

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:

```
1.1.py > ...
1  from random import randint
2
3  class RandomNumberIterator:
4      def __init__(self, amount, start, end):
5          self.range_start = start
6          self.range_end = end
7          self.amount = amount
8          self.current = -1
9          self.rand_list = list(randint(self.range_start, self.range_end) for i in range(self.amount))
10
11     def __iter__(self):
12         return self
13
14     def __next__(self):
15         if (self.current < self.amount - 1):
16             self.current += 1
17             return self.rand_list[self.current]
18         raise StopIteration
19
20 for i in RandomNumberIterator(10, 1, 10):
21     print(i)
```

Результат:

```
9
7
9
10
10
4
10
4
4
3
```

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:

```
1.2.py > ...
1  from random import randint
2
3  def RandomNumberGenerator(start, end, amount):
4      for _ in range(amount):
5          yield randint(start, end)
6
7  for i in RandomNumberGenerator(0, 10, 10):
8      print(i)
```

Результат:

```
7
7
5
9
9
3
5
5
4
3
```

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

Код программы:

```
1.3.py > ...
1  def fib_gen(num):
2      count, a, b = 0, 0, 1
3      while (count < num):
4          yield a
5          a, b, count = b, a + b, count + 1
6
7  def plus_10_gen(seq):
8      for i in seq:
9          yield i + 10
10
11  for i in plus_10_gen(fib_gen(10)):
12      print(i)
```

Результат:

```
10
11
11
12
13
15
18
23
31
44
```

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:

```
1.4.py > ...
1  import csv
2
3  countries = {}
4
5  with open('world-cities.csv', encoding='utf-8') as f:
6      reader = csv.DictReader(f)
7      for i in reader:
8          if i["country"] not in countries.keys():
9              countries[i["country"]] = [i["name"]]
10         else:
11             countries[i["country"]].append(i["name"])
12
13  while True:
14      cities_to_find= input("Введите города для поиска (через пробел): ").split()
15      for city_to_find in cities_to_find:
16          for country, cities in countries.items():
17              if (city_to_find.lower() in [city.lower() for city in cities]):
18                  print(f"{city_to_find} находится в {country}")
```

Результат:

```
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 функции генератора.

Код программы:

```
2.1.py > ...
1  import time
2
3  class Timer:
4      def __init__(self):
5          self.start_time = 0
6          self.end_time = 0
7          self.elapsed_time = 0
8
9      def __enter__(self):
10         self.start_time = time.perf_counter()
11         return self
12
13     def __exit__(self, exc_type, exc_value, exc_traceback):
14         self.end_time = time.perf_counter()
15         self.elapsed_time = self.end_time - self.start_time
16
17
18     def fib_gen(num):
19         count, a, b = 0, 0, 1
20         while (count < num):
21             yield a
22             a, b, count = b, a + b, count + 1
23
24     fib_list = list()
25     with Timer() as timer1:
26         #fib_list = list(fib_gen(1_000_000))
27         print(fib_gen(1_000_000))
28
29
30     #print(fib_list)
31     print(timer1.elapsed_time)
```

Результат:

```
<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 строке 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 строки текстового файла и вызовите нужное число раз функцию calculate(...) вашего калькулятора, чтобы распечатать все результаты на экране.

Код программы:

```
2.2.py > ...
1  import math
2  import unittest
3  import numpy
4
5  def convert_precision(tolerance):
6      if tolerance == 0:
7          return 0
8      return abs(math.floor(math.log10(abs(tolerance))))
9
10 def calculate(action, *args, tolerance=1e-6):
11     if action not in ['+', '-', '*', '/', 'medium', \
12         'variance', 'std_deviation', 'median', 'q2', 'q3']:
13         raise ValueError("Неверная операция")
14
15     if (action == '+'):
16         result = sum(args)
17     elif (action == '-'):
18         result = args[0]
19         for i in range(1, len(args)):
20             result -= args[i]
21     elif (action == '*'):
22         result = 1
23         for i in range(len(args)):
24             result *= args[i]
25     elif (action == '/'):
26         result = args[0]
27         for i in range(1, len(args)):
28             result /= args[i]
29
30     elif (action == 'medium'):
31         return numpy.mean(args)
32     elif (action == 'variance'):
33         return numpy.var(args)
34     elif (action == 'std_deviation'):
35         return numpy.std(args)
36     elif (action == 'median'):
37         return numpy.median(args)
38     elif (action == 'q2'):
39         return numpy.median(args)
40     elif (action == 'q3'):
41         return numpy.percentile(args, 75)
```

```

42         else:
43             raise ValueError("Неверная операция")
44
45         precision = convert_precision(tolerance)
46         rounded_result = round(result, precision)
47
48         return rounded_result
49
50     class BatchCalculatorContextManager:
51         def __init__(self, file):
52             self.filename = file
53             self.file = None
54             self.lines = None
55             self.line_count = None
56
57         def __enter__(self):
58             self.file = open(self.filename, "r")
59             self.lines = self.file.readlines()
60             self.line_count = len(self.lines)
61             return self
62
63         def __exit__(self, exc_type, exc_val, exc_tb):
64             self.file.close()
65             if exc_val:
66                 raise
67
68         def perform_calculation(self):
69             for line in self.lines:
70                 a, op, b = line.split()
71                 yield calculate(op, float(a), float(b))
72
73
74     with BatchCalculatorContextManager('file.txt') as calc:
75         for i in calc.perform_calculation():
76             print(f"{i}")

```

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 `pip` или `conda`, в зависимости от того чем вы пользуетесь, пакет `pymongo` для подключения к базам данных MongoDB. Например команда для `pip`: `pip install pymongo`. 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({
    user: "myUserAdmin",
    pwd: "abc123",
    roles: [
        { role: "userAdminAnyDatabase", db: "admin" },
        "readWriteAnyDatabase"
    ]
})
```

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 залогинитесь 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 MongoDB 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

Результат: