Лабораторная работа №3

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
```

Задача 4 (файл sort.py)

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. Пример:

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

Вывод: [123, 100, -100, -30, 30, 4, -4, 1, -1, 0]

Необходимо решить задачу двумя способами:

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

Шаблон реализации:

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


if __name__ == '__main__':
    result = ...
    print(result)
```

```
result_with_lambda = ...

print(result_with_lambda)
```

Задача 1 (файл field.py)

Текст программы

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

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

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

# Проверка для одного аргумента
for title in field(goods, 'title'):
   print(title)

# Проверка для двух аргументов
for item in field(goods, 'title', 'price'):
   print(item)
```

Пример выполнения программы

```
/Users/karinapodgornova/PycharmProj
Ковер
Диван для отдыха
{'title': 'Ковер', 'price': 2000}
{'title': 'Диван для отдыха'}

Process finished with exit code 0
```

Задача 2 (файл gen_random.py)

Текст программы

```
# 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
import random

def gen_random(count, begin, end):
    for i in range(count):
        yield random.randint(begin, end)

for num in gen_random(5, 1, 3):
    print(num)
```

Пример выполнения программы

```
/Users/karinapodgornova/PycharmProjec
1
1
2
3
2

Process finished with exit code 0
```

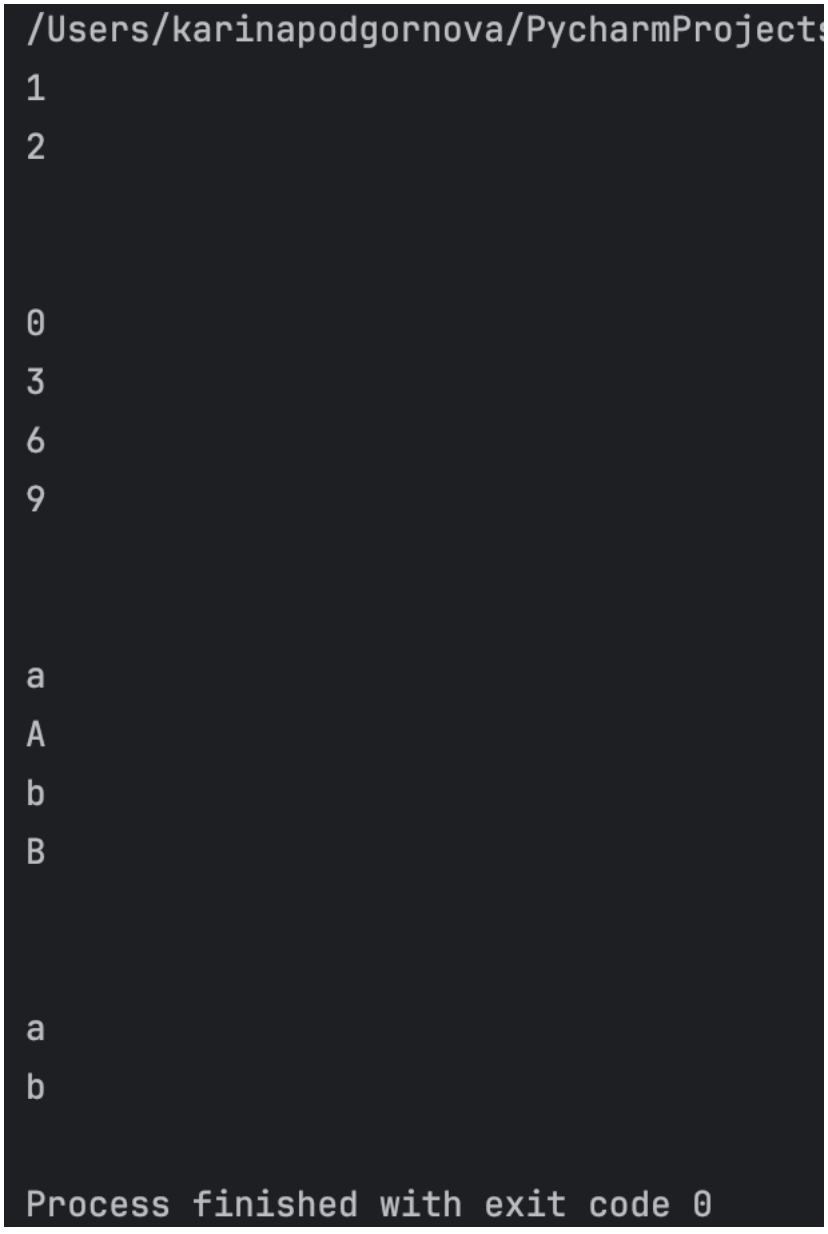
Задача 3 (файл unique.py)

Текст программы

```
# Итератор для удаления дубликатов
class Unique(object):
    def __init__(self, items, **kwargs):
        self.items = iter(items)  # Преобразуем входные данные в итератор
        self.ignore_case = kwargs.get('ignore_case', False)  # Получаем ignore_case
из kwargs, по умолчанию False
        self.seen = set()

    def __next__(self):
        while True:
            try:
                item = next(self.items)
                if self.ignore_case and isinstance(item, str):
                    item = item.lower()  # При ignore_case=True приводим строки к
нижнему регистру
                if item not in self.seen:
                    self.seen.add(item)
                    return item
            except StopIteration:
                raise StopIteration

    def __iter__(self):
        return self

# Примеры использования:
data1 = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
for item in Unique(data1):
    print(item)

print("\n")

data2 = (i for i in range(10) if i % 3 == 0) # Генератор
for item in Unique(data2):
    print(item)

print("\n")

data3 = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
for item in Unique(data3):
```

```
    print(item)

print("\n")

for item in Unique(data3, ignore_case=True):
    print(item)
```

Пример выполнения программы



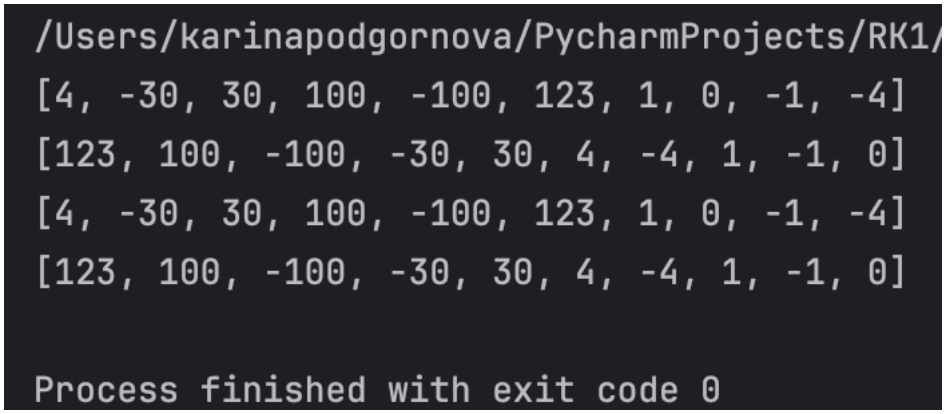
Задача 4 (файл sort.py)

Текст программы

```
# with Lamda
data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]
print(data)
```

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

# without Lamda
data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]
print(data)
result_without_lambda = sorted(data, key=abs, reverse=True)
print(result_without_lambda)
```

Пример выполнения программы



Вывод

Я изучила возможности функционального программирования в языке Python.