Практическое задание №6 – Составление программ со списками в IDE PyCharm Community

Автор: Кузнецов Владислав ИС-26

Цели практического занятия

Цель практического занятия: Закрепить знания по созданию и использованию функций в Python, освоить обработку исключений и работу с IDE PyCharm Community. Создать программы, использующие функции, обработку ошибок, а также соответствующие требованиям PEP 8.

Вариант 18. Задания

Условие 1: Дан целочисленный список A размера 10. Необходимо вывести порядковый номер последнего из тех его элементов AkA_kAk, которые удовлетворяют двойному неравенству: A1<Ak<A10A_1 < A_k < A_{10}A1<Ak<A10. Если таких элементов нет, вывести 0.

Условие 2: Дано число RRR и список размера NNN. Нужно найти два различных элемента списка, сумма которых наиболее близка к числу RRR, и вывести эти элементы в порядке возрастания их индексов.

Ход работы

Настройка проекта в PyCharm Community: - Открыл IDE PyCharm Community. - Создал новый проект по пути: C:\Документы\PycharmProjects\IS26\Proj\Kuz. - Назвал проект PR6.

Создание пакета и файла: - Внутри проекта создал пакет PZ6, где будет размещена работа. - В пакете PZ6 создал файл PZ6_18.py, соответствующий задачам варианта.

Задача 1. Порядковый номер последнего элемента, удовлетворяющего двойному неравенству:

Разработка алгоритма и блок-схема:

Начало

-
- **Ввод списка A из 10 элементов**.
- **Присвоить переменной значение первого элемента списка**.
- **Присвоить переменной значение последнего элемента списка**.
- **Для каждого элемента списка A, начиная с 2-го по 9-й, выполнить проверку:**

  Вывести значение. Если элементов не найдено, вывести 0.

Конец

Код:

```python
#Кузнецов Влад

# Сначала создаем пустой список
A = []

# Используем цикл для ввода 10 элементов
for i in range(10):
    num = int(input(f"Введите элемент {i + 1}: "))
    A.append(num)  # Добавляем число в список

print("Ваш список:", A)

# Получаем первое и последнее числа из списка
first_element = A[0]
last_element = A[-1]

index = 0

# Перебираем от 1 до 8, по тому что ищем между первым и последним
for i in range(1, 9):
    if first_element < A[i] < last_element:  # Проверяем условие A1 < AK <
        index = i + 1

print("Порядковый номер последнего подходящего элемента:", index)

```

Пример выполнения программы:

- Введите элемент 1: 5
  Введите элемент 2: 7
  Введите элемент 3: 6
  Введите элемент 4: 8
  Введите элемент 5: 4
  Введите элемент 6: 2
  Введите элемент 7: 9

Введите элемент 8: 10
Введите элемент 9: 3
Введите элемент 10: 15
Последний элемент, удовлетворяющий условию: 8

Задача 2. Найти 2 различных элементов списка:

Разработка алгоритма и блок-схема:

Начало

-
- Вводим число R и список чисел.
- Применяем перебор всех пар чисел и находим пару с минимальной разницей между их суммой и числом R.
- Выводим два числа с минимальной разницей в сумме от числа R.

Конец

Код:

```python
#Кузнецов Влад

N = int(input("Введите размер списка: "))

R = int(input("Введите число R: "))

A = []

# Вводим элементы списка
for i in range(N):
    num = int(input(f"Введите элемент {i + 1}: "))
    A.append(num)  # Добавляем число в список

print("Ваш список:", A)

min_diff = 999999999

best_num1 = None
best_num2 = None

for i in range(N):  # Перебираем первый элемент пары
    for j in range(i + 1, N):  # Перебираем второй элемент
        diff = abs(A[i] + A[j] - R)  # Считаем разницу между суммой пары чисел и чи
        if diff < min_diff:
            min_diff = diff  # Новая минимальная разница
            best_num1 = A[i]  # Первое число
            best_num2 = A[j]  # Второе число

print(f"Элементы списка, чья сумма наиболее близка к числу {R}: {best_num1} и {best

```

Пример выполнения программы:

```
Введите число R: 15
Введите размер списка N: 6
Введите элемент 1: 4
Введите элемент 2: 8
Введите элемент 3: 3
Введите элемент 4: 10
Введите элемент 5: 7
Введите элемент 6: 5
Два числа, сумма которых наиболее близка к числу R: 8 и 7
```

Выводы

- Закреплены знания и навыки работы со списками в Python.
- Усовершенствована практика обработки различных типов данных и условий с использованием циклов и логических операторов.
- Программы были оформлены в соответствии с PEP 8, с добавлением обработки исключений и корректных комментариев.