**Magic Methods In Python**

- [April 06, 2022](https://magicmethodsinpython.blogspot.com/2022/04/magic-methods-in-python.html)

**[](https://blogger.googleusercontent.com/img/a/AVvXsEjt6YCyGJsBrtF7S76uNfJTEU4aWxpiTvDD5eS7tznrmQ_x4eXxajHo3lj5itRGYpZ4ES-bj1pKVdSdv15MGH57HBA6N_BC-mZuP8jOJcK2bysJc_BmKOpQ4rkpnbasCKW22Jbs41c7RkYt2SGVX6qJ6EJYxWUTw-czJv65WphGUncDJ0iQ9tzp_5uU)**

**Magic Methods**

**INTRODUCTION**

What do you mean by magic methods? They're all written in Python. They're special methods that you can add to your classes to add "magic." Double underscores usually surround them (e.g., \_\_init\_\_ or \_\_lt\_\_). They aren't well recorded as well as they should be. In the Python documentation, all of the magic methods are listed in the same section, although they're dispersed and poorly arranged. There aren't many examples in that area (which could be by design, given that they're all included in the language reference, along with tedious syntax descriptions and so on).

**Types of Magic Methods**

There are many magic methods available. All of them are as follows:

1. **Constructor and Initialization**

The most fundamental magic technique, \_\_init\_\_, is well-known. It's a method for defining an object's initialization behaviour. When I call x = SomeClass(), however, \_\_init\_\_ is not the first thing that happens. Actually, it's a method called \_\_new\_\_ that creates the instance and then provides any parameters to the initializer at the time of creation. There's \_\_del\_\_ at the other end of the object's lifecycle. Let's take a closer look at each of these three magical techniques:

* \_\_new\_\_ is the first method to get called in an object's instantiation. It takes the class, then any other arguments that it will pass along to \_\_init\_\_. \_\_new\_\_ is used fairly rarely, but it does have its purposes, particularly when sub classing an immutable type like a tuple or a string. \_\_new\_\_ is not useful that much as \_\_init\_\_ is used.
* \_\_init\_\_ is the initializer for the class. It gets passed whatever the primary constructor was called with (so, for example, if we called x = SomeClass(10, 'foo'), \_\_init\_\_ would get passed 10 and 'foo' as arguments. \_\_init\_\_ is almost universally used in Python class definitions.

**Destructor and deletion**

* If \_\_new\_\_ and \_\_init\_\_ formed the constructor of the object, \_\_del\_\_ is the destructor. It doesn't implement behavior for the statement del x (so that code would not translate to x.\_\_del\_\_()). Rather, it defines behavior for when an object is garbage collected. It can be quite useful for objects that might require extra cleanup upon deletion, like sockets or file objects. Be careful, however, as there is no guarantee that \_\_del\_\_ will be executed if the object is still alive when the interpreter exits, so \_\_del\_\_ can't serve as a replacement for good coding practices (like always closing a connection when you're done with it. In fact, \_\_del\_\_ should almost never be used because of the precarious circumstances under which it is called; use it with caution!

**Comparison methods**

Python provides a host of magic methods for doing easy object comparisons using operators rather than cumbersome method calls. They also give a mechanism to change Python's default behavior for object comparisons (by reference). The following is a list of the methods and what they do:

* \_\_cmp\_\_ is the most basic of the comparison magic methods. It actually implements behavior for all of the comparison operators (<, ==, !=, etc.), but it might not do it the way you want (for example, if whether one instance was equal to another were determined by one criterion and and whether an instance is greater than another were determined by something else). \_\_cmp\_\_ should return a negative integer if self < other, zero if self == other, and positive if self > other.
* \_\_eq\_\_(self, other)

Defines behavior for the equality operator, ==.

* \_\_ne\_\_(self, other)

Defines behavior for the inequality operator, !=.

* \_\_lt\_\_(self, other)

Defines behavior for the less-than operator, <.

* \_\_gt\_\_(self, other)

Defines behavior for the greater-than operator, >.

* \_\_le\_\_(self, other)

Defines behavior for the less-than-or-equal-to operator, <=.

* \_\_ge\_\_(self, other)

Defines behavior for the greater-than-or-equal-to operator, >=.

