# Лабораторная работа №5:

# Комплект 1: Итераторы. Генераторы.

1.1: Создайте свой класс-итератор class RandomNumberIterator, который, в ходе итерирования по такому итератору, генерирует случайные числа в количестве и в диапазоне, которые передаются в конструктор в виде списка параметров.

# Код программы:

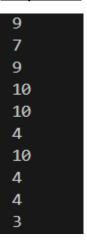
```
from random import randint

class RandomNumberIterator:
    def __init__(self, amount, start, end):
        self.range_start = start
        self.range_end = end
        self.amount = amount
        self.rand_list = list(randint(self.range_start, self.range_end) for i in range(self.amount))

def __iter__(self):
    return self

def __next__(self):
    if (self.current < self.amount - 1):
        self.current += 1
        return self.rand_list[self.current]
        raise StopIteration

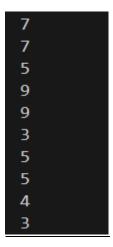
for i in RandomNumberIterator(10, 1, 10):
    print(i)</pre>
```



1.2: Решите задачу 1.1 уже с использованием генераторной функции, использующей ключевое слово yield. В качестве аргументов она должна принимать количество элементов и диапазон.

## Код программы:

### Результат:



1.3: Сделайте две функции-генератора. Первый генератор создаёт ряд Фибоначчи, а второй генератор добавляет значение 10 к каждому числу. Вызовете эти генераторы так, чтобы сгенерировать некоторое количество чисел Фибоначчи с добавлением числа 10 к каждому числу.

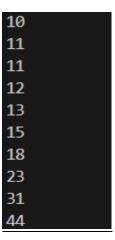
## Код программы:

```
1.3.py > ...
def fib_gen(num):
count, a, b = 0, 0, 1
while (count < num):
    yield a
    a, b, count = b, a + b, count + 1

def plus_10_gen(seq):
    for i in seq:
        yield i + 10

for i in plus_10_gen(fib_gen(10)):
    print(i)</pre>
```

### Результат:



1.4: Напишите программу, на вход к которой подается список стран и городов для каждой страны. Затем по названиям городов из ещё одного списка выводится в какой стране расположен каждый город.

## Код программы:

```
Введите города для поиска (через пробел): Dubai Lend
Dubai находится в United Arab Emirates
Lend находится в Austria
```

## Комплект 2: Менеджеры контекста.

2.1: Напишите класс менеджера контекста Timer, который умеет считать время в секундах, затраченное на некоторые вычисления внутри соответствующего блока with с помощью функции perf\_counter модуля time. Используйте этот менеджер контекста для определения времени на вычисления достаточно большого количества чисел Фибоначчи (например миллиона) в цикле с помощью отдельной функциигенератора.

### Код программы:

```
🅏 2.1.py > ...
      import time
      class Timer:
          def init (self):
             self.start time = 0
              self.end time = 0
             self.elapsed time = 0
          def enter (self):
              self.start time = time.perf counter()
              return self
12
          def exit (self, exc_type, exc_value, exc_traceback):
              self.end time = time.perf counter()
              self.elapsed time = self.end time - self.start time
      def fib gen(num):
          count, a, b = 0, 0, 1
          while (count < num):
              yield a
              a, b, count = b, a + b, count + 1
      fib list = list()
      with Timer() as timer1:
          #fib list = list(fib gen(1 000 000))
          print(fib gen(1 000 000))
      #print(fib list)
31
      print(timer1.elapsed_time)
```

```
<generator object fib_gen at 0x000001FE2EC302E0>
0.00020490004681050777
```

2.2: Напишите класс менеджера контекста BatchCalculatorContextManager, для вашего проекта калькулятора из предыдущих лабораторных работ. Этот менеджер контекста должен уметь открывать и закрывать текстовый файл, в каждой строчке которого записана пара чисел в сочетании с арифметической операцией над ними в виде простого арифметического выражения без пробелов. В сочетании с дополнительной функцией генератором и вашим менеджером контекста прочитайте все строчки текстового файла и вызовите нужное число раз функцию calculate(...) вашего калькулятора, чтобы распечатать все результаты на экране.

### Код программы:

```
🥏 2.2.py > ...
      import math
      import unittest
      import numpy
 4
     def convert precision(tolerance):
         if tolerance == 0:
             return 0
         return abs(math.floor(math.log10(abs(tolerance))))
     def calculate(action, *args, tolerance=1e-6):
          if action not in ['+', '-', '*', '/', 'medium', \
                  'variance', 'std_deviation', 'median', 'q2', 'q3']:
             raise ValueError("Неверная операция")
          if (action == '+'):
             result = sum(args)
          elif (action == '-'):
             result = args[0]
              for i in range(1, len(args)):
                result -= args[i]
          elif (action == '*'):
              result = 1
              for i in range(len(args)):
                result *= args[i]
          elif (action == '/'):
             result = args[0]
              for i in range(1, len(args)):
                 result /= args[i]
          elif (action == 'medium'):
             return numpy.mean(args)
          elif (action == 'variance'):
             return numpy.var(args)
          elif (action == 'std deviation'):
             return numpy.std(args)
          elif (action == 'median'):
              return numpy.median(args)
          elif (action == 'q2'):
              return numpy.median(args)
          elif (action == 'q3'):
              return numpy.percentile(args, 75)
```

```
42
         else:
             raise ValueError("Неверная операция")
         precision = convert precision(tolerance)
         rounded result = round(result, precision)
         return rounded result
     class BatchCalculatorContextManager:
         def init (self, file):
             self.filename = file
             self.file = None
             self.lines = None
             self.line count = None
         def __enter__(self):
             self.file = open(self.filename, "r")
             self.lines = self.file.readlines()
             self.line count = len(self.lines)
             return self
         def exit (self, exc type, exc val, exc tb):
             self.file.close()
             if exc val:
                 raise
         def perform calculation(self):
             for line in self.lines:
                 a, op, b = line.split()
70
                 yield calculate(op, float(a), float(b))
71
72
     with BatchCalculatorContextManager('file.txt') as calc:
         for i in calc.perform calculation():
             print(f"{i}")
76
```

### Результат:

23.0 5.0 27.0 5.0 2.3: Установите локально на свой компьютер объектную базу данных MongoDB. Установите с помощью менеджера пакетов рір или conda, в зависимости от того чем вы пользуетесь, пакет рутопдо для подключения к базам данных MongoDB. Например команда для рір: рір install рутопдо. С помощью инструмента MongoDB Shell создайте нового пользователя с правами админа, к примеру. Введите в командной строке mongosh без аргументов и уже в командной строке внутри MongoDB Shell введите: db.createUser({

Затем выйдите из MongoDB Shell (Введите exit или нажмите CtrlD). Перезайдите снова в MongoDB Shell с помощью команды mongosh -u myUserAdmin в командной строке и введя пароль abc123. Тем самым вы залогинетесь в базу MongoDB под новой учётной записью. Создайте пустую базу данных myshinynewdb с помощью команды use myshinynewdb. Добавьте коллекцию user в эту базу данных с одной единственной записью: db.user.insert({name: "Ada Lovelace", age: 205}). Коллекция будет создана автоматически. Напишите класс менеджера контекста для управляемого подключения к MondoDB и отключения от неё. Внутри блока with с помощью вызова метода user\_collection.find({'age': 205}) найдите вашу запись об "Ada Lovelace" и распечатайте её в терминале.

### Код программы:

Не выполнил