**Теоретический материал к занятию Циклы**

В языке Python существует два типа циклов:

- счетные циклы, когда мы заранее знаем количество повторений;
- условные циклы, когда мы заранее не знаем количество повторений, но знаем условие, согласно которому нужно что-либо повторять.

Счетный цикл в языке Python (как и в большинстве других языков программирования) называется for.

Представьте, что надо написать программу, выводящую на экран слово “Привет” один раз. Очевидно, команда print(), запросто справится с поставленной задачей. А что если слово “Привет” нужно вывести 5 раз? Можно написать 5 строк кода. А если 10 раз? Неужели писать 10 одинаковых строк кода? Это достаточно частый сценарий: повторить что-либо определенное количество раз, поэтому можно предположить, что в языках программирования должны быть средства, позволяющие решить поставленную задачу проще и быстрее. И действительно, они есть, и называются циклами.
Рассмотрим код, использующий цикл for:

```
for i in range(10):
   print('Привет')
```

Обратите внимание, программа состоит из 2 строчек кода вместо 10.

Общая структура цикла for в Python имеет вид:

```
for название переменной in range(количество повторений):
    блок кода
```

Мы пишем ключевое слово for, затем название переменной цикла (об этом чуть позже), далее слова in range() и в скобках указываем количество повторений. Далее следует обязательный символ двоеточия, а затем на новой строке, с отступом - блок кода, который и будет повторяться заданное количество раз.

```
for i in range(5):
   num = int(input())
   print('Квадрат вашего числа равен:', num*num)
```

```
print('Цикл завершен')
```

Эта программа, используя цикл for, считывает 5 чисел и выводит их квадраты с поясняющим текстом.
Тело цикла состоит из двух строк кода, они выделены отступом. Не забывайте, что в Python отступы имеют большое значение!
Четвертая строка кода, печатающая текст “Цикл завершен” не содержит отступа, так как не является частью цикла.

1. однократное выполнение тела цикла называется **итерацией цикла;**
2. слово **for** пишется маленькими буквами;
3. первая строка цикла заканчивается двоеточием (:);
4. тело цикла выделяют отступом.

```
for _ in range(5):
  print('Python')
```

Мы заменили имя переменной на символ нижнего подчеркивания. Очень часто это применяют, если переменная внутри цикла не имеет какого то весомого значения в программе.

Функция range(), на самом деле позволяет нам намного больше. В примерах выше мы указывали только одно значение внутри скобок. Давайте рассмотрим другие варианты.

```
for i in range(1, 10):
  print(i)
```

Данный пример выведет числа от 1 до 10.

```
for i in range(1, 10, 2):
  print(i)
```

Данный же пример выведет каждое второе число от 1 до 10.

Синтаксис range()

```
range(начало, конец,  шаг)
```

Циклы также могут перебирать последовательности. Строка, является последовательностью. Рассмотрим пример. Пользователь вводит строку и программа выводит все символы данной строки.

```
word = input()
for char in word:
    print(char)
```

В данном случае мы не используем функцию range(). Мы условно говорим что у нас создается переменная char, внутри цикла, а каждый символ строки (элемент, если это список) перезаписывается поочередно в переменную char. Таким образом у нас выведутся поочередно все символы нашей строки (каждый на новой строке).

Также можно перебрать последовательность, используя и функцию range(), но для этого в качестве количества повторов цикла, нам надо указать длину последовательности, а обращаться к элементам последовательность по индексу (порядковому номеру, который начинается с 0). Индекс указывается в квадратных скобках.

```
word = input()
for i in range(len(word)):
    print(word[i])
```

Рассмотрим структуру цикла while:

```
while условие:
    блок кода
```

Двоеточие (:) в конце строки сообщает Python, что дальше находится блок кода, называемый телом цикла, именно он и будет повторяться, пока истинно условие.
Условие это любое логическое выражение, которое принимает два значения True/ False.

```
i = 0
while i < 10:
   print('Привет')
   i += 1
```

Мы создали переменную i с начальным значением 0, и поставили условие в цикле while: пока значение переменной i < 10. Тело цикла содержит две команды:

1. печать текста “Привет”
2. увеличение значения переменной на 1

Тем самым цикл выполнится РОВНО 10 раз, поскольку спустя 10 итераций, значение переменной i будет равно 10 и условие i < 10 становится ложным (ведь 10 = 10, а не меньше 10).

```
num = int(input())
while num != -1:
    print('Квадрат вашего числа равен:', num * num)
    num = int(input())
```

Мы поставили условие: пока считанное число не равно −1. И пока это условие истинно (равно True) выполняется тело цикла, которое в этом случае печатает квадрат считанного числа с поясняющей надписью и переходит к следующему числу (считывает следующее число).

Оператор break прерывает **ближайший** цикл for или while

```
result = 0
for i in range(10)
    num = int(input())
    if num < 0:
        break
    result += num
print(result)
```

Программа считывает 10 чисел и суммирует их, пока не обнаружит отрицательное число. В этом случае выполнение цикла прерывается командой break.

Оператор break открывает еще и возможность работы с бесконечными циклами. Бесконечным называется цикл while, не имеющий возможности завершиться — его условие всегда истинно. В Python можно создать бесконечный цикл следующим образом:

```
while True:
    print('Python!')
```

Такая программа будет печатать текст Python бесконечное количество раз (пока не будет прервана).

```
while True:
    if условие 1:  # условие для остановки цикла
        break
    ...
    if условие 2:  # еще одно условие для остановки цикла
        break
    ...
```

Следующий оператор называется оператором пропуска отдельных итерации continue. Оператор continue позволяет перейти к следующей итерации цикла for или while до завершения всех команд в теле цикла.

```
for i in range(1, 101):
    if i == 7 or i == 17 or i == 29 or i == 78:
        continue  # переходим на следующую итерацию
    print(i)
```

Циклы, аналогично условиям могут быть вложенными

```
for hours in range(24):
    for minutes in range(60):
        for seconds in range(60):
            print(hours, ':', minutes, ':', seconds)
```

- вложенный цикл выполняет все свои итерации для каждой отдельной итерации внешнего цикла;
- вложенные циклы завершают свои итерации быстрее, чем внешние;
- чтобы получить общее количество итераций вложенного цикла, надо **перемножить количество итераций всех циклов**.

Можно вкладывать друг в друга циклы for и while в произвольном порядке.