

From Nothing to Python

I promise that I love you

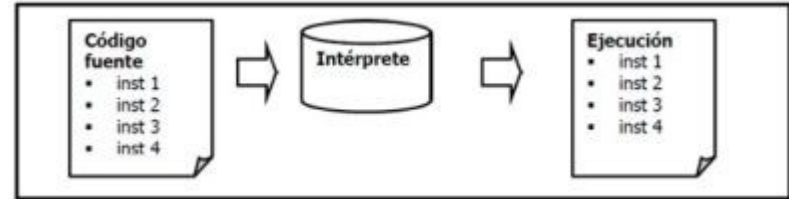
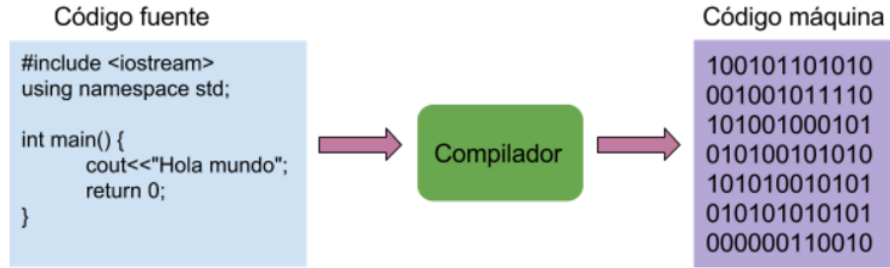
t.me/cyberh99

DevUco Group and projects :-)

<https://t.me/joinchat/EWwrOk34IP-Qit7TLHoutg>



Lenguajes interpretados vs Lenguajes compilados



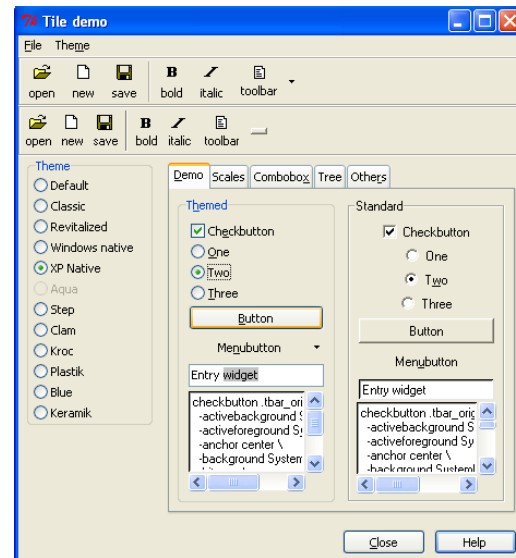
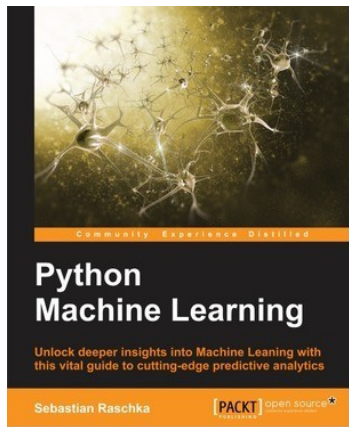
Ventajas de lenguajes interpretados

- Pueden ser ejecutados en cualquier plataforma.
- Ocupan menos espacio en la memoria.
- El framework es el que se encarga de que el hardware ejecute las instrucciones.
- Las variables de datos son dinámicas y no se restringen a un solo tipo.
- Son más utilizados en desarrollo web y en electrónica.

