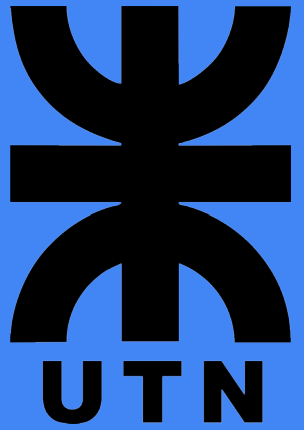
Cátedra de Ingeniería de Software de Fuentes Abiertas / Libres

Reglas de diseño y programación



Integrantes:

Chincho, Braian Ismael	63672	braianchincho@gmail.com
Prado Macat, Federico	65221	fpradomacat@gmail.com
Pucheta Moyano, Fernanda Noel	67356	puchetafernanda@gmail.com

Temas a tratar:

- Código de conducta
- Reglas de diseño
- Reglas de programación

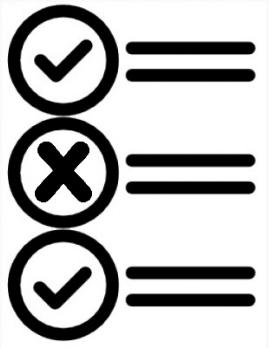
Código de conducta

¿Qué “reglas” creen que deberían existir en una comunidad de Software Libre para permitir el trabajo en equipo?

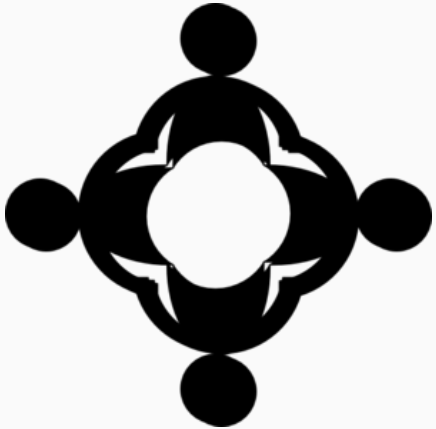
Código de conducta

¿Qué es?

Documento



Normas, violaciones y sanciones

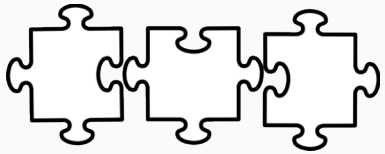


Comunidad

Código de conducta - Introducción

¿Por qué?

- Trabajo en equipo

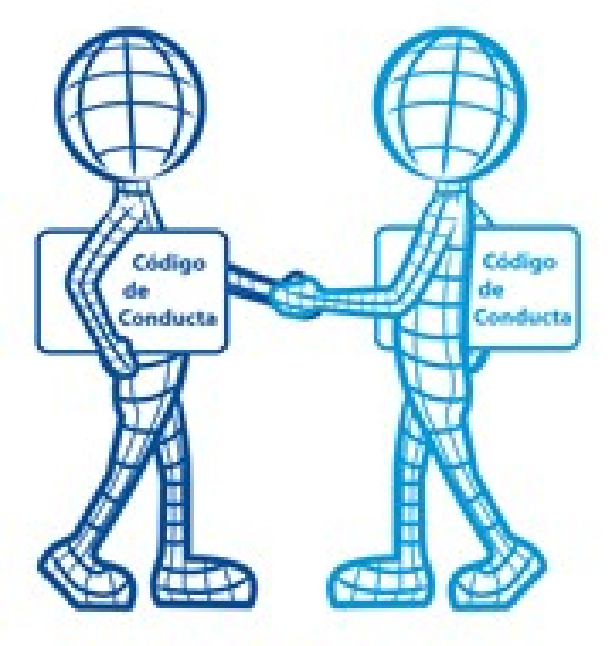


- Trato con respeto, saludable y constructivo


- Tomar medidas


Código de conducta - Reglas de Debian

- Ser respetuoso
- Asumir buena fe
- Ser colaborador
- Expresarse concisamente
- Ser abierto



Código de conducta - Reglas de FreeBSD

1. Respetar otros *commiters*
2. Respetar otros *contribuyentes*
3. Discutir cualquier cambio antes de *commitear*
4. Respete a los *mantenedores* existentes
5. Cualquier cambio disputado debe ser *vuelto atrás* en espera de la resolución de la disputa si lo solicita un mantenedor.
6. Los cambios van a FreeBSD-CURRENT antes que a FreeBSD-STABLE.

