Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №3
«Функциональные возможности языка Python»

Выполнил:
студент группы ИУ5-32Б
Давиташвили Шако

Москва, 2021 г.

Общее описание задания

Задание лабораторной работы состоит из решения нескольких задач.

Файлы, содержащие решения отдельных задач, должны располагаться в пакете lab_python_fp. Решение каждой задачи должно располагаться в отдельном файле.

При запуске каждого файла выдаются тестовые результаты выполнения соответствующего задания.

Задача 1 (файл field.py)

Описание задания:

Необходимо реализовать генератор `field`, который последовательно выдает значения ключей словаря.

- В качестве первого аргумента генератор принимает список словарей, дальше через *args генератор принимает неограниченное количество 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.

Текст программы:

```
def field(items, *args):

assert len(args) > 0
count=0
if (len(args) == 1 ):
for i in items:
mean = i.get(args[0])
if( mean != None):
yield mean

else:
for i in items:
ans = {}
for j in args:
mean = i.get(j)
if mean != None:
ans[j] = mean
count+=1
if count > 0:
yield ans

if __name__ == "__main__":
```

```
goods = [
{'title': 'Ковер', 'price': 2000, 'color': 'green'},
{'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
]

ans1 = []
ans2 = []

for i in field(goods,"title"):
ans1.append(i)
print(ans1)

for i in field(goods,"title","price"):
ans2.append(i)
print(ans2)
```

Примеры выполнения программы:

```
hakodavitahvili@MacBook-Pro-Shako lab_python_fp % python3 field.py
['Ковер', 'Диван для отдыха']
[{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}
```

Задача 2 (файл gen_random.py)

Описание задания:

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.

Текст программы:

```
import random

def gen_random(value, min, max):
    for i in range(value):
        yield random.randint(min,max)

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

Примеры выполнения программы:

```
hakodavitahvili@MacBook-Pro-Shako lab_python_fp % python3 gen_random.py
2
3
3
2
1
```

Задача 3 (файл unique.py)

Описание задания:

- 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.

Текст программы:

```
from gen_random import gen_random

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

    def __next__(self):
        it =  iter(self.items)
        while True:
            try:
                current = next(it)
            except StopIteration:
                raise
            else:
                if self.ic == True and isinstance(current, str):
                    temp = current[:]
                    if temp.lower() not in self.seen:
                        self.seen.add(temp.lower())
                        return current
                elif current not in self.seen:
                    self.seen.add(current)
                    return current

    def __iter__(self):
        return self

if __name__ == "__main__":

    data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
    print(list(Unique(data)))

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

    data = gen_random(1, 3, 10)
    print(list(Unique(data)))
```

Примеры выполнения программы:

```
hakodavitahvili@MacBook-Pro-Shako lab_python_fp % python3 unique.py
[1, 2]
['a', 'A', 'b', 'B']
[9]
```

Задача 4 (файл sort.py)

Описание задания:

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.

Текст программы:

```
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),reverse = True)
    print(result_with_lambda)
```

Примеры выполнения программы:

```
hakodavitahvili@MacBook-Pro-Shako lab_python_fp % python3 sort.py
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]
```

Задача 5 (файл print_result.py)

Описание задания:

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.

Текст программы:

```
def print_result(func):
    def wrapper(*args,**kwargs):
        print(func.__name__)
        return_value = func(*args, **kwargs)

        if isinstance(return_value,list):
            for i in return_value:
                print(i)

        elif isinstance(return_value, dict):
            for i in return_value:
                print(i, "=", return_value[i])
        else:
            print(return_value)

        return return_value
    return wrapper

@print_result
def test_1():
    return 1


@print_result
def test_2():
    return 'iu5'
```

```python
@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()
```

Примеры выполнения программы:

```
hakodavitahvili@MacBook-Pro-Shako lab_python_fp % python3 print_result.py
!!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
```

Задача 6 (файл cm_timer.py)

Описание задания:

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`.

`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`).

Текст программы:

```
from contextlib import contextmanager
import time

class cm_timer_1():

    def __init__(self):
        self.start_time = 0

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

    def __exit__(self, type, value, traceback):
        print(time.time()-self.start_time)

@contextmanager
def cm_timer_2():
    start_time = time.time()
    yield
    print(time.time()-start_time)

if __name__ == '__main__':
    with cm_timer_1():
        time.sleep(1)

    with cm_timer_2():
        time.sleep(1)
```

Примеры выполнения программы:

```
hakodavitahvili@MacBook-Pro-Shako lab_python_fp % python3 cm_timer.py
1.0044171810150146
1.0047879219055176
```

Задача 7 (файл process_data.py)

Описание задания:

- 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.

- Функция `f1` должна вывести отсортированный список профессий без повторений (строки в разном регистре считать равными). Сортировка должна игнорировать регистр.

- Функция `f2` должна фильтровать входной массив и возвращать только те элементы, которые начинаются со слова “программист”.

- Функция `f3` должна модифицировать каждый элемент массива, добавив строку “с опытом Python” (все программисты должны быть знакомы с Python).

- Функция `f4` должна сгенерировать для каждой специальности зарплату от 100 000 до 200 000 рублей и присоединить её к названию специальности.

Текст программы:

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


path = "../data_light.json"

# Необходимо в переменную path сохранить путь к файлу, который был передан при запуске сценария

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

# Далее необходимо реализовать все функции по заданию, заменив `raise NotImplemented`
# Предполагается, что функции f1, f2, f3 будут реализованы в одну строку
# В реализации функции f4 может быть до 3 строк

@print_result
def f1(arg):
    return list(Unique(field(arg, "job-name"), True))


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


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


@print_result
def f4(arg):
    return dict(zip(arg, ['зарплата {} руб.'.format(x) for x in gen_random(len(arg), 1000000, 2000000)]))

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

Пример выполнения программы:

```
hakodavitahvili@MacBook-Pro-Shako lab_python_fp % python3 process_data.py
f1
Администратор на телефоне
Медицинская сестра
Охранник сутки-день-ночь-вахта
ВРАЧ АНЕСТЕЗИОЛОГ РЕАНИМАТОЛОГ
теплотехник
разнорабочий
Электро-газосварщик
Водитель Gett/Гетт и Yandex/Яндекс такси на личном автомобиле
Монолитные работы
Организатор — тренер
Помощник руководителя
Автоэлектрик
Врач ультразвуковой диагностики в детскую поликлинику
Менеджер по продажам ИТ услуг (B2B)
Менеджер по персоналу
Аналитик
Воспитатель группы продленного дня
Инженер по качеству
Инженер по качеству 2 категории (класса)
Водитель автомобиля
Пекарь
Переводчик
Терапевт
врач-анестезиолог-реаниматолог
Инженер-конструктор в наружной рекламе
Монтажник-сборщик рекламных конструкций
Оператор фрезерно-гравировального станка
Зоотехник
Сварщик
Рабочий-строитель
врач-трансфузиолог
Юрисконсульт
Специалист отдела автоматизации
Растворщик реагентов
Бармен
Официант
Технолог
Фельдшер-лаборант
Медицинская сестра по физиотерапии
врач функциональной диагностики
Рентгенолаборант
диспетчер по навигации
водитель погрузчика,штабелер
Машинист автогрейдера
наладчик ЧПУ
```

```
f2
Программист
Программист C++/C#/Java
Программист 1C
Программистр-разработчик  информационных систем
Программист C++
Программист/ Junior Developer
Программист / Senior Developer
Программист/ технический специалист
Программист C#
f3
Программист с опытом Python
Программист C++/C#/Java с опытом Python
Программист 1C с опытом Python
Программистр-разработчик  информационных систем с опытом Python
Программист C++ с опытом Python
Программист/ Junior Developer с опытом Python
Программист / Senior Developer с опытом Python
Программист/ технический специалист с опытом Python
Программист C# с опытом Python
f4
Программист с опытом Python = зарплата 1903025 руб.
Программист C++/C#/Java с опытом Python = зарплата 1158570 руб.
Программист 1C с опытом Python = зарплата 1325581 руб.
Программистр-разработчик  информационных систем с опытом Python = зарплата 1772533 руб.
Программист C++ с опытом Python = зарплата 1393564 руб.
Программист/ Junior Developer с опытом Python = зарплата 1787841 руб.
Программист / Senior Developer с опытом Python = зарплата 1365118 руб.
Программист/ технический специалист с опытом Python = зарплата 1320520 руб.
Программист C# с опытом Python = зарплата 1773499 руб.
0.015811681747436523
```