LibreOffice

Guía de Calc

Capítulo 12
Macros de Calc

Automatizar tareas repetitivas

Derechos de autor

Colaboradores

Este libro está adaptado de versiones anteriores del mismo.

De esta edición

Steve Fanning
Jean Hollis Weber
José María López Sáez
Juan C. Sanz

De ediciones previas

Andrew Pitonyak
Barbara Duprey
Jean Hollis Weber
Simon Brydon

Comentarios y sugerencias

Puede dirigir cualquier clase de comentario o sugerencia acerca de este documento a: documentation@es.libreoffice.org.

Nota

Todo lo que envíe a la lista de correo, incluyendo su dirección de correo y cualquier otra información personal que escriba en el mensaje se archiva públicamente y no puede ser borrada

Fecha de publicación y versión del programa

Versión en español publicada el 3 de marzo de 2021. Basada en la versión 6.2 de LibreOffice.

Uso de LibreOffice en macOS

Algunas pulsaciones de teclado y opciones de menú son diferentes en macOS de las usadas en Windows y Linux. La siguiente tabla muestra algunas sustituciones comunes para las instrucciones dadas en este capítulo. Para una lista detallada vea la ayuda de la aplicación.

Windows o Linux	*Equivalente en Mac*	*Efecto*
Herramientas > Opciones opción de menú	**LibreOffice > Preferencias**	Acceso a las opciones de configuración
Clic con el botón derecho	*Control+clic* o *clic derecho* depende de la configuración del equipo	Abre menú contextual
Ctrl (Control)	⌘ *(Comando)*	Utilizado con otras teclas
F5	*Mayúscula*+⌘+*F5*	Abre el navegador
F11	⌘+*T*	Abre la ventana de estilos y formato

Contenido

Introducción

El *capítulo 13* de la *Guía de primeros pasos* (titulado *Primeros pasos con Macros*) es una introducción a las características de las macros que están disponibles en LibreOffice. El presente capítulo aporta más información introductoria sobre el uso de las macros dentro de una hoja de cálculo de Calc.

Una macro es un conjunto de comandos o pulsaciones de teclas que son almacenadas para un uso futuro. Un ejemplo de una simple macro es una en la que se ingresa la dirección en la celda actual de una hoja de cálculo abierta. También se pueden usar las macros para automatizar tareas, tanto simples como complejas, y permiten introducir nuevas funciones que no están incluidas en Calc.

La manera más sencilla de crear una macro es grabar una serie de acciones por medio de la interfaz de usuario de Calc. Calc guarda las macros grabadas empleando el lenguaje de programación de código abierto LibreOffice Basic, que es un dialecto del famoso lenguaje de programación BASIC. Estas macros pueden editarse y mejorarse, después ser grabadas, con el Entorno de Desarrollo integrado (IDE, por sus siglas en inglés) incorporado.

Las macros de Calc más poderosas se crean utilizando el código escrito por medio de uno de los cuatro lenguajes de programación compatibles (*LibreOffice Basic*, *BeanShell*, *JavaScript* y *Python*). Este capítulo ofrece un resumen de las características de las macros en Calc y se enfoca principalmente en el lenguaje de programación de macros por defecto, *LibreOffice Basic*. Se incluyen ejemplos para los lenguajes de programación *BeanShell*, *JavaScript* y *Python*, pero descripciones más completas de las características para estos lenguajes están fuera del alcance de este documento.

Cómo utilizar la grabadora de macros

El *capítulo 13* de la *Guía de primeros pasos* incluye ejemplos que muestran cómo utilizar la grabadora de macros y ayudan a comprender las secuencias de comandos generadas por *LibreOffice Basic*. Los siguientes pasos dan otro ejemplo más específico de una hoja de cálculo Calc, sin las explicaciones tan detalladas de la *Guía de los primeros pasos*. Se crea y se guarda una macro que lleva a cabo un pegado especial con la operación multiplicar a lo largo de un rango de celdas de hojas de cálculo.

1) Seleccione **Herramientas > Opciones > LibreOffice > Avanzado** desde la *barra de menú* y seleccione la opción **Activar grabación de macros (limitada)** para permitir la grabación de la macro. Haga clic en el botón **Aceptar**.
2) Seleccione **Archivo > Nuevo > Hoja de cálculo** de la *barra de menú* para crear una nueva hoja de cálculo.
3) Ingrese los números que se muestran en la *Figura 1* en las celdas A1:C3 en la primera hoja de la hoja de cálculo nueva.

	A	B	C
1	1	8	9
2	2	7	10
3	3	6	11

Figura 1: Ingrese los números en las celdas A1:C3

4) Seleccione la celda A3, que contiene el número 3, y seleccione **Editar > Copiar** en la *barra de menú* para copiar dicho valor en el portapapeles.
5) Seleccione todas las celdas en el rango A1:C3.

6) Seleccione **Herramientas > Macros > Grabar macro** de la *barra de menú* para comenzar a grabar la macro. Calc muestra el cuadro de diálogo *Grabar macro*, que incluye un botón **Finalizar grabación** (*Figura 2*).

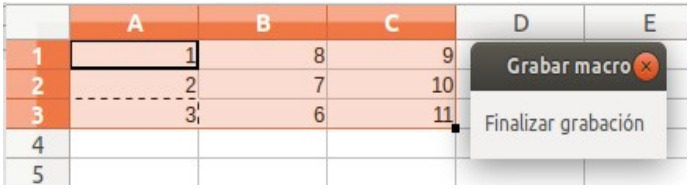


Figura 2: Cuadro de diálogo Grabar macro con el botón Finalizar grabación

7) Seleccione **Editar > Pegado especial** de la *barra de menú* para abrir el cuadro de diálogo de *Pegado especial* (*Figura 3*).