[](https://blogger.googleusercontent.com/img/a/AVvXsEgHr6m-04jwmBS3-Lh036Nb_XOWiqJ6FEjP4d3qCay6T85fUtjbavGWqG_RXYQh4Fovwu-GrRJh1CfqxjqoqGhAGhOlRMAzwTznGHuyl5lI66lhlPPea_6nt_yatvM_g8popVViDfRXTqlSgHwIaIN4w633-m9ECyYhneRTmnsFB1ymfbsSkYDhVEW-)

[](https://blogger.googleusercontent.com/img/a/AVvXsEhFZ7eBwSPNoNzyr4f5kKlULC3k2LsE8X4S5p6gXL7wXMMvQGa6yOGjrwayHEYmy4lK_Y3aihfaOqXEdKvSasDRQ3iYUMBYEEpFien-5WdfNbtQ15Uo3Omf98vgsdTR_kMZdJNI-D-dhsQfDaxMdqgNvw3OYGBS3PIjCRXkuWlbKyGnKKk32ENiiJcM)

**Numeric Magic Methods**

**Unary Operators**

* \_\_pos\_\_(self)

Implements behavior for unary positive (e.g. +some\_object)

* \_\_neg\_\_(self)

Implements behavior for negation (e.g. -some\_object)

* \_\_abs\_\_(self)

Implements behavior for the built-in abs() function.

* \_\_invert\_\_(self)

Implements behavior for inversion using the ~ operator.

* \_\_round\_\_(self, n)

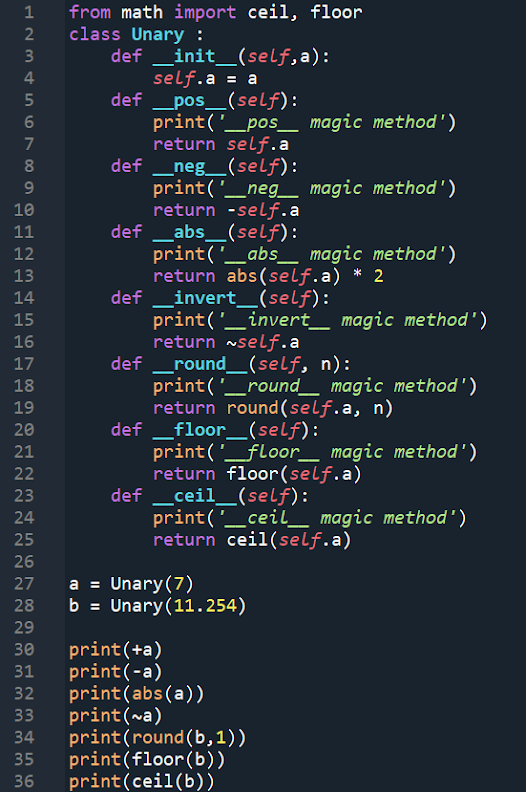
Implements behavior for the built-in round() function. n is the number of decimal places to round to.

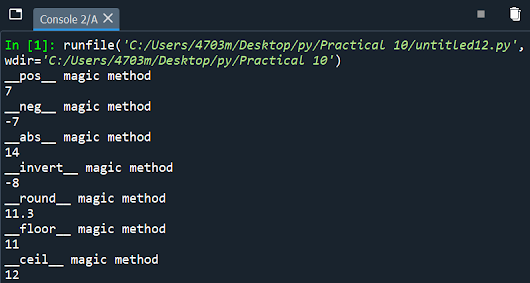
* \_\_floor\_\_(self)

Implements behavior for math.floor(), i.e., rounding down to the nearest integer.

* \_\_ceil\_\_(self)

Implements behavior for math.ceil(), i.e., rounding up to the nearest integer.

[](https://blogger.googleusercontent.com/img/a/AVvXsEjUdt9xSwNLqkIDhIm8MLOxhoeg44GTH1h4G4vyirg2FvlYhnm5_RSvYU9e7vfi6UnH3jVNbju4pbfBoey1OqAFTWtJMBdsfnH0tLIIQQONYAEmQv8SpQNDIKAkEKJmJfrlPmzWzBZlVNH_GIyQeFX2MH-1Rm_yVlTmD6uQIk7wkPmgHX8DFmM4q4zi)

[](https://blogger.googleusercontent.com/img/a/AVvXsEgpQWmQbZJGuh4NGp4rBuKeewJyxXTEygrqHUh6JxcHEgKtxkhwG0R8QuoRZ_Uk55QhRNIPP7gqFmFRMWxPeybLT3W4eInjvAcI20iKBH9PIrswUm5UkydpRNt2bOooUYRh2r1Z366H2f398vZK5mTfUCyRHXsuZWxUo5Oo-nlC_YjgRRroTwMuPAbk)

**Binary Operators**

* \_\_add\_\_(self, other)

    Implements addition.