Código de conducta - Reglas de FreeBSD

7. No pelear en público con otros committers, se ve mal
8. Respetar los *code freezes* y estar al tanto
9. Cuando haya dudas de un procedimiento, preguntar primero!
10. Testear cambios antes de *subirlos*
11. No commitear en *src/contrib*, *src/crypto*, and *src/sys/contrib*

Reglas de diseño
Filosofía Unix

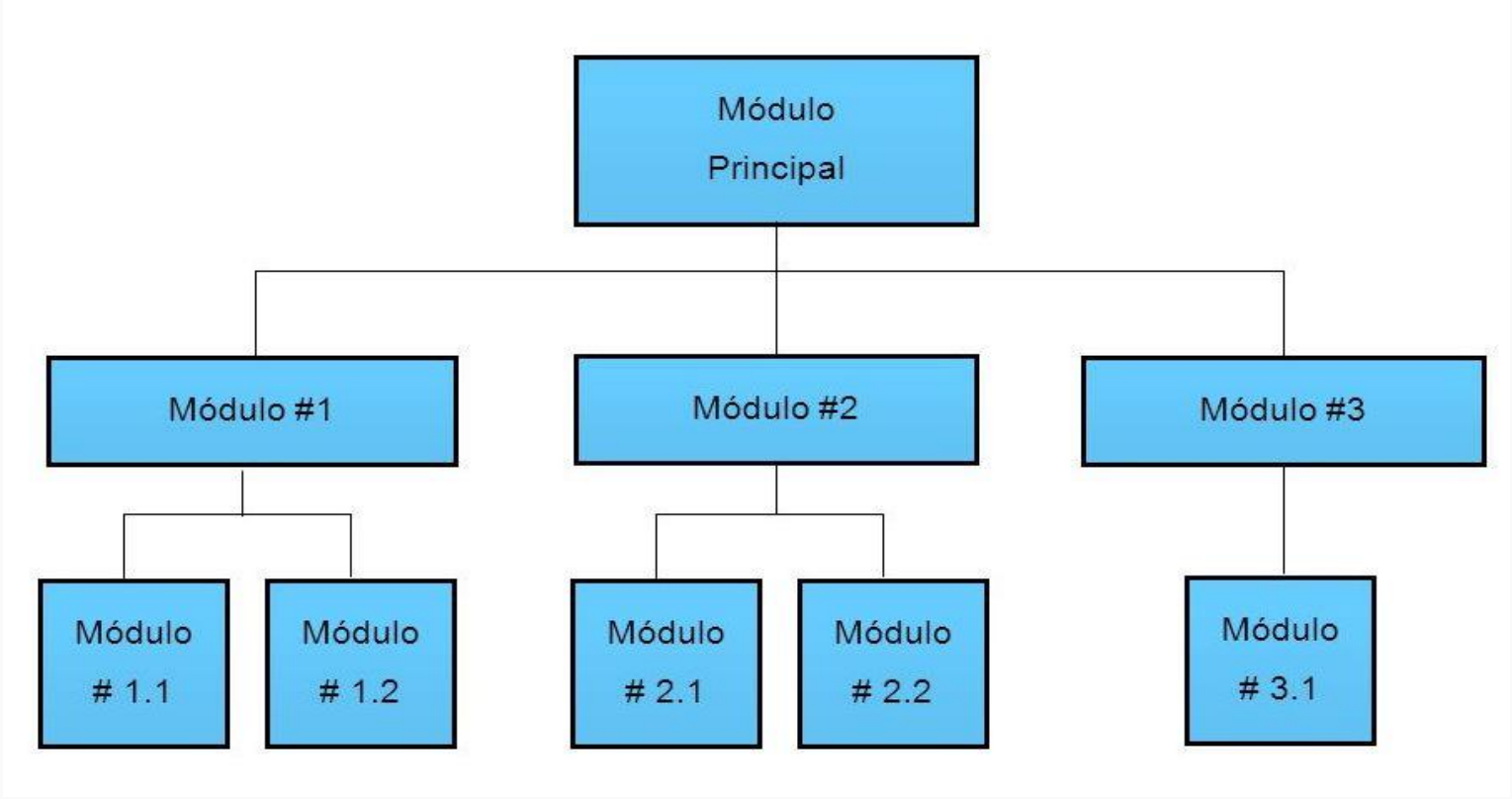
Guía para la buena programación de software.