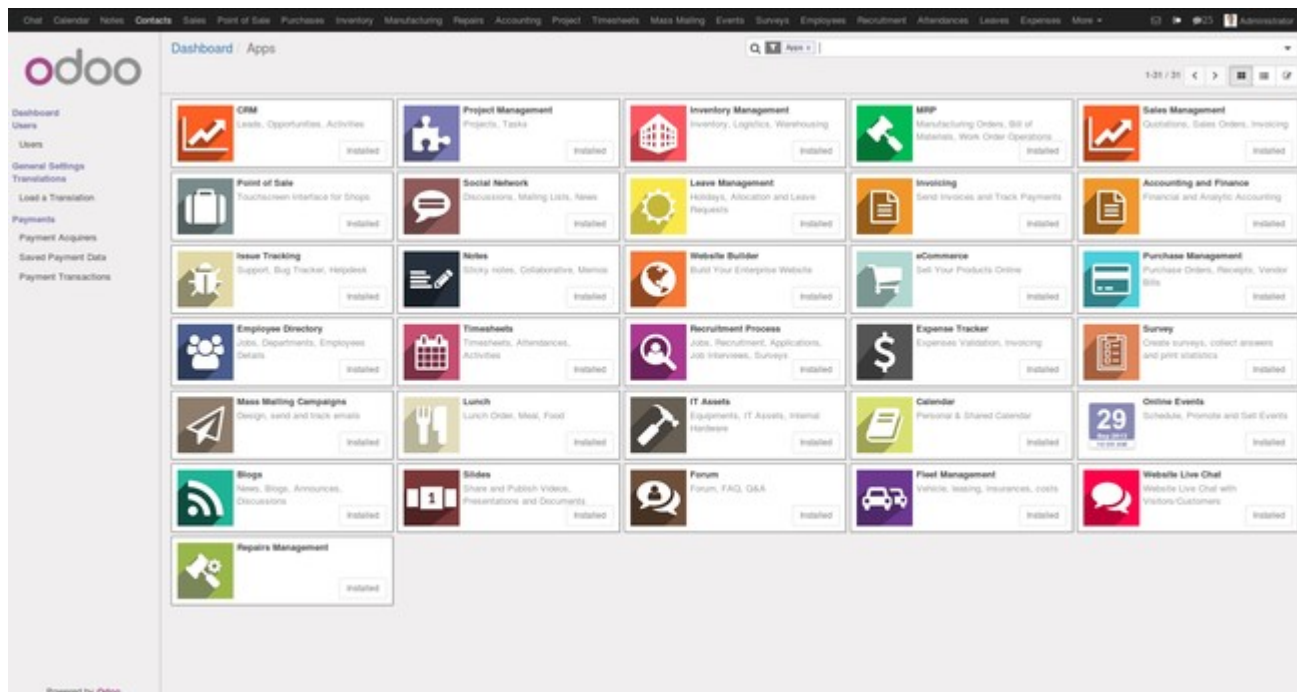
Desventajas de lenguajes interpretados

- Su ejecución es más lenta con respecto a los lenguajes compilados.
- Son más difíciles de debuggear.
- Requieren de una máquina virtual que sirva como intérprete las instrucciones y el procesador.
- No todas las máquinas virtuales están disponibles para todas las plataformas.