Figura 3: Cuadro de diálogo Pegado especial

8) Seleccione la opción **Pegar todo** en el área *Selección* y la opción **Multiplicar** en el área *Operaciones*, y haga clic en **Aceptar**. Los valores en las celdas A1:C3 ahora se multiplican por 3 (*Figura 4*).

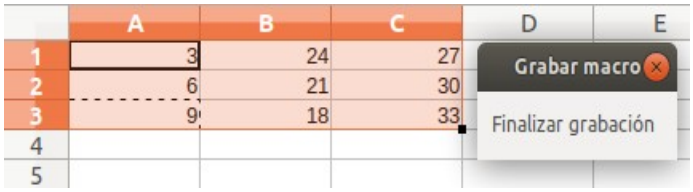
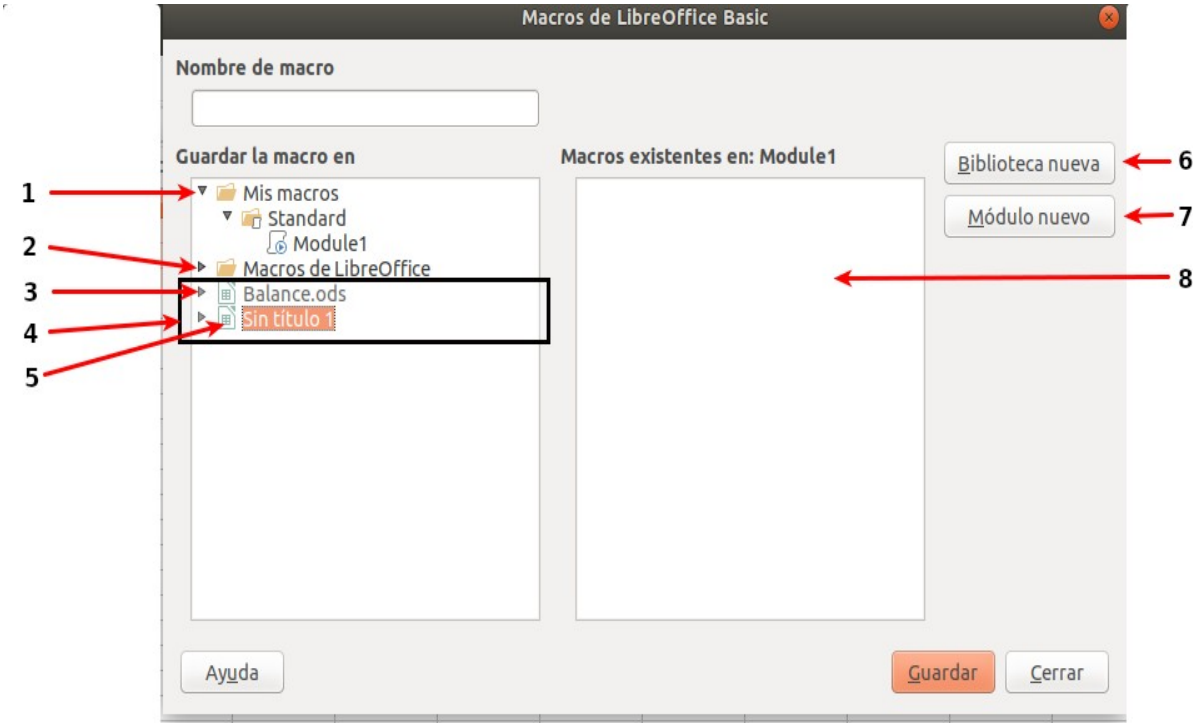


Figura 4: Celdas A1:C3 se multiplican por 3

9) Haga clic en el botón **Finalizar grabación** para finalizar la grabación de la macro. Calc muestra una variante del cuadro de diálogo *Macros de LibreOffice Basic* (*Figura 5*).

Advertencia

Un antiguo error hacía que las versiones de LibreOffice 6.2 colapsaran al hacer clic en el botón **Finalizar grabación**, pero este error está corregido en LibreOffice 6.3. Ver entrada #122598 en el sistema de seguimiento de errores *Bugzilla* en *The Document Foundation* para obtener más información.



1	*Mis Macros*	**5**	Documento actual
2	*Macros de LibreOffice*	**6**	Crear nueva biblioteca
3	Ícono expandir/colapsar	**7**	Crear nuevo módulo en la biblioteca
4	Abrir documentos	**8**	Macros en el módulo seleccionado

Figura 5: Partes del cuadro de diálogo Macros de LibreOffice Basic

Nota

El área *Guardar la macro en* del cuadro de diálogo *Macros de LibreOffice Basic* muestra las macros *LibreOffice Basic* existentes, estructuradas jerárquicamente en contenedores de bibliotecas, bibliotecas, módulos y macros tal como se describe en el *Capítulo 13* de la *Guía de los primeros pasos*. La *Figura 5* muestra el contenedor de bibliotecas *Mis Macros*, el contenedor de bibliotecas *Macros de LibreOffice*, el contenedor de bibliotecas para el archivo abierto *Balance.ods* y el contenedor de bibliotecas para el archivo *Sin título 1* creado en el paso 2. Seleccione los íconos *expandir / colapsar* a la izquierda de cada nombre de contenedor de biblioteca para visualizar las bibliotecas, los módulos y las macros dentro de ese contenedor.

10) Seleccione la entrada para el documento actual en el área *Guardar la macro en*. Como el documento actual en este ejemplo no se ha guardado, se lo llama por su nombre por defecto *Sin título 1*.

Los documentos que han sido guardados incluyen una biblioteca de macro llamada *Standard*. Esta biblioteca no se crea hasta que se guarda el documento o se necesita la

biblioteca, así que en este punto del procedimiento de ejemplo, el nuevo documento no contiene una biblioteca. Se puede crear una nueva biblioteca que contenga la macro recién creada, pero no es necesario.