Es una serie de reglas básicas a tener en cuenta ante cualquier diseño de software

Reglas de diseño

¿Conocen alguno de los principios de la filosofía de Unix o reglas de diseño?

Modularidad



Claridad

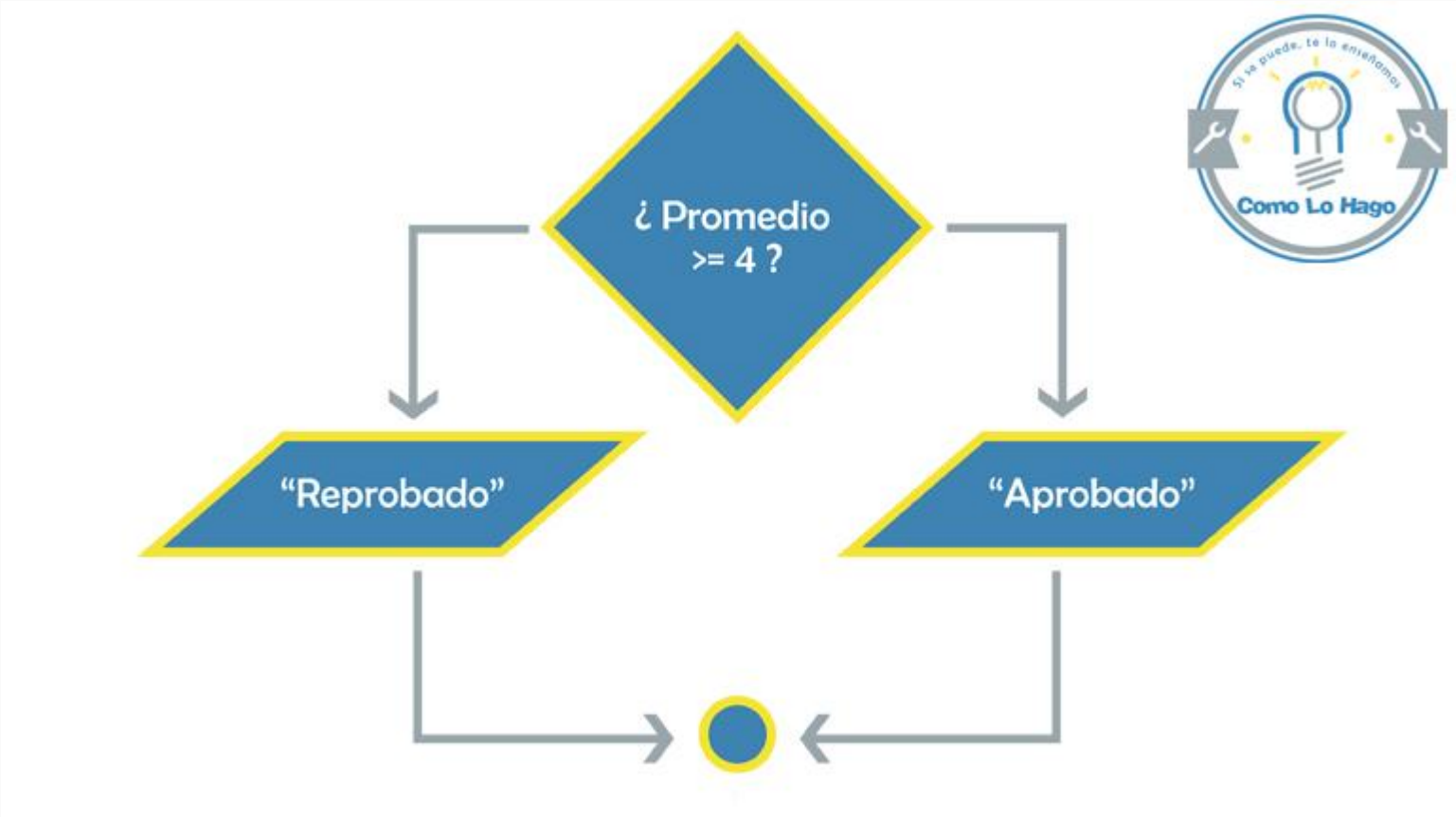
Economía

Separación

FRONTEND

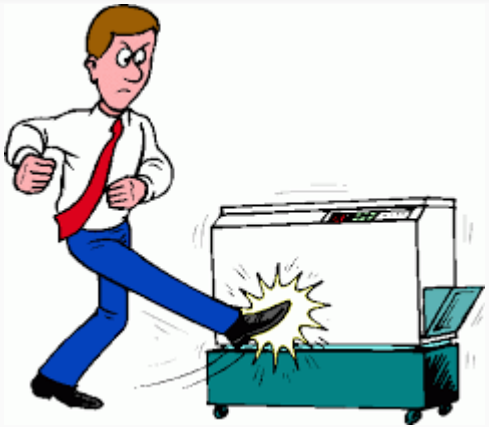
BACKEND

Simplicidad

