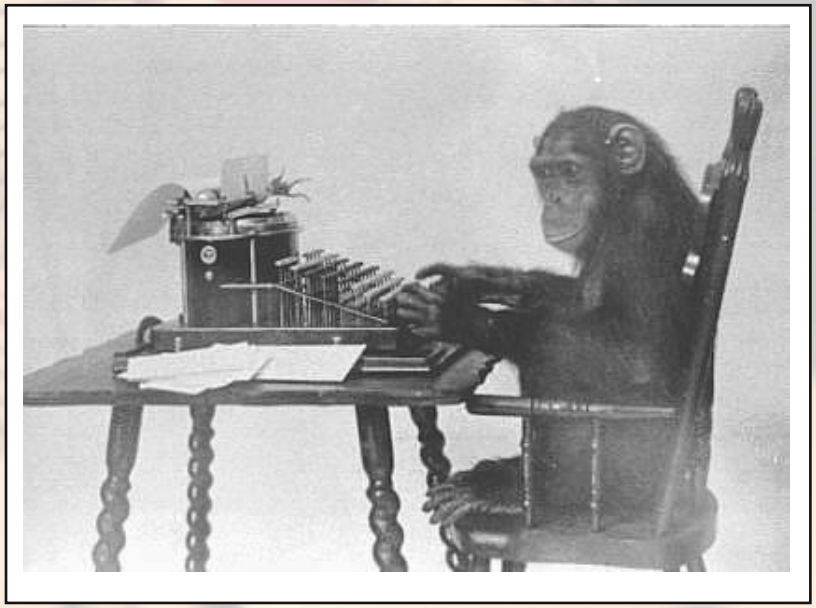
Generación dinámica de documentos con Appy/POD

Juan Ignacio Rodríguez de León ~ @jileon en twitter ~ euribates@gmail.com



¿Qué es POD?

POD (Python Open Document) es una librería python que permite generar documentos con contenidos dinámicos.

Es parte de un *framework* llamando Appy (*Applications in python*)

euribates@gmail.com

¿Cómo funciona?

1) Se crea una plantilla en formato ODF (Open Document Format)

¿Cómo funciona?

1) Se crea una plantilla en formato ODF (Open Document Format)

2) Insertamos código dentro

¿Cómo funciona?

1) Se crea una plantilla en formato ODF (Open Document Format)
2) Insertamos código dentro
3) Llamamos a pod con la plantilla y un conjunto de objetos Python, obteniendo un documento ODF con el resultado deseado

euribates@gmail.com

Otros formatos

- Se puede usar el LibreOffice en modo de servidor para convertir el ODT resultante en cualquier formato que LibreOffice pueda exportar: RTF, Documentos Word, PDF, etc...
- No lo veremos en esta presentación, pero se puede hacer
- Para generar en documento en formato ODT, sin embargo, **NO** es necesario ni siquiera tener instalado OpenOffice.

Hola, Mundo

Primero creamos un documento con OpenOffice/LibreOffice. Vamos a incluir una variable que definiremos por programa. Para incluir la variable en el texto del documento:

Hola, Mundo

Primero creamos un documento con OpenOffice/LibreOffice. Vamos a incluir una variable que definiremos por programa. Para incluir la variable en el texto del documento:

1) Seleccionamos “grabar cambios” (Menú Editar->Cambios->Grabar)

Hola, Mundo

Primero creamos un documento con OpenOffice/LibreOffice. Vamos a incluir una variable que definiremos por programa. Para incluir la variable en el texto del documento:

1) Seleccionamos “grabar cambios” (Menú Editar->Cambios->Grabar)

