Студент группы ИС-23 Камилов С. Н.

Практическое задание №6

Тема: “Составление программ со списками в IDE PyCharm Community.”

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

Постановка задачи 1: 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.

Тип алгоритма: функция с ветвлением и циклом.

Текст программы:

```
  # Программа для вывода четных чисел по возрастанию и нечетных по убыванию в порядке индексов

  spisok = [47, 64, 8, 85, 36, 32, 45, 5, 43, 63]


  def go_back_out(spis):  # Функция с принятием аргумента списка
     print('Прямой вывод четных чисел: ')
     for i in range(len(spis)):  # цикл с выводом четных чисел их списка в порядке возрастания
         if spis[i] % 2 == 0:
             print(spis[i], end=' ')
     print('\n\nРеверсивный вывод нетных чисел:')
     for i in range((len(spis)-1), -1, -1):  # цикл в выводом нечетных чисел в порядке убывания
         if spis[i] % 2 != 0:
             print(spis[i], end=' ')


  try:
     go_back_out(spisok)  # Вызов функции
  except TypeError:
     print('Неправильный аргумент! Должен быть список')
```

Протокол работы программы:

```
Прямой вывод четных чисел:
64 8 36 32
```

Реверсивный вывод нечетных чисел:
63 43 5 45 85 47
Process finished with exit code 0

Постановка задачи 2: Дан список размера N. Найти количество участков, на которых его элементы монотонно убывают.

Тип алгоритма: функция с циклом и ветвлением.

Текст программы:

```
# Программа для подсчета количества монотонно убывающих участков

a = [3, 5, 92, 5, 100, 2, 50, 35, 79, 60, 98, 47, 92, 40, 32, 58, 24]


def compute(spisok):  # Функция считает количество участков с монотонным убыванием
    score = 0
    check_list = 0
    for i in range(len(spisok)):  # Цикл для перебора списка
        if spisok[i-1] < spisok[i]:  # Проверка на монотонность участка
            score += 1
        else:
            check_list += 1  # Здесь будет записывать количество исключений,
сработавших при проверке
    return 'Количество участков с монотонным убыванием: ' + str(check_list)


try:
    print(compute(a))  # Вызов функции
except TypeError:
    print('Неправильный аргумент! Нужно передавать список!')
```

Протокол работы программы:
Количество участков с монотонным убыванием: 9

Process finished with exit code 0

Постановка задачи 3: 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.

Тип алгоритма: функция с циклом и ветвлением.

Текст программы:

```
# Программа для нахождения ближней точки b в множестве a

import random
import math

try:
    n = int(input('Размер множества a: '))
    a = [[random.randint(1, 100) for i in range(0, n)], [random.randint(1, 100) for
i in range(0, n)]]  # Множество a
    b = [[], []] # Список для значений точки B
    s = [[100],[],[]]
    bx = int(input('Введите x B: '))
    by = int(input('Введите y B: '))
    b[0].append(bx)
    b[1].append(by)
    print(b)
    for i in range(0, n):
        r = math.sqrt((b[0][0]-a[0][i])**2+(b[1][0]-a[1][i])**2)
        if r < s[0][0]:
            s[0][0].clear()
            s[0].append(r)
            s[1].append(a[0][i])
            s[2].append(a[1][i])
    print(s)
except Exception as e:
    print('Вы ввели не целочисленный тип данных!', e)
```

Протокол работы программы:

Количество участков с монотонным убыванием: 9

Process finished with exit code 0

Вывод: в процессе выполнения практического задания, мною были выработаны навыки составления программ в IDE PyCharm Community со списками. Были

использованы языковые конструкции языка программирования Python: функции, try-except, if, for.

Выполнены разработка кода, отладка, тестирование, оптимизация программного кода.

Готовые программные коды выложены на GitHub.