Московский государственный технический

университет им. Н. Э. Баумана

Отчёт по лабораторной работе №3 по курсу «Разработка интернет приложений».

«Функциональные возможности языка Python».

Выполнил:
Анцифров Н. С.
студент группы ИУ5-51Б

Проверил:
Гапанюк Ю. Е.

Подпись и дата:

Подпись и дата:

Москва, 2021 г.

Содержание

1. Задание лабораторной работы.

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.

2. Задача 1.

Задание:

Необходимо реализовать генератор field. Генератор field последовательно выдаёт 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.

Текст программы:

field.py

```
def field(items, *args):
    assert len(args) > 0
    if len(args) == 1:
```

```
            res = []
            for item in items:
                if args[0] in item.keys():
                    res.append(item[args[0]])
            print(res)
        else:
            for item in items:
                res = dict()
                for key in args:
                    if key in item.keys():
                        res[key] = item[key]
                print(res)

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

if __name__ == "__main__":
    field(goods, 'title')
    field(goods, 'title', 'price')
```

Результат работы программы:

```
PS C:\Users\antsi\OneDrive\Документы\МГТУ\5 семестр\РИП>  c:; cd 'c:\Users\antsi\OneDrive\Документы\МГТУ\5 семестр\РИП'; & 'C:\Users\antsi\AppData\Local\Programs\P
ython\Python310\python.exe' 'c:\Users\antsi\.vscode\extensions\ms-python.python-2021.10.1365161279\pythonFiles\lib\python\debugpy\launcher' '63187' '--' 'c:\Users\
antsi\OneDrive\Документы\МГТУ\5 семестр\РИП\Ла63\lab_python_fp\field.py'
['Ковер', 'Диван для отдыха']
{'title': 'Ковер', 'price': 2000}
{'title': 'Диван для отдыха', 'price': 5300}
```

3. Задача 2.

Задание:

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:

gen_random.py

```
from random import randrange
def gen_random(count, min, max):
    return [randrange(min, max+1) for i in range(count)]
if __name__ == "__main__":
    print (gen_random(5, 1, 3))
```

Результат выполнения программы:

```
PS C:\Users\antsi\OneDrive\Документы\МГТУ\5 семестр\РИП>  c:; cd 'c:\Users\antsi\OneDrive\Документы\МГТУ\5 семестр\РИП'; & 'C:\Users\antsi\AppData\Local\Programs\
ython\Python310\python.exe' 'c:\Users\antsi\.vscode\extensions\ms-python.python-2021.10.1365161279\pythonFiles\lib\python\debugpy\launcher' '49583' '--' 'c:\Users
antsi\OneDrive\Документы\МГТУ\5 семестр\РИП\Ла63\lab_python_fp\gen_random.py'
[1, 1, 1, 1, 3]
```

4. Задача 3.

Задание:

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:

unique.py

```
from gen_random import gen_random

class Unique(object):
    def __init__(self, items, **kwargs):
        self.elems = set()
        self.data = items
        self.index = 0

        if 'ignore_case' not in kwargs.keys():
            self.ignore_case = False
        else:
            self.ignore_case = kwargs['ignore_case']
    def __iter__(self):
        return self
    def __next__(self):
        while True:
            if self.index < len(self.data):
                current = self.data[self.index]
                self.index = self.index + 1
                if self.ignore_case:
                    if current.lower() not in self.elems:
                        self.elems.add(current.lower())
                        return current
                else:
                    if current not in self.elems:
                        self.elems.add(current)
                        return current
            else:
                raise StopIteration
```

```
if __name__ == "__main__":
    print ('Тест 1 - [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]')
    data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
    for el in Unique(data):
        print(el)
    print ('Тест 2 - gen_random(1, 3, 10)')
    data = gen_random(1, 3, 10)
    for el in Unique(data):
        print(el)
    print ('Тест 3 - [\'a\', \'A\', \'b\', \'B\', \'a\', \'A\', \'b\', \'B\']
и Unique(data)')
    data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    for el in Unique(data):
        print(el)
    print ('Тест 4 - [\'a\', \'A\', \'b\', \'B\', \'a\', \'A\', \'b\', \'B\']
и Unique(data, ignore_case=True)')
    for el in Unique(data, ignore_case=True):
        print(el)
```

Результат выполнения программы:

```
PS C:\Users\antsi\OneDrive\Документы\МГТУ\5 семестр\РИП>  c:; cd 'c:\Users\antsi\OneDrive\Документы\МГТУ\5 семестр\РИП'; & 'C:\Users\antsi\AppData\Local\Programs\P
ython\Python310\python.exe' 'c:\Users\antsi\.vscode\extensions\ms-python.python-2021.10.1365161279\pythonFiles\lib\python\debugpy\launcher' '64646' '--' 'c:\Users\
antsi\OneDrive\Документы\МГТУ\5 семестр\РИП\Лаб3\lab_python_fp\unique.py'
Тест 1 - [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
1
2
Тест 2 - gen_random(1, 3, 10)
5
Тест 3 - ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B'] и Unique(data)
a
A
b
B
Тест 4 - ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B'] и Unique(data, ignore_case=True)
a
b
```

5. Задача 4.

Задание:

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:

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

Текст программы:

sort.py

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

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

```
    print(result)

    print ("\nС lambda")
    result_with_lambda = (lambda d : sorted(d, key=abs, reverse=True))(data)
    print(result_with_lambda)
```

Результат выполнения программы:

```
PS C:\Users\antsi\OneDrive\Документы\МГТУ\5 семестр\РИП>  c:; cd 'c:\Users\antsi\OneDrive\Документы\МГТУ\5 семестр\РИП'; & 'C:\Users\antsi\AppData\Local\Programs\P
ython\Python310\python.exe' 'c:\Users\antsi\.vscode\extensions\ms-python.python-2021.10.1365161279\pythonFiles\lib\python\debugpy\launcher' '49358' '--' 'c:\Users\
antsi\OneDrive\Документы\МГТУ\5 семестр\РИП\Лаб3\lab_python_fp\sort.py'
Без lambda
[123, 100, -100, -30, 4, -4, 1, -1, 0]

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

6. Задача 5.

Задание:

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:

print_result.py

```
def print_result(func):
    def myfunc(*args):
        pars = func(*args)
        print(func.__name__)
        if isinstance(pars, dict):
            for key, p in pars.items():
                print('{} = {}'.format(key, p))
        elif isinstance(pars, list):
            for p in pars:
                print(p)
        else:
            print(pars)
        return pars
    return myfunc


@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 программы:

```
PS C:\Users\antsi\OneDrive\Документы\МГТУ\5 семестр\РИП>  c:; cd 'c:\Users\antsi\OneDrive\Документы\МГТУ\5 семестр\РИП'; & 'C:\Users\antsi\AppData\Local\Programs\P
ython\Python310\python.exe' 'c:\Users\antsi\.vscode\extensions\ms-python.python-2021.10.1365161279\pythonFiles\lib\python\debugpy\launcher' '55491' '--' 'c:\Users\
antsi\OneDrive\Документы\МГТУ\5 семестр\РИП\Лаб3\lab_python_fp\print_result.py'
!!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
```

7. Задача 6.

Задание:

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).

Текст программы:

cm_timer.py

```
import time
from contextlib import contextmanager
```

```python
class cm_timer_1:

    def __init__(self):
        self.timer = time.time()

    def __enter__(self):
        pass

    def __exit__(self, exp_type, exp_value, traceback):
        self.timer = time.time() - self.timer
        print('time: {}'.format(self.timer))


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


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

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

Результат выполнения программы:

```
PS C:\Users\antsi\OneDrive\Документы\МГТУ\5 семестр\РИП>  c:; cd 'c:\Users\antsi\OneDrive\Документы\МГТУ\5 семестр\РИП'; & 'C:\Users\antsi\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\antsi\.vscode\extensions\ms-python.python-2021.10.1365161279\pythonFiles\lib\python\debugpy\launcher' '62464' '--' 'c:\Users\antsi\OneDrive\Документы\МГТУ\5 семестр\РИП\Лаб3\lab_python_fp\cm_timer.py'
time: 5.507565259933472
time: 5.511245250701904
```

8. Задача 7.

Задание:

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 счёт 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:

process_data.py

```
import json
import sys
import cm_timer
from print_result import print_result
from gen_random import gen_random

with open('C:\\Users\\antsi\\OneDrive\\Документы\\МГТУ\\5
семестр\\РИП\\Лаб3\\lab_python_fp\\data_light.json', encoding='utf8') as a:
    data = json.load(a)

@print_result
def f1(d):
     return sorted(set([val.lower() for val in d]), key=str.lower)


@print_result
def f2(d):
    return list(filter(lambda a: str.startswith(a, 'программист'), d))


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


@print_result
def f4(d):
    temp = list(zip(d, [(', зарплата '+ str(el) + ' руб.') for el in
list(gen_random(len(d), 100000, 200000))]))
    return [(el[0]+el[1]) for el in temp]
```

```
if __name__ == '__main__':
    with cm_timer.cm_timer_1():
        f4(f3(f2(f1([el['job-name'] for el in data]))))
```

Результат выполнения программы:

```
эндокринолог
энергетик
энергетик литейного производства
энтомолог
юрисконсульт
юрисконсульт 2 категории
юрисконсульт. контрактный управляющий
юрист
юрист (специалист по сопровождению международных договоров, английский - разговорный)
юрист волонтер
юристконсульт
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 156384 руб.
программист / senior developer с опытом Python, зарплата 155131 руб.
программист 1с с опытом Python, зарплата 145171 руб.
программист c# с опытом Python, зарплата 145557 руб.
программист c++ с опытом Python, зарплата 122778 руб.
программист c++/c#/java с опытом Python, зарплата 182824 руб.
программист/ junior developer с опытом Python, зарплата 131172 руб.
программист/ технический специалист с опытом Python, зарплата 151726 руб.
программистр-разработчик  информационных систем с опытом Python, зарплата 112897 руб.
time: 1.0631208419799805
```