Python for all



I mean all



Instalando Python

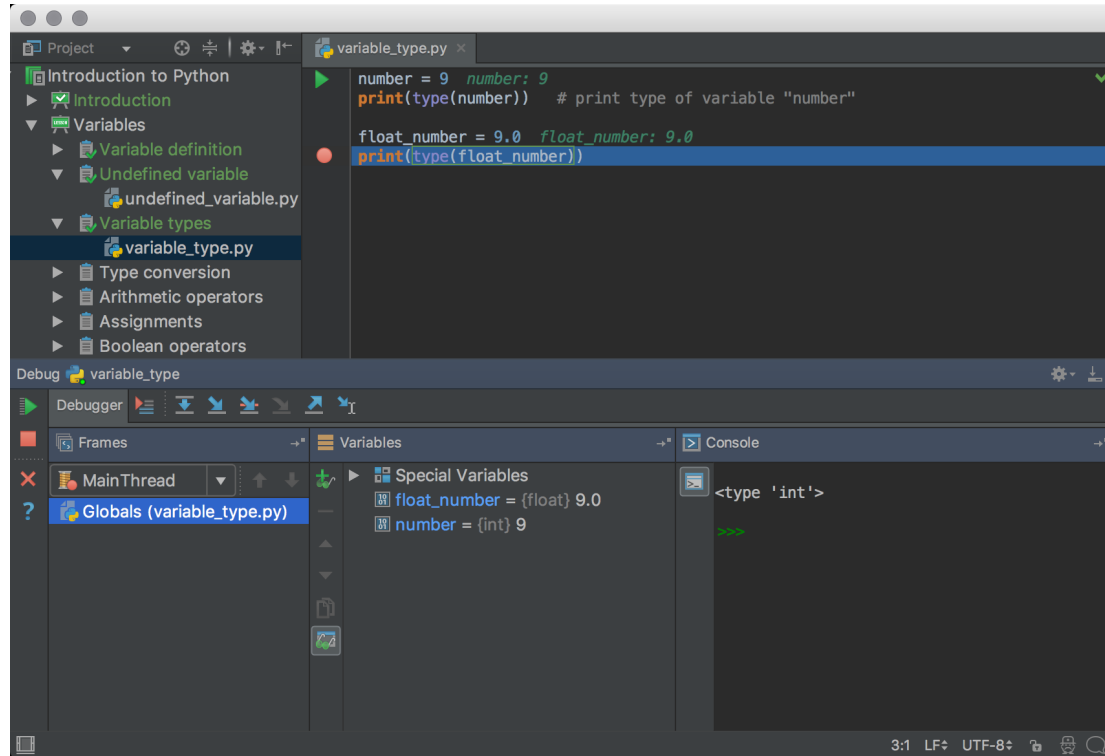
From terminal (Linux system):

- apt install python

From Windows:

- <https://www.python.org/>

IDE para python



Tipos de variables

- Int
- Long
- Float
- Bool
- string
- Unicode

- Listas
- Tuplas
- Diccionarios

Imagen de asignación múltiple de variables

Listas / Tuplas / Diccionarios

★ Listas:

- Almacenan datos de mismo (o diferente) tipo
- Los datos pueden ser modificados sobre la marcha
- lista = ["asd",2,3,True]

★ Tuplas:

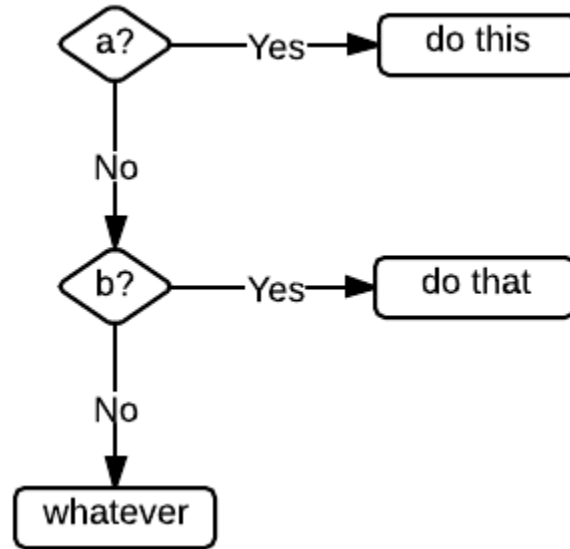
- Almacenan datos de mismo (o diferente tipo)
- Los datos NO pueden modificarse sobre la marcha
- tupla = ("AD",2,3,False)

★ Diccionario:

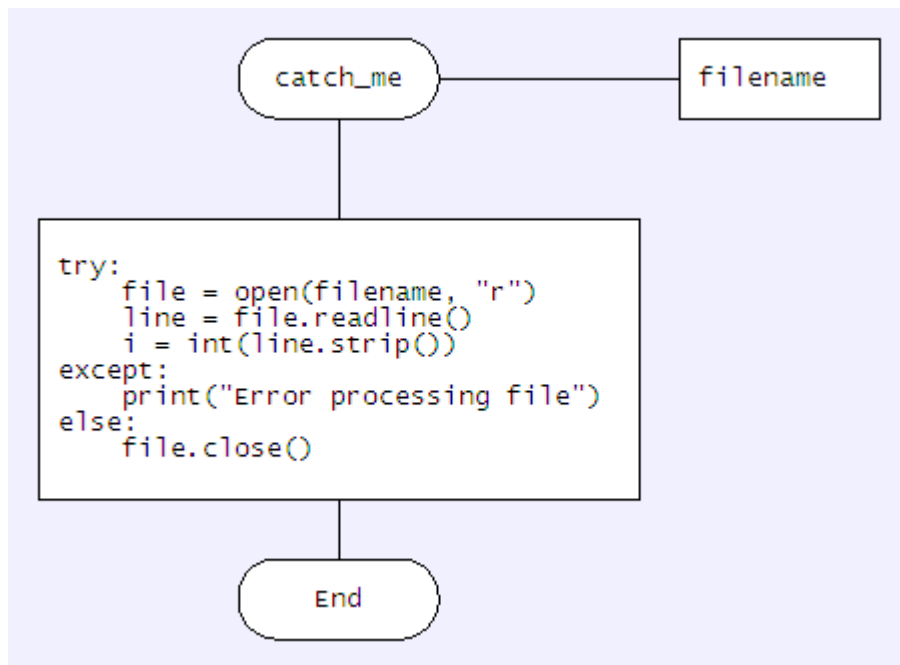
- Almacenan valores clave:valor
- diccionario = {"hola":"pepe", "Adiós":"tito"}

If / else / elif

```
if a:  
    do this  
elif b:  
    do that  
else:  
    whatever
```



Try / Except



For Loop

```
string = "Hello World"  
for x in string:  
    print x
```

```
collection = ['hey', 5, 'd']  
for x in collection:  
    print x
```

```
list_of_lists = [ [1, 2, 3], [4, 5, 6], [7, 8, 9] ]  
for list in list_of_lists:  
    for x in list:  
        print x
```



While Loop

```
n = raw_input("Please enter 'hello':")
while n.strip() != 'hello':
    n = raw_input("Please enter 'hello':")
```

```
while True:
    n = raw_input("Please enter 'hello':")
    if n.strip() == 'hello':
        break
```

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-

print "Hello World"
```

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-

class Hello:
    def __init__(self,nombre):
        self.nombre = nombre

    def say_hello(self):
        print "Hello %s"%(self.nombre)

Hello("Hector").say_hello()
```

```
#include <iostream>

int main(int argc, char const *argv[]) {
    /* code */
}
```

```
#include <iostream>

class hello{

public:
    void say_hello(){
        std::cout<<"Hello World \n";
    }
};

int main(int argc, char const *argv[]) {
    /* code */

    hello h = hello();
    h.say_hello();

    return 0;
}
```


Strings Stuff

Accediendo al contenido

- Accediendo por índices:
 - ◆ `string[0]`
- Accediendo rangos:
 - ◆ `string[0:6]`
- A la inversa:
 - ◆ `string[-2]`

Strip y Split

`split(".")` → Separa las palabras según un carácter

`strip("0")` → Elimina el carácter especificado de la cadena

Contenido dinámico

`nombre = raw_input()`

`saludos = "Buenas tardes %s" % nombre`

Conversion	Meaning
'd'	Signed integer decimal.
'i'	Signed integer decimal.
'o'	Signed octal value.
'u'	Obsolete type – it is identical to 'd'.
'x'	Signed hexadecimal (lowercase).
'X'	Signed hexadecimal (uppercase).
'e'	Floating point exponential format (lowercase).
'E'	Floating point exponential format (uppercase).
'f'	Floating point decimal format.
'F'	Floating point decimal format.
'g'	Floating point format. Uses lowercase exponential format if exponent is less than -4 or not less than precision, decimal format otherwise.
'G'	Floating point format. Uses uppercase exponential format if exponent is less than -4 or not less than precision, decimal format otherwise.
'c'	Single character (accepts integer or single character string).
'r'	String (converts any Python object using <code>repr()</code>).
's'	String (converts any Python object using <code>str()</code>).
'%'	No argument is converted, results in a '%' character in the result.

