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»

ОТЧЁТ ПО

Лабораторной работе №5

Выполнил: Богуславский Андрей

студент группы ИУ5-31Б

_____

Подпись и дата:

Проверил:

преподаватель каф. ИУ5

Нардид А.Н.

Подпись и дата:

Москва

2024

Лабораторная работа №5

Функциональные возможности языка Python.
Цель лабораторной работы: изучение возможностей функционального программирования в языке Python.

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.

Задача 1 (файл field.py)
Необходимо реализовать генератор field. Генератор field последовательно выдает значения ключей словаря. Пример:

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

- 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.

Шаблон для реализации генератора:

```
# Пример:
# 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(items, *args):
    assert len(args) > 0
    # Необходимо реализовать генератор
```

Задача 2 (файл gen_random.py)

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

Шаблон для реализации генератора:

```
# Пример:
# gen_random(5, 1, 3) должен выдать выдать 5 случайных чисел
# в диапазоне от 1 до 3, например 2, 2, 3, 2, 1
# Hint: типовая реализация занимает 2 строки
def gen_random(num_count, begin, end):
    pass
    # Необходимо реализовать генератор
```

Задача 3 (файл unique.py)

- 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.

Пример:

data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
Unique(data) будет последовательно возвращать только 1 и 2.

data = gen_random(10, 1, 3)
Unique(data) будет последовательно возвращать только 1, 2 и 3.

data = [‘a’, ‘A’, ‘b’, ‘B’, ‘a’, ‘A’, ‘b’, ‘B’]
Unique(data) будет последовательно возвращать только a, A, b, B.

Unique(data, ignore_case=True) будет последовательно возвращать только a, b.

Шаблон для реализации класса-итератора:

```
# Итератор для удаления дубликатов
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
        pass

    def __next__(self):
        # Нужно реализовать __next__
        pass

    def __iter__(self):
        return self
```

Текст программы:

Файл field.py:

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

    if len(args) == 1:
        for item in items:
            if args[0] in item and item[args[0]] is not None:
                yield item[args[0]]
    else:
        for item in items:
```

```
            dict = {}
            all_none = True
            for key in args:
                if key in item and item[key] is not None:
                    dict[key] = item[key]
                    all_none = False
            if not all_none:
                yield dict

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

print(str(list(field(goods, 'title')))[1:-1]) #должен выдавать 'Ковер', 'Диван
для отдыха'
print(str(list(field(goods, 'title', 'price')))[1:-1]) #должен выдавать {'title':
'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}
```

Файл gen_random.py:

```
from random import randint
def gen_random(num_count, begin, end):
    ans = []
    for i in range(num_count):
        ans.append(randint(begin,end))
    yield ans

print(str(list(gen_random(5,1,3)))[1:-1])
```

Файл unique.py:

```
from gen_random import gen_random

class Unique(object):
    def __init__(self, items, **kwargs):
        self.data = iter(items)
        self.ignore_case = kwargs.get('ignore_case', False)
        self.unique_items = set()

    def __next__(self):
```

```
        while True:
            item = next(self.data)
            check_item = item.lower() if self.ignore_case else item
            if check_item not in self.unique_items:
                self.unique_items.add(check_item)
                return item

    def __iter__(self):
        return self

data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
unique_data = Unique(data)
print(list(unique_data))
# Output: [1, 2]

data = gen_random(10, 1, 3)
unique_data = Unique(data)
print(list(unique_data))
# Output: [1, 2, 3]

data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
unique_data = Unique(data)
print(list(unique_data))
# Output: ['a', 'A', 'b', 'B']

unique_data_ignore_case = Unique(data, ignore_case=True)
print(list(unique_data_ignore_case))
# Output: ['a', 'b']
```

Экранные формы с выводом:

```
PS C:\Users\Andrey\Desktop\инфа> & C:/Users/Andrey/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/Andrey/Desktop/инфа/field.py
'Ковер', 'Диван для отдыха'
{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}
PS C:\Users\Andrey\Desktop\инфа>
```

```
PS C:\Users\Andrey\Desktop\инфа> & C:/Users/Andrey/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/Andrey/Desktop/инфа/gen_random.py
[1, 1, 3, 3, 2]
PS C:\Users\Andrey\Desktop\инфа>
```

```
PS C:\Users\Andrey\Desktop\инфа> & C:/Users/Andrey/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/Andrey/Desktop/инфа/unique.py
[3, 3, 1, 1, 2]
[1, 2]
```