ПМ2 Прикладное программирование.

РО 2.1. Создавать консольные приложения на Python.

Тема 4. Цикл While.

Цель занятия:

Сформировать у студентов понимание принципа работы цикла while, умение применять его для решения задач, связанных с повторяющимися действиями: подсчётом, накоплением сумм и поиском значений.

Учебные вопросы:

1. Понятие цикла.
2. Цикл while в Python.
3. Типовые алгоритмы с while.

1. Понятие цикла.

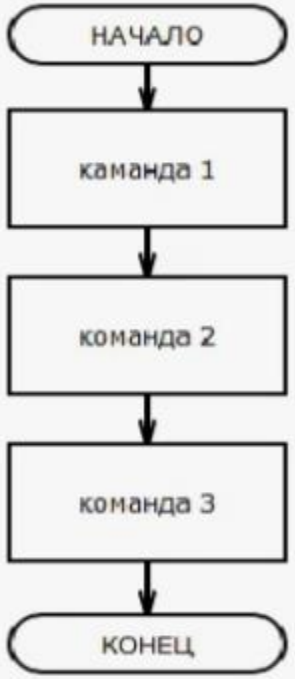
Цикл — это конструкция в программировании, которая позволяет многократно выполнять один и тот же фрагмент кода, пока выполняется определённое условие.

Цикл реализует повторение действий без необходимости вручную писать одни и те же команды много раз.

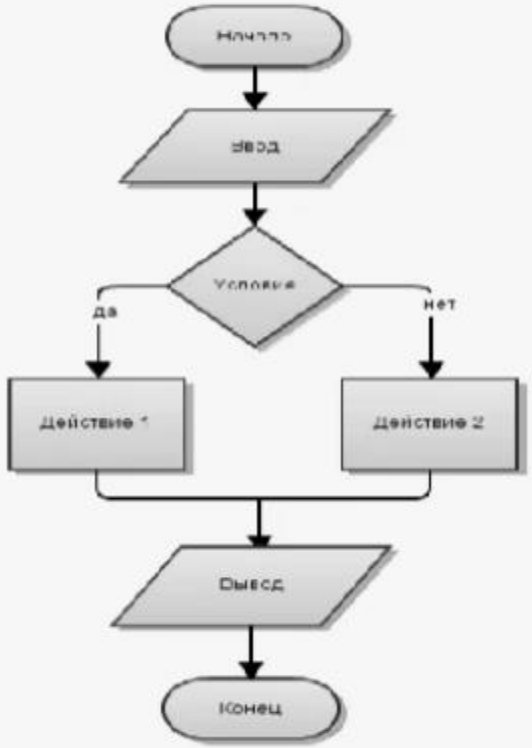
В Python существует два типа циклов: цикл **while** и цикл **for**