* \_\_sub\_\_(self, other)

    Implements subtraction.

* \_\_mul\_\_(self, other)

    Implements multiplication.

* \_\_floordiv\_\_(self, other)

    Implements integer division using the // operator.

* \_\_truediv\_\_(self, other)

    Implements division using the / operator.

* \_\_mod\_\_(self, other)

    Implements modulo using the % operator.

* \_\_pow\_\_(self, other)

    Implements behavior for exponents using the \*\* operator.

* \_\_lshift\_\_(self, other)

    Implements left bitwise shift using the << operator.

* \_\_rshift\_\_(self, other)

    Implements right bitwise shift using the >> operator.

* \_\_and\_\_(self, other)

    Implements bitwise and using the & operator.

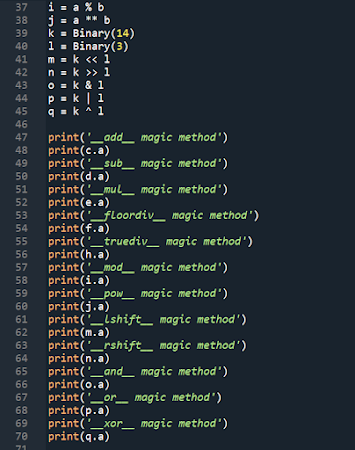
* \_\_or\_\_(self, other)

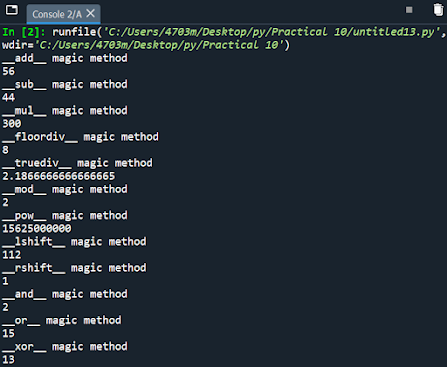
    Implements bitwise or using the | operator.

* \_\_xor\_\_(self, other)

    Implements bitwise xor using the ^ operator.

[](https://blogger.googleusercontent.com/img/a/AVvXsEix2PoLlhTiX4lE_ndkasOLJu_qsRY_rHNTPsPPDuOBi5-26KwqcyXgeNZwaJDURcKY3J0vicSGbl7UALBeUL5Vk52W9Rdz7DK3hGGrfM0QMWV0yg75uoysyjOEDHdqIy8xakNQ-6XAK5A0gZ-TskMahHDnRLtqq9ilSI34foEvP1TxQ_MAqn0NFeGd)

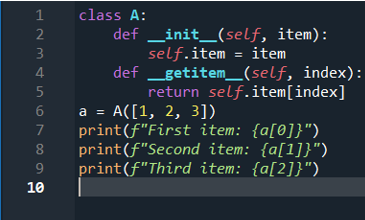
[](https://blogger.googleusercontent.com/img/a/AVvXsEgRS-_AQfQVwdLrbElQqXt4L2AZJLHxahf4O3dZrXWgHij4FjDKt9WlIYZqVUX1Z1jxjVK3zHI8dDEkVO_3FPb0QUYJ29AJQ8p4aWEzETUpXGz9oSzC_XUpaFZzpgDY_g4sxjFUgOWuUy38SAjuQzm692NCwHqEMwyHlAxKukn3Qu1ujHRGa1tTbttF)

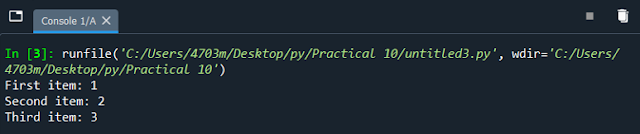
[](https://blogger.googleusercontent.com/img/a/AVvXsEjNaIEsSxq_aKO7XzCQDCHKFaYbhyDAIkTxgYT3ZSLeG46dJuIu9hXVfqiReoqrBmMI685npnwQrKZ3rXqAl6Ao5Gw85myZiOxDAlugtmPi5aK-SgSCDMwgoM1J4SNWkvtVWMgm1I4atXyGd9ZWL488TvEXw9xL-QDsuKwaYwqUqysH5USfJZsMAUSe)

