PYTHON ДЛЯ СЕТЕВЫХ ИНЖЕНЕРОВ

WELCOME TO ПРОДЛЕНКА :)

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

СПЕЦИАЛЬНЫЕ МЕТОДЫ

`__str__`

`__str__`

Метод `__str__` отвечает за строковое отображение информации об объекте. Он вызывается при использовании str и print:

```
class Switch:
    def __init__(self, hostname, model):
        self.hostname = hostname
        self.model = model

    def __str__(self):
        return 'Swicth: {}'.format(self.hostname)

In [5]: sw1 = Switch('sw1', 'Cisco 3850')

In [6]: print(sw1)
Swicth: sw1

In [7]: str(sw1)
Out[7]: 'Swicth: sw1'
```

`__del__`

`__del__`

Метод `__del__` вызывается перед удалением объекта. В этот метод, как правило выносятся такие действия как, например, корректное завершение сессии.

```
class Switch:
    def __init__(self, hostname, model):
        self.hostname = hostname
        self.model = model

    def __str__(self):
        return 'Swicth: {}'.format(self.hostname)

    def __del__(self):
        print('Я умираю....')

In [9]: sw1 = Switch('sw1', 'Cisco 3850')

In [10]: del sw1
Я умираю....
```

__del__

```
import netmiko

DEVICE_PARAMS = {
        'device_type': 'cisco_ios',
        'ip': '192.168.100.1',
        'username': 'cisco',
        'password': 'cisco',
        'secret': 'cisco'
}

class CiscoSSH:
    def __init__(self, **device_params):
        self.ssh = netmiko.ConnectHandler(**device_params)
        self.ssh.enable()

    def __del__(self):
        print('Закрываю сессию')
        self.ssh.disconnect()


In [4]: r1 = CiscoSSH(**DEVICE_PARAMS)

In [5]: del r1
Закрываю сессию
```

МЕНЕДЖЕР КОНТЕКСТА

МЕНЕДЖЕР КОНТЕКСТА

Для использования объекта в менеджере контекста, в классе должны быть определены методы `__enter__` и `__exit__`:

- `__enter__` выполняется в начале блока with и, если метод возвращает значение, оно присваивается в переменную, которая стоит после as.
- `__exit__` гарантированно вызывается после блока with, даже если в блоке возникло исключение.

МЕНЕДЖЕР КОНТЕКСТА

```
class CiscoSSH:
    def __init__(self, **device_params):
        self.ssh = netmiko.ConnectHandler(**device_params)
        self.ssh.enable()
        print('CiscoSSH __init__ called')

    def __enter__(self):
        print('CiscoSSH __enter__ called')
        return self

    def __exit__(self, exc_type, exc_value, traceback):
        print('CiscoSSH __exit__ called')
        self.ssh.disconnect()
```

МЕНЕДЖЕР КОНТЕКСТА

```
In [8]: with CiscoSSH(**DEVICE_PARAMS) as r1:
   ...:     print('Внутри with')
   ...:     print(r1.ssh.send_command('sh ip int br'))
   ...:
CiscoSSH __init__ called
CiscoSSH __enter__ called
Внутри with
Interface                  IP-Address      OK? Method Status                Protocol
Ethernet0/0                192.168.100.1   YES NVRAM  up                    up
Ethernet0/1                192.168.200.1   YES NVRAM  up                    up
Ethernet0/2                190.16.200.1    YES NVRAM  up                    up
Ethernet0/3                192.168.230.1   YES NVRAM  up                    up
Ethernet0/3.100            10.100.0.1      YES NVRAM  up                    up
Ethernet0/3.200            10.200.0.1      YES NVRAM  up                    up
Ethernet0/3.300            10.30.0.1       YES NVRAM  up                    up
CiscoSSH __exit__ called
```

МЕНЕДЖЕР КОНТЕКСТА

Если внутри блока with возникает исключение, оно будет сгенерировано после выполнения метода `__exit__`:

```
In [10]: with CiscoSSH(**DEVICE_PARAMS) as r1:
    ...:     print('Внутри with')
    ...:     print(r1.ssh.send_command('sh ip int br'))
    ...:     raise ValueError('Ошибка')
    ...:
CiscoSSH __init__ called
CiscoSSH __enter__ called
Внутри with
Interface                  IP-Address      OK? Method Status                Protocol
Ethernet0/0                192.168.100.1   YES NVRAM  up                    up
Ethernet0/1                192.168.200.1   YES NVRAM  up                    up
Ethernet0/2                190.16.200.1    YES NVRAM  up                    up
Ethernet0/3                192.168.230.1   YES NVRAM  up                    up
Ethernet0/3.100            10.100.0.1      YES NVRAM  up                    up
Ethernet0/3.200            10.200.0.1      YES NVRAM  up                    up
Ethernet0/3.300            10.30.0.1       YES NVRAM  up                    up
CiscoSSH __exit__ called
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-10-4e8b17370785> in <module>()
      2     print('Внутри with')
      3     print(r1.ssh.send_command('sh ip int br'))
----> 4     raise ValueError('Ошибка')

ValueError: Ошибка
```

МЕНЕДЖЕР КОНТЕКСТА

Если метод `__exit__` возвращает истинное значение, исключение не генерируется:

```
class CiscoSSH:
    def __init__(self, **device_params):
        self.ssh = netmiko.ConnectHandler(**device_params)
        self.ssh.enable()
        print('CiscoSSH __init__ called')

    def __enter__(self):
        print('CiscoSSH __enter__ called')
        return self

    def __exit__(self, exc_type, exc_value, traceback):
        print('CiscoSSH __exit__ called')
        self.ssh.disconnect()
        return True
```

МЕНЕДЖЕР КОНТЕКСТА

```
In [17]: with CiscoSSH(**DEVICE_PARAMS) as r1:
    ...:     print('Внутри with')
    ...:     print(r1.ssh.send_command('sh ip int br'))
    ...:     raise ValueError('Ошибка')
    ...:
CiscoSSH __init__ called
CiscoSSH __enter__ called
Внутри with
Interface                  IP-Address      OK? Method Status                Protocol
Ethernet0/0                192.168.100.1   YES NVRAM  up                    up
Ethernet0/1                192.168.200.1   YES NVRAM  up                    up
Ethernet0/2                190.16.200.1    YES NVRAM  up                    up
Ethernet0/3                192.168.230.1   YES NVRAM  up                    up
Ethernet0/3.100            10.100.0.1      YES NVRAM  up                    up
Ethernet0/3.200            10.200.0.1      YES NVRAM  up                    up
Ethernet0/3.300            10.30.0.1       YES NVRAM  up                    up
CiscoSSH __exit__ called
```

МЕТОД `__getattr__`

МЕТОД `__getattr__`

Метод `__getattr__` вызывается только для не существующих атрибутов:

```
In [1]: class CiscoSSH:
   ...:     def __init__(self, **device_params):
   ...:         self._device_params = device_params
   ...:         self.ssh = netmiko.ConnectHandler(**device_params)
   ...:         self.ssh.enable()
   ...:

In [2]: r1 = CiscoSSH(**DEVICE_PARAMS)

In [3]: r1.test
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-20-d2cff4242cf5> in <module>()
----> 1 r1.test

AttributeError: 'CiscoSSH' object has no attribute 'test'
```

МЕТОД `__getattr__`

Этот метод позволяет динамически создавать атрибуты, при первом вызове:

```
class CiscoSSH:
    def __init__(self, **device_params):
        self.ssh = netmiko.ConnectHandler(**device_params)
        self.ssh.enable()

    def __getattr__(self, attr):
        if attr == 'sh_version':
            print('Создаю атрибут sh_version')
            output = self.ssh.send_command('sh version')
            self.sh_version = re.search('Version (.*?),', output).group(1)
            return self.sh_version

In [8]: r1 = CiscoSSH(**DEVICE_PARAMS)

In [9]: r1.sh_version
Создаю атрибут sh_version
Out[9]: '15.2(2.3)T'

In [10]: r1.sh_version
Out[10]: '15.2(2.3)T'
```

МЕТОД `__getattr__`

При таком варианте метода `__getattr__` не генерируется исключение при доступе к несуществующим атрибутам. Чтобы вернуть этот функционал, добавлена генерация исключения:

```
class CiscoSSH:
    def __init__(self, **device_params):
        self.ssh = netmiko.ConnectHandler(**device_params)
        self.ssh.enable()

    def __getattr__(self, attr):
        if attr == 'sh_version':
            print('Создаю атрибут sh_version')
            output = self.ssh.send_command('sh version')
            self.sh_version = re.search('Version (.*?),', output).group(1)
            return self.sh_version
        else:
            raise AttributeError("'CiscoSSH' object has no attribute '{}'".format(attr))
```

МЕТОД `__getattr__`

```
In [3]: r1 = CiscoSSH(**DEVICE_PARAMS)

In [4]: r1.sh_version
Создаю атрибут sh_version
Out[4]: '15.2(2.3)T'

In [5]: r1.sh_clock
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-55-0e09b642cfdd> in <module>()
----> 1 r1.sh_clock

<ipython-input-52-e5278c4462df> in __getattr__(self, attr)
     11             return self.sh_version
     12         else:
---> 13             raise AttributeError("'CiscoSSH' object has no attribute '{}'".format(attr))

AttributeError: 'CiscoSSH' object has no attribute 'sh_clock'
```