<https://docs.python.org/2.7/library/stdtypes.html#string-formatting-operations>

Listas Stuff

- ❖ Midiendo tamaños de listas:
 - `len(lista)`
- ❖ Accediendo valores de la lista:
 - `lista[0]`
- ❖ Agregando valores de la lista:
 - `lista[1] = "Python"`
- ❖ Agregando elementos al final de la lista:
 - `lista.append(variable)`
- ❖ Eliminando elementos de la lista:
 - `del lista[9]`

Diccionarios Stuff

- Listando las claves de los diccionarios:
 - ◆ `diccionario.keys()`
- Accediendo a los valores de diccionarios:
 - ◆ `diccionario[key]`
- [RE]Asignando valores:
 - ◆ `diccionario[key] = "Adios"`
- Obteniendo la longitud de los diccionarios:
 - ◆ `len(diccionario)`

Let's do the terminal stuff





I don't know what to do

- ❖ Pedir entrada por teclado hasta que el usuario introduzca la palabra exit.
- ❖ Pedir una serie de datos por teclado y almacenarlos en una lista. Imprimiendo esta última antes de salir del programa.
- ❖ Crear un programa que lleve el registro del nombre, apellidos y peso de los alumnos de un instituto.

Funciones

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-

def ooo_some_code():
    print ("Hello World \n")

ooo_some_code()
```

```
#!/usr/bin/env python
#-*- coding:utf-8 -*-

name=raw_input("¿Cual es tu nombre?")

def say_hello(nombre):
    print ("Hello %s"%(nombre))

say_hello(name)
```

Variables / For / listas / diccionarios/ TODO JUNTO :-)

Crea un programa que imprima los datos de nombre de los alumnos con sus índices de masa corporal. Este programa tiene que incluir:

- Utilizar funciones
- Utilizar diccionarios
- :-)

0.5	0.25	0.5	0.25	1.00	SUSPENSO
0.75	0.5	0.5	N.P.	1.75	SUSPENSO
0	N.P.	0	N.P.	0.00	SUSPENSO
0.25	0.25	0.25	N.P.	0.75	SUSPENSO
1.25	0.25	1	N.P.	2.50	SUSPENSO
1	0	0	N.P.	1.00	SUSPENSO
0.75	0	0.5	0.25	1.50	SUSPENSO
0.75	0	0.75	N.P.	1.50	SUSPENSO
0.25	0.75	0.75	N.P.	1.75	SUSPENSO
2.25	0.75	0.75	N.P.	3.75	SUSPENSO
1.75	N.P.	0	N.P.	1.75	SUSPENSO
0.25	N.P.	1	0	1.25	SUSPENSO
1.5	1.5	0.75	N.P.	3.75	SUSPENSO
0	N.P.	N.P.	N.P.	0.00	SUSPENSO
N.P.	0	0	N.P.	0.00	SUSPENSO
0.25	0	0.75	N.P.	1.00	SUSPENSO
1.75	0.25	1	1.5	4.50	SUSPENSO
2.5	1	0.5	N.P.	4.00	SUSPENSO
0.75	0.25	0	0	1.50	SUSPENSO
1.25	0.25	0.25	0.5	3.25	SUSPENSO
2.5	0.5	0.5	0.75	4.25	SUSPENSO
1.25	0.25	0	N.P.	1.50	SUSPENSO
0	0.25	0.75	N.P.	1.00	SUSPENSO
1.75	1	2	1	5.75	APROBADO
0	0	0.75	N.P.	0.75	SUSPENSO
0	0.25	1	N.P.	1.25	SUSPENSO
1	N.P.	1	0	2.00	SUSPENSO
1.5	0.25	1.25	0.25	3.25	SUSPENSO
0	N.P.	N.P.	N.P.	0.00	SUSPENSO
0.5	N.P.	N.P.	N.P.	0.50	SUSPENSO
1.25	0.25	2	N.P.	3.50	SUSPENSO
1.75	0.75	0.75	N.P.	3.25	SUSPENSO
1.25	0.25	1.75	0.25	3.50	SUSPENSO
2.25	1.5	0.25	N.P.	4.00	SUSPENSO
0.5	0.25	0.75	0	1.50	SUSPENSO
0.75	0	N.P.	N.P.	0.75	SUSPENSO
0	0	0.25	0	0.25	SUSPENSO
0.25	1	0.25	N.P.	1.50	SUSPENSO
1.75	0.75	1	N.P.	3.50	SUSPENSO
1.25	0.25	1.75	0	3.25	SUSPENSO
1.75	1.25	1.75	N.P.	4.75	SUSPENSO
0.5	0.25	1.5	1	3.25	SUSPENSO
0.5	0.5	0	0	1.00	SUSPENSO
0	N.P.	N.P.	N.P.	0.00	SUSPENSO
0.75	0.25	0	N.P.	1.00	SUSPENSO
2.25	0	1.75	0.5	4.50	SUSPENSO
0.25	N.P.	0	N.P.	0.25	SUSPENSO
1	0.25	1	0	2.25	SUSPENSO
0.75	0.75	0	N.P.	1.50	SUSPENSO
0.75	0.25	1	N.P.	2.00	SUSPENSO

Meme by viki xd

Mr Chicken as a class

