**Управляющие структуры.**

В этой главе рассматриваются управляющие структуры. В языке Python условное ветвление реализуется с помощью оператора if, а циклическая обработка с помощью операторов while и for … in.

**Условное ветвления. Оператор if**

Ход выполнения программы может быть линейным, то есть таким, когда инструкции выполняются друг за другом, начиная с первой и заканчивая последней. Однако, на практике требуется пропускать участки кода или изменять значения переменных в зависимости от некоторых условий. Иными словами, в программе может присутствовать ветвление, которое реализуется в языке Python оператором if.

В самом простом случаи синтаксис оператора if выглядит так:

If выражение:

инструкция\_1

инструкция\_2

….

инструкция\_n

После оператора if записывается логическое выражение. Если это выражение истинно, то выполняются инструкции, определяемые данным оператором. После логического выражения требуется поставить двоеточие. Блок кода, который необходимо выполнить, в случае истинности выражения, отделяется четырьмя пробелами слева.

Примеры:

# Пример 1  
# Выведет на консоль: Hello World  
if 1:  
 print("Hello World")  
  
# Пример 2  
# Выведет на консоль: Hello Python  
if 1 == True:  
 print('Hello Python')  
  
  
# Пример 3  
a = 10  
b = 15  
  
# Ничего не выведет  
if a > b:  
 print('Hello')

Бывают случаи, когда необходимо предусмотреть альтернативный вариант выполнения программы. Т.е. при истинном условии нужно выполнить один набор инструкций, при ложном – другой. Для этого используется конструкция if – else. Общий вид конструкции if – else в языке Python имеет вид:

if выражение:

набор инструкций\_1

else:

набор инструкций\_2

Синтаксис оператора else повторяет синтаксис оператора if, за исключением того, что между ключевым словом else и двоеточием не требуется записывать логическое выражение.

Примеры:

# Пример 1  
# Выведет на консоль: Hello World  
if 1:  
 print("Hello World")  
else:  
 print("Python is cool")  
# Пример 2  
# Выведет на консоль: Hello Python  
if 1 == True:  
 print('Hello Python')  
else:  
 print("Python is cool")  
  
  
# Пример 3  
a = 10  
b = 15  
  
# Выведет на консоль: Python is cool  
if a > b:  
 print('Hello')  
else:  
 print("Python is cool")

Для реализации выбора из нескольких альтернативных вариантов используется конструкция if – elif – else, где оператор else является не обязательным. Общий вид конструкции if – elif – else в языке Python имеет вид:

if выражение\_1:

набор инструкций\_1

elif выражение\_2:

набор инструкций\_2

elif выражение\_3:

набор инструкций\_3

…

else:

набор инструкций\_n

Синтаксис оператора elif полностью повторяет синтаксис оператора if.

Примеры:

# Пример 1  
# Выведет на консоль: 3  
lst = [1, 2, 3]  
if not lst:  
 print("1")  
elif len(lst) > 10:  
 print("2")  
else:  
 print("3")  
  
# Пример 2  
a = 10  
b = 15  
  
# Ничего не выведет  
if a > b:  
 print('1')  
elif a == b:  
 print('2')  
elif a+2 >= b:  
 print('3')

**Циклы**

Цикл —это управляющая конструкция, предназначенная для организации многократного исполнения набора инструкций. В языке Python есть два оператора циклов – while и for … in.

**Циклы while**

Оператор цикла while выполняет указанный набор инструкций до тех пор, пока условие цикла истинно. Синтаксис оператора while полностью повторяет синтаксис оператора if и имеет вид:

while выражение:

набор инструкций

Истинность логического выражения проверяется в начале цикла while и после каждого выполнения всего набора инструкций. Выполнение набора инструкций внутри цикла называется – итерацией.

Примеры:

# Пример 1  
a = 10  
while a > 0:  
 print(a, end=' ')  
 a -= 1  
# На консоль выведет: 10 9 8 7 6 5 4 3 2 1  
print()  
  
# Пример 2  
lst = []  
i = 0  
while len(lst) < 3:  
 lst.append(i)  
 i += 1  
print(lst)  
# На консоль выведет: [0, 1, 2]

**Циклы for**

Оператор цикла for в языке Python предназначен для выполнения набора инструкций заданное количество раз, которое задано набором элементов в коллекции. Синтаксис оператора for в общем случаи имеет вид:

for элемент in коллекция:

набор инструкций

Для того, чтобы цикл for выполнился N раз требуется после оператора in вызвать встроенную функцию range(N), которая возвращает список из N элементов с 0 до N-1.

Примеры:

# Пример 1  
for i in range(5):  
 print(i, end=' ')  
# Выведет на консоль: 0 1 2 3 4

Для перебора всех элементов коллекции в цикле for, требуется после оператора in указать имя коллекции:

# Пример 1  
lst = list('python')  
for ch in lst:  
 print(ch, end=' ')  
# Выведет на консоль: p y t h o n

Элементы коллекции в цикле for можно изменять, но при этом изменения, применяемые к перебираемым элементам, например, присваивание, не отразятся на коллекции. Чтобы изменить элемент коллекции требуется обратиться к ней по индексу (если коллекция поддерживает оператор []).

Примеры:

# Пример 1  
lst = list('python')  
for ch in lst:  
 ch += '10'  
print(lst)  
# Выведет на консоль: ['p', 'y', 't', 'h', 'o', 'n']  
  
# Пример 2  
lst = list('python')  
for i in range(len(lst)):  
 lst[i] += "!"  
print(lst)  
# Выведет на консоль: ['p!', 'y!', 't!', 'h!', 'o!', 'n!']

**Оператор break**

В языке Python для досрочного прерывания работы цикла используется оператор break. Важно отметить, что оператор break применяется для ближайшего описанного выше оператора цикла.

Примеры:

# Пример 1  
a = 10  
while a > 1:  
 print(a, end=' ')  
 if a == 5:  
 break  
 a -= 1  
# Выведет на консоль: 10 9 8 7 6 5  
print()  
# Пример 2  
for i in range(10):  
 print(i)  
 break  
# Выведет на консоль: 0  
  
# Пример 3  
for i in range(10):  
 while i > 0:  
 print(i, end=' ')  
 i -= 1  
 break  
# Выведет на консоль: 1 2 3 4 5 6 7 8 9