Московский государственный технический университет им. Н.Э. Баумана

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

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе «Функциональные возможности языка Python»

Выполнила:

студентка группы ИУ5-34Б:

Теряева Ксения Владимировна
14.12.2022г

Проверил:

преподаватель каф. ИУ5

Гапанюк Ю.Е.

Москва, 2022 г

Задание:

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.

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

Текст программы:

```
#1
# Пример:
goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
 ]

def field(items, *args):
    assert len(args) > 0
    if len(args) == 1:
        for i in items:
            if args[0] in i.keys():
                #print i[args[0]]
                yield i[args[0]]
    else:
        for i in items:
            dict1 = {}
            for j in range(len(args)):
                if args[j] in i.keys():
                    dict1[args[j]]=i[args[j]]
            #print dict1
            yield dict1
```

```
print(list(field(goods, 'title'))) #должен выдавать 'Ковер', 'Диван для отдыха'
list(field(goods, 'title', 'price')) #должен выдавать {'title': 'Ковер', 'price':2000}, {'title': 'Диван для отдыха', 'price': 5300}
```

Результат работы программы:

```
['Ковер', 'Диван для отдыха']
[{'title': 'Ковер', 'price': 2000},
 {'title': 'Диван для отдыха', 'price': 5300}]
```

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. Пример: gen_random(5, 1, 3) должен выдать 5 случайных чисел в диапазоне от 1 до 3, например 2, 2, 3, 2, 1.

Текст программы:

```python
#2
import random as rd
def gen_random(num_count, begin, end):
    i=0
    while (i<num_count):
        a = rd.randint(begin,end)
        yield a
        i+=1
print(list(gen_random(5, 1, 3)))
```

Результат работы программы:

```
[3, 2, 1, 1, 3]
```

Задача 3.

1. Необходимо реализовать итератор Unique(данные), который принимает на вход массив или генератор и итерируется по элементам, пропуская дубликаты.
2. 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.
3. При реализации необходимо использовать конструкцию **kwargs.
4. Итератор должен поддерживать работу как со списками, так и с генераторами.
5. Итератор не должен модифицировать возвращаемые значения.

Текст программы:

```python
#3
#data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
#data = gen_random(10, 1, 3)
mas1 = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
# Итератор для удаления дубликатов
class Unique(object):
    def __init__(self, items, **kwargs):
        mas_new = []
        mas2 = []
        if kwargs.get("ignore_case") == False:
            self.arr = items
            self.arr = list(set(items))
        else:
            self.arr = set(items)  # ['a', 'A', 'b', 'B']
            for i in self.arr:
                if type(i) == str:
                    if not i.casefold() in mas_new:
                        mas_new.append(i.casefold())
                        mas2.append(i)
            self.arr = mas2
```

```python
    def __next__(self):
      try:
        return self.arr.pop(0)
      except:
        raise StopIteration

    def __iter__(self):
        return self

print(list(Unique(mas1,ignore_case = True)))
```

Результат работы программы:

```
['b', 'a']
```

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.

Текст программы:

```
#4
def sort_with_lambda(arr):
    return sorted(arr, key = lambda x: -abs(x))
mas1 = [4, -30, 100, -100, 123, 1, 0, -1, -4]
if __name__ == '__main__':
    mas3 = sorted(mas1,key = lambda x: -abs(x))
    print(mas3)

    mas2 = sorted(mas1, key = abs, reverse = True)
    print(mas2)
```

Результат работы программы:

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

Задача 5.

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

1. Декоратор должен принимать на вход функцию, вызывать её, печатать в консоль имя функции и результат выполнения, после чего возвращать результат выполнения.
2. Если функция вернула список (list), то значения элементов списка должны выводиться в столбик.
3. Если функция вернула словарь (dict), то ключи и значения должны выводить в столбик через знак равенства.

Текст программы:

```
#5
def print_result(func):
    def func_2(*args,**kargs):
      a = func(*args,**kargs)
      print(func.__name__)
      if type(a) == list:
        for i in a:
          print(i)
      elif type(a) == dict:
        for k,v in a.items():
          print(k, " = ", v)
      else:
          print(a)
      return a
    return func_2
```

```
@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 работы программы:

```
!!!!!!!!
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.