11) Haga clic en el botón **Módulo nuevo**. Calc muestra el cuadro de diálogo *Módulo nuevo* (*Figura 6*). Ingrese un nombre para el nuevo módulo o deje el nombre por defecto *Module1*.

Figura 6: Cuadro de diálogo Módulo nuevo

Nota

Las bibliotecas, los módulos y los nombres de macros deben seguir algunas reglas estrictas. De acuerdo a las reglas principales, los nombres deben:

- Comenzar con una letra
- Constar de letras minúsculas (a..z), letras mayúsculas (A..Z), dígitos (0..9), y guiones bajos (_)
- No deben contener ningún espacio, símbolos de puntuación o caracteres especiales (incluidas tildes).

12) Haga clic en el botón **Aceptar** para crear un nuevo módulo. Como no hay ninguna biblioteca en el documento actual, Calc crea y utiliza la biblioteca *Standard* automáticamente.

13) En el cuadro de diálogo *Macros de LibreOffice Basic*, seleccione la entrada para el módulo recientemente creado en el área *Guardar la macro en*, ingrese el texto **PasteMultiply** en la caja *Nombre de macro*, y haga clic en el botón **Guardar** (*Figura 7*).

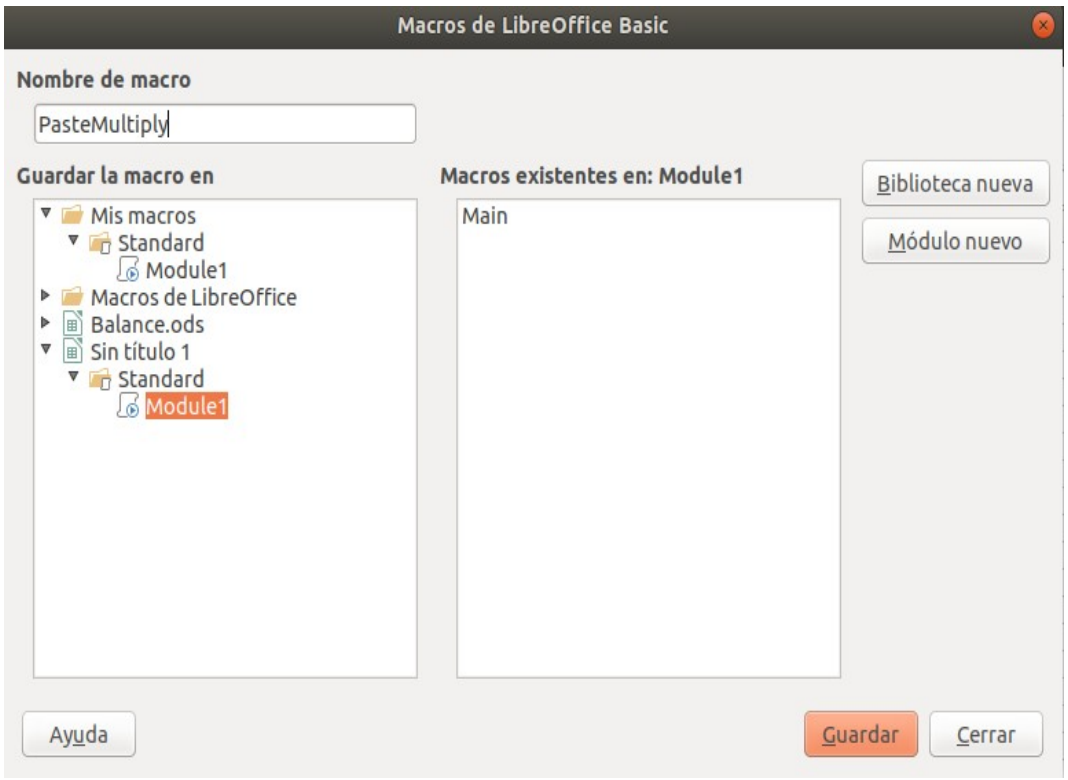


Figura 7: Seleccione el módulo y nombre la macro

La macro se guarda con el nombre **PasteMultiply** en el módulo recientemente creado dentro de la biblioteca *Standard* del documento *Sin título 1*. El listado 1 muestra el contenido de la macro.

Listado 1. PasteMultiply - Pegado especial con la macro multiplicar

```
sub PasteMultiply
  rem ----------------------------------------------------------------
  rem define variables
  dim document   as object
  dim dispatcher as object
  rem ----------------------------------------------------------------
  rem get access to the document
  document   = ThisComponent.CurrentController.Frame
  dispatcher = createUnoService("com.sun.star.frame.DispatchHelper")

  rem ----------------------------------------------------------------
  dim args1(5) as new com.sun.star.beans.PropertyValue
  args1(0).Name = "Flags"
  args1(0).Value = "A"
  args1(1).Name = "FormulaCommand"
  args1(1).Value = 3
  args1(2).Name = "SkipEmptyCells"
  args1(2).Value = false
  args1(3).Name = "Transpose"
  args1(3).Value = false
  args1(4).Name = "AsLink"
  args1(4).Value = false
  args1(5).Name = "MoveMode"
  args1(5).Value = 4

  dispatcher.executeDispatch(document, ".uno:InsertContents", "", 0,
```

```
args1())

end sub
```

Cómo escribir funciones propias

Crear una macro función

Se puede escribir una macro y luego nombrarla como se nombraría a una función Calc. Siga los siguientes pasos para crear una macro función sencilla:

1) Cree una nueva hoja de cálculo, guárdela con el nombre *CalcTestMacros.ods* y déjela abierta en Calc.