2) Escribimos el nombre de la variable

euribates@gmail.com

Hola, Mundo

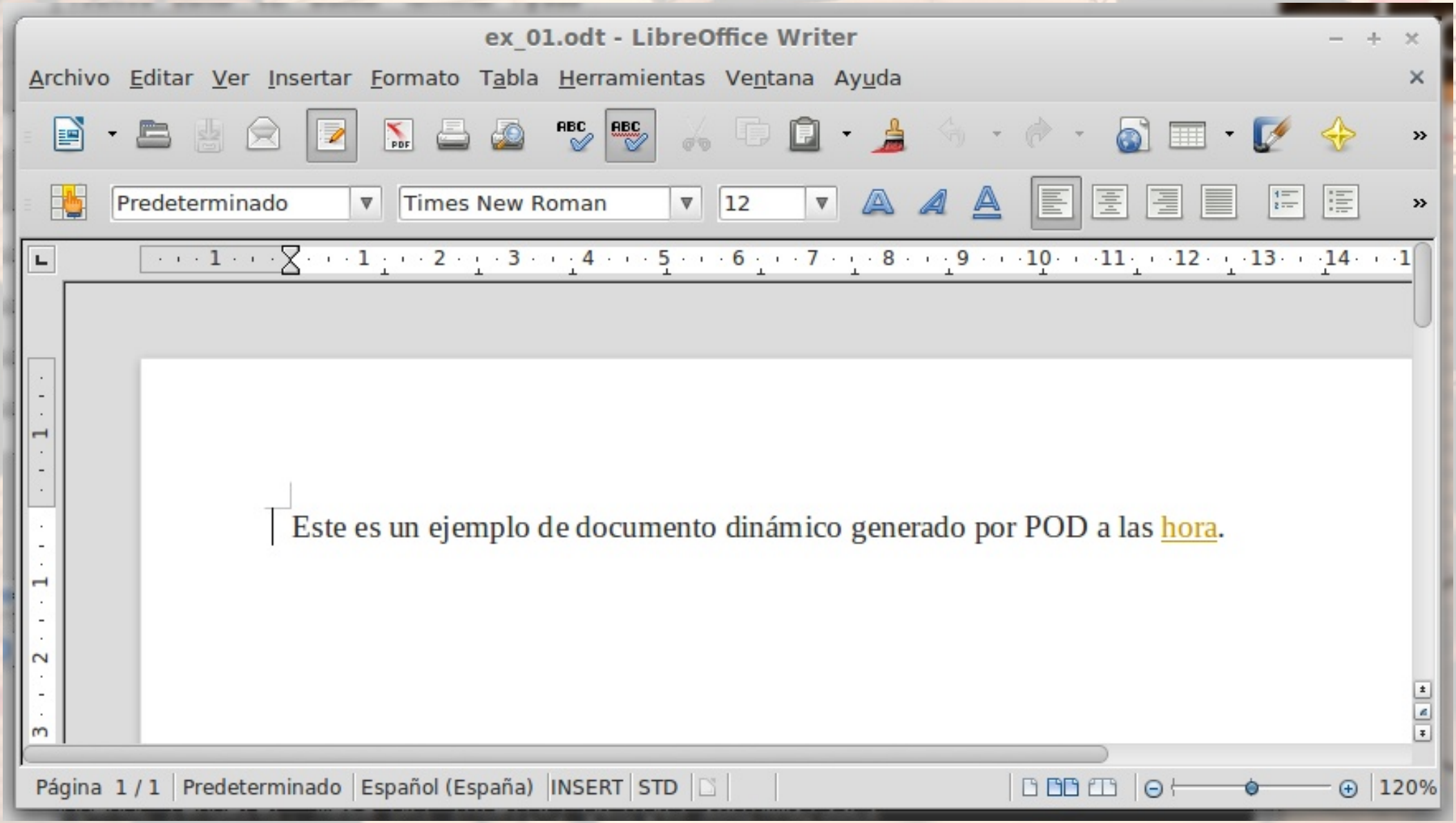
Primero creamos un documento con OpenOffice/LibreOffice. Vamos a incluir una variable que definiremos por programa. Para incluir la variable en el texto del documento:

1) Seleccionamos “grabar cambios” (Menú Editar->Cambios->Grabar)

2) Escribimos el nombre de la variable

3) Volvemos al modo normal con la misma opción del menú.

