Практическое занятие 4 Разработка программы с использованием цикла while

1. Тема работы

Разработка программы с использованием цикла `while`. Организация повторяющихся действий и обработка данных в циклах.

2. Цель работы

Изучить работу цикла `while` в Python, научиться применять его для организации повторяющихся действий, освоить использование операторов `break` и `continue`, изучить организацию бесконечных циклов и условия выхода из них.

3. Задачи работы

- Научиться использовать цикл `while` для решения задач
- Освоить операторы управления циклом: `break` и `continue`
- Научиться организовывать корректный выход из цикла
- Написать 4 программы с использованием цикла `while` согласно индивидуальному варианту

4. Необходимое оборудование и ПО

- Компьютер с ОС Windows 10
- Установленные: VS Code, Python
- Установленные расширения VS Code: Python, Python Debugger

---

5. Ход работы

Часть 1: Повторение и подготовка

1. **Откройте вашу папку проекта** в VS Code (например, `Иванов_СА-501`)
2. **Создайте новую папку** `Lab_4` внутри вашей основной папки
3. **Создайте файлы** для заданий: `while1.py`, `while2.py`, `while3.py`, `while4.py`

Часть 2: Разбор примера программы

Пример 1: Программа "Сумма чисел до нуля"

```
# Программа выводит на экран текст, пока не будет введён q

s = ""

while s != "q":
    s = input("Введите текст (q для выхода): ")
    print(f"Вы ввели: {s}")

print("Работа программы завершена")
```

Пример 2: Программа с использованием `break` и `continue`

```
# Программа считает сумму положительных чисел, игнорируя отрицательные

total = 0

while True:  # бесконечный цикл
    number = int(input("Введите число (999 для выхода): "))

    if number == 999:
        break  # выход из цикла

    if number < 0:
        print("Отрицательные числа игнорируются")
        continue  # переход к следующей итерации

    total += number

print(f"Сумма положительных чисел: {total}")
```

Пошаговый разбор:

1. **Цикл `while`** - выполняется, пока условие истинно
2. **Бесконечный цикл** - `while True:` выполняется всегда, пока не встретится `break`
3. **Оператор `break`** - немедленный выход из цикла
4. **Оператор `continue`** - переход к следующей итерации цикла
5. **Накопительная переменная** - `total` накапливает сумму

Запуск программы:

- Сохраните файл (Ctrl+S)
- Запустите через Ctrl+F5 или в терминале:

```
python while1.py
```

Часть 3: Выполнение задания по варианту

1. **Получите задание** у преподавателя согласно вашему варианту
2. **Реализуйте программы** в соответствующих файлах
3. **Протестируйте программы** на разных входных данных
4. **Убедитесь**, что циклы работают корректно и есть условия выхода

---

6. Варианты заданий

Вариант 1:

1. Напишите программу, которая запрашивает числа до тех пор, пока не будет введён 0, затем выводит сумму всех введённых чисел.
2. Напишите программу, которая вычисляет факториал числа N с использованием цикла `while`.
3. Напишите программу, которая определяет, является ли введённое число простым.
4. * Напишите программу, которая реализует игру "Угадай число": компьютер загадывает число от 1 до 100, а пользователь пытается угадать.

Вариант 2:

1. Напишите программу, которая запрашивает числа до тех пор, пока не будет введено отрицательное число, затем выводит количество введённых положительных чисел.
2. Напишите программу, которая выводит все чётные числа от 2 до N с использованием цикла `while`.
3. Напишите программу, которая вычисляет сумму цифр введённого числа произвольной длины.
4. * Напишите программу, которая реализует простой калькулятор с операциями +, -, *, /, который работает до тех пор, пока пользователь не введёт "выход".

Вариант 3:

1. Напишите программу, которая запрашивает пароль до тех пор, пока не будет введён правильный пароль.

2. Напишите программу, которая выводит числа Фибоначчи до N-го члена с использованием цикла `while`.
3. Напишите программу, которая определяет, является ли введённое число простым.
4. * Напишите программу, которая реализует игру "Угадай число": компьютер загадывает число от 1 до 100, а пользователь пытается угадать.

Вариант 4:

1. Напишите программу, которая запрашивает числа и находит их среднее арифметическое до тех пор, пока не будет введён 0.
2. Напишите программу, которая находит наибольший общий делитель (НОД) двух чисел с использованием алгоритма Евклида.
3. Напишите программу, которая вычисляет сумму цифр введённого числа произвольной длины.
4. * Напишите программу, которая реализует банкомат: начальный баланс 1000, пользователь может снимать деньги, пока баланс не станет отрицательным.

Вариант 5:

1. Напишите программу, которая выводит таблицу умножения для числа N до 10.
2. Напишите программу, которая находит сумму ряда 1 + 1/2 + 1/3 + ... + 1/N.
3. Напишите программу, которая определяет, является ли введённое число простым.
4. * Напишите программу, которая реализует простой калькулятор с операциями +, -, *, /, который работает до тех пор, пока пользователь не введёт "выход".

---

7. Отчёт по выполнению работы

Отчёт должен содержать:

1. **Цель работы**
2. **Задание по варианту** (текст всех 4 заданий)
3. **Листинги программ** (код всех 4 программ)
4. **Скриншоты выполнения программ** (результаты работы в терминале)
5. **Выводы**:
   - С какими трудностями столкнулись при написании программ?
   - В каких задачах целесообразно использовать цикл `while` вместо `for`?
   - Какая программа была самой сложной и почему?
   - Как избежать бесконечного цикла при использовании `while`?

8. Контрольные вопросы

1. Чем цикл `while` отличается от цикла `for`?
2. Что такое бесконечный цикл и как его организовать?
3. Какие операторы позволяют управлять выполнением цикла?
4. В чём разница между `break` и `continue`?
5. Как организовать цикл с постусловием в Python?
6. Можно ли использовать `else` с циклом `while`? Если да, то в каких случаях он выполняется?
7. Как избежать зацикливания программы?
8. В каких практических задачах предпочтительнее использовать цикл `while`?