Magic or Dunder Methods

Magic methods in Python are the special methods that start and end with the double underscores. They are also called dunder methods. Magic methods are not meant to be invoked directly by you, but the invocation happens internally from the class on a certain action.

For example: '__abs__', '__add__', '__and__', '__bool__', '__ceil__'

Use the dir() function to see the number of magic methods inherited by a class.

Add two numbers : using the + operator. Consider the following example in Notebook

__new__() method:

Languages such as Java and C# use the new operator to create a new instance of a class. In Python the __new__() magic method is implicitly called before the __init__() method. The __new__() method returns a new object, which is then initialized by __init__().

Abstraction in Python

In Python, an abstraction is used to hide the irrelevant data/class in order to reduce the complexity. It also enhances the application efficiency.

Abstraction classes in Python

A class that consists of one or more abstract method is called the abstract class. Abstract methods do not contain their implementation. Abstract class can be inherited by the subclass and abstract method gets its definition in the subclass.

➔Example in Notebook

An abstract base class is the common application program of the interface for a set of subclasses. It can be used by the third-party, which will provide the implementations such as with plugins. It is also beneficial when we work with the large code-base hard to remember all the classes.

Encapsulation in Python

Encapsulation is one of the fundamental concepts in object-oriented programming (OOP). It describes the idea of wrapping data and the methods that work on data within one unit. This puts restrictions on accessing variables and methods directly and can prevent the accidental modification of data. To prevent accidental change, an object's variable can only be changed by an object's method. Those types of variables are known as private variables.

Protected members:

Protected members (in C++ and JAVA) are those members of the class that cannot be accessed outside the class but can be accessed from within the class and its subclasses. To accomplish this in Python, just follow **the convention** by prefixing the name of the member by a **single underscore "_"**.

Private members:

In Python, there is no existence of Private instance variables that cannot be accessed except inside a class

*args and **kwargs in Python

use the "wildcard" or "*" notation like this – *args OR **kwargs – as our function's argument when we have doubts about the number of arguments we should pass in a function

Python *args :

The special syntax *args in function definitions in python is used to pass a variable number of arguments to a function. It is used to pass a non-key worded, variable-length argument list.

Python **kwargs:

The special syntax ***kwargs* in function definitions in python is used to pass a keyworded, variable-length argument list. We use the name *kwargs* with the double star. The reason is that the double star allows us to pass through keyword arguments (and any number of them).