Analytics Vidhya

Top 10 common MAGIC FUNCTIONS in python

__init__

__str__

__eq__

__repr__

__ne__

__add__

__getitem__

__setitem__

__len__

__del__

__init__: This is the constructor method for a class. It's called when an object is instantiated and allows you to set initial values for attributes.

```
def __init__(self, x):
    self.x = x
```

__str__: Defines a human-readable string representation of the object, which is what you'll see when you use the print() function.

```
def __str__(self):
    return f"An object with x: {self.x}"
```

__eq__: Allows you to use the == operator to compare two objects of the class.

```
def __eq__(self, other):
    return self.x == other.x
```

__repr__: Defines an "official" string representation of the object, useful for debugging. This is what you'll see when you look at the object in a Python shell.

```
def __repr__(self):
    return f"MyClass(x={self.x})"
```

__ne__: Allows you to use the != operator to compare two objects of the class.

```
def __ne__(self, other):
    return not self.__eq__(other)
```

__add__: Allows you to define custom behavior for the + operator for the class instances.

```
def __add__(self, other):
    return MyClass(self.x + other.x)
```

__getitem__: Allows you to use indexing to access elements in the object, like obj[i].

```
def __getitem__(self, index):
    return self.some_list[index]
```

__setitem__: Allows you to use indexing to set elements in the object, like obj[i] = x.

```
def __setitem__(self, index, value):
    self.some_list[index] = value
```

__len__: Allows you to use the len() built-in function with the object.

```
def __len__(self):
    return len(self.some_list)
```

__del__: The destructor method for a class, called when an object is about to be destroyed. Use it cautiously, as Python's garbage collector usually handles resource cleanup.

```
def __del__(self):
    print("Object is being deleted")
```