2) Seleccione **Herramientas > Macros > Organizar Macros > LibreOffice Basic** de la *barra de menú* para abrir el cuadro de diálogo *Macros de LibreOffice Basic* (*Figura 8*). Tenga en cuenta que la interfaz del cuadro de diálogo *Macros de LibreOffice Basic* en este contexto es diferente de la versión que se ve en Calc cuando el usuario hace clic en el botón **Finalizar grabación** en el cuadro de diálogo *Grabar macro* (vea *Figura 5*).

   El área *Macro de* muestra una lista de todos los contenedores de bibliotecas disponibles, incluyendo aquellos relacionados con cualquier documento de LibreOffice que esté abierto actualmente. *Mis Macros* contiene macros que pueden escribirse o agregarse a LibreOffice y están disponibles en más de un documento. *Macros de LibreOffice* contiene macros que se incluyeron en la instalación de LibreOffice y no deben cambiarse.

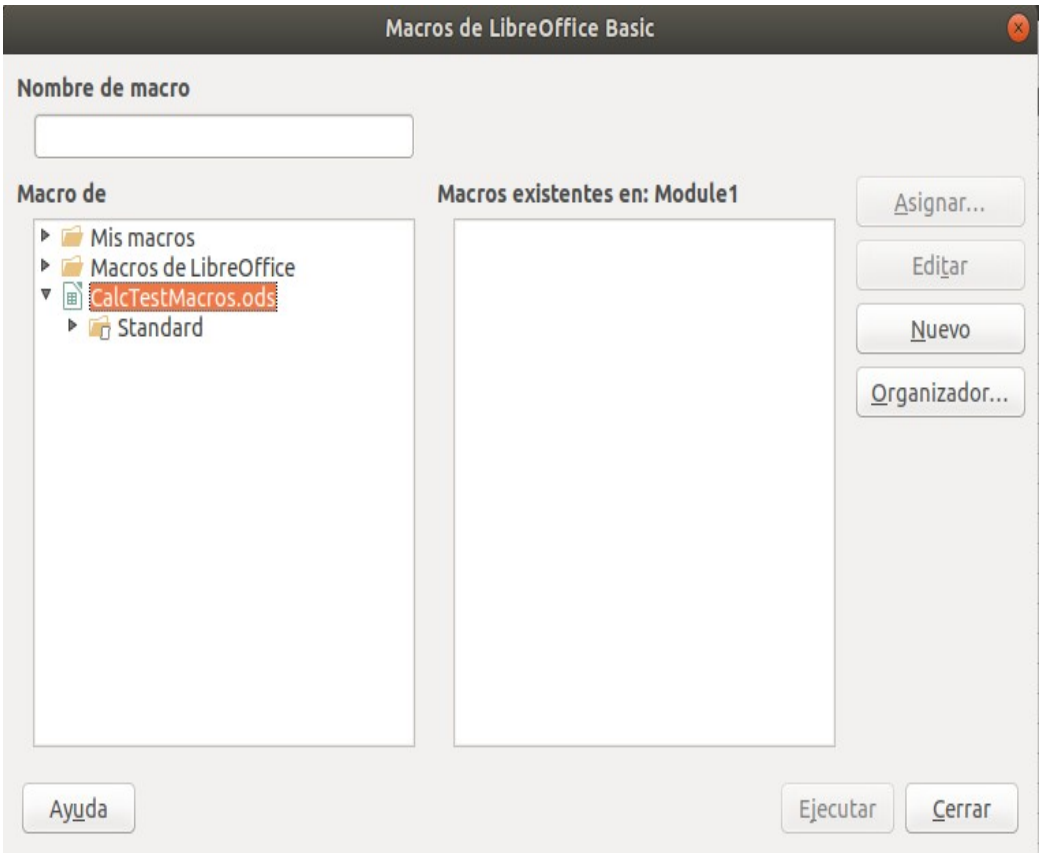


Figura 8: Cuadro de diálogo Macros de LibreOffice Basic

3) Haga clic en **Organizador** para abrir el cuadro de diálogo *Organizador de macros de LibreOffice Basic* (*Figura 9*).

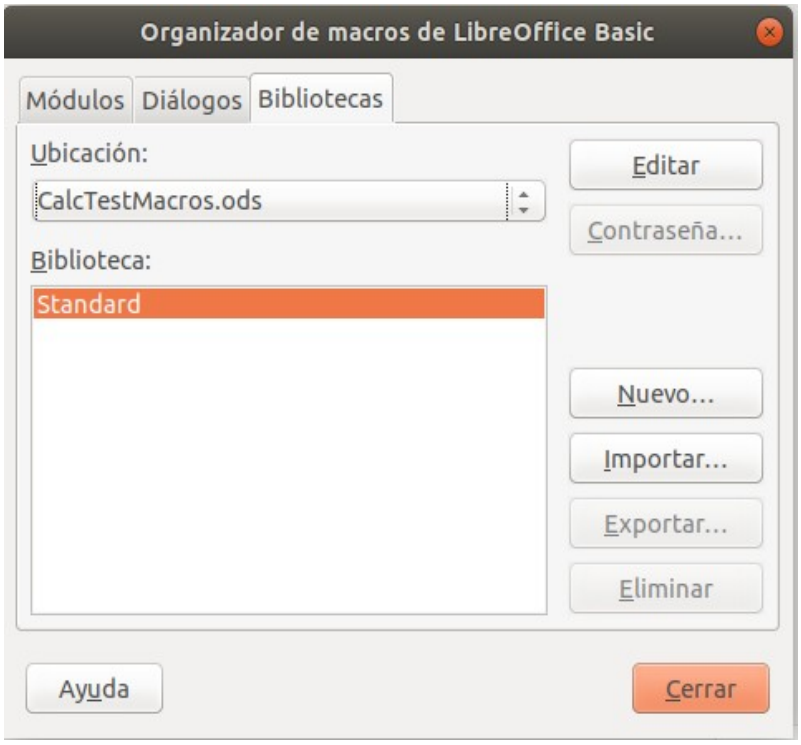


Figura 9: Organizador de macros de LibreOffice Basic

Haga clic en la pestaña *Bibliotecas* y, en el área *Ubicación*, seleccione la entrada para el nombre del documento actual. El área *Biblioteca* se actualiza para mostrar el nombre de la biblioteca vacía *Standard*.

