Занятие 23. Циклы и итерации. Оператор for

План занятия

1. Определение оператора цикла 2
2. Синтаксис оператора цикла for 2
3. Использование оператора else с циклом while и for 2
4. Прерывание цикла 3
5. Практическая работа 4

1. Определение оператора цикла

Циклы — это инструкции, выполняющие одну и ту же последовательность действий, пока действует заданное условие.

2. Синтаксис оператора цикла for

Конструкция **for** на языке Python описывается следующей схемой:

```
for i in s:
    инструкции
```

Здесь i – переменная цикла Инструкция for выполняет итерации по всем элементам s, пока не исчерпаются доступные элементы.

На каждой итерации она принимает новое значение из объекта s. Область видимости переменной цикла не ограничивается инструкцией for. Если перед инструкцией for была объявлена переменная с тем же именем, ее предыдущее значение будет затерто. Более того, по окончании цикла эта переменная будет хранить последнее значение, полученное в цикле.

Инструкция for может работать с любыми объектами, поддерживающими итерации. К числу этих объектов относятся встроенные типы последовательностей, такие как списки, кортежи и строки, а также любые другие объекты, реализующие протокол итераторов.

```
s = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
i = 0
for element in s:
    s[i] = element + 2
    print s[i]
    i = i + 1
```

3. Использование оператора else с циклом while и for

В Python допустимо использование оператора else, связанного с циклом while:

– если оператор else используется с циклом for – он будет выполнен, когда список для цикла будет завершен;

– при использовании else вместе с циклом while – он будет использован, когда условие примет значение ложь (false).

Основное назначение инструкции `else` в циклах заключается в том, чтобы избежать установки или проверки какого-либо флага или условия, когда работа цикла прерывается преждевременно. В следующем примере демонстрируется использование оператора else с циклом while, который выводит сообщение до тех пор, пока счётчик меньше 5, после чего будет выполнение действие оператора else:

```
count = 0
while count < 5:
  print count, " is  less than 5"
  count  += 1
else:
  print count, " is not less than 5"
```

4. Прерывание цикла

Прервать цикл можно с помощью инструкции break. Например, в следующем фрагменте выполняется чтение элементов списка, пока не будет встречен 0:

```
d=[1,11,45,25,34,75,12,0, 101,100,23,45]
for c in d:
   if c == 0: break
print c
```

Перейти к следующей итерации (пропустив оставшиеся инструкции в

теле цикла) можно с помощью инструкции continue. Эта инструкция используется не часто, но иногда она может быть полезна, – когда оформление еще одного уровня вложенности программного кода в условной инструкции могло бы привести к нежелательному усложнению программы. Например, следующий цикл пропускает все 0 в списке:

```
d=[1,11,45,25,34,0,75,12,0, 101,100,0,23,0,0,45]
for c in d:
   if c == 0: continue
   print c
```

Инструкции `break` и `continue` воздействуют только на самый внутренний цикл. Если необходимо организовать выход из глубоко вложенной иерархии циклов, можно воспользоваться исключением.

5. Практическая работа

1. Создайте список, состоящий из четырех строк. Затем, с помощью цикла for, выведите строки поочередно на экран.

2. Измените предыдущую программу так, чтобы в конце каждой буквы строки добавлялось тире. (Подсказка: цикл for может быть вложен в другой цикл.)

3. Создайте список, содержащий элементы целочисленного типа, затем с помощью цикла перебора измените тип данных элементов на числа с плавающей точкой. (Подсказка: используйте встроенную функцию float().)