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 «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Лабораторная работа № 3

по курсу “Разработка интернет-приложений”

“Функциональные возможности языка Python.”

Выполнил:

студент группы ИУ5-53Б

Волгина А. Д.

4.10.21

Проверил:

Гапанюк Ю. Е.

2021 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 раполагаться в отдельном файле.

При запуске каждого файла выдаются тестовые результаты выполнения соответствующего задания.

Текст программы

Main.py:

```
from random import randrange
def field(items, *args):
    assert len(args) > 0
    result = []
    if(len(args) == 1):
        for item in items:
            if(item[args[0]] == None):
                continue
            result.append(item[args[0]])
        return result
    for item in items:
        dic = {}
        for arg in args:
            if(item[arg] == None):
                continue
            dic[arg] = item[arg]
        if(dic != {}):
            result.append(dic)
    return result
def gen_random(amount, beg, end):
    result = []
    for i in range(amount):
        result.append(randrange(beg, end + 1, 1))
    return result
class Unique(object):
    def __init__(self, items, **kwargs):
        # Нужно реализовать конструктор
        # В качестве ключевого аргумента, конструктор должен принимать bool-параметр ignore_case,
        # в зависимости от значения которого будут считаться одинаковыми строки в разном регистре
        # Например: ignore_case = True, Абв и АБВ - разные строки
        #           ignore_case = False, Абв и АБВ - одинаковые строки, одна из которых удалится
        # По-умолчанию ignore_case = False
        self.ignore_case = False
        try:
            for key, value in kwargs.items():
                self.ignore_case = kwargs[key]
        except:
            self.ignore_case = False
```

```
        self.items = items
        self.used_elements = set()
        self.index = 0

    def __next__(self):
        # Нужно реализовать __next__
        while True:
            if self.index >= len(self.items):
                raise StopIteration
            else:
                current = self.items[self.index]
                self.index = self.index + 1
                try:
                    if(type(current) == str and self.ignore_case):
                        current = current.lower()
                except:
                    pass
                if current not in self.used_elements:
                    # Добавление в множество производится
                    # с помощью метода add
                    self.used_elements.add(current)
                    return current

    def __iter__(self):
        return self

def main():
    goods = [
   {'title': 'Ковер', 'price': 2000, 'color': 'green'},
   {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},
   {'title': None, 'price': None, 'color': 'black'}
 ]
    print(field(goods, 'title'))
    lst1 = gen_random(10, 1, 3)
    print(lst1)
    lst2 = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    for i in Unique(lst2, ignore_case=True):
        print(i)
if __name__ == "__main__":
    main()
```

unique.py:

```
class Unique(object):
    def __init__(self, items, **kwargs):
        # Нужно реализовать конструктор
        # В качестве ключевого аргумента, конструктор должен принимать bool-параметр ignore_case,
        # в зависимости от значения которого будут считаться одинаковыми строки в разном регистре
        # Например: ignore_case = True, Абв и АБВ - разные строки
        #           ignore_case = False, Абв и АБВ - одинаковые строки, одна из которых удалится
        # По-умолчанию ignore_case = False
```

```
        self.ignore_case = False
        try:
            for key, value in kwargs.items():
                self.ignore_case = kwargs[key]
        except:
            self.ignore_case = False
        self.items = items
        self.used_elements = set()
        self.index = 0

    def __next__(self):
        # Нужно реализовать __next__
        while True:
            if self.index >= len(self.items):
                raise StopIteration
            else:
                current = self.items[self.index]
                self.index = self.index + 1
                try:
                    if(type(current) == str and self.ignore_case):
                        current = current.lower()
                except:
                    pass
                if current not in self.used_elements:
                    # Добавление в множество производится
                    # с помощью метода add
                    self.used_elements.add(current)
                    return current

    def __iter__(self):
        return self
def main():
    """
    lst2 = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    for i in Unique(lst2, ignore_case=True):
        print(i)
     """

if __name__ == "__main__":
    main()
```

sort.py:

