Комарькова Анастасия группа ПОКС-21, вариант 9

Практическое занятие № 6

Тема: составление программ со списками в IDE PyCharm Community.

Цель: Закрепить усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрести навыки составления программ со списками в IDE PyCharm Community.

Постановка задачи: Дан целочисленный список размера 10. Вывести вначале все содержащиеся в данном списке четные числа в порядке возрастания их индексов, а затем — все нечетные числа в порядке убывания их индексов. (для удобства проверки все списки выводятся на экран)

Текст программы:

```
"""
Дан целочисленный список размера 10. Вывести вначале все содержащиеся в данном списке четные числа в порядке возрастания
их индексов, а затем — все нечетные числа в порядке убывания их индексов. (для удобства проверки все списки выводятся на
экран)
"""

count = 0
list = []

while count < 10:                # заполнение списка числами пользователя
  list.append(int(input('Введите число для заполнения списка: ')))
  count += 1
print('Ваш список:  ',list)

print('Четные элементы Вашего списка')
for element in list:             # перебор элементов. Вывод четных
  if element % 2 == 0:
    print(element)

print('Нечетные элементы Вашего списка')
for element in list[::-1]:       # перебор элементов. Вывод нечетных с конца
  if element % 2 == 1:
    print(element)
```

Протокол работы программы:

```
Введите число для заполнения списка: 12
Введите число для заполнения списка: 13
Введите число для заполнения списка: 10
Введите число для заполнения списка: 11
Введите число для заполнения списка: 465
Введите число для заполнения списка: 87
Введите число для заполнения списка: 6544
Введите число для заполнения списка: 55
Введите число для заполнения списка: 78
```

Комарькова Анастасия группа ПОКС-21, вариант 9

```
Введите число для заполнения списка: 0
Ваш список:   [12, 13, 10, 11, 465, 87, 6544, 55, 78, 0]
Четные элементы Вашего списка
12
10
6544
78
0
Нечетные элементы Вашего списка
55
87
465
11
13

Process finished with exit code 0
```

Постановка задачи: Дан список размера N. Найти количество участков, на которых его элементы монотонно убывают.(для удобства проверки все списки выводятся на экран)

Текст программы:

```
"""
Дан список размера N. Найти количество участков, на которых его элементы монотонно убывают.(для удобства проверки все списки выводятся на экран)
"""
from random import randint


list = []
list_count = []

n = int(input('Введите размер списка:  '))
max_num = int(input('Введите максимальное число:  '))

count = 0
while count < n:            # заполнение списка числами от 0 до маскимального значения пользователя
   list.append(randint(0, max_num))
   count += 1
print('Ваш список:  ', list)

"""
подробно о следующем шаге: здесь обрабатывается список пользователя, если последующий элемент больше предыдущего,
в другой список - list_count, заносится 1, иначе 0. Такой шаг я сделала для того, чтобы обозначить начало и конец
```

```
убывающей последовательности в списке пользователя.
"""
for i in range(1, len(list)):
   if list[i] > list[i - 1]:
      list_count.append(0)
   else:
      list_count.append(1)

print('Список с отображенными началами и концами убывания элементов, где 1 - убывание
элементов: ', list_count)

a = 0
"""
подробно о следующем шаге: здесь идет работа со списком list_count - последовательностью
единиц (которая означает, что
элементы в пользовательском списке на этих позициях убывали) и нулей. Так, например, для
последовательности 87, 45, 33 в
list_count занесется 1, 1. Но вышеприведенная последовательность убывания одна, хотя чисел 3.
Следующий цикл решает
эту проблему - после того, как встретилась последовательность единиц, она заменяется на одну
единицу = один ПРОМЕЖУТОК
убывания.
"""
while a < len(list_count) - 1:
   if list_count[a] and list_count[a + 1] == 1:
      list_count[a] = 0
   a += 1
print('Список с отображеннными промежутками убывания - единицы: ', list_count)

print('Количество промежутков убывания:  ', sum(list_count))     # Вывод суммы списка - количества
промежутков убывания
```

Протокол работы программы:

```
Введите размер списка:  10
Введите максимальное число:  10
Ваш список:   [7, 4, 10, 4, 2, 4, 7, 5, 3, 6]
Список с отображенными началами и концами убывания элементов, где 1 - убывание элементов:
[1, 0, 1, 1, 0, 0, 1, 1, 0]
Список с отображенными промежутками убывания - единицы:  [1, 0, 0, 1, 0, 0, 0, 1, 0]
Количество промежутков убывания:   3

Process finished with exit code 0
```

Постановка задачи: Дано множество A из N точек на плоскости и точка B (точки заданы своими координатами x, y). Найти точку из множества A,наиболее близкую к точке B. Расстояние R между точками с координатами (x1, y1) и (x2, y2) вычисляется по формуле: R = √(x2 – x1)2+ (y2 – y1)2. Для хранения данных о каждом наборе точек следует использовать по два список: первый список для

хранения абсцисс, второй — для хранения ординат.(для удобства проверки все списки выводятся на экран)

Текст программы:

```
"""
Дано множество A из N точек на плоскости и точка B (точки заданы своими координатами x, y). Найти точку из множества A, наиболее близкую к точке B. Расстояние R между точками с координатами (x1, y1) и (x2, y2) вычисляется по формуле:R = √(x2 – x1)2+ (y2 – y1)2. Для хранения данных о каждом наборе точек следует использовать по два список: первый список для хранения абсцисс, второй — для хранения ординат.(для удобства проверки все списки выводятся на экран)
"""
from math import sqrt
from random import randint

x = int(input('Введите значение абсциссы точки B '))
y = int(input('Введите значение ординаты точки B '))
n = int(input('Введите  количество точек  x и y:  '))


max_num = int(input('Введите максимальное значение для точек:  ')) + 1
abs_x = []      # список для абсцисс
ord_y = []      # список для ординат

count = 0

while count < n:          # заполнение списков
   abs_x.append(randint(0, max_num))
   ord_y.append(randint(0, max_num))
   count += 1
print('Список абсцисс:  ', abs_x)

print('Список ординат:  ', ord_y)

min_x = 0

min_y = 0

for i in range(0, len(abs_x) - 1):

   r = sqrt(((x - abs_x[i]) ** 2) + ((y - ord_y[i]) ** 2))  # расчет расстояния между точками по формуле

   if r < max_num:

      max_num = r

      min_x = abs_x[i]

      min_y = ord_y[i]
```

Комарькова Анастасия группа ПОКС-21, вариант 9

```
print('Минимальное расстояние:  ', max_num)

print('Абсцисса самой близкой точки:  ',  min_x)

print('Ордината самой близкой точки:  ', min_y)
```

Протокол работы программы:

```
Введите значение абсциссы точки B 0
Введите значение ординаты точки B 0
Введите  количество точек  x и y:  10
Введите максимальное значение для точек:  10
Список абсцисс:   [10, 0, 6, 3, 8, 5, 11, 5, 6, 11]
Список ординат:   [1, 5, 1, 10, 2, 0, 1, 4, 2, 3]
Минимальное расстояние:   5.0
Абсцисса самой близкой точки:   0
Ордината самой близкой точки:   5

Process finished with exit code 0
```

Вывод: в процессе выполнения практического занятия я выработала навыки составления программ со списками в IDE PyCharm Community.Были использованы языковые конструкции while, if, for, append.
Выполнены разработка кода, отладка, тестирование, оптимизация программного кода.
Готовые программные коды выложены на GitHub.