
Программист/ Junior Developer
Программист/ технический специалист
Программистр-разработчик  информационных систем
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
Программист/ Junior Developer с опытом Python
Программист/ технический специалист с опытом Python
Программистр-разработчик  информационных систем с опытом Python
f4
Программист с опытом Python, зарплата 164528 руб.
Программист / Senior Developer с опытом Python, зарплата 106361 руб.
Программист 1С с опытом Python, зарплата 199068 руб.
Программист C# с опытом Python, зарплата 152831 руб.
Программист C++ с опытом Python, зарплата 152012 руб.
Программист C++/C#/Java с опытом Python, зарплата 136901 руб.
Программист/ Junior Developer с опытом Python, зарплата 181547 руб.
Программист/ технический специалист с опытом Python, зарплата 176088 руб.
Программистр-разработчик  информационных систем с опытом Python, зарплата 120723 руб.
time: 0.9762928485870361
```