4) Haga clic en **Nuevo** para abrir el cuadro de diálogo *Biblioteca nueva* para crear una nueva biblioteca para este documento (*Figura 10*).

Figura 10: Cuadro de diálogo Biblioteca nueva

5) Ingrese un nombre de biblioteca descriptivo (por ejemplo, **AuthorsCalcMacros**) y haga clic en **Aceptar** para crear la biblioteca. El área *Biblioteca:* del cuadro de diálogo del *Organizador de macros de LibreOffice Basic* se actualiza para incluir el nombre de la biblioteca creada recientemente. Un nombre de biblioteca puede tener hasta 30 caracteres. Tenga en cuenta que, en algunos casos, el cuadro de diálogo podría mostrar una parte del nombre solamente.

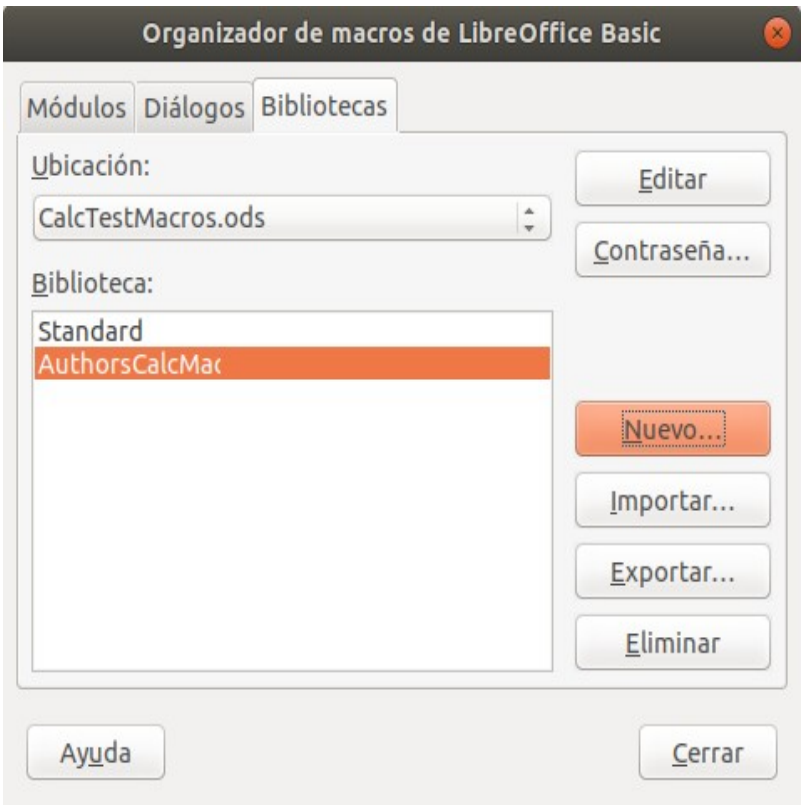


Figura 11: La nueva biblioteca se muestra en el área Biblioteca:

Seleccione la entrada **AuthorsCalcMacros** en el área *Biblioteca:* y haga clic en **Editar** para editar la biblioteca. Calc automáticamente crea un módulo llamado *Module1* y una macro llamada *Main*. Calc muestra el *IDE (LibreOffice Basic Integrated Development Environment)*, como se muestra en la *Figura 12*.