**\_\_getitem\_\_ and \_\_setitem\_\_**

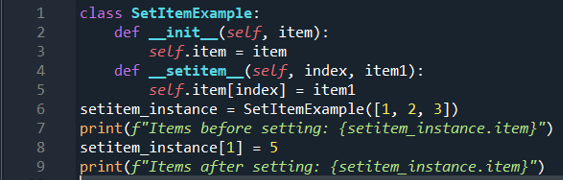
**magic method**

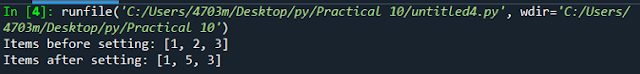
* \_\_getitem\_\_ method is used to get an item from the invoked instances’ attribute. \_\_getitem\_\_ is commonly used with containers like list, tuple, etc.

[](https://blogger.googleusercontent.com/img/a/AVvXsEgVrxUW4_-Zt_nKI5zx_sh9XvBQbh2WLl-LOWjvVhov_PIZKY0lQY-cpIDYt3Dg8Y2vhiNVHkOrEYH2IiLeYBX25sJx91xg0ihQWDJQFEWtNzJbxVhIM1dftpEhILnA-_7nd6wBuqrf94nyf2ijPzBAynZJwkfTJgBOiHfmeuU74o-joNRA_m1JKDwE)

[](https://blogger.googleusercontent.com/img/a/AVvXsEgPZs747sejAYavyxusH7_8RGTAdF7tU_W-1jSQJyrFyBa9u-buwB4eCNdDarEAcA01yV0Q0h5oghtiFNr8v5KdTbeJs5uv5mJkcJTj_GOeE4N5fURhybySNa46Ss3Nk6vAHTxcJbzzPV8eAA79wSXFFT1evIHusQ58-3_yowmhpSJP0jQpVQIJkNIz)

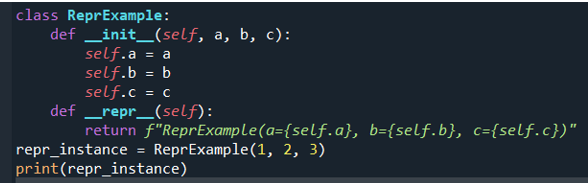
* \_\_setitem\_\_ method is used to set the item into a specific index of the invoked instances’ attribute. Similar to \_\_getitem\_\_, \_\_setitem\_\_ is also used with containers.

[](https://blogger.googleusercontent.com/img/a/AVvXsEi5afQhvYQMou6pgP83eJPGIj9AftKhJXxNIqJOPJxH7yQd9d1uKkR6f8Iix2jRrgXy2-faSUb_4m8y5vbb5aYTnKQ-Ma_cc_4MMtStXTqQJm1-8NYwGub6ACMwJqy8J4V98YgpxB8A2eKSJD8c8NWTxrptyK9FEvPuCHdy0mXTFFCbZ4UzREEGfDcb)

[](https://blogger.googleusercontent.com/img/a/AVvXsEhsYWiwb_wplO9EPvjYIhas_MDYaYND751yxuvTOx90HIsdES6Z5NUFtPRvvmf5Ma6Tc3RJxCLF2Obwf84ZczCQyrMCsARDx_vSvH_ULAxRpAzxON3i-_WnXZZSy9NeXjhtKoi9E6ywW7ki0Dy6dzFToDBid2Kxq7rPTgEt4TCcnp81TOC1zlsoeF6M)

**\_\_repr\_\_ magic method**

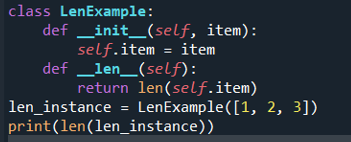
\_\_repr\_\_ magic method is used to represent the instance of a class in string. When we try to print the object of a class the \_\_repr\_\_ method is implicitly invoked that returns a string.