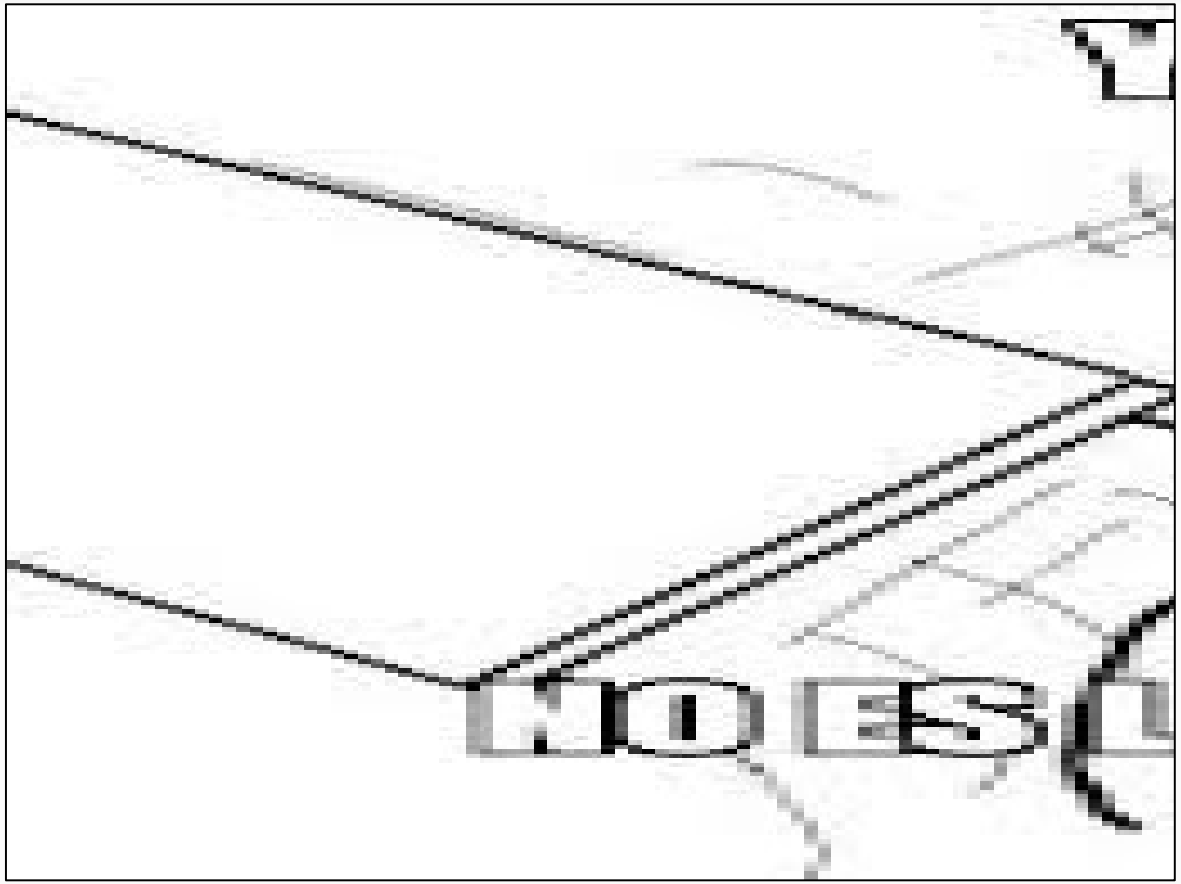


Mínima sorpresa

Reparación



Diversidad



Extensibilidad

Todo esto puede resumirse en...

Principio KISS:

“Keep it short and simple”

or…

"Aquellos que no pueden entender Unix, están condenados a reinventarlo, pobremente"

- Henry Spencer, 1987

Reglas de programación

¿En qué piensan cuando hablamos de reglas de programación?