Алгоритмы

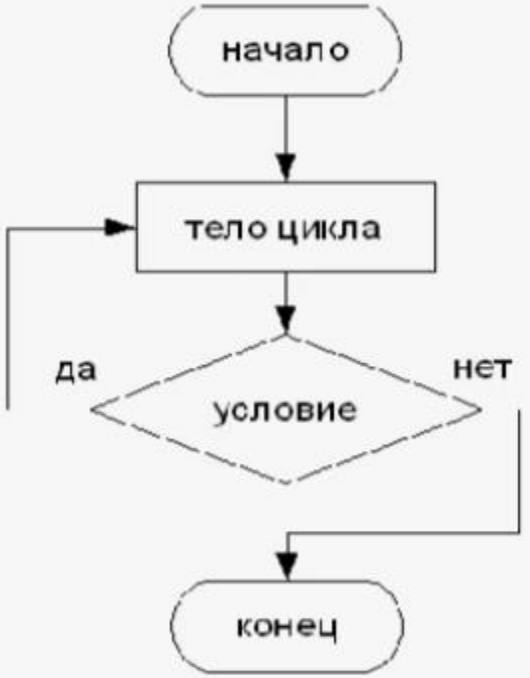
Линейные

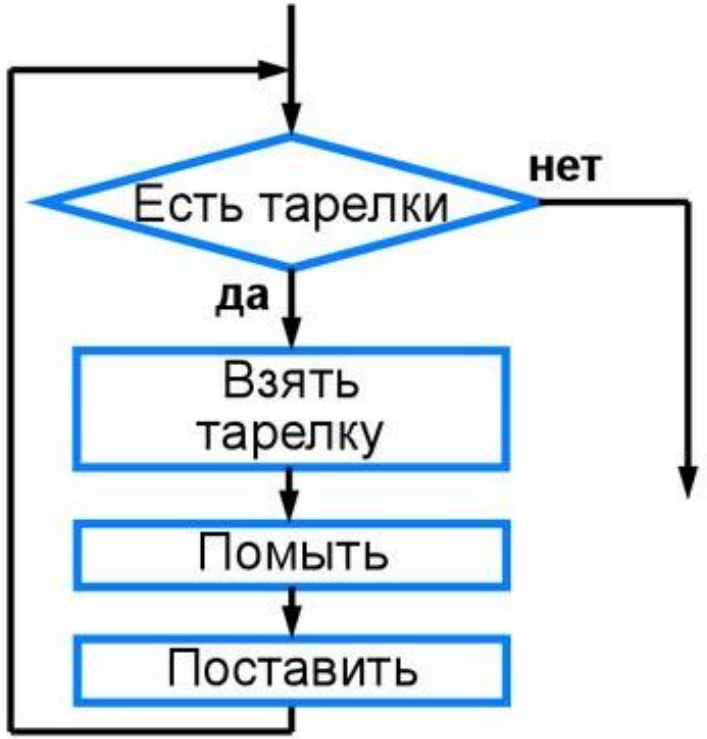


Разветвляющийся



Циклический




Есть тарелки
нет
да
Взять тарелку
Помыть
Поставить

Циклы позволяют:

- Автоматизировать повторяющиеся операции.

  Без цикла программисту пришлось бы копировать команды десятки или сотни раз.
- Обрабатывать большие объёмы данных.

  Например, вывод всех чисел от 1 до 100, обработка списка учащихся, чтение файлов.
- Реагировать на условия, меняющиеся в процессе работы программы.

  Программа может работать, пока пользователь не введёт нужные данные или пока значение не достигнет результата.
- Уменьшить количество кода и снизить вероятность ошибок.

  Один блок кода выполняется многократно, что делает программу короче и понятнее.

Вывод:

Цикл — это один из фундаментальных инструментов программирования.

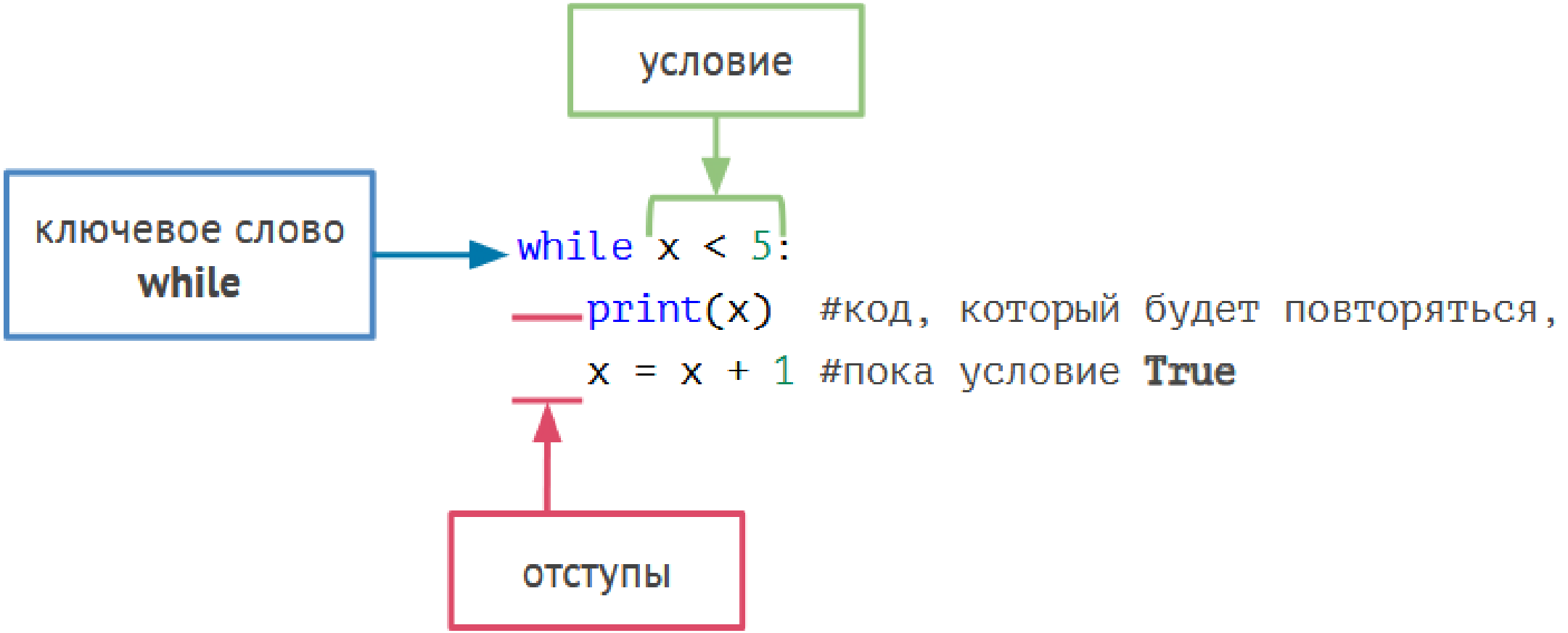
Он позволяет компьютеру выполнять повторяющиеся действия автоматически, что значительно упрощает решение задач, связанные с перебором, расчётами и анализом данных.