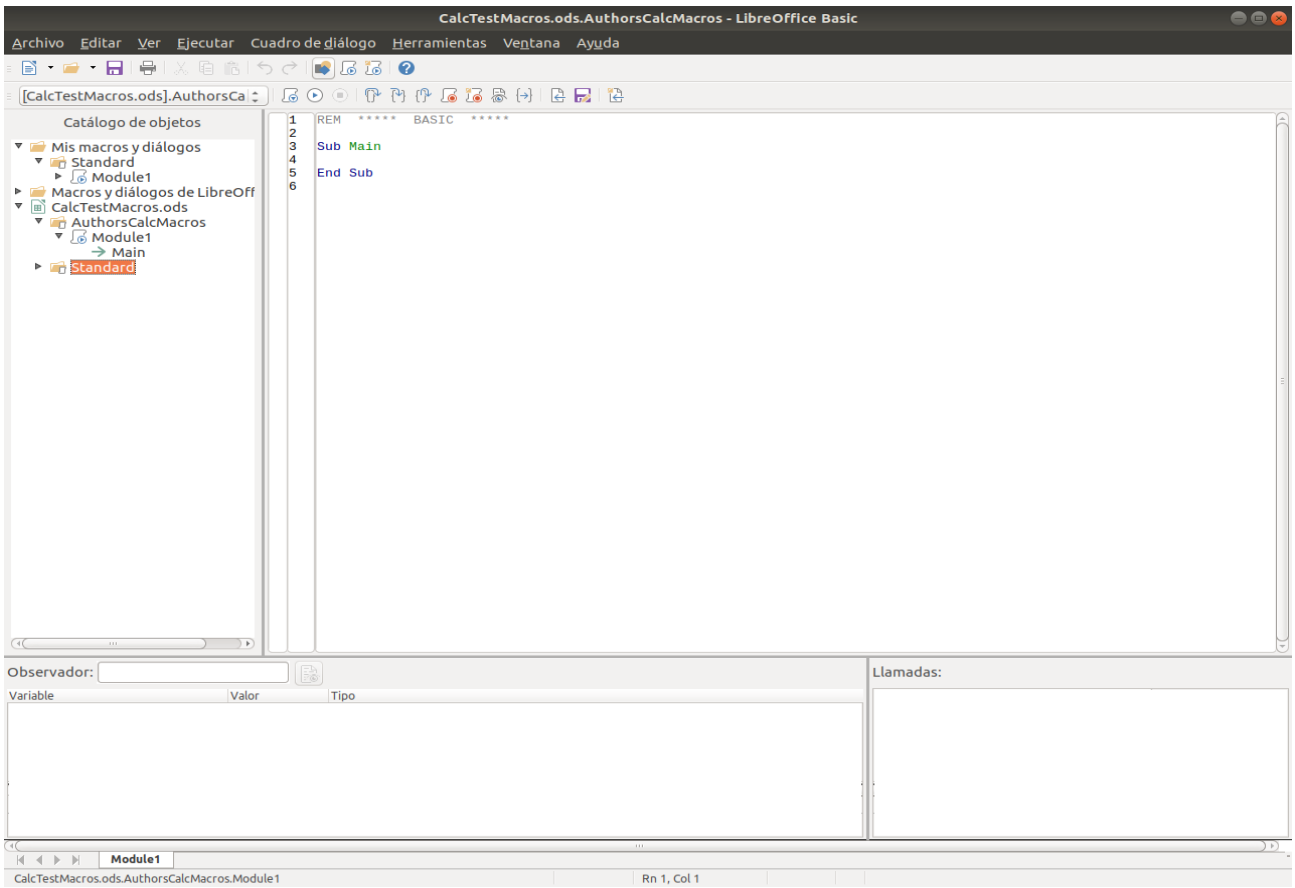


Figura 12: Entorno de desarrollo integrado LibreOffice Basic

La Figura 12 muestra la configuración predeterminada para el *IDE de LibreOffice Basic*. Esto consta de:

- Una barra de menú
- Tres barras de herramientas (*Idioma*, *Macro* y *Estándar*). La barra de herramientas *Macro* proporciona varios íconos para editar y probar programas.
- El *Catálogo de objetos* permite la selección del contenedor de bibliotecas, biblioteca, módulo y macro requeridos.
- La *ventana del editor*, en el que se puede editar el código del programa *LibreOffice Basic*. La columna en el lado izquierdo se usa para establecer puntos de ruptura en el código del programa.
- La ventana *Observador* (ubicada a la izquierda, debajo del *Catálogo de objetos* y de la *ventana del editor*) muestra los contenidos de las variables o matrices durante un paso del proceso en particular.
- La ventana *Llamadas* (ubicada a la derecha, debajo del *Catálogo de objetos* y de la *ventana del editor*) proporciona información sobre la pila de llamadas de procedimientos y funciones cuando se ejecuta un programa.
- Un área de control de pestañas
- Una *Barra de estado*.

El *IDE* de *LibreOffice Basic* proporciona características poderosas para el desarrollo y la depuración de las macros de *LibreOffice Basic*. Una descripción más completa de estas características está fuera del alcance de este documento.

6) En la *ventana del editor*, cambie el código para que sea el mismo que se muestra en el Listado 2. La adición importante es la creación de la función **NumberFive**, que regresa el valor 5.

Sugerencia

La instrucción *Option Explicit* obliga a todas las *variables* a declararse antes de que se usen. Si se omite la *Option Explicit*, las variables se definen automáticamente en el primer uso como tipo *Variant*.

Listado 2. Función que regresa el valor 5.

```
REM  *****  BASIC  *****
Option Explicit

Sub Main

End Sub

Function NumberFive ()
  NumberFive = 5
End Function
```

7) Haga clic en el botón **Guardar** de la barra de herramientas Estándar dentro del *IDE* de *LibreOffice Basic* para guardar el *Module1* modificado.

Cómo utilizar la macro como una función

Cuando utilice la hoja de cálculo *CalcTestMacros.ods* recientemente creada, seleccione una celda e ingrese la fórmula **=NumberFive()** (*Figura 13*). Calc encuentra la macro, la nombra y muestra el resultado (5) en esa celda.

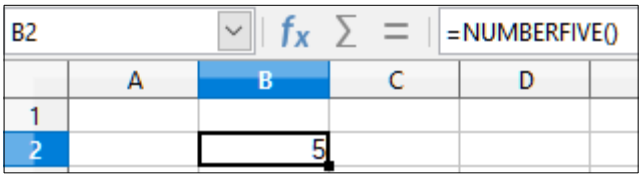


Figura 13: Utilice la macro NumberFive como una función Calc

Sugerencia

Los nombres de las funciones no son sensibles a mayúsculas y minúsculas. En la *Figura 13*, el nombre de la función se ingresa como *NumberFive()*, pero Calc lo muestra como **NUMBERFIVE()** en la barra de *Fórmulas*.

Advertencia de seguridad de las macros

Ahora debería guardar el documento Calc, cerrarlo y abrirlo nuevamente. Dependiendo de las configuraciones en el cuadro de diálogo *Seguridad de macros* al que se puede acceder seleccionando **Herramientas > Opciones > LibreOffice > Seguridad > Seguridad de macros** de la *barra de menú*, se podría ver una de las advertencias de Calc en las *Figuras 14* y *15*.

En el caso de la advertencia de la *Figura 14*, deberá hacer clic en **Activar macros** o Calc no permitirá que se ejecute ninguna macro en el documento. Si no cree que un documento contenga una macro, es más seguro hacer clic en **Desactivar macros** en el caso de que la macro sea un virus.