Nombre apropiado para las variables

```
2  if a < 24 and b < 60 and c < 60
3    return true
4  else
5    return false
6
7  if horas < 24 and minutos < 60 and segundos < 60
8    return true
9  else
10   return false
11
12
```

NO

SI

Código comentado

```
12  class Boton(BaseInterfaz):
13      """Representa un botón que se puede pulsar y disparar una acción."""
14
15      def __init__(self, texto="Sin texto", x=0, y=0, icono=None):
16          """Inicializa al actor.
17
18          :param texto: Texto que se mostrará dentro de botón.
19          :param x: Posición horizontal inicial.
20          :param y: Posición vertical inicial.
21          :param icono: Imagen que se mostrará sobre el botón.
22          """
23          BaseInterfaz.__init__(self, x=x, y=y)
24          self.texto = texto
25          self._crear_imagenes_de_botones()
26          self.centro = ("centro", "centro")
27          self.funcion = None
28          self.fijo = True
29
30          if icono:
31              self.icono = pilas.imagenes.cargar(icono)
32          else:
33              self.icono = None
```

Código Indentado

Reglas de prc

```
for (exp1;exp2;exp3) {
    sentencia1;
    sentencia2;
    ...
    while (condición_1) {
        sentencia1;
        sentencia2;
        ...
        if (condición) {
            sentencia1;
            sentencia2;
            ...
            while (condición_2) {
              sentencia1;
              sentencia2;
              ...
            } /*while (condición_2)*/

        } /*if*/
        else {
            sentencia1;
            sentencia2;¿

        } /*else*/
    } /*while (condición_1)*/
} /*for*/
```

Longitud de línea menor a 80 columnas

Reglas de

```
# NINJA-IDE is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with NINJA-IDE; If not, see <http://www.gnu.org/licenses/>.
from __future__ import absolute_import
from __future__ import unicode_literals

import os
import re
import uuid

from PyQt5.QtWidgets import (
    QWidget,
    QVBoxLayout
)
from PyQt5.QtCore import Qt, QTimer
from PyQt5.QtQuickWidgets import QQuickWidget

from ninja_ide import resources
from ninja_ide.gui.ide import IDE
from ninja_ide.tools import ui_tools
from ninja_ide.tools.locator import locator
from ninja_ide.tools.logger import NinjaLogger
logger = NinjaLogger(__name__)
```

Regla de Nombrado

Reglas de programación

```
int ThisVariableIsATemporaryCounter = 0;
```

```
int tmp = 0;
```

```
int count_active_users()
```

```
int cau()
```

Guías de estilos

- **PEP 8 (python enhancement proposal)-- Style Guide for Python Code**
- **AmbySoft Inc. Coding Standards for Java v17.01d**
- **Philips Healthcare - C# Coding Standard**
- **Mozilla Coding Style Guide**

Zen of Python

```
   import this
   """The Zen of Python, by Tim Peters. (poster by Joachim Jablon)"""

 1 Beautiful is better than ugly.
 2 Explicit is better than impl..
 3 Simple is better than cσмplεx.
 4 Cσмplεx is better than cOmp1|c@ted.
 5 Flat is better than nested.
 6 S p a r s e  is better than dense.
 7 Readability counts.
 8 Special cases aren't special enough to break the rules.
 9 Although practicality beats purity.
10 raise PythonicError("Errors should never pass silently.")
11 # Unless explicitly silenced.
12 In the face of ambiguity, refuse the temptation to guess.
13 There should be one-- and preferably only one --obvious way to do it.
14 # Although that way may not be obvious at first unless you're Dutch.
15 Now is better than ...                    never.
16 Although never is often better thanrightnow.
17 If the implementation is hard to explain, it's a bad idea.
18 If the implementation is easy to explain, it may be a good idea.
19 Namespaces are one honking great idea -- let's do more of those!
```

¿Preguntas?

