**Условные ветвления: if, elif, else**

x = 10  
  
if x == 10:  
 print(“1 case!”) # первая “ветка”: выполнится, если 1 условие - True

else:

print(“2 case!”) # вторая “ветка”: выполнится, если 1 условие – False  
  
# напечатает “1 case”  
  
  
if x < 10:  
 print(“1 case!”) # первая “ветка”: выполнится, если 1 условие - True

elif x > 10:

print(“2 case!”) # вторая “ветка”: выполнится, если 2 условие - True

else:  
 print(“3 case!”) # вторая “ветка”: выполнится, если условия 1 и 2 - False  
  
# напечатает“3case”

- Поток выполнения программы входит только в одну “ветку”  
- выражений **elif** может быть сколь угодно много  
- **else** выполняется тогда и только тогда, когда все условия, описанные в **if/elif** возвращают False

- Условные ветвления могут быть вложенными  
  
if x <= 10:  
 print(x)

if x == 10:  
 ...

**Логические операторы and, or, not**

1. **and**   
  
Если хотя бы один из операндов выражения с **and** равен False, то резуьтат всего выражения будет False

x = True

y = False

z = False

>> x and y and z  
False

2. **or**   
  
Если хотя бы один из операндов выражения с **or** равен True, то резуьтат всего выражения будет True

>> x or y or z  
True

3. **not**

оператор not инвертирует значение bool-типа, превращая False в True и наоборот  
  
>>> not (x and y and z)

True

**Циклы**

1. Цикл for – используется для итерирования по какой-либо конечной последовательности (например, список или строка)

my\_list = [1, 2, 3, 4, 5]

# итерирование по элементам самого списка my\_list

for element in my\_list:  
 print(element)

# итерирование по счетчику, величина которого равна длине списка – 1,

и получение каждого элемента по индексу

for x in range(len(my\_list)):  
 print(my\_list[x])  
  
# если у нас есть два списка равной длины, и мы хотим, например, суммировать элементы обоих списков, стоящих на одинаковых позициях  
  
my\_list2 = [10, 11, 12, 13, 14, 15]

for idx, element in enumerate(my\_list):  
 print(element + my\_list2[idx])  
  
\* функция **enumerate** создает пары **индекс элемента – сам элемент**

**2. Цикл while**

Позволяет реализовать бесконечный цикл, который может прерваться в случае наступления определенного события.

# Обход списка в цикле while

my\_list = [1, 2, 3, 4, 5]

counter = -1

condition = True

while condition:

counter += 1

if counter >= len(my\_list) - 1:

condition = False

print(my\_list[counter])

# ИЛИ проще

while True:

counter += 1

if counter > len(my\_list) - 1:

break # оператор break немедленно прерывает цикл

print(my\_list[counter])