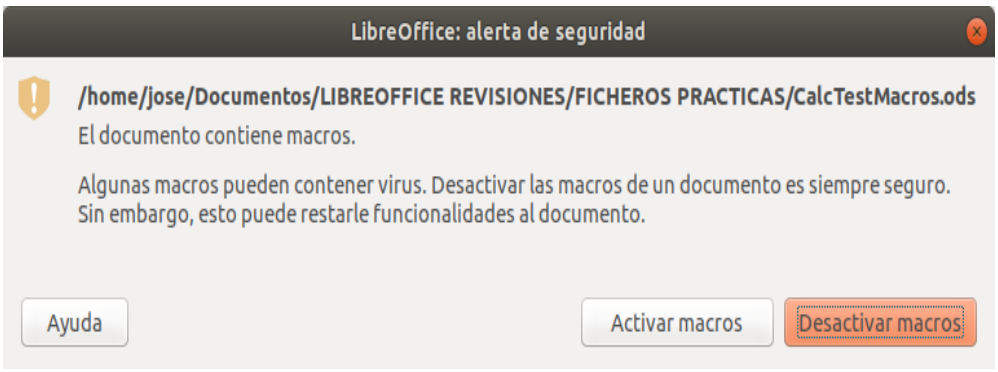


Figura 14: Advertencia de que un documento contiene macros

En el caso de la advertencia que se muestra en la *Figura 15*, Calc no permitirá que se ejecute ninguna macro en el documento y deberá hacer clic en el botón **Aceptar** para eliminar la advertencia de su pantalla.

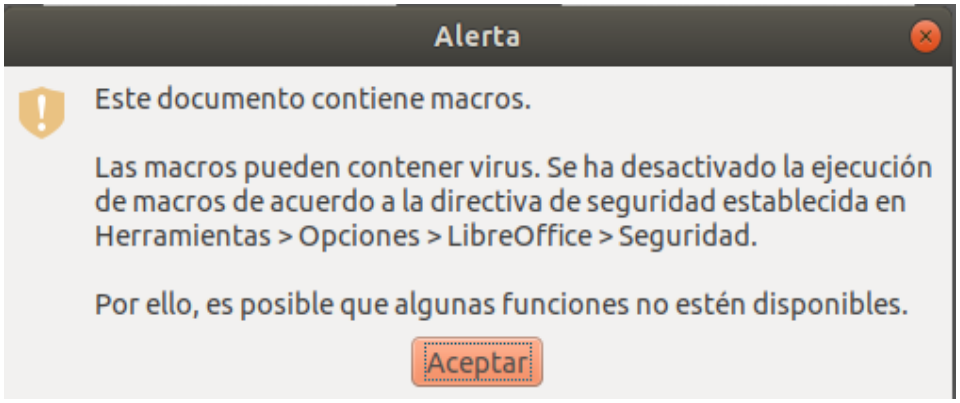


Figura 15: Advertencia de que las macros en el documento están deshabilitadas.

Cuando el documento se carga con las macros desactivadas, Calc no podrá encontrar ninguna función de macro e indicará un error en cualquier celda afectada con el texto *#¿NOMBRE?* en esa celda.

Bibliotecas cargadas / descargadas

Al abrir una hoja de cálculo, Calc no abre todas las bibliotecas que encuentra en los contenedores de bibliotecas disponibles porque sería un desperdicio de recursos. En su lugar, Calc automáticamente carga solamente la biblioteca *Standard* dentro del contenedor de bibliotecas *Mis Macros* y la propia biblioteca *Standard* del documento. Ninguna otra biblioteca se carga de manera automática.

Al abrir la hoja de cálculo *CalcTestMacros.ods* nuevamente, Calc revisa todas las bibliotecas visibles y cargadas para la función ya que no contiene una función llamada *NumberFive()*. Las bibliotecas cargadas en *Macros de LibreOffice*, *Mis Macros* y el documento se revisan para que la función sea nombrada correctamente. En la primera implementación, la función *NumberFive()* se almacena en la biblioteca *AuthorsCalcMacros*, que no carga automáticamente cuando se abre el documento. Por lo tanto, no se encuentra la función *NumberFive()* y aparece una condición de error en la celda donde se la nombra (*Figura 16*).

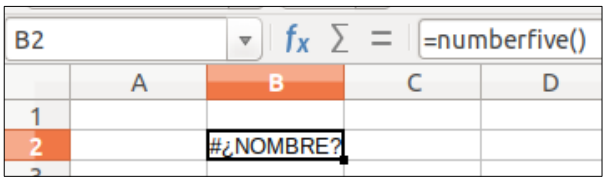


Figura 16: La función de la macro no está disponible

Seleccione **Herramientas > Macros > Organizar Macros > LibreOffice Basic** de la *barra de menú* para abrir el cuadro de diálogo *Macros de LibreOffice Basic* (*Figura 17*). El ícono de una biblioteca cargada (por ejemplo, *Standard*) tiene un aspecto diferente al ícono de una biblioteca que no está cargada (por ejemplo, *AuthorsCalcMacros*).

Figura 17: Diferentes símbolos para bibliotecas cargadas y no cargadas

Haga clic en el ícono para expandir al lado de *AuthorsCalcMacros* para cargar la biblioteca. El ícono cambia de apariencia para indicar que la biblioteca ya está cargada. Haga clic en **Cerrar** para cerrar el cuadro de diálogo *Macros de LibreOffice Basic*.

Desafortunadamente, la celda que contiene **=NumberFive()** en la implementación inicial todavía marca error. Calc no recalcula las celdas que marcan error a menos que se las edite o cambie de alguna manera. La solución a menudo es almacenar las macros utilizadas como funciones en la biblioteca *Standard*. Si la macro es grande o hay muchas macros, un *resguardo (stub)* con el nombre deseado se almacena en la biblioteca *Standard*. El *resguardo* (*stub*) carga la biblioteca que contiene la implementación y luego llama la implementación. Los siguientes pasos ilustran el método.

