# Урок 04.01. Оператор while

В нашей жизни мы очень часто прибегаем к повторениям. Например, чтобы выучить стихотворение нам необходимо его повторять до тех пора пока не выучим. Множественное повторение одних и тех же действий называется циклом, а для программиста это одна из главных основ кода.

Рассмотрим такую задачу: мы включаем компьютер и после загрузки у нас запрашивается ввод пароля. Если мы ввели неправильный пароль, то запрос ввода пароль повторится. И запрос пароля будет до тех пор пока мы не введем правильный пароль. С точки зрения условного оператора, который мы знаем такой алгоритм будет выглядеть следующим образом:

Если пароль неверный, то

 Вывести ошибку

 Запросить пароль заново

Иначе

 Войти в учетную запись

Но тут есть проблема, если пароль ввели неправильно, то нужно его запросить еще раз и еще раз, и так до тех пор пока не будет введен правильный пароль. Тогда наша программа немного видоизменится:

Пока пароль неверный выполнять

 Вывод ошибки

 Ввод пароля заново

Войти в учетную запись

И для таких программу у python’а есть специальный оператор, который с английского так и переводится «пока», а на английском – «while». С этими знаниями давайте создадим такую программу

password = input(‘Введите пароль: ’)

while password != ‘zxc’:

 print(‘Неверный пароль’)

 password = = input(‘Введите пароль еще раз: ’)

print(‘Вы вошли в учетную запись’)

Обращаю ваше внимание на отступы – они обязательны. Этот код выполняется каждый раз, когда условие выполняется. А каждое выполнение цикла называется итерацией.

Часто так получается, что цикл должен выполняться заранее неизвестное число раз. И давайте рассмотрим такую задачку. У нас есть определенная сумма денег, и мы попали в большой торговый центр, в котором можно что-то купить, поесть, сходить в игрой клуб и так далее. Но денег у нас ограничено, и они могут кончиться, и если мы увлечемся, то можем не заметить, как они у нас кончились. Для того, чтобы это не стало для нас неожиданностью напишем программу, которая выводит сообщение, если у нас становится мало денег. Такая программа будет выглядеть так:

balance = int(input(‘Сколько есть денег: ’))

while balance > 500:

 cost = int(input(‘Сколько потратили: ’))

 balance -= cost

print(‘На счету осталось мало денег!’)

print(‘Баланс равен:’, balance)

Запустим программу и убедимся в её работе. И может произойти такая ситуация, когда наш цикл не сработает, если условие не сработало сразу. Такое срабатывание может быть если в начале ввести число меньше 500, например, 300.

## Цикл while, его прерывание, оператор break

Мы уже познакомились с циклом *while*, итерация, тело цикла, но знаем ли мы как работает цикл на самом деле и можно ли выйти из него раньше, чем требуется?

Чтобы найти ответы представим, что сегодня солнечный день, и мы решили побегать. Сначала мы бежим по прямой дороге и никуда не сворачиваем, и через некоторое время мы прибегаем к старту на стадионе и начинаем наворачивать круги. Мы формулируем свое пребывание на стадионе такое: пока температура больше 15 мы будем бегать. Думаю, вы догадались, что тут используется цикл *while*, у которого условие «температура больше 15», а тело цикло – «сделать шаг». И вот мы пробегаем два шага, т.е. делаем несколько команд тела цикла, внезапно похолодало, и полил дождь. Внимание вопрос знатокам: мы сразу покинем стадион или продолжим бежать. В реальной жизни мы, конечно же, прервались и пошли домой, однако мы с вами программисты и должны думать как программисты, а именно: бегун будет бегать до тех пор, пока выполняется условие цикла – температура выше 15 градусов.

Давайте напишем такую программу. Задание будет звучать так: напишите программу, которая запрашивает начальную температуру воздуха, и определяет расстояние, которое пробежал бегун, если температура падает на один градус каждые полкруга, при этом бегун узнает о температуре только в начале нового круга.

temp = int(input('Введите начальное значение температуры: '))

distance = 0

circle\_distance = 400

while temp > 15:

 distance += circle\_distance / 2

 temp -= 1

 print('Пробежал дистанцию: ', distance)

 distance += circle\_distance / 2

 temp -= 1

 print('Пробежал дистанцию: ', distance)

 print('Температура: ', temp)

print('Закончить бегать')

print('Итоговая дистанция: ', distance)

print('Температура: ', temp)

Теперь запустим программу и введем число 17. Хорошо. Пробежали 400м и температура 15, и мы ушли. А теперь еще раз запустим и введем уже 18 градусов. И что же мы видим: мы пробежали 800м, а температура, при которой мы закончили – 14 градусов. Конечно, это соответствует условию задачи, но не хотелось бы закончить бегать пораньше. Для этого наша программа должна была прерваться сразу, как только температура опустилась до 15 градусов. Давайте посмотрим в отладчике как наша программа работала.

Мы увидели, что причину нашей проблемы. Чтобы исправить нашу программу нам надо прерваться сразу после понижения температуры. В python для прерывания оператора цикла есть специальная команда ***break***. Исправим нашу программу со следующим условием: после уменьшения температуры необходимо проверить если температура опустилась до 15 или ниже градусов, то прервать выполнение цикла.

temp = int(input('Введите начальное значение температуры: '))

distance = 0

circle\_distance = 400

while temp > 15:

 distance += circle\_distance / 2

 temp -= 1

 if temp < 15:

 break

 print('Пробежал дистанцию: ', distance)

 distance += circle\_distance / 2

 temp -= 1

 print('Пробежал дистанцию: ', distance)

 print('Температура: ', temp)

print('Закончить бегать')

print('Итоговая дистанция: ', distance)

print('Температура: ', temp)

Теперь если мы введем 18 градусов, то наш бегун закончит бегать, сразу после понижения температуры. Поздравляю всех, мы спасли замерзающего бегуна.

Давайте для закрепления разберем ещё одну задачу.

Нам нужно расшифровать определенный код в виде одного большого числа. Для этого нам нужно посчитать сумму цифр справа налево. Для удобства расшифровки сообщения в числе может встречаться цифра 5, она отделяет одну часть сообщения от другой, как пробел между словами. И теперь, когда мы встречаем цифру 5, то выводим сообщение «обнаружен разрыв», заканчиваем считать сумму, и выводим её на экран. Напишем программу:

number = int(input('Введите число: '))

sum = 0

while number != 0:

 last\_number = number % 10

 sum += last\_number

 if last\_number == 5:

 print('Обнаружен разрыв!')

 break

print('Сумма:', summ)

Давайте проверим нашу программу. Введем 145 – всё хорошо. Еще раз, и уже введем 123. Так, что-то пошло не так, наша программа «зависла». Давайте попробуем еще один прием проверки программы – добавим в программу команды *print*, которые будут печатать нам промежуточные результаты вычислений.

number = int(input('Введите число: '))

summ = 0

while number != 0:

 last\_number = number % 10

 summ += last\_number

 print(number, last\_number, summ)

 if last\_number == 5:

 print('Обнаружен разрыв!')

 break

print('Сумма:', summ)

Запустим, и увидим, что не так с нашей программой. Наш цикл оказался бесконечным. Теперь нам нужно это исправить. А именно, нам нужно уменьшать наше число *number*.

number = int(input('Введите число: '))

summ = 0

while number != 0:

 last\_number = number % 10

 summ += last\_number

 print(number, last\_number, summ)

 if last\_number == 5:

 print('Обнаружен разрыв!')

 break

 number //= 10

print('Сумма:', summ)

Далее необходимо выполнить следующие задания самостоятельно.