2. Цикл **while** в Python.

Цикл **while** используется для многократного выполнения блока команд до тех пор, пока условие истинно (**True**).

Он подходит для случаев, когда количество повторений заранее неизвестно, и цикл должен работать до достижения определённого результата.

Синтаксис конструкции **while**



Условие продолжения цикла (True / False)

Цикл запускается только если условие истинно.

На каждой итерации происходит повторная проверка условия:

```
1  x = 1
2  while x <= 5:  # условие True, пока i не больше 5
3      print(x, end=" ")
4      x += 1
5
```

```
1 2 3 4 5
```

Как только i становится 6 → условие i <= 5 ложно → цикл прекращается.

Работа с условием цикла

- Условие может быть простым: x < 10, n != 0, flag == True
- Или сложным, с логическими операторами: x > 0 and x % 2 == 0
- Чтобы цикл завершился, необходимо менять значения переменных, участвующих в условии.

Бесконечный цикл

Бесконечный цикл — это цикл, который никогда не заканчивается, потому что условие всегда остаётся True.

```
x = 1
while x <= 5:  # условие True, пока i не больше 5
    print(x, end=" ")
    # x += 1
```

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Причины возникновения:

- ошибка программиста
- неправильно написанное условие
- забыли изменить переменную в цикле

Как избежать:

- тщательно проверять условие
- изменять переменные внутри цикла
- по необходимости использовать **break**

Ключевые слова break, continue и pass

break — завершает цикл досрочно

```
while True:
    x = int(input("Введите число: "))
    if x == 0:
        break  # выход из цикла
```

```
Введите число: 1
Введите число: 2
Введите число: 0

>>>
```

continue — пропускает оставшийся код данной итерации и переходит к следующей

```
i = 0
while i < 5:
    i += 1
    if i == 3:
        continue
    print(i, end=" ")  # 3 будет пропущено
```