Quedaría algo así



Fusionar plantilla con datos

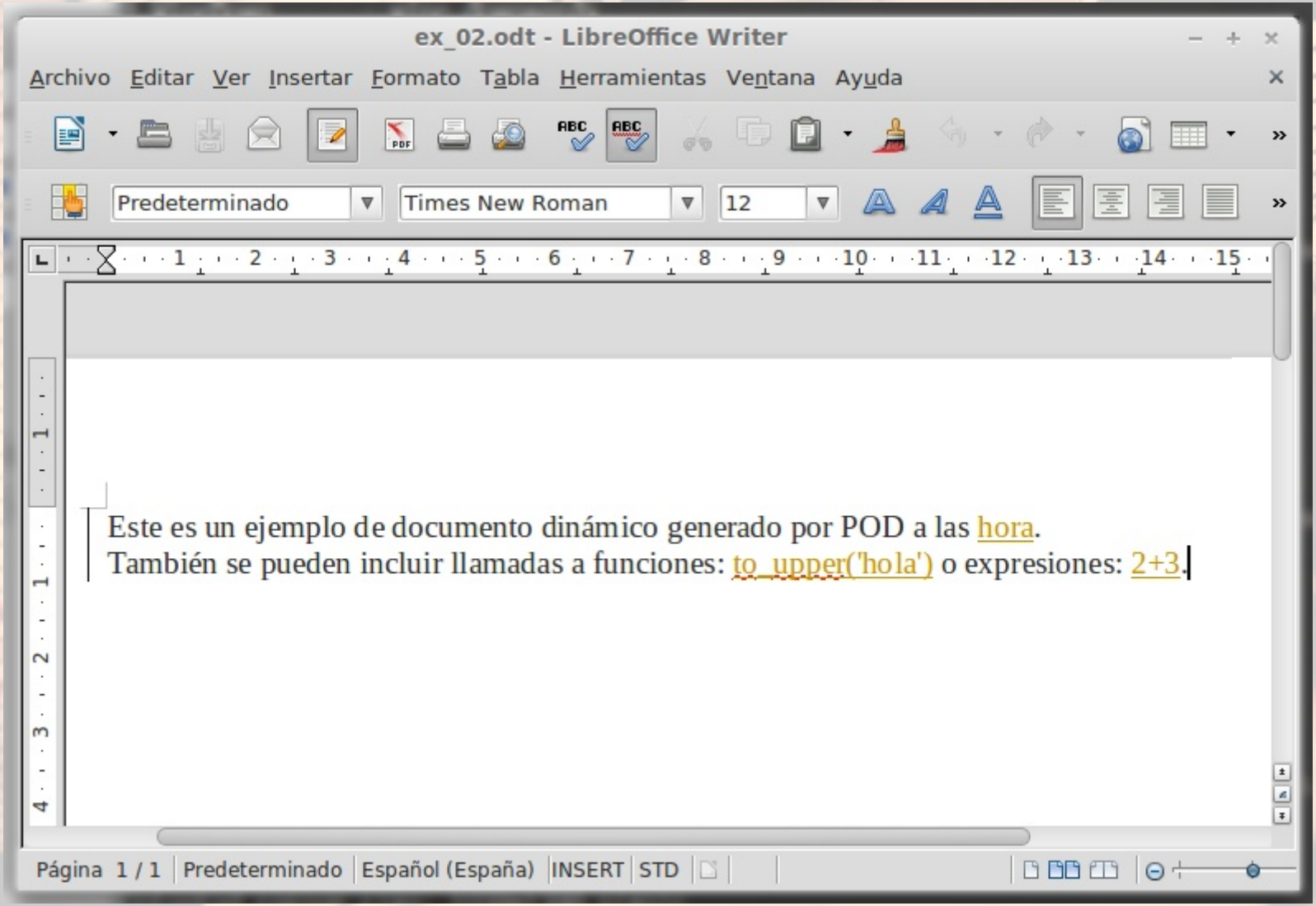
```
#!/usr/bin/env python

from appy.pod.renderer import Renderer
import datetime

hora = datetime.datetime.now()
renderer = Renderer(
    'ex_01.odt',      # Plantilla
    {'hora':hora},    # Contexto
    'out_01.odt'     # Salida
    )
renderer.run()
```

No solo variables

Podemos incluir otras cosas, además de variables:

- Valores retornados por funciones
- Expresiones
- Valores retornados por métodos de objetos
- Estructuras de datos complejas

