**Занятие 6. Цикл while**

**1. Цикл while**

Цикл **while (“пока”)** позволяет выполнить одну и ту же последовательность действий, пока проверяемое условие истинно. Условие записывается до тела цикла и проверяется до выполнения тела цикла. Как правило, цикл while используется, когда невозможно определить точное значение количества проходов исполнения цикла.

Синтаксис цикла while в простейшем случае выглядит так:

**while условие:**

**блок инструкций**

При выполнении цикла while сначала проверяется условие. Если оно ложно, то выполнение цикла прекращается и управление передается на следующую инструкцию после тела цикла while. Если условие истинно, то выполняется инструкция, после чего условие проверяется снова и снова выполняется инструкция. Так продолжается до тех пор, пока условие будет истинно. Как только условие станет ложно, работа цикла завершится и управление передастся следующей инструкции после цикла.

Например, следующий фрагмент программы напечатает на экран квадраты всех целых чисел от 1 до 10. Видно, что цикл while может заменять цикл for ... in range(...):

i = 1

while i <= 10:

print(i \*\* 2)

i += 1

В этом примере переменная i внутри цикла изменяется от 1 до 10. Такая переменная, значение которой меняется с каждым новым проходом цикла, называется счетчиком. Заметим, что после выполнения этого фрагмента значение переменной i будет равно 11, поскольку именно при i == 11 условие i <= 10 впервые перестанет выполняться.

Вот еще один пример использования цикла while для определения количества цифр натурального числа n:

n = int(input())

length = 0

while n > 0:

n //= 10 # это эквивалентно n = n // 10

length += 1

print(length)

В этом цикле мы отбрасываем по одной цифре числа, начиная с конца, что эквивалентно целочисленному делению на 10 (n //= 10), при этом считаем в переменной length, сколько раз это было сделано.

В языке Питон есть и другой способ решения этой задачи: length = len(str(i)).

### 2. Инструкции управления циклом

После тела цикла можно написать слово else: и после него блок операций, который будет выполнен *один раз* после окончания цикла, когда проверяемое условие станет неверно:

i = 1

while i <= 10:

print(i)

i += 1

else:

print('Цикл окончен, i =', i)

Казалось бы, никакого смысла в этом нет, ведь эту же инструкцию можно просто написать *после* окончания цикла. Смысл появляется только вместе с инструкцией break. Если во время выполнения Питон встречает инструкцию break внутри цикла, то он сразу же прекращает выполнение этого цикла и выходит из него. При этом ветка else исполняться не будет. Разумеется, инструкцию break осмыленно вызывать только внутри инструкции if, то есть она должна выполняться только при выполнении какого-то особенного условия.

Приведем пример программы, которая считывает числа до тех пор, пока не встретит отрицательное число. При появлении отрицательного числа программа завершается. В первом варианте последовательность чисел завершается числом 0 (при считывании которого надо остановиться).

a = int(input())

while a != 0:

if a < 0:

print('Встретилось отрицательное число', a)

break

a = int(input())

else:

print('Ни одного отрицательного числа не встретилось')

Во втором варианте программы сначала на вход подается количество элементов последовательности, а затем и сами элементы. В таком случае удобно воспользоваться циклом for. Цикл for также может иметь ветку else и содержать инструкции break внутри себя.

n = int(input())

for i in range(n):

a = int(input())

if a < 0:

print('Встретилось отрицательное число', a)

break

else:

print('Ни одного отрицательного числа не встретилось')

Другая инструкция управления циклом — continue (продолжение цикла). Если эта инструкция встречается где-то посередине цикла, то пропускаются все оставшиеся инструкции до конца цикла, и исполнение цикла продолжается со следующей итерации.