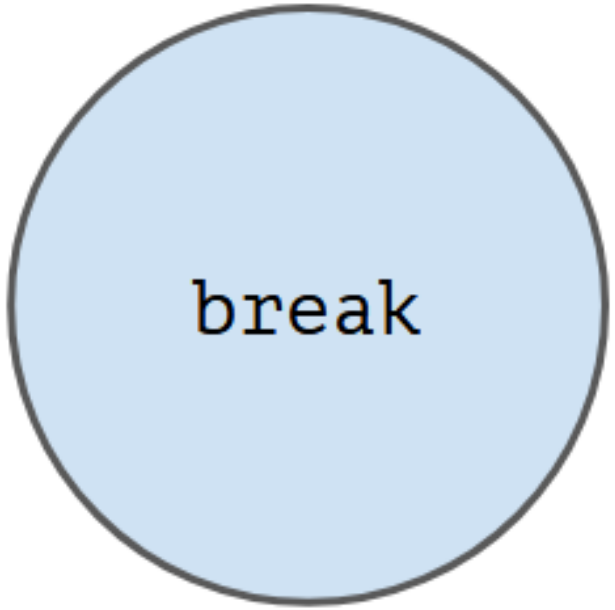
```
1 2 4 5

>>>
```

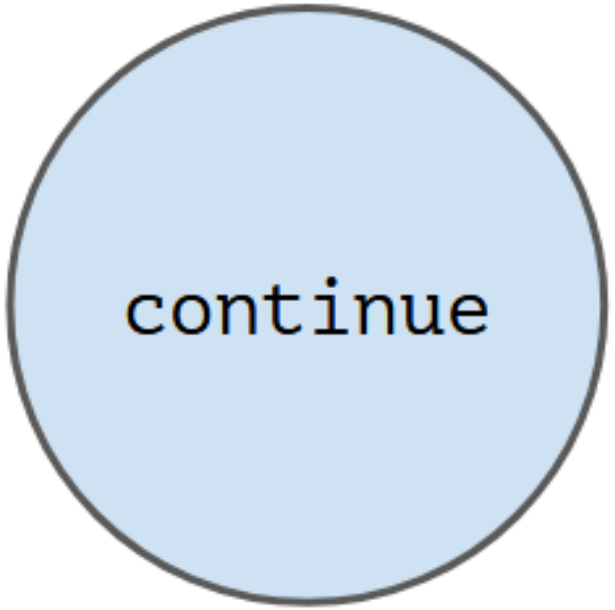
pass — заглушка, ничего не делает

Используется, когда синтаксически блок кода нужен, но команды ещё не готовы.

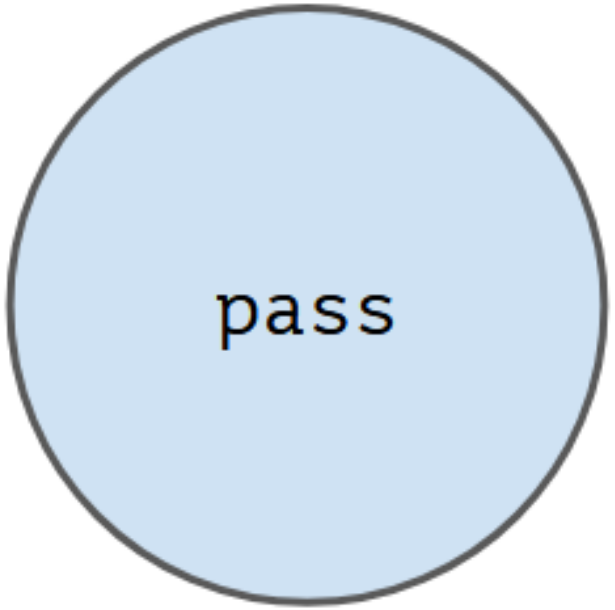
```
while False:
    pass  # код будет добавлен позже
```



Прерывает исполнение цикла



Завершает исполнение текущей итерации цикла и переходит к следующей итерации



Игнорирует условие и продолжает исполнение цикла

Вывод:

Цикл **while** — универсальный инструмент для выполнения действий до достижения нужного условия.

Важно правильно формулировать условие, изменять переменные и контролировать выполнение цикла.

3. Типовые алгоритмы с while.

Ввод данных до выполнения условия.

Алгоритм повторяет ввод, пока данные некорректны.

```
passwd = "1234"
user_passwd = input("Введите пароль: ")
while user_passwd != passwd:
    print("Ошибка!")
    user_passwd = input("Введите пароль ещё раз: ")

print("Вход выполнен!")
```

Бесконечный цикл с выходом через **break**.

Используется, когда условие выхода определяется внутри цикла.

```
passwd = "1234"
while True:
    user_passwd = input("Введите пароль: ")
    if user_passwd == passwd:
        break
print("Вход выполнен!")
```

Поиск значения, удовлетворяющего условию.

Суть алгоритма: цикл продолжается, пока не найдётся значение, подходящее под условие.

Пример: найти первое число, делящееся на 7 и большее 100

```
1  x = 101
2  while x % 7 != 0:
3      x += 1
4
5  print("Найдено:", x)
```

Контрольные вопросы:

- Что такое цикл и для чего он используется?
- Как записывается синтаксис цикла while в Python?
- В каком случае цикл while завершает своё выполнение?
- Что произойдёт, если переменная в условии цикла не изменяется?
- В чём отличие цикла while от условного оператора if?
- Приведите пример задачи, для которой логично использовать цикл while.
- Как можно прервать выполнение цикла досрочно?

Домашнее задание:

https://ru.hexlet.io/programs/python-basics-free

Материалы лекций:

https://github.com/ShViktor72/education2025

ПМ2_Прикладное_программирование

Обратная связь:

colledge20education23@gmail.com