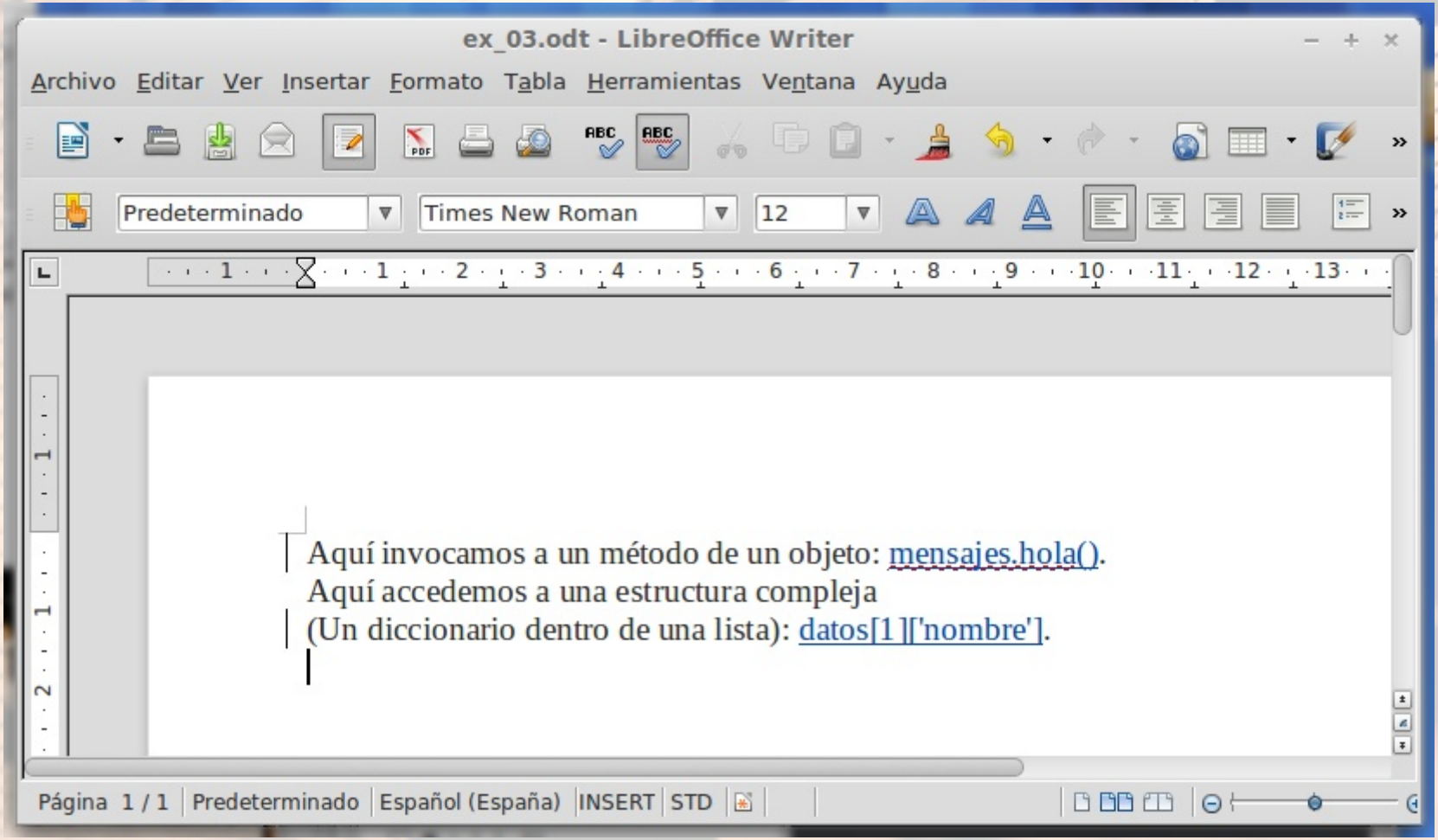
Plantilla con funciones/expresiones



Fusionar con funciones/expresiones

```
#!/usr/bin/env python

from appy.pod.renderer import Renderer
import datetime

hora = datetime.datetime.now()
def to_upper(s):
    return s.upper()
renderer = Renderer(
    'ex_02.odt',
    globals(),
    'out_02.odt'
    )
renderer.run()
```

Plantilla con métodos/estructuras de datos



Fusionar con métodos/estructuras de datos

```
#!/usr/bin/env python

from appy.pod.renderer import Renderer

class Mensajes:
    def __init__(self, msg):
        self.msg = msg

    def hola(self):
        return self.msg

mensajes = Mensajes('hola, mundo')

datos = [
    {'nombre':'Matt Murdock'},
    {'nombre':'Peter Parker'},
    ]

renderer = Renderer('ex_03.odt', globals(), 'out_03.odt')
renderer.run()
```

