Control structures

if / elif / else

if/elif/else

Конструкция if/elif/else дает возможность выполнять различные действия в зависимости от условий.

В этой конструкции только if является обязательным, elif и else опциональны:

- Проверка if всегда идет первой. После оператора if должен быть какой-то тест, который в случае результата True, приводит к выполнению действий в этом блоке.
- С помощью elif можно сделать несколько разветвлений. То есть, проверять входящие данные на разные условия.
- Блок else выполняется в том случае, если ни одно из условий if или elif не было истинным.

```
In [1]: a = 9

In [2]: if a == 0:
   ...:     print 'a равно 0'
   ...: elif a < 10:
   ...:     print 'a меньше 10'
   ...: else:
   ...:     print 'a больше 10'
   ...:
a меньше 10
```

if/elif/else

В Python:

- True (истина)
  - любое ненулевое число
  - любая ненулевая строка
  - любой ненулевой объект
- False (ложь)
  - 0
  - None
  - пустая строка
  - пустой объект

Иногда удобнее использовать тернарный оператор (ternary expressions), а не развернутую форму:

```
In [25]: s = [1, 2, 3, 4]

In [26]: result = True if len(s) > 5 else False

In [27]: result
Out[27]: False
```

Операторы сравнения, которые могут использоваться в тестах

```
In [3]: 5 > 6
Out[3]: False

In [4]: 5 > 2
Out[4]: True

In [5]: 5 < 2
Out[5]: False

In [6]: 5 == 2
Out[6]: False

In [7]: 5 == 5
Out[7]: True

In [8]: 5 >= 5
Out[8]: True

In [9]: 5 <= 10
Out[9]: True

In [9]: 5 != 10
Out[9]: True
```

Оператор **in**

Оператор in позволяет выполнять проверку на наличие элемента в списке или подстроки в строке:

```
In [8]: 'ello' in 'Hello'
Out[8]: True

In [10]: vlan = [10, 20, 30, 40]

In [11]: 10 in vlan
Out[11]: True

In [12]: 50 in vlan
Out[12]: False
```

При использовании со словарями условие in выполняет проверку по ключам словаря:

```
In [15]: r1 = {
    ....:  'IOS': '15.4',
    ....:  'IP': '10.255.0.1',
    ....:  'hostname': 'london_r1',
    ....:  'location': '21 New Globe Walk',
    ....:  'model': '4451',
    ....:  'vendor': 'Cisco'}

In [16]: 'IOS' in r1
Out[16]: True

In [17]: '4451' in r1
Out[17]: False
```

Операторы **and, or, not**

```
In [18]: vlan = [10, 20, 30, 40]

In [19]: 5 > 1 and 10 in vlan
Out[19]: True

In [20]: 5 > 1 and 50 in vlan
Out[20]: False

In [21]: 5 > 1 or 50 in vlan
Out[21]: True

In [22]: not 5 > 10
Out[22]: True

In [23]: 50 not in vlan
Out[23]: True
```

Пример использования конструкции if/elif/else

```
# -*- coding: utf-8 -*-

username = raw_input('Введите имя пользователя: ' )
password = raw_input('Введите пароль: ' )

if len(password) < 8:
    print 'Пароль слишком короткий'
elif username in password:
    print 'Пароль содержит имя пользователя'
else:
    print 'Пароль для пользователя %s установлен' % username
```

```
(test)nata@lab1:~$ python file7.py
Введите имя пользователя: nata
Введите пароль: 123nata123
Пароль содержит имя пользователя

(test)nata@lab1:~$ python file7.py
Введите имя пользователя: nata
Введите пароль: 1234
Пароль слишком короткий

(test)nata@lab1:~$ python file7.py
Введите имя пользователя: nata
Введите пароль: 123456789
Пароль для пользователя nata установлен
```

Цикл for

Цикл for

Цикл for проходится по указанной последовательности и выполняет действия, которые указаны в блоке for.
Примеры последовательностей, по которым может проходиться цикл for:

- строка
- список
- словарь
- функция range() или итератор xrange()
- любой другой итератор (например, sorted(), enumerate())

```
In [2]: for i in xrange(10):
   ...:     print 'interface FastEthernet0/' + str(i)
   ...:
interface FastEthernet0/0
interface FastEthernet0/1
interface FastEthernet0/2
interface FastEthernet0/3
interface FastEthernet0/4
interface FastEthernet0/5
interface FastEthernet0/6
interface FastEthernet0/7
interface FastEthernet0/8
interface FastEthernet0/9
```

Цикл for

```
In [3]: vlans = [10, 20, 30, 40, 100]

In [4]: for vlan in vlans:
   ...:     print 'vlan %d' % vlan
   ...:     print ' name VLAN_%d' % vlan
   ...:
vlan 10
 name VLAN_10
vlan 20
 name VLAN_20
vlan 30
 name VLAN_30
vlan 40
 name VLAN_40
vlan 100
 name VLAN_100

In [5]: vlan_name = {10:'Mngmt', 20:'Marketing', 30:'IT', 40:'Voice'}

In [6]: for vlan in vlan_name:
   ....:     print 'vlan %d' % vlan
   ....:     print ' name %s' % vlan_name[vlan]
   ....:
vlan 40
 name Voice
vlan 10
 name Mngmt
vlan 20
 name Marketing
vlan 30
 name IT
```

Пример вложенных циклов for

```
In [7]: commands = ['switchport mode access',
                    'spanning-tree portfast',
                    'spanning-tree bpduguard enable']

In [8]: fast_int = ['0/1','0/3','0/4','0/7','0/9','0/10','0/11']

In [9]: for int in fast_int:
   ...:     print 'interface FastEthernet ' + int
   ...:     for command in commands:
   ...:         print ' %s' % command
   ...:

interface FastEthernet 0/1
 switchport mode access
 spanning-tree portfast
 spanning-tree bpduguard enable
interface FastEthernet 0/3
 switchport mode access
 spanning-tree portfast
 spanning-tree bpduguard enable
interface FastEthernet 0/4
 switchport mode access
 spanning-tree portfast
 spanning-tree bpduguard enable
...
```

Совмещение for и if

```
In [10]: access_template = ['switchport mode access', 'switchport access vlan',
'spanning-tree portfast', 'spanning-tree bpduguard enable']

In [11]: fast_int = {'access':
{'0/12':'10','0/14':'11','0/16':'17','0/17':'150'}}

In [12]: for int in fast_int['access']:
   ....:     print 'interface FastEthernet' + int
   ....:     for command in access_template:
   ....:         if command.endswith('access vlan'):
   ....:             print ' %s %s' % (command, fast_int['access'][int])
   ....:         else:
   ....:             print ' %s' % command
   ....:
interface FastEthernet0/12
 switchport mode access
 switchport access vlan 10
 spanning-tree portfast
 spanning-tree bpduguard enable
interface FastEthernet0/14
 switchport mode access
 switchport access vlan 11
 spanning-tree portfast
 spanning-tree bpduguard enable
interface FastEthernet0/16
 switchport mode access
 switchport access vlan 17
 spanning-tree portfast
 spanning-tree bpduguard enable
```

Итератор enumerate()

Иногда, при переборе объектов в цикле for, нужно не только получить сам объект, но и его порядковый номер. Это можно сделать, создав дополнительную переменную, которая будет расти на единицу с каждым прохождением цикла.
Но, гораздо удобнее это делать с помощью итератора enumerate().

```
In [1]: list1 = ['str1', 'str2', 'str3']

In [2]: for position, string in enumerate(list1):
   ...:     print position, string
   ...:
0 str1
1 str2
2 str3
```

enumerate() умеет считать не только с нуля, но и с любого значение, которое ему указали после объекта:

```
In [1]: list1 = ['str1', 'str2', 'str3']

In [2]: for position, string in enumerate(list1, 100):
   ...:     print position, string
   ...:
100 str1
101 str2
102 str3
```

Цикл while

Цикл while

В цикле while мы пишем выражение, как и в if. И, если это выражение истина, то действия внутри этого блока выполняются. Но, в отличии от if, while возвращается после выполнения в начало цикла.