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

if __name__ == '__main__':
    data_abs = {}
    for i in range(len(data)):
        data_abs[i] = abs(data[i])
    data_abs = sorted(data_abs.items(), key=operator.itemgetter(1), reverse=True)
```

```
    result = [data[x[0]] for x in data_abs]
    print(result)

    result_with_lambda = sorted(data, key=lambda x: abs(x), reverse=True)
    print(result_with_lambda)
```

process_data.py:

```
import json
import sys
#import '.print_result.py'
from print_result import print_result
from cm_timer import cm_timer_1
from unique import Unique
from gen_random import gen_random
# 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, 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):
    res = []
    for el in data:
        res.append(el['job-name'])
    #print(sorted(Unique(res, ignore_case=True)))
    return sorted(Unique(res, ignore_case=True))


@print_result
def f2(arg):
    #print(list(filter(lambda x: x.startswith('программист'), arg)))
    return list(filter(lambda x: x.startswith('программист'), f1(data)))


@print_result
def f3(arg):
    return list(map(lambda x: x + ' с опытом Python', f2(f1(data))))


@print_result
def f4(arg):
```

```
    res = f3(f2(f1(data)))
    salaries = gen_random(len(f3(f2(f1(data)))), 100000, 200000)
    for i in range(len(f3(f2(f1(data))))):
        res[i] += ", зарплата {0}".format(salaries[i])
    return res


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

print_result.py:

```
def print_result(some_func, arg=[]):
    print(some_func.__name__)
    result = some_func(arg)
    if(type(result) == list):
        for el in result:
            print(el)
        return some_func
    if(type(result) == dict):
        for k, v in result.items():
            print("{0} = {1}".format(k, v))
        return some_func
    print(result)
    return some_func

@print_result
def test_1(arg=[]):
    return 1


@print_result
def test_2(arg=[]):
    return 'iu5'


@print_result
def test_3(arg=[]):
    return {'a': 1, 'b': 2}


@print_result
def test_4(arg=[]):
    return [1, 2]


if __name__ == '__main__':

    print('!!!!!!!!')
```

```
    test_1()
    test_2()
    test_3()
    test_4()
```

gen_random.py:

```
from random import randrange
def gen_random(amount, beg, end):
    result = []
    for i in range(amount):
        result.append(randrange(beg, end + 1, 1))
    return result
def main():
    lst1 = gen_random(10, 1, 3)
    print(lst1)
if __name__ == "__main__":
    main()
```

field.py:

```
def field(items, *args):
    assert len(args) > 0
    result = []
    if(len(args) == 1):
        for item in items:
            if(item[args[0]] == None):
                continue
            result.append(item[args[0]])
        return result
    for item in items:
        dic = {}
        for arg in args:
            if(item[arg] == None):
                continue
            dic[arg] = item[arg]
        if(dic != {}):
            result.append(dic)
    return result
def main():
    goods = [
   {'title': 'Ковер', 'price': 2000, 'color': 'green'},
   {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'},
   {'title': None, 'price': None, 'color': 'black'}
    ]
    print(field(goods, 'title'))
if __name__ == "__main__":
    main()
```

cm_timer:

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

```
class cm_timer_1:

    def __init__(self):
        self.begin = time.time()
    def __enter__(self):
        print("Начало работы: {0}".format(self.begin))
    def __exit__(self, exp_type, exp_value, traceback):
        if exp_type is not None:
            print(exp_type, exp_value, traceback)
        else:
            self.end = time.time()
            print("Конец работы: {0}".format(self.end))
            print("Время работы: {0}".format(self.end - self.begin))
"""
with cm_timer_1():
    time.sleep(5.5)
"""
@contextmanager
def cm_timer_2():
    beg = time.time()
    print("Начало работы: {0}".format(beg))
    yield
    end = time.time()
    print("Конец работы: {0}".format(end))
    print("Время работы: {0}".format(end - beg))
"""
with cm_timer_2():
    time.sleep(5.5)
"""
```

Скриншоты: