Лекция №2 Циклы

Прежде чем мы начнем циклы, надо бы посмотреть и разобраться с одной функцией, а именно range().

В общем виде если напечатать print(range(5)), то ответ будет объект типа range и в консоли будет следующее range(0, 5), а не 0,1,2,3,4. Если мы хотим получить именно список значений необходимо обернуть range в list() и получить следующую конструкцию list(range(0, 5)), при выводе сего счастья мы получим 0,1,2,3,4. Но как-то меня немного понесло вперед, надо бы немного рассказать про циферки внутри и можно бежать дальше.

И так синтаксис функции range имеет три варианта исполнения и выглядит так:

range(x), где х-1 это последние обрабатываемое значение

range(x,y), где х это начальный элемент, а у-1 конечный элемент

range(x,y,z), где х и у такие же как и в варианте 2, но при этом еще добавляется шаг изменения z

Теперь давайте немного примеров и пойдем к циклам:

print(list(range(5))) #[0,1,2,3,4]

print(list(range(2,7))) #[2,3,4,5,6]

print(list(range(3,11,4))) #[3,7]

*Ну а теперь циклы*

For \_\_ in \_\_(в течении условия)

Самый просто пример применения цикла for in это следующее конструкция, она есть во всех языках и даже смысл один и тот же, но у for in есть другие прикольные возможности, но о них не в этот раз, а когда я расскажу массивы. И так давайте посмотрим на вот такую конструкцию:

for i in range(5):

print(i)

вывод по итогу, будет следующий:

0 1 2 3 4

В общем-то нам этого кажется даже будет достаточно, вместо простого range(x) можно использовать все вышеописанные варианты. Зачем? Ну вам виднее.

While (пока условие)

Цикл while также используется для повторения частей кода, но вместо зацикливания на какое-то количество раз, он выполняет работу до тех пор, пока выполняется условие. Давайте взглянем на простой пример со следующим условием:

Выводим х прибавляя каждый раз 0,5 пока он не станет больше 3, давайте напишем, что нам мешает). Самое главное не забудьте определить переменную которой мы будем пользоваться, а то питон будет ругаться ошибками по типу (name ‘x’ is not defined).

x=0

while x<3:

print(x)

x+=0.5

Готово! Запустим сей прекрасный код мы увидим следующую последовательность

0 0.5 1.0 1.5 2.0 2.5

Если бы мы написали сначала плюс потом вывод, то ноля бы не было, а в конце добавилось бы еще 3.0

Ну вот и вроде всё что хотелось по этому поводу рассказать, сегодня лекция много меньше, чем предыдущая, но практики будет больше))).