Estructuras de control

- Las estructuras de control se realizan mediante notas (En LibreOffice: Insertar -> Comentarios [Ctrl-Shift-C])
- El código va dentro de la nota
- Solo hay dos estructuras de control: IF y FOR

La sentencia IF

La sentencia *if* tiene la siguiente forma:

```
do <área afectada> [if <condicion python>]
```

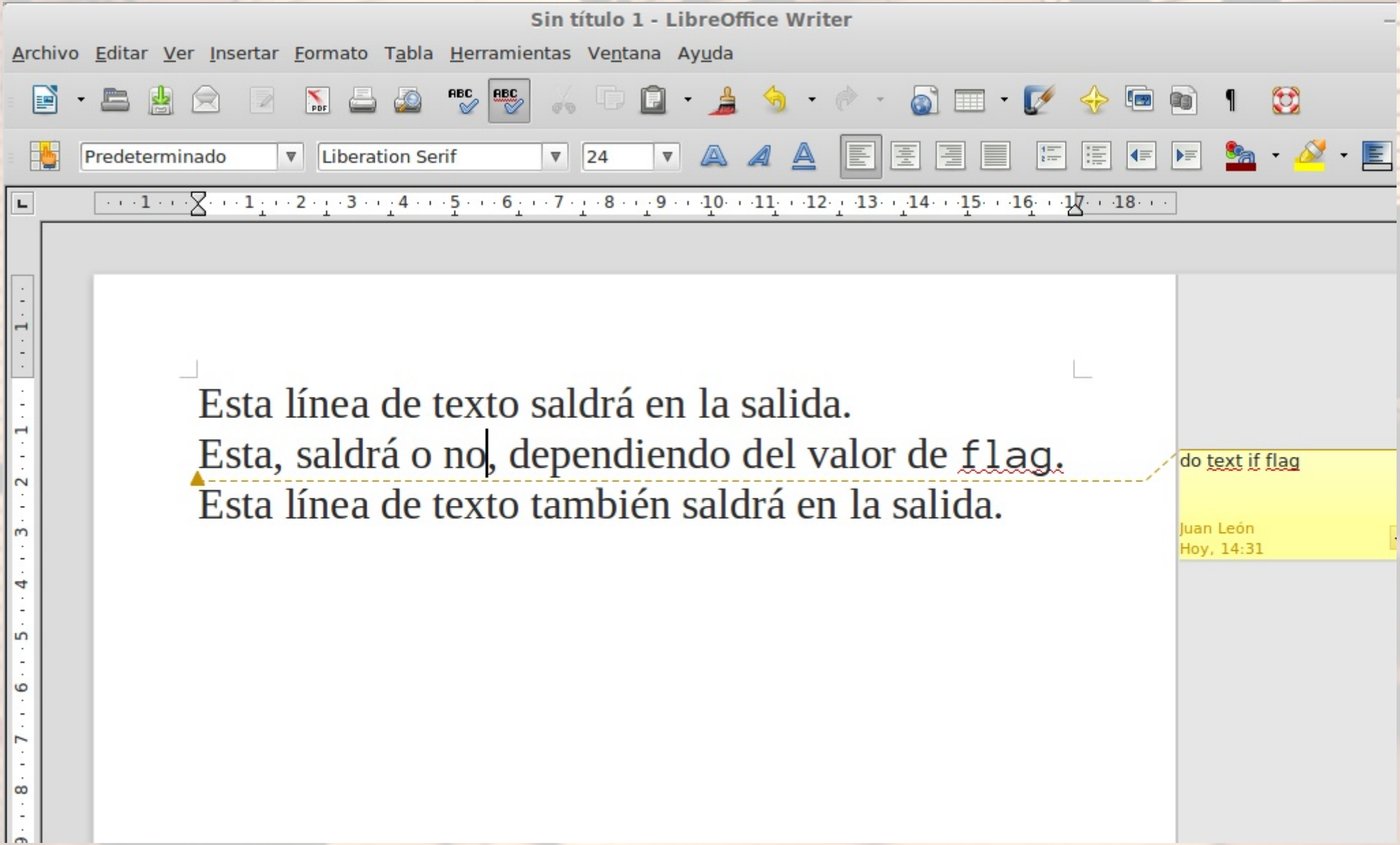
Donde el área afectada puede ser:

- `text` Todo el párrafo
- `title` El título
- `section` Toda la sección
- `table` Toda la tabla
- `row` Toda una fila de una tabla
- `cell` Una celda dentro de una tabla

La sentencia IF

Después del IF, puede venir cualquier expresión en Python, que se evaluará como un valor booleano. Si es verdadero, el área afectada aparecerá en el documento generado, si es falsa, no aparecerá.

La sentencia IF



La sentencia IF

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

from appy.pod.renderer import Renderer

flag = False

renderer = Renderer(
    'ex_04.odt',
    locals(),
    'out_04.odt',
    )
renderer.run()
```

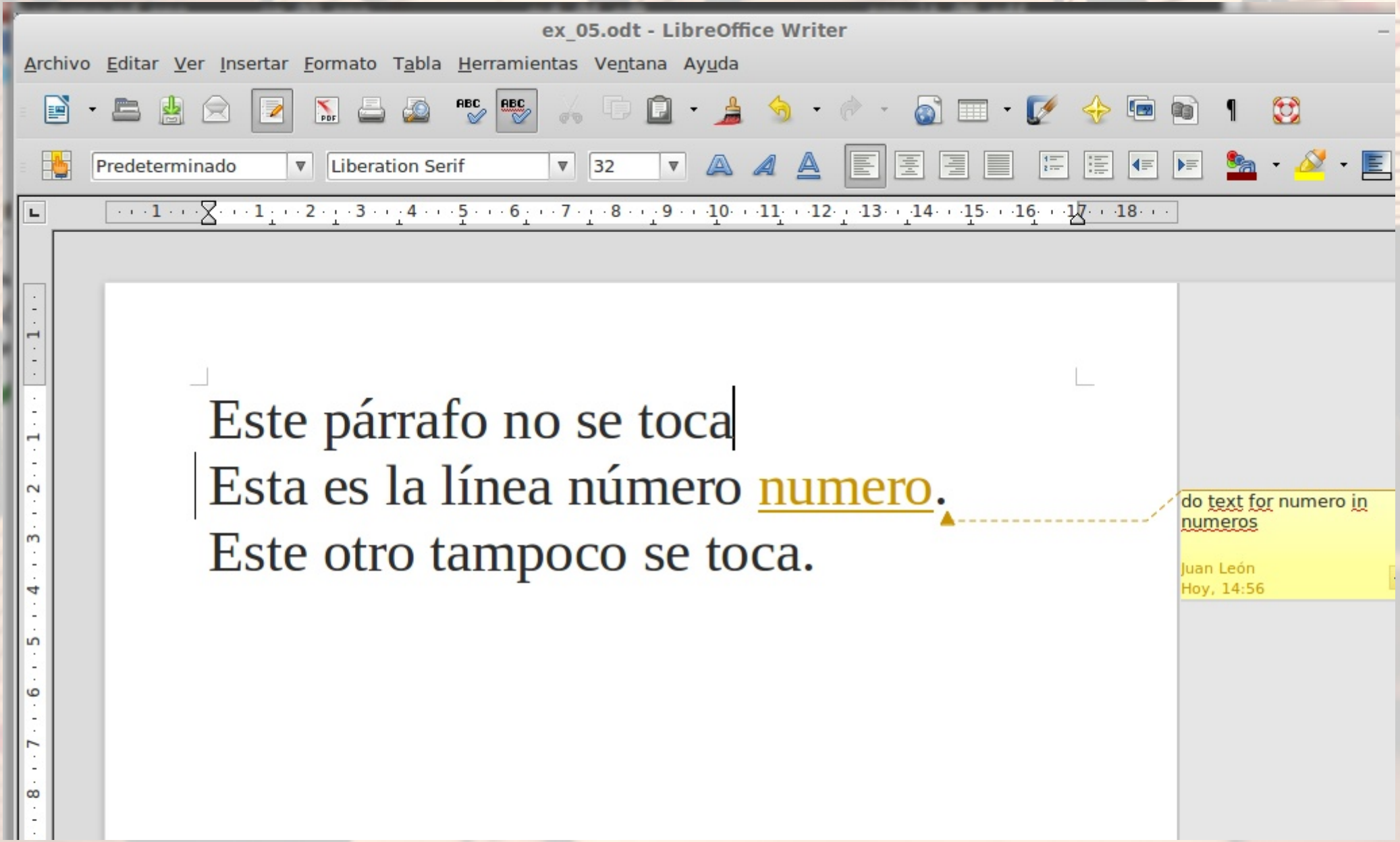
La sentencia FOR

La sentencia *for* tiene la siguiente forma:

```
do <área afectada> for <variable> in <iterador>
```

- En variable podemos usar cualquier nombre que sea válido en Python
- Servirá como la variable de la iteración, es decir, el valor que cambia en cada ciclo. Se inserta en el contexto para que sea accesible.
- En iterador nos servirá cualquier variable iterable en Python: listas, texto, tuplas, etc...
- El área de actuación es igual que en el if.

La sentencia FOR



euribates@gmail.com

La sentencia FOR

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

from appy.pod.renderer import Renderer

numeros = range(1,101)

renderer = Renderer(
    'ex_05.odt',
    locals(),
    'out_05.odt',
    )
renderer.run()
```

Insertar contenido arbitrario

- Existe una clausula especial **`from`**, que se puede añadir a las sentencias `if`/`for`
- Permite sustituir el área afectada por el resultado de la cláusula `from`

Para usar la cláusula from

- La cláusula from debe empezar en una nueva línea (Si no, pensará que es parte de la sentencia y probablemente de error)
- La cláusula debe ser una expresión válida en Python, que debe devolver un fragmento correctamente formateado en ODT.

Ejemplo de uso de la cláusula from

Ejemplo con un if

```
do text
from '<text:p>Hola, mundo</text:p>'
```

Ejemplo con un for

```
do text for i in range(3)
from '<text:p>Hola, mundo %d</text:p>' % i
```

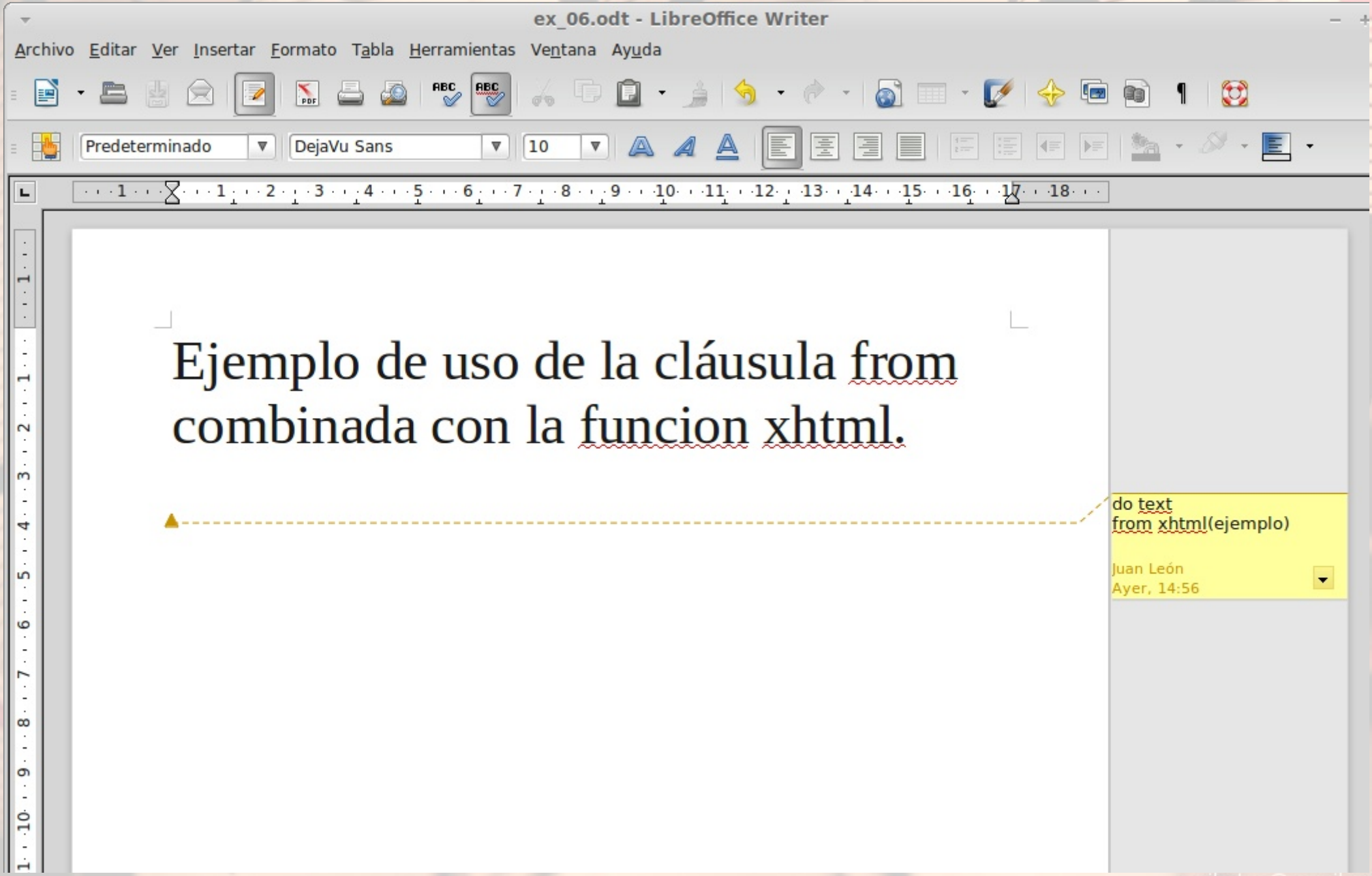
La pega: Hay que conocer el formato interno de ODT (El *resumen* tiene 303 páginas)