При использовании цикла while главная опасность заключается в том, что он может выполняться бесконечно. Конечно, условно бесконечно, но все же стоит, при использовании циклов while, обращать внимание на то, будет ли такое состояние, при котором условие в цикла перейдет в состояние ложь.

```
In [1]: a = 5

In [2]: while a > 0:
   ...:     print a
   ...:     a -= 1
   ...:
5
4
3
2
1
```

Пример использования while

```
# -*- coding: utf-8 -*-

username = raw_input('Введите имя пользователя: ' )
password = raw_input('Введите пароль: ' )

pass_OK = False

while not pass_OK:
    if len(password) < 8:
        print 'Пароль слишком короткий\n'
        password = raw_input('Введите пароль еще раз: ' )
    elif username in password:
        print 'Пароль содержит имя пользователя\n'
        password = raw_input('Введите пароль еще раз: ' )
    else:
        print 'Пароль для пользователя %s установлен' % username
        pass_OK = True
```

```
(test)nata@lab1:~$ python file9.py
Введите имя пользователя: nata
Введите пароль: nata
Пароль слишком короткий

Введите пароль еще раз: natanata
Пароль содержит имя пользователя

Введите пароль еще раз: 123345345345
Пароль для пользователя nata установлен
```

Работа с файлами

Открытие файла

Прежде чем мы сможем работать с файлом, его надо открыть.
Для открытия файлов, чаще всего, используется функция open():

```
file = open('file_name.txt', 'r')
```

В функции open():

- 'file_name.txt' – это имя файла
- тут может указывать не только имя, но и путь (абсолютный или относительный)
- 'r' – режим открытия файла

Функция open() создает объект file, к которому потом мы сможем применять различные методы, для работы с ним.

Режимы открытия файлов:

- r – открыть файл только для чтения (значение по умолчанию)
- w – открыть файл для записи
  - если файл существет, то его содержимое удаляется
  - если файл не существует, то создается новый
- a – открыть файл для дополнение записи. Данные добавляются в конец файла

Чтение файла

В Python есть несколько методов чтения файла:

- read() - этот метод считывает содержимое файла в строку
- readline() - считывает файл построчно
- readlines() - этот метод считывает строки файла и создает список из строк

```
In [1]: f = open('test.txt')

In [2]: f.read()
Out[2]: 'This is line 1\nanother line\none more line\n'

In [3]: f.read()
Out[3]: ''
```

Метод readline():

```
In [4]: f = open('test.txt')

In [5]: f.readline()
Out[5]: 'This is line 1\n'

In [6]: f.readline()
Out[6]: 'another line\n'
```

Чтение файла

Метод readlines():

```
In [9]: f = open('test.txt')

In [10]: f.readlines()
Out[10]: ['This is line 1\n', 'another line\n', 'one more line\n']
```

Метод readlines() и удаление спецсимволов:

```
In [11]: f = open('test.txt')

In [12]: [line.strip() for line in f.readlines()]
Out[12]: ['This is line 1', 'another line', 'one more line']
```

Запись файлов

При записи очень важно определиться с режимом открытия файла (чтобы случайно его не удалить):

- append - добавить строки в существующий файл
- write - перезаписать файл
- оба режима создают файл, если он не существует

Для записи в файл используются такие методы:

- write() - записать в файл одну строку
- writelines() - позволяет передавать в качестве аргумента список строк

Обратите внимание, что после выполнения операций с файлом, обязательно надо закрыть файл!

Запись файлов

Метод write()

```
In [1]: f = open('test2.txt', 'a')
In [2]: f.write('line1\n')
In [3]: f.write('line2\n')
In [4]: f.write('line3\n')
In [5]: f.close()

In [6]: cat test2.txt
line1
line2
line3
```

Метод writelines()

```
In [8]: list1 = ['line4\n','line5\n','line6\n']
In [9]: f = open('test2.txt', 'w')
In [10]: f.writelines(list1)
In [11]: f.close()

In [12]: cat test2.txt
line4
line5
line6
```

Конструкция with

Конструкция with ... as

В Python существует более удобный способ работы с файлами, который гарантирует закрытие файла автоматически: конструкция with ... as:

```
In [13]: with open('test2.txt') as f:
    ....:     for line in f:
    ....:         print line
    ....:
line4
line5
line6

In [14]: f.closed
Out[14]: True
```

Python для сетевых инженеров

Автор курса: Наташа Самойленко
nataliya.samoylenko@gmail.com