

**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 ИУ5-31Б
Фонин Максим Алексеевич

Москва, 2021 г.

Описание задания

1. 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 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.
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.
3. Необходимо реализовать итератор `Unique(данные)`, который принимает на вход массив или генератор и итерируется по элементам, пропуская дубликаты.
 - a. 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`.
 - b. При реализации необходимо использовать конструкцию `**kwargs`.
 - c. Итератор должен поддерживать работу как со списками, так и с генераторами.
 - d. Итератор не должен модифицировать возвращаемые значения.
4. Дан массив 1, содержащий положительные и отрицательные числа. Необходимо одной строкой кода вывести на экран массив 2, который 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 и без использования `lambda`-функции.
5. Необходимо реализовать декоратор `print_result`, который выводит на экран результат выполнения функции.
 - a. Декоратор должен принимать на вход функцию, вызывать её, печатать в консоль имя функции и результат выполнения, после чего возвращать результат выполнения.

- b. Если функция вернула список (list), то значения элементов списка должны выводиться в столбик.
 - c. Если функция вернула словарь (dict), то ключи и значения должны выводиться в столбик через знак равенства.
6. Необходимо написать контекстные менеджеры `cm_timer_1` и `cm_timer_2`, которые считают время работы блока кода и выводят его на экран.
7. В предыдущих задачах были написаны все требуемые инструменты для работы с данными. Применим их на реальном примере.
- a. В файле `data_light.json` содержится фрагмент списка вакансий.
 - b. Структура данных представляет собой список словарей с множеством полей: название работы, место, уровень зарплаты и т.д.
 - c. 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.
 - d. Предполагается, что функции `f1`, `f2`, `f3` будут реализованы в одну строку. В реализации функции `f4` может быть до 3 строк.
 - e. Функция `f1` должна вывести отсортированный список профессий без повторений (строки в разном регистре считать равными). Сортировка должна игнорировать регистр. Используйте наработки из предыдущих задач.
 - f. Функция `f2` должна фильтровать входной массив и возвращать только те элементы, которые начинаются со слова “программист”. Для фильтрации используйте функцию `filter`.
 - g. 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`.
 - h. 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

```
# field.py
goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
]
# field(goods, 'title') должен выдавать 'Ковер', 'Диван для отдыха'
# field(goods, 'title', 'price') должен выдавать {'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}

def field(dicts, *keys):
    assert len(keys) > 0
    if len(keys) == 1:
        for d in dicts:
            val = d.get(keys[0])
            if val != None:
                yield val
    else:
        for d in dicts:
            res_dict = dict()
            for key in keys:
                val = d.get(key)
                if val != None:
                    res_dict[key] = val
            if len(res_dict) != 0:
                yield res_dict

# gen_random.py
from random import randint

def gen_random(amount, minimal, maximum):
    for _ in range(amount):
        yield randint(minimal, maximum)

# unique.py
# Итератор для удаления дубликатов
class Unique(object):
    '''Итератор для удаления дубликатов'''

    def __init__(self, items, **kwargs):
        self.items = items
        self.ignore_case = kwargs.get('ignore_case')
        self.used_elements = set()

    def __iter__(self):
        return self

    def __next__(self):
        it = iter(self.items)
        while True:
            try:
                curr = next(it)
            except StopIteration:
                raise
```

```

        else:
            if self.ignore_case and isinstance(curr, str):
                if curr.lower() not in self.used_elements:
                    self.used_elements.add(curr.lower())
                    return curr
            elif curr not in self.used_elements:
                self.used_elements.add(curr)
                return curr

# 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 c: abs(c), reverse=True)
    print(result_with_lambda)

# print_result.py
def print_result(func):
    def decorated_func(*args, **kwargs):
        res = func(*args, **kwargs)
        print(func.__name__)
        if isinstance(res, list):
            for i in res:
                print(i)
        elif isinstance(res, dict):
            for key, value in res.items():
                print(f"{key} = {value}")
        else:
            print(res)
        return 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()

# cm_timer.py
from contextlib import contextmanager
import time

class cm_timer_1():
    def __init__(self):
        self.start_time = None

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

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

```

```

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

```

```

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

    with cm_timer_2():
        time.sleep(1)

```

```

# process_data.py
import json
import sys
from lab_python_fp.field import field
from lab_python_fp.gen_random import gen_random
from lab_python_fp.unique import Unique
from lab_python_fp.print_result import print_result
from lab_python_fp.cm_timer import cm_timer_1

```

```

try:
    path = "../data_light.json" #sys.argv[1]
except IndexError:
    raise ValueError("Не указан путь к файлу")
else:
    with open(path, encoding='utf-8') as f:
        data = json.load(f)

```

```

@print_result

```

```

def f1(lst):
    return sorted(list(Unique(field(lst, 'job-name'), ignore_case=True)),
key=str.lower)

@print_result
def f2(lst):
    return list(filter(lambda s: str.startswith(str.lower(s), 'программист'),
lst))

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

@print_result
def f4(lst):
    return dict(zip(lst, [f"зарплата {num} руб." for num in
gen_random(len(lst), 1000000, 2000000)]))

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

```

Результат выполнения программы

```
f0max@f0max:~/Code/laba_3/code$ python3 ./process_data.py
*** Package Initialization ***

f1
1С программист
2-ой механик
3-ий механик
4-ий механик
4-ый электромеханик
[химик-эксперт
ASIC специалист
JavaScript разработчик
RTL специалист
Web-программист
web-разработчик
Автожестянщик
Автоинструктор
Автомоящик
Автор студенческих работ по различным дисциплинам
автослесарь
Автослесарь - моторист
Автоэлектрик
Агент
Агент банка
Агент нпф
Агент по гос. закупкам недвижимости
Агент по недвижимости
Агент по недвижимости (стакер)
Агент по недвижимости / Риэлтор
Агент по привлечению юридических лиц
Агент по продажам (интернет, ТВ, телефония) в ПАО Ростелеком в населенных пунктах Амурской области: г. Благовещенск, г. Белогорск, г. Свободный, г. Шимановск, г. Зея, г. Тында
Агент торговый
агрегатчик-топливник KOMATSU
агроном
агроном по защите растений
Агроном-полевод
агрохимик почвовед
Администратор
Администратор (удаленно)
Администратор Active Directory
Администратор в парикмахерский салон
Администратор зала (предприятий общественного питания)
Администратор кофейни
Администратор на ресепшен
Администратор на телефоне
Администратор по информационной безопасности
Администратор ресторана
Администратор сайта
Администратор ярмарок выходного дня
Администратор-кассир
Аккомпаниатор на 0,5 ст.
аккумуляторщик 4 разряда
Акушерка

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 1012464 руб.
Программист / Senior Developer с опытом Python = зарплата 1388245 руб.
Программист 1С с опытом Python = зарплата 1652039 руб.
Программист C# с опытом Python = зарплата 1600304 руб.
Программист C++ с опытом Python = зарплата 1091388 руб.
Программист C++/C#/Java с опытом Python = зарплата 1066206 руб.
Программист/ Junior Developer с опытом Python = зарплата 1153393 руб.
Программист/ технический специалист с опытом Python = зарплата 1711066 руб.
Программистр-разработчик информационных систем с опытом Python = зарплата 1221854 руб.
time: 0.07022833824157715
f0max@f0max:~/Code/laba_3/code$
```