```
class Chicken(object):  
    "Describes a chicken."  
    def __init__(self, name):  
        self.name = name  
    def make_sounds(self):  
        "Print chicken sound."  
        print "squeaaaak!"
```

P00

CLASE

Propiedad y comportamiento
de un objeto concreto

ATRIBUTO

Propiedad del objeto

MÉTODO

Lo que un objeto
puede hacer
(algoritmo)

OBJETO

Instancia de una clase

MENSAJE

Comunicación dirigida a un
objeto ordenándole que
ejecute uno de sus métodos

Clases, Objetos etc...

```
class dog():  
    '''code '''
```

A. Definición de una clase en Python

```
class dog():  
    def new_dog(self):  
        self.born()  
        self.breathe()
```

1. Definición de funciones dentro de clases (método)

```
c = dog()  
c.new_dog()
```

I. Llamada a la clase(objeto) y a una función dentro de la misma

Our friend *self*

[show 10 more comments](#)

18 Answers

active

oldest

votes



535



The reason you need to use `self.` is because Python does not use the `@` syntax to refer to instance attributes. Python decided to do methods in a way that makes the instance to which the method belongs be *passed* automatically, but not *received* automatically: the first parameter of methods is the instance the method is called on. That makes methods entirely the same as functions, and leaves the actual name to use up to you (although `self` is the convention, and people will generally frown at you when you use something else.) `self` is not special to the code, it's just another object.

Python could have done something else to distinguish normal names from attributes -- special syntax like Ruby has, or requiring declarations like C++ and Java do, or perhaps something yet more different -- but it didn't. Python's all for making things explicit, making it obvious what's what, and although it doesn't do it entirely everywhere, it does do it for instance attributes. That's why assigning to an instance attribute needs to know what instance to assign to, and that's why it needs `self.`

Sr.No.	Method, Description & Sample Call
1	<code>__init__ (self [,args...])</code> Constructor (with any optional arguments) Sample Call : <i>obj = className(args)</i>
2	<code>__del__(self)</code> Destructor, deletes an object Sample Call : <i>del obj</i>
3	<code>__repr__(self)</code> Evaluable string representation Sample Call : <i>repr(obj)</i>
4	<code>__str__(self)</code> Printable string representation Sample Call : <i>str(obj)</i>
5	<code>__cmp__ (self, x)</code> Object comparison Sample Call : <i>cmp(obj, x)</i>

