Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМУПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра Инновационных

технологий (ФИТ)

Отчет

по лабораторной работе №7

**«ЦИКЛЫ FOR И WHILE»**

по дисциплине «Алгоритмы и структуры данных»

Студент гр. 023

\_\_\_\_\_\_\_\_\_\_\_\_ Иптышев В.А

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2025 г.

Доцент кафедры УИ

\_\_\_\_\_\_\_\_\_\_ Лобода Ю.О.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2025 г.

Томск 2025

**Содержание**

[Введение 3](#_Toc192806723)

[I ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРИМЕНЕНИЯ УСЛОВНОГО ОПЕРАТОРА В PYTHON 4](#_Toc192806724)

[II ПРАКТИЧЕСКАЯ ЧАСТЬ 7](#_Toc192806725)

[2.1. Решения задач на тему «Выбор из двух» 7](#_Toc192806726)

[2.2 Решения задач на тему «Логические операции» 10](#_Toc192806727)

[2.3 Решения задач на тему «Вложенные и каскадные условия» 12](#_Toc192806728)

[2.4 Итоговая работа на условный оператор 16](#_Toc192806729)

[Заключение 19](#_Toc192806730)

# Введение

В жизни часто встречаются ситуации, где приходится выполнять однотипные, монотонные действия. Например, сортировка деталей, выставление оценок в журнале, рассада моркови на грядке – все эти занятия требуют повторения определенных операций. Для решения этой задачи были придуманы циклы. В любой ситуации, где приходится что-либо повторять, они являются хорошими помощниками. Использование циклов может сильно сократить написанный код, и сделать его более понятным и лаконичным для чтения.

В Python существует две основных разновидности циклов: *for* и *while*. Цикл *for* замечательно работает, если мы заранее знаем, сколько повторений (итераций) нам требуется сделать. В противном случае стоит использовать *while*.

**Целью** лабораторной работы является изучение циклов в Python.

**Задачи**:

1. Изучить теорию по циклам в Python;
2. Решить задачи на тему циклы;
3. Сделать вывод по применению циклов в Python.

# I ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРИМЕНЕНИЯ ЦИКЛОВ В PYTHON

Чтобы написать цикл с заранее известным количеством повторений можно использовать оператор *for*.

Структура цикла *for* в Python выглядит так:

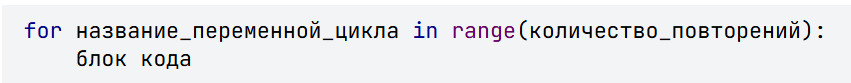


Рисунок 1.1 — структура цикла *for* в Python.

Рассмотрим следующий программный код:

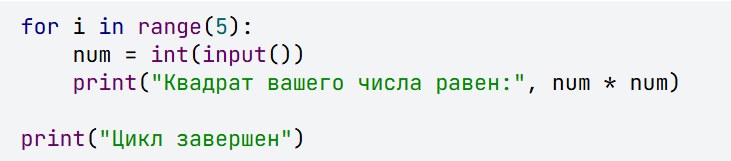


Рисунок 1.2 — код, выполняющий некую логику с использованием цикла *for*.

Такая программа считывает 5 чисел и выводит на экран их квадраты вместе с поясняющей надписью. Поскольку вторая и третья строки выделены отступом, Python считает, что это тело цикла, которое выполняется 5 раз. Четвертая строка не содержит отступа, поэтому не является частью цикла и будет выполнена всего один раз, после того как цикл завершится.

Цикл *for* замечательно работает, если мы заранее знаем, сколько повторений (итераций) нам потребуется сделать. Но иногда нужно, чтобы цикл выполнялся до наступления некоторого события, и количество итераций в этом случае заранее оценить просто невозможно. И здесь на помощь приходит цикл *while*.

Структура цикла *while* в Python выглядит так:

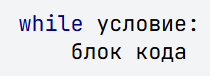


Рисунок 1.2 — структура цикла while.

Двоеточие (:) в конце строки с инструкцией while сообщает Python, что дальше находится блок команд. В блок входят все строки, расположенные с отступом от строки с инструкцией while, вплоть до следующей строки без отступа.

Блок команд, который выполняется в цикле while, называется телом цикла.

Рассмотрим код, использующий цикл while, который распечатает 10 раз слово Привет:

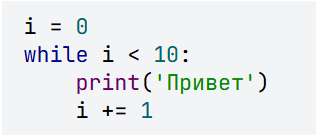


Рисунок 1.2 — код, печатающий 10 раз слово «Привет».

Такой код можно легко заменить циклом for, поскольку мы заранее знаем, сколько раз нужно выполнить тело цикла. Однако так бывает не всегда.

Напишем программу, которая считывает числа и выводит их квадраты, пока не будет введено −1. При такой постановке задачи мы не можем воспользоваться циклом for, так как не знаем, сколько чисел будет предшествовать числу −1.

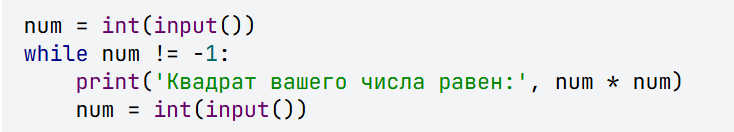


Рисунок 1.2 — код, печатающий квадраты введенного числа до того момента, пока не было введено -1.

# II ПРАКТИЧЕСКАЯ ЧАСТЬ