Por eso se inventó...

La función xhtml

- Esta función nos permite convertir de Html a formato ODT, preparado para ser incrustado con la cláusula `from`.
- Si utilizamos clases en nuestro Html, y existe un estilo con el mismo nombre en la plantilla, se usará ese estilo.
- Los tags h1..h6 se mapean a los estilos Heading1..Heading6.

Ejemplo 6 ~ Uso de from/xhtml ~ Plantilla



Ejemplo 6 ~ Uso de from/xhtml ~ Programa

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

from appy.pod.renderer import Renderer
ejemplo = '''
<h2>Ejemplo</h2>
<p>Un párrafo con <b>negritas</b>, <i>itálicas</i>, subíndices:
H<sub>2</sub>O, exponentes:  2·π·r<sup>2</sup> y de postre un enlace:
<a href="http://www.python.org/">www.python.org</a>.</p>
<ul>
    <li>Uno</li>
    <li>Dos</li>
    <li>Tres</li>
      <ul>
      <li>Tres.Uno</li>
      <li>Tres.Dos</li>
      </ul>
</ul>
'''
renderer = Renderer('ex_06.odt', locals(), 'out_06.odt',)
renderer.run()
```

la función document

- La función document permite integrar dentro del resultado imágenes o ficheros externos.
- La función tiene la siguiente signatura:

```
document(content=None, at=None, format=None, anchor='as-char')
```

- Se puede usar para incrustar contenidos que estén en memoria (porque los recuperamos de una base de datos, por ejemplo) o que estén en el sistema de ficheros.

la función document para contenidos en memoria

Si el contenido está en memoria:

- **`content`** puede ser o una variable cuyo valor son los datos binarios, o un objeto de tipo file, abierto
- **`format`** puede ser o un tipo MIME o un extensión de fichero (sin punto) de alguno de los formatos aceptables: Un documento ODT, un documento PDF, una imagen PNG, una imagen JPEG o una imagen GIF

la función document para contenidos en ficheros

Si el contenido está en un fichero:

- El parámetro **`at`** indica la ruta dentro del sistema de ficheros de la imagen o documento a incrustar.
- los parámetros `content` y `format` no son necesarios.

El parámetro anchor de la función document

Este parámetro se usa solo para las imágenes, y determina la forma en que la imagen se ancla en el documento resultante. Los posibles valores son:

Valor del parámetro	Descripción
`page`	A la página
`paragrap`	Al párrafo
`char`	Al carácter
`as-char`	Como un carácter

El código de los ejemplos está disponible

El contenido de esta presentación, tanto en pdf como en su fuente original para scribus, los ficheros auxiliares, documentos y scripts de ejemplo están disponibles públicamente en:

```
hg clone https://bitbucket.org/euribates/presentacion-appy
```

Créditos y agradecimientos

La fotografía usada en el fondo es de:

Luke Robinson

(mortalcoil)

http://www.flickr.com/photos/mortalcoil/