Если инструкции break и continue содержатся внутри нескольких вложенных циклов, то они влияют лишь на исполнение самого внутреннего цикла. Вот не самый интеллектуальный пример, который это демонстрирует:

for i in range(3):

for j in range(5):

if j > i:

break

print(i, j)

**3. Множественное присваивание**

В Питоне можно за одну инструкцию присваивания изменять значение сразу нескольких переменных. Делается это так:

a, b = 0, 1

Этот код можно записать и так:

a = 0

b = 1

Множественное присваивание удобно использовать, когда нужно обменять значения двух переменных. В обычных языках программирования без использования специальных функций это делается так:

a = 1

b = 2

tmp = a

a = b

b = tmp

print(a, b)

# 2 1

В Питоне то же действие записывается в одну строчку:

a = 1

b = 2

a, b = b, a

print(a, b)

# 2 1

**Задачи:**

### Задача «Список квадратов»

По данному целому числу N распечатайте все квадраты натуральных чисел, не превосходящие N, в порядке возрастания.

### Задача «Минимальный делитель»

Дано целое число, не меньшее 2. Выведите его наименьший натуральный делитель, отличный от 1.

### Задача «Степень двойки»

По данному натуральному числу N найдите наибольшую целую степень двойки, не превосходящую N. Выведите показатель степени и саму степень.

Операцией возведения в степень пользоваться нельзя!

### Задача «Утренняя пробежка»

В первый день спортсмен пробежал *x* километров, а затем он каждый день увеличивал пробег на 10% от предыдущего значения. По данному числу *y* определите номер дня, на который пробег спортсмена составит не менее *y* километров.

Программа получает на вход действительные числа *x* и *y* и должна вывести одно натуральное число.

### Задача «Длина последовательности»

Программа получает на вход последовательность целых неотрицательных чисел, каждое число записано в отдельной строке. Последовательность завершается числом 0, при считывании которого программа должна закончить свою работу и вывести количество членов последовательности (не считая завершающего числа 0). Числа, следующие за числом 0, считывать не нужно.

### Задача «Сумма последовательности»

Определите сумму всех элементов последовательности, завершающейся числом 0. В этой и во всех следующих задачах числа, следующие за первым нулем, учитывать не нужно.

### Задача «Среднее значение последовательности»

Определите среднее значение всех элементов последовательности, завершающейся числом 0.

### Задача «Максимум последовательности»

Последовательность состоит из натуральных чисел и завершается числом 0. Определите значение наибольшего элемента последовательности.

### Задача «Индекс максимума последовательности»

Последовательность состоит из натуральных чисел и завершается числом 0. Определите индекс наибольшего элемента последовательности. Если наибольших элементов несколько, выведите индекс первого из них. Нумерация элементов начинается с нуля.

### Задача «Количество четных элементов последовательности»

Определите количество четных элементов в последовательности, завершающейся числом 0.

### Задача «Количество элементов, которые больше предыдущего»

Последовательность состоит из натуральных чисел и завершается числом 0. Определите, сколько элементов этой последовательности больше предыдущего элемента.

### Задача «Второй максимум»

Последовательность состоит из **различных** натуральных чисел и завершается числом 0. Определите значение второго по величине элемента в этой последовательности. Гарантируется, что в последовательности есть хотя бы два элемента.

### Задача «Количество элементов, равных максимуму»

Последовательность состоит из натуральных чисел и завершается числом 0. Определите, сколько элементов этой последовательности равны ее наибольшему элементу.

### Задача «Числа Фибоначчи»

Последовательность Фибоначчи определяется так:

*φ*0 = 0,  *φ*1 = 1,  *φn* = *φn*−1 + *φn*−2.

По данному числу *n* определите *n*-е число Фибоначчи *φn*.

Эту задачу можно решать и циклом for.

### Задача «Номер числа Фибоначчи»

Дано натуральное число *A*. Определите, каким по счету числом Фибоначчи оно является, то есть выведите такое число *n*, что *φn = A*. Если *А* не является числом Фибоначчи, выведите число -1.

### Задача «Максимальное число идущих подряд равных элементов»

Дана последовательность натуральных чисел, завершающаяся числом 0. Определите, какое наибольшее число подряд идущих элементов этой последовательности равны друг другу.