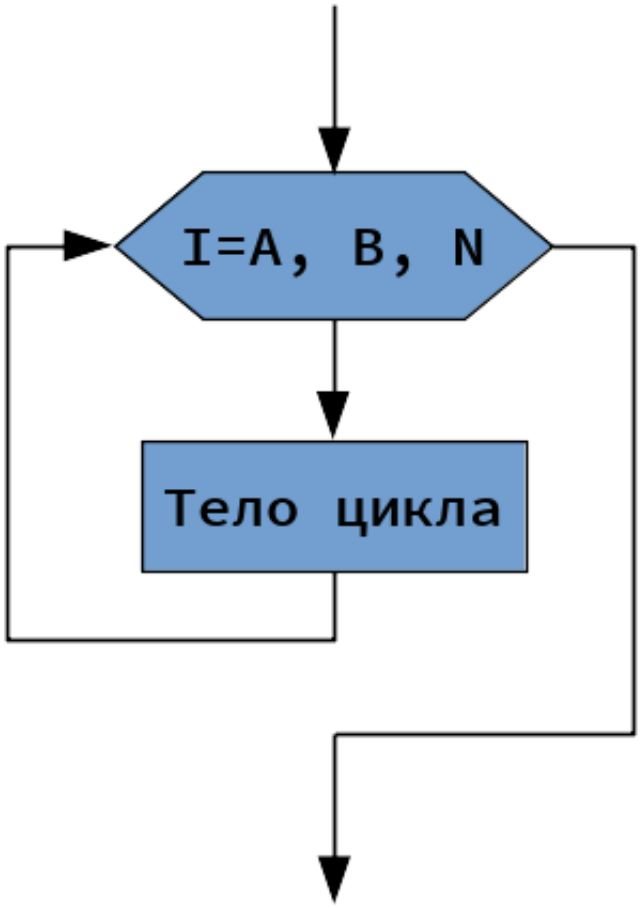
6. Циклы в Python, работа и устройство цикла for, типичное применение range и enumerate в цикле for.

Цикл for используется в тех случаях, когда нам необходимо повторить какое-либо действие определенное количество раз (в отличии от цикла while это число итераций известно)
Пример:

```
i = 0
for i in range(5):
    i += 1
print(i) #выдаст 5
```



Также в цикле for возможно обращаться к элементам коллекции двумя способами:
№1 - по индексу (не для словарей):

```
mylist = [6,3,8,1]
for i in range(len(mylist)):
print(mylist[i])
```

№2 - автоматически представляя каждый элемент как переменную

```
mylist = [6,3,8,1]
for e in mylist:
    print(e)
```

Range() позволяет генерировать ряд чисел в рамках заданного диапазона. В зависимости от того, как много аргументов передаётся функции, можно решить, где этот ряд чисел начнется и закончится, а также насколько велика разница будет между двумя числами.
Используется чаще всего для цикла for. При необходимости можно конвертировать в tuple/list/set

```
range(3) # генерация чисел 0, 1, 2
range(1,6) # генерация чисел 1, 2, 3, 4, 5
range(1,10,2) # генерация чисел 1, 3, 5, 7, 9
```

Встроенная в Python функция enumerate() применяется для итерируемых коллекций (строки, списки, словари и др.) и создает объект, который генерирует кортежи, состоящие из двух элементов - индекса элемента и самого элемента.

```
>>> a = [10, 20, 30, 40]
>>> for i in enumerate(a):
...     print(i)
...
(0, 10)
(1, 20)
(2, 30)
(3, 40)
```

Выход из цикла можно осуществить при помощи инструкции break
Также можно пропустить текущую итерацию при помощи инструкции continue

У циклах for/while по синтаксису также есть else. Else проверяет, был ли произведен выход из цикла инструкцией break, или же "естественным" образом. Блок инструкций внутри else выполнится только в том случае, если выход из цикла произошел без помощи break.

```
for i in "hello world":
    if i == "a":
        break
else:
    print("буквы a в строке нет")
```