PYTHON ДЛЯ СЕТЕВЫХ ИНЖЕНЕРОВ

WELCOME TO ПРОДЛЕНКА :)

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ (ООП)

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Wikipedia:

Объектно-ориентированное программирование (ООП) - методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования

ТЕРМИНЫ

- Класс (Class) - элемент программы, который описывает какой-то тип данных. Класс описывает шаблон для создания объектов, как правило, указывает переменные этого объекта и действия, которые можно выполнять применимо к объекту.
- Экземпляр класса (Instance) - объект, который является представителем класса.
- Метод (Method) - функция, которая определена внутри класса и описывает какое-то действие, которое поддерживает класс

СОЗДАНИЕ КЛАССА

СОЗДАНИЕ КЛАССА

```
In [1]: class Switch:
   ...:     pass
   ...:

In [2]: sw1 = Switch()
```

АТРИБУТЫ

```
In [3]: sw1 = Switch()
In [4]: sw2 = Switch()

In [5]: sw1.hostname = 'sw1'
In [6]: sw1.model = 'Cisco 3850'

In [7]: sw2.hostname = 'sw2'
In [8]: sw2.model = 'Cisco 3750'
```

АТРИБУТЫ

```
In [13]: sw1.model
Out[13]: 'Cisco 3850'

In [14]: sw2.model
Out[14]: 'Cisco 3750'
```

МЕТОДЫ

```
In [15]: class Switch:
    ...:     def info(self):
    ...:         print('Hostname: {}\nModel: {}'.format(self.hostname, self.model))
    ...:
```

МЕТОДЫ

```
In [16]: sw1 = Switch()

In [17]: sw1.hostname = 'sw1'

In [18]: sw1.model = 'Cisco 3850'

In [19]: sw1.info()
Hostname: sw1
Model: Cisco 3850
```

__init__

```
In [32]: class Switch:
    ...:     def __init__(self, hostname, model):
    ...:         self.hostname = hostname
    ...:         self.model = model
    ...:
    ...:     def info(self):
    ...:         print('Hostname: {}\nModel: {}'.format(self.hostname, self.model))
    ...:
```

__init__

```
In [33]: sw1 = Switch('sw1', 'Cisco 3850')

In [36]: sw1.info()
Hostname: sw1
Model: Cisco 3850

In [37]: sw1.hostname
Out[37]: 'sw1'
```

SELF

Эти варианты равнозначны:

```
In [38]: Switch.info(sw1)
Hostname: sw1
Model: Cisco 3850

In [39]: sw1.info()
Hostname: sw1
Model: Cisco 3850
```

SELF

```
In [40]: class Switch:
    ...:     def __init__(self, hostname, model):
    ...:         self.hostname = hostname
    ...:         self.model = model
    ...:

In [41]: def info(sw_obj):
    ...:     print('Hostname: {}\nModel: {}'.format(sw_obj.hostname, sw_obj.model))
    ...:

In [42]: sw1 = Switch('sw1', 'Cisco 3850')

In [43]: info(sw1)
Hostname: sw1
Model: Cisco 3850
```

СПЕЦИАЛЬНЫЕ МЕТОДЫ

__str__

```
In [45]: class Switch:
    ...:     def __init__(self, hostname, model):
    ...:         self.hostname = hostname
    ...:         self.model = model
    ...:

In [46]: sw1 = Switch('sw1', 'Cisco 3850')

In [47]: print(sw1)
<__main__.Switch object at 0xb4e47d8c>
```

__str__

```
In [52]: class Switch:
    ...:     def __init__(self, hostname, model):
    ...:         self.hostname = hostname
    ...:         self.model = model
    ...:
    ...:     def __str__(self):
    ...:         return 'Hostname: {}, Model: {}'.format(self.hostname, self.model)
    ...:

In [53]: sw1 = Switch('sw1', 'Cisco 3850')

In [54]: print(sw1)
Hostname: sw1, Model: Cisco 3850

In [55]: str(sw1)
Out[55]: 'Hostname: sw1, Model: Cisco 3850'
```

ОДНО ПОДЧЕРКИВАНИЕ ПЕРЕД ИМЕНЕМ

Одно подчеркивание перед именем указывает, что имя используется как внутреннее, что этот объект является внутренней особенностью реализации и не стоит его использовать напрямую.

```
class Switch:
    def _get_display_str(self):
        hostname = getattr(self, 'hostname', None)
        model = getattr(self, 'model', None)
        return 'Hostname: {}, Model: {}'.format(hostname, model)

    def __str__(self):
        return self._get_display_str()
```

ДВА ПОДЧЕРКИВАНИЯ ПЕРЕД ИМЕНЕМ

Два подчеркивания перед именем метода или аргумента используются не просто как договоренность. Такие имена трансформируются в формат "имя класса + имя метода". Это позволяет создавать уникальные методы и атрибуты классов.

> *Такое преобразование выполняется только в том случае, если в конце менее двух подчеркиваний или нет подчеркиваний*

ДВА ПОДЧЕРКИВАНИЯ ПЕРЕД ИМЕНЕМ

```
class Switch:
    def __get_display_str(self):
        hostname = getattr(self, 'hostname', None)
        model = getattr(self, 'model', None)
        return 'Hostname: {}, Model: {}'.format(hostname, model)

    def __str__(self):
        return self.__get_display_str()
```

ДВА ПОДЧЕРКИВАНИЯ ПЕРЕД ИМЕНЕМ

```
In [1]: sw1 = Switch()

In [2]: dir(sw1)
Out[2]:
['_Switch__get_display_str',
 '__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
...
 '__weakref__']

In [3]: print(sw1)
Hostname: None, Model: None
```

ДВА ПОДЧЕРКИВАНИЯ ПЕРЕД И ПОСЛЕ ИМЕНИ

Таким образом обозначаются специальные переменные и методы.

Эти методы вызываются при использовании функций и операторов Python и позволяют реализовать определенный функционал.

Как правило, такие методы не нужно вызывать напрямую. Но, например, при создании своего класса может понадобиться описать такой метод, чтобы объект поддерживал какие-то операции в Python.

ДВА ПОДЧЕРКИВАНИЯ ПЕРЕД И ПОСЛЕ ИМЕНИ

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

    def __str__(self):
        return 'Hostname: {}, Model: {}'.format(self.hostname, self.model)

    def __del__(self):
        print("Я умираю....")
```

```
In [2]: sw1 = Switch('sw1', 'cisco')

In [3]: del sw1
Я умираю....

In [4]: sw1
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-4-924218d7093c> in <module>()
----> 1 sw1

NameError: name 'sw1' is not defined
```