**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и системы управления»

Кафедра ИУ5. Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №3

«Функциональные возможности языка Python ч.1»

|  |  |  |
| --- | --- | --- |
| Выполнил:  Каженец Д.Н.  ИУ5-31Б |  | Проверил:  Гапанюк Е.Ю. |
| Подпись и дата: |  | Подпись и дата: |

Постановка задачи

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**

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'}

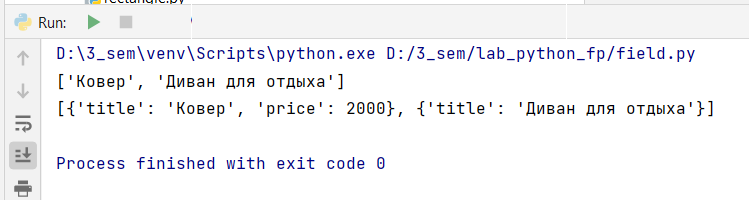
]

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.

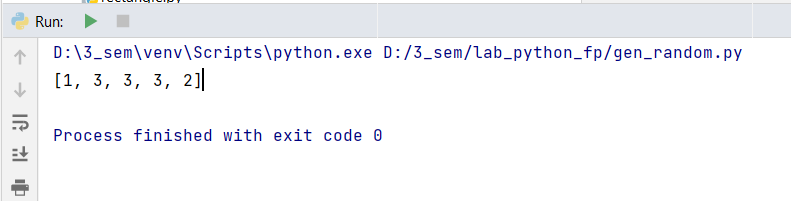
def field(items, \*args):  
 assert len(args) > 0  
  
 if len(args) == 1:  
 for i in items:  
 if i[args[0]] is not None:  
 yield i[args[0]]  
 else:  
 for i in items:  
 d = {}  
 for j in args:  
 if j in i:  
 d[j] = i[j]  
 yield d  
  
goods = [{'title': 'Ковер', 'price': 2000, 'color': 'green'},  
 {'title': 'Диван для отдыха', 'color': 'black'}]  
  
print(list(field(goods, 'title')))  
print(list(field(goods, 'title', 'price')))

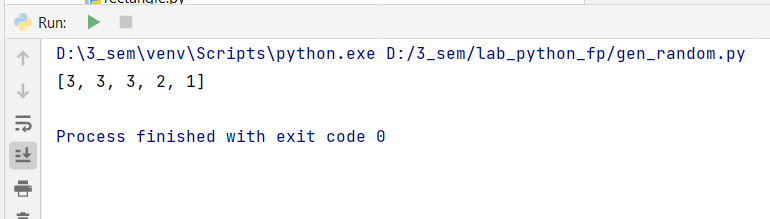


**Задача 2**

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

from random import randrange  
  
  
def gen\_random(num\_count, begin, end):  
 for i in range(num\_count):  
 yield randrange(begin, end + 1)  
  
print(list(gen\_random(5, 1, 3)))



**Задача 3**

* 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.

class Unique(object):  
 def \_\_init\_\_(self, items, \*\*kwargs):  
 self.ignore\_case = kwargs.get('ignore\_case', False)  
 self.items = list(items)  
 self.received = set()  
 self.index = 0  
  
 def \_\_next\_\_(self):  
 while self.index < len(self.items):  
 item = self.items[self.index]  
 self.index += 1  
  
 if self.ignore\_case and isinstance(item, str):  
 item\_key = item.lower()  
 else:  
 item\_key = item  
  
 if item\_key not in self.received:  
 self.received.add(item\_key)  
 return item  
 raise StopIteration  
  
 def \_\_iter\_\_(self):  
 return self  
  
data1 = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]  
unique1 = Unique(data1)  
print(list(unique1))  
data2 = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']  
unique2 = Unique(data2)  
print(list(unique2))  
unique3 = Unique(data2, ignore\_case=True)  
print(list(unique3))