```
#!/usr/bin/env python
#-*- coding: utf-8 -*-

class Simple_class:

    homer = "ouch"

    def __init__(self, nombre):
        self.nombre = nombre
        print "Obviously it's a simple class"

    def say_hello(self):
        print "Hello %s"%(self.nombre)

    def say_goodbye(self):
        print "Goodbye %s"%(self.nombre)

    def flirt(self):
        self.say_hello()
        self.say_goodbye()
        print self.homer

objet = Simple_class("tito")
objet.flirt()
```

Herencia

La herencia nos permite que una clase herede las variables y métodos a varias subclases.

Estas subclases, aparte de los métodos propios, tiene incorporados los atributos y métodos heredados de la anterior clase (superclase)



```
1  #!/usr/bin/env python
2  # -*- coding:utf-8 -*-
3
4  class Animal:
5      def __init__(self):
6
7          print "It's an animal"
8
9      def born(self):
10         print "It's borning a new animal"
11
12     def breathe(self):
13         print "It's breathing"
14
15 class dog(Animal):
16     def new_dog(self):
17         self.born()
18         self.breathe()
19
20 c = dog()
21
22 c.new_dog()
23
```


Poliformismo

En el ámbito de la orientación a objetos el polimorfismo hace referencia a la habilidad que tienen los objetos de diferentes clases a responder a métodos con el mismo nombre, pero con implementaciones diferentes.



```
#!/usr/bin/env python
#-*- coding:utf-8 -*-

class Animal:
    def __init__(self):

        print "It's an animal"

    def born(self):
        print "It's borning a new animal"

    def breathe(self):
        print "It's breathing"

class dog(Animal):
    def __init__(self):
        self.born()
        self.breathe()
        print "It's a dog :-)"

    def look_a_cat(self):
        print "The dog try to catch the cat"

class cat(Animal):
    def __init__(self):
        self.born()
        self.breathe()
        print "It's a cat"

    def look_a_cat(self):
        print "Tha cat fell in 'love'"

rayo = dog()
rayo.look_a_cat()
bigotes = cat()
bigotes.look_a_cat()
```

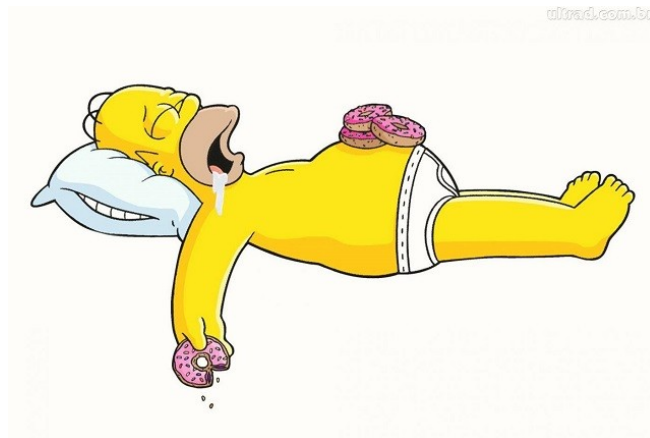
Encapsulación

La encapsulación nos permite crear funciones privadas dentro de nuestras clases, de forma que no pueden ser accesibles desde el exterior de dicha clase.

```
class Fecha
{
    public:
        /* Comentado por ineficiente
        int anho;
        int mes;
        int dia;    */

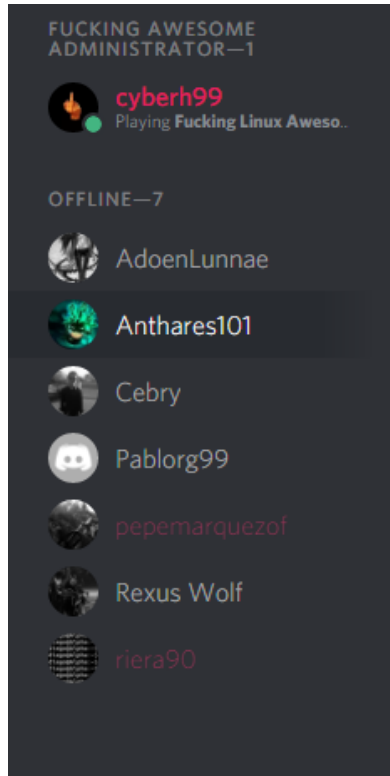
        long numeroSegundos;