[](https://blogger.googleusercontent.com/img/a/AVvXsEgBWrC44vZVaEMQ-gfuu-eOqsWIfIKtYqucIC_nO-7CtaMqWC6k5syERRrm5IuXdVDHNIublslqwJQlHd25MZBYCxDzFZfvf72q15MuMK3tnkerbLW0Hfi1vSRcTxtIEvqWz8lAQp6vAGj6y8Opj1TcGKIsZk7596HRLqCgd8PaMtoRlaDmgQoJQU2R)

[](https://blogger.googleusercontent.com/img/a/AVvXsEjTrgLidx2-C9e--fNuRholjaCHFZL2pBdrxCEnUMFQyP4HhoPCb4InbfPn-MoyIyzsJf4Dc2wPjdMqibCQghRPDDUYHR7m3-kjMCb6eNRY1R7upoOyrW2mcRujvxio_K2GLtfFl9Jz6SasRNvpvDpNIx7kI1t_vWa2hyBIzkmHI0XGvR3h0VtgEssG)

**\_\_len\_\_ magic method**

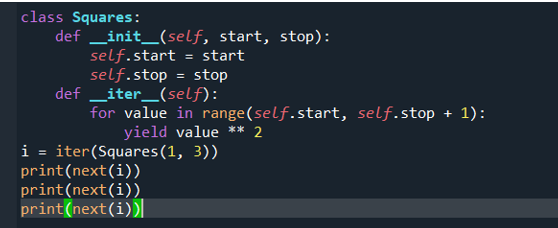
\_\_len\_\_ magic method is used to find the length of the instance attributes. When we use len(instance), it returns the length of the instance attribute which is usually containers.

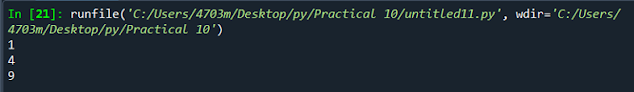
[](https://blogger.googleusercontent.com/img/a/AVvXsEgJi7sJbEMx1F10ME52ev-YtWsbxPechqhXiynaOZRNC-hKD67Oe55Jkz72ht8x0faqcGoNyp95UK283l2HzRxgkwxEDt4hB5_sm9CRLEO69btCjNz4jDZ0lapkq4C0juN8GPk1EBTyevYisGbp9jvl0ydxAE1gvPIyA3LZC0fSUqR3eR2OQuY_e_Ni)

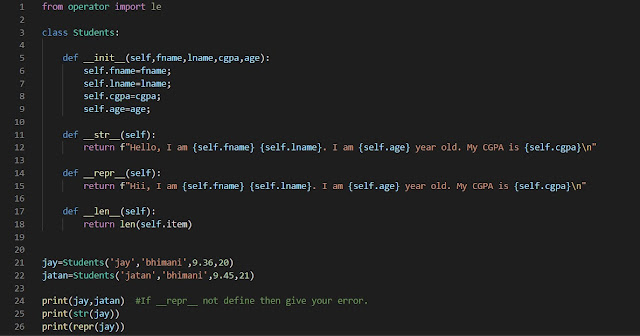
[](https://blogger.googleusercontent.com/img/a/AVvXsEjI7Pm6uEtgcgcN1arJn_q2dvJVDnEwjjc9bUbhRNjuvqnDM3PFUapuFp2t7_ZtKJ4djW-zjAErY1Kdg4wDPRxyZPkfdh3kBEXaSh17iMOo_zELpqx5SVfFWfH-XKNH6Diw2dofYz3K9kMgs9pN8uKZpwFBx1r4zBWw-KbCcinMoJ03L02nwF3NpXOY)

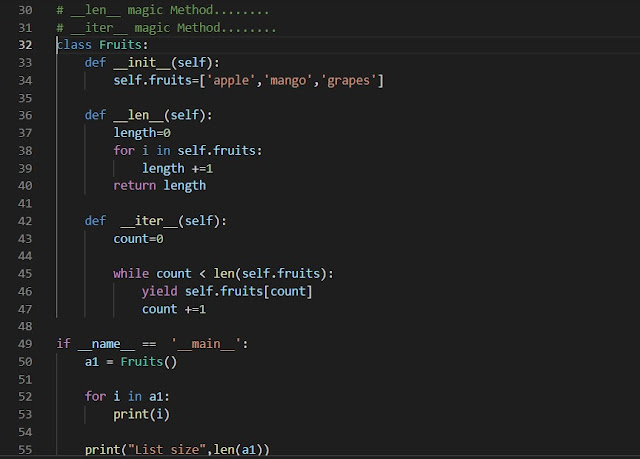
**\_\_iter\_\_ magic method**

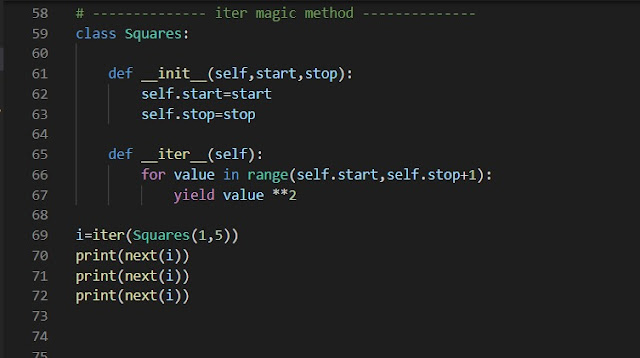
\_\_iter\_\_ method is used to provide a generator object for the provided instance. We can make use of iter() and next() method to leverage \_\_iter\_\_ method.

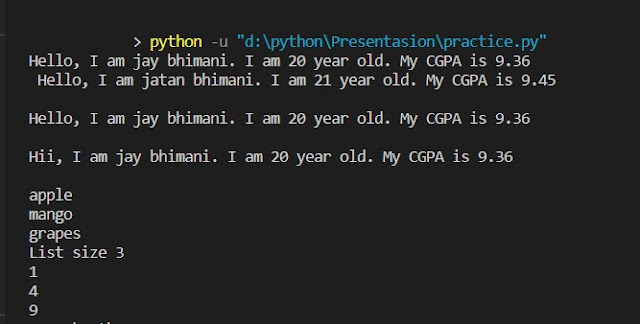
[](https://blogger.googleusercontent.com/img/a/AVvXsEhRGnIM1cF8JPWcgcWLT7iRnXkz3GoE5tBK4K8_GcMQWCBAjJv_xFfisPS00FhHC-tEESxWJGfWNBGEg6DxCyA3zuyhk30ZUvKpD2OaXyo04KHqmtu9wB3qqXA8aClLroL2KbfKkO8tZh60T8zeeijaus1d87wxJGnablKifZDjg26UVpj9GZ08q55X)

[](https://blogger.googleusercontent.com/img/a/AVvXsEgLcYt86a-zoMOV3hCEo-hsoDtOT19rcy56L41tD4wl2cnYjehKiMYRZSp2SS_rZwBB_bX6LU_On4r6Mjmx7mFPj3P9IpA7Tk5ws7mRjDFVbjVTUZBybRtPfnBxQqnmqQA_HHXnTWVvrtKgUaRiVnLSJM-n1-3Pi12NM6Glnm0fOPVPKWXxdrGas-r7)

[](https://blogger.googleusercontent.com/img/a/AVvXsEiP7MAyjkUlMtNfJu063KgfBj5f97wbkCOAI5Um2uQvVPk4avc7rPgumZpSHLJ7EHnEU4YIcx1lzxF78HBnUuT1FLTioWk48mIpuiIbt8PGio4zPBq6Kgd3WI2uhIUNoMnjHqiqq-3B1Q9wKMTuWC-AIMVMOHN6bIfrzUal5IzHf1UGzzjfX_qEi6Wr)

[](https://blogger.googleusercontent.com/img/a/AVvXsEh5wjKQ7F1y734SAKpWKL3wM9kuJ9tkbWT0juwtMprn_9G2XGsgeU0Z3qm1DJ_pcqkCFKOwtBWEeFNc81pBntWI5dtzqX03eWtLjj4niWpq1EKi8lVaiENHLp2pVgdVrJL3elzCOLfrP7Vines5wJ8N9jSbcSXNJu5odnTrkwF5nVz1GWwebFxgltxy)

[](https://blogger.googleusercontent.com/img/a/AVvXsEhexcugpjW5VCo4BguqbDO04cT0StTduLcni4usUAeMQtfJ0OHgaKMMhcL0la0Qef0eMmuJgn03ICucu_5CmMiVbVywAQlmcO-WTuiCqZW2XSWwrojlkAevIvICM878J0r-NGy-2SYfG3GfxI9a3MQW4gWYfYarctg9CXYpRr-T5BZ2b2JD9n3skVUt)

[](https://blogger.googleusercontent.com/img/a/AVvXsEgKMxrC01_5p_OONSJeuFbZpJxZwLgjEjfG4NkHWkz6CaLIOLedlor5AdHU8oJpA6DfNa1BfiTyZ6y5lQV_gmWv1cm7jlzNW3GuFxf9TMEGUeU9fUb60HEjV67ztIQaM9V5cn3r5Qun6zjc5yV-fv3KOfAZxh84DtgZpbVLduWOEeENjzZhbwkhb1cj)