Bibliografía

Código de Conducta

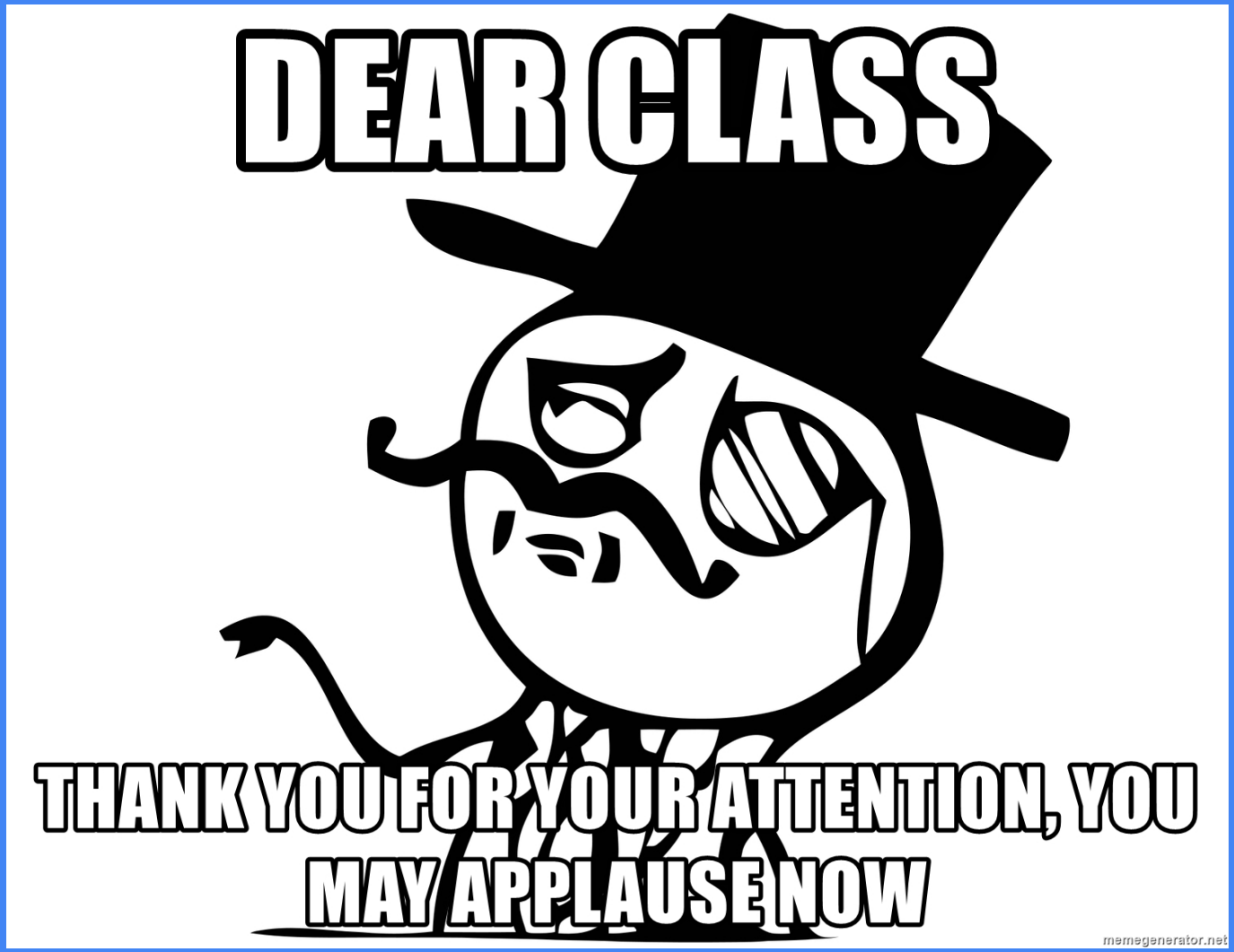
- Contributor Covenant : https://www.contributor-covenant.org/
- Code of Conduct (GitHub): https://opensource.guide/es/code-of-conduct/
- FreeBSD Committers' Big List of Rules: https://www.freebsd.org/doc/en_US.ISO8859-1/articles/committers-guide/rules.html
- Django Code of Conduct: https://www.djangoproject.com/conduct/

Reglas de diseño

- Filosofia unix: https://www.educadictos.com/la-filosofia-unix/

Guías de estilos de programación

- https://www.python.org/dev/peps/pep-0008/#public-and-internal-interfaces
- https://web.archive.org/web/20070814042731/http://www.mozilla.org/hacking/mozilla-style-guide.html
- http://tics.tiobe.com/viewerCS/index.php?CSTD=General
- http://www.ambysoft.com/essays/javaCodingStandards.html


DEAR CLASS
THANK YOU FOR YOUR ATTENTION, YOU MAY APPLAUSE NOW
memegenerator.net