        void metodoMaravilloso1();
        void metodoMaravilloso2();
};
```



```
2
3  #!/usr/bin/env python
4  #-*- coding: utf-8 -*-
5
6
7  class super_calculator:
8      def __init__(self,num1,num2):
9          print "It's too simple"
10         self.num1 = num1
11         self.num2 = num2
12
13     def __mult(self):
14
15         f_number = self.num1 * self.num2
16         return f_number
17
18     def operate(self):
19         print self.__mult()
20
21
22 casio = super_calculator(2,2)
23 casio.operate()
```

You Think something



Jurassic park python's version

```
## Animals_2a3.py -- a complete OO zoo !!           DMQ 6/10/04
'''
# Animal    -->  Reptile
# -----
# Animal    -->  Mammal    -->  Bovine
# -----
# Animal    -->  Mammal    -->  Canine
# -----
# Animal    -->  Mammal    -->  Feline    -->  Cat
# -----
# _numAnimals    _numMammals    _numFelines    _numCats
# home          genus
# __init__()    __init__()    __init__()    __init__()
#              .sound          .sound
#              .name
# show()        show()        show()        show()
#              talk()         talk()
#
'''
```

Python modules

getpass

Permite hacer capturas de texto, pero se encarga de que la entrada no se muestre por pantalla

```
#!/usr/bin/env python
#-*- coding: utf-8 -*-

from getpass import getpass

password = getpass()

print password
```


platform

Nos permite obtener
cierta información
sobre el sistema en el
cual se ejecuta el
intérprete

```
root@zeus:~/python-s-initiations/Examples# python platform_example.py
Arquitectura:
x86_64
Version de Python:

Compilador de python
GCC 7.2.0
Sistema:
Linux
Uname information:
('Linux', 'zeus', '4.14.0-kali3-amd64', '#1 SMP Debian 4.14.13-1kali1 (2018-01-25)', 'x86_64', '')
root@zeus:~/python-s-initiations/Examples#
```

Módulos para trabajar en redes

- ❖ Conexiones hacia cualquier servidor
 - socket
 - scrapy
- ❖ Conexiones web (trabajo con web)
 - urllib2
 - request
 - beautifulshoup

Graphical interface	wxPython	http://wxpython.org
Graphical interface	pyGtk	http://www.pygtk.org
Graphical interface	Pmw	http://pmw.sourceforge.net/
Graphical interface	Tkinter 3000	http://effbot.org/zone/wck.htm
Graphical interface	Tix	http://tix.sourceforge.net/
Database	MySQLdb	http://sourceforge.net/projects/mysql-python
Database	PyGreSQL	http://www.pygresql.org/
Database	Gadfly	http://gadfly.sourceforge.net/
Database	SQLAlchemy	http://www.sqlalchemy.org/
Database	kinterbasdb	http://kinterbasdb.sourceforge.net/
MSN Messenger	msnlib	http://auriga.wearlab.de/~alb/msnlib/
MSN Messenger	pymn	http://telepathy.freedesktop.org/wiki/Pymn
MSN Messenger	msnp	http://msnp.sourceforge.net/
Network	Twisted	http://twistedmatrix.com/
Images	PIL	http://www.pythonware.com/products/pil/
Images	gdmodule	http://newcenturycomputers.net/projects/gdmodule.html
Images	VideoCapture	http://videocapture.sourceforge.net/
Sciences and Maths	scipy	http://www.scipy.org/

<https://www.catswhocode.com/blog/python-50-modules-for-all-needs>



Relax.. Last diapo