Текст программы:

```
#6
from time import time, sleep
import contextlib
class cm_timer_1():
  def __enter__(self):
    self.a = time()
  def __exit__(self, type, value, traceback):
    print(time() - self.a)


with cm_timer_1():
    sleep(2)
```

```
2.00211763381958
```

```python
@contextlib.contextmanager
def cm_timer_2():
  a = time()
  try:
    yield a
  finally:
    print(time() - a)

with cm_timer_2():
    sleep(2)
```

```
2.0021729469299316
```

Результат работы программы:

```
2.00211763381958
```

```
2.0021729469299316
```

Задача 7.

В предыдущих задачах были написаны все требуемые инструменты для работы с данными. Применим их на реальном примере.

1. В файле data_light.json содержится фрагмент списка вакансий.
2. Структура данных представляет собой список словарей с множеством полей: название работы, место, уровень зарплаты и т.д.
3. 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.
4. Предполагается, что функции f1, f2, f3 будут реализованы в одну строку. В реализации функции f4 может быть до 3 строк.
5. Функция f1 должна вывести отсортированный список профессий без повторений (строки в разном регистре считать равными). Сортировка должна игнорировать регистр. Используйте наработки из предыдущих задач.
6. Функция f2 должна фильтровать входной массив и возвращать только те элементы, которые начинаются со слова “программист”. Для фильтрации используйте функцию filter.
7. 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.
8. 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:

```
!wget https://raw.githubusercontent.com/ugapanyuk/BKIT_2021/main/notebooks/fp/files/data_light.json
```

```
--2022-11-07 14:18:41--  https://raw.githubusercontent.com/ugapanyuk/BKIT_2021/main/notebooks/fp/files/data_light.json
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.110.133, 185.199.111.133, 185.199.108.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 14232722 (14M) [text/plain]
Saving to: ‘data_light.json’

data_light.json     100%[===================>]  13.57M  --.-KB/s    in 0.1s

2022-11-07 14:18:42 (126 MB/s) - ‘data_light.json’ saved [14232722/14232722]
```

```
#7
import json
import sys
# Сделаем другие необходимые импорты

path = 'data_light.json'

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

with open(path) 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(data)

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

```
@print_result
def f2(arg):
    return list(filter(lambda a: a.startswith("Программист") or a.startswith("программист"), arg))


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


@print_result
def f4(arg):
    salary = list(zip(list(gen_random(len(arg), 100000, 200000)),arg))
    return [f"{name}, зарплата {saly} руб." for saly, name in salary]


if __name__ == '__main__':
    print(data[0])
    with cm_timer_1():
        f4(f3(f2(f1(data))))
```

Результат работы программы:

```
Инженер по стандартизации Технического отдела (Сергиево-Посадский филиал)
Инженер по тепловым расчетам
Инженер по холодоснабжению
Инженер по эксплуатации зданий
Инженер разработчик РЭА
Инженер расчетчик
Инженер сервисного центра
Инженер средств радио и телевидения
Инженер технического надзора
Инженер технолог
Инженер технолог ОГТ (отдел главного технолога)
Инженер технолог печати
Инженер технолог по сборке
Инженер электроник
Инженер – нормировщик
Инженер – технолог
Инженер, ведущий
Инженер-аналитик
```

И еще много …

```
 f2
 Программист / Senior Developer
 Программист 1С
 Программист C#
 Программист C++
 Программист C++/C#/Java
 Программист/ Junior Developer
 Программист/ технический специалист
 Программистр-разработчик  информационных систем
 программист
f3
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
программист с опытом Python
f4
Программист / Senior Developer с опытом Python, зарплата 179754 руб.
Программист 1С с опытом Python, зарплата 127199 руб.
Программист C# с опытом Python, зарплата 173498 руб.
Программист C++ с опытом Python, зарплата 113660 руб.
Программист C++/C#/Java с опытом Python, зарплата 144874 руб.
Программист/ Junior Developer с опытом Python, зарплата 178152 руб.
Программист/ технический специалист с опытом Python, зарплата 118949 руб.
Программистр-разработчик  информационных систем с опытом Python, зарплата 113296 руб.
программист с опытом Python, зарплата 136944 руб.
```