1) Seleccione **Herramientas > Macros > Organizar Macros > LibreOffice Basic** en la *barra de menú* para abrir el cuadro de diálogo *Macros de LibreOffice Basic*. Seleccione la macro *NumberFive* y haga clic en **Editar** para abrir la macro y editarla (*Figura 18*).

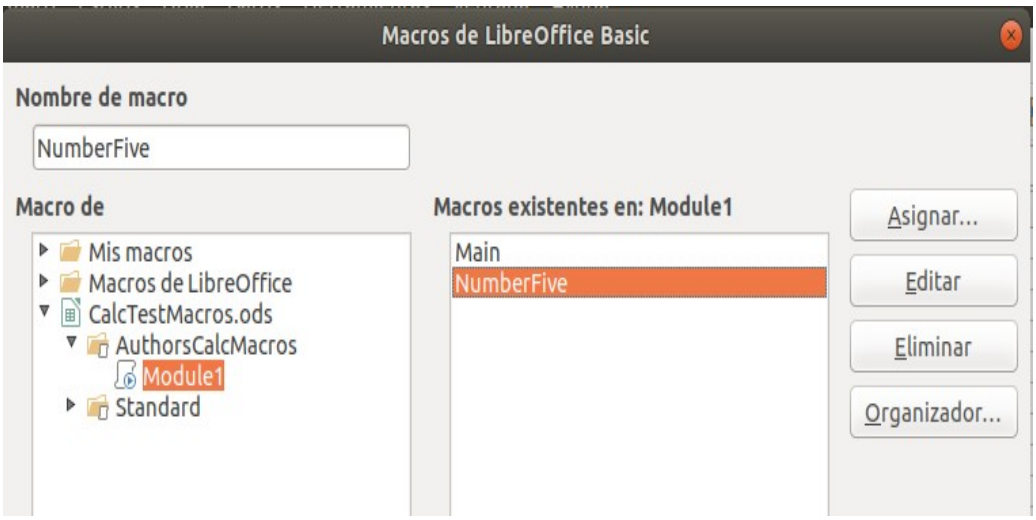


Figura 18: Seleccione una macro y haga clic en Editar.

2) Calc muestra el *IDE* de *LibreOffice Basic* (*Figura 12*), con el cursor de entrada en la ventana del editor en la línea Function NumberFive (). Cambie el nombre de *NumberFive* a *NumberFive_Implementation* para que el código de la función coincida con el del Listado 3.

   Listado 3. Cambie el nombre de NumberFive a NumberFive_Implementation

```
Function NumberFive_Implementation ()
  NumberFive_Implementation = 5
End Function
```

3) Haga clic en el botón **Seleccionar Macro** en la barra de herramientas *Standard* del *IDE de LibreOffice Basic* para abrir el cuadro de diálogo *Macros de LibreOffice Basic* (*Figura 18*).

4) Seleccione la biblioteca *Standard* en el documento *CalcTestMacros.ods* y haga clic en el botón **Nuevo** para crear un nuevo módulo. Ingrese un nombre significativo tal como *CalcFunctions* y haga clic en **Aceptar**. Calc crea una macro llamada *Main* de manera automática y abre el módulo para su edición.

5) Cree una macro en el módulo *CalcFunctions* de la biblioteca *Standard* que cargue la biblioteca *AuthorsCalcMacros* si no está ya cargada y luego llame la función de la implementación. Vea Listado 4. Aquí la función *NumberFive* es el resguardo (stub).

   Listado 4. Cree una nueva función NumberFive para llamar la función NumberFive_Implementation

```
Function NumberFive()
  If NOT BasicLibraries.isLibraryLoaded("AuthorsCalcMacros") Then
    BasicLibraries.LoadLibrary("AuthorsCalcMacros")
  End If
  NumberFive = NumberFive_Implementation()
End Function
```

6) Guarde, cierre y abra el documento Calc nuevamente. Esta vez, si las macros están habilitadas, la función *NumberFive()* funcionará como se espera.

Cómo pasar argumentos a una macro

Para ilustrar una función que acepte argumentos, escribiremos una macro que calcule la suma de los argumentos que son positivos. Los argumentos que sean inferiores a cero serán ignorados (vea Listado 5).

Listado 5. PositiveSum calcula la suma de los argumentos positivos

```
Function PositiveSum(Optional x)
  Dim TheSum As Double
  Dim iRow As Integer
  Dim iCol As Integer

  TheSum = 0.0
  If NOT IsMissing(x) Then
    If NOT IsArray(x) Then
      If x > 0 Then TheSum = x
    Else
      For iRow = LBound(x, 1) To UBound(x, 1)
        For iCol = LBound(x, 2) To UBound(x, 2)
          If x(iRow, iCol) > 0 Then TheSum = TheSum + x(iRow, iCol)
        Next
      Next
    End If
  End If
  PositiveSum = TheSum
End Function
```

La macro que se encuentra en el Listado 5 muestra algunas técnicas importantes:

1) El argumento **x** es opcional. Cuando un argumento no es opcional y la función se llama sin él, Calc genera un mensaje de advertencia cada vez que se llama una macro. Si Calc llama la función muchas veces, entonces el error se muestra muchas veces.
2) La función *IsMissing* revisa que se haya pasado un argumento antes de que se utilice.
3) La función *IsArray* revisa que el argumento tenga un solo valor o una matriz. Por ejemplo, *=PositiveSum(7)* o *=PositiveSum(A4)*. En el primer caso, el número 7 se pasa como argumento, y en el segundo caso, el valor de la celda **A4** se pasa a la función. En ambos casos, *IsArray* regresa el valor *Falso*.
4) Si se pasa un rango a la función, se pasa como una matriz bidimensional de valores; por ejemplo *=PositiveSum(A2:B5)*. Las funciones *LBