dunder methods:

are the special methods that start and end with the double underscores.

the invocation happens internally from the class on a certain action. For example, when you add two numbers using the + operator, internally, the `__add__()` method will be called.

Examples:

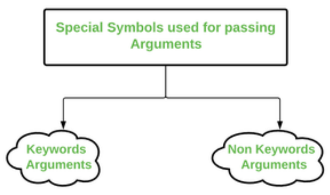
```
'__abs__', '__add__', '__and__', '__bool__', '__ceil__', '__class__',
'__delattr__', '__dir__', '__divmod__', '__doc__', '__eq__', '__float__',
'__floor__', '__floordiv__', '__format__', '__ge__', '__getattribute__',
'__getnewargs__', '__gt__', '__hash__', '__index__', '__init__',
'__init_subclass__', '__int__', '__invert__', '__le__', '__lshift__',
'__lt__', '__mod__', '__mul__', '__ne__', '__neg__', '__new__', '__or__',
'__pos__', '__pow__', '__radd__', '__rand__', '__rdivmod__', '__reduce__',
'__reduce_ex__', '__repr__', '__rfloordiv__', '__rlshift__', '__rmod__',
'__rmul__', '__ror__', '__round__', '__rpow__', '__rrshift__',
'__rshift__', '__rsub__', '__rtruediv__', '__rxor__', '__setattr__',
'__sizeof__',
```

For example, the `__add__` method is a magic method which gets called when we add two numbers using the + operator. Consider the following example.

*args and **kwargs in Python:

We can pass a variable number of arguments to a function using special symbols.

There are two special symbols:



Special Symbols Used for passing arguments:-

- *args (Non-Keyword Arguments)
- **kwargs (Keyword Arguments)

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.

- The syntax is to use the symbol * to take in a variable number of arguments; by convention, it is often used with the word args.
- What **args* allows you to do is take in more arguments than the number of formal arguments that you previously defined. With **args*, any number of extra arguments can be tacked on to your current formal parameters (including zero extra arguments).
- For example, we want to make a multiply function that takes any number of arguments and is able to multiply them all together. It can be done using *args.
- Using the *, the variable that we associate with the * becomes an iterable meaning you can do things like iterate over it, run some higher-order functions such as map and filter, etc.

For example:

```
def myFun(*argv):
    for arg in argv:
        print(arg)


myFun('Hello', 'Welcome', 'to', 'MuhammadWassel')
```

output:

```
Hello

Welcome

to

MuhammadWassel
```

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).

- A keyword argument is where you provide a name to the variable as you pass it into the function.
- One can think of the *kwargs* as being a dictionary that maps each keyword to the value that we pass alongside it. That is why when we iterate over the *kwargs* there doesn’t seem to be any order in which they were printed out.

Example:

```
def myFun(**kwargs):
    for key, value in kwargs.items():
        print("%s == %s" % (key, value))


# Driver code
myFun(first='Mu', mid='Abd', last='elfattah')
```

```
last == elfattah

mid == abd

first == Muhammad
```

Python- Encapsulation Vs Abstraction:

In this tutorial you will master everything about the OOPs concept of encapsulation and abstraction in python and how they differ from each other. Also you will walk through the important access modifiers in object oriented programming such as private ,public and protected in detail with examples.

In your previous tutorial you have grasped the concept of inheritance in python. If you are new to OOPS , we would suggest you to learn our tutorial of inheritance to acquaint yourself with OOPS.

Encapsulation:

Encapsulation is one of the important building blocks of object oriented programming (OOP) which enables the wrapping of all data (attributes and methods) into a single component(class) thereby preventing the accidental modification of data by imposing some restrictions on accessing variables or methods directly.

Abstraction:

Like encapsulation,Abstraction is one of the basic building blocks of Object oriented programming.

Abstraction is the process of hiding internal implementation of a process from its end user,showing only essential features of data to the external world in order to reduce complexity and increase efficiency. In the context of programming, isolating the signature of a method from its definition is known as abstraction.It generally focuses on ideas rather than the functionality.

In our daily life, we use appliances like TV, Refrigerator, Washing Machine etc to meet some specific needs. We are not aware of its internal working or we are not interested to know about. We just operate them, profoundly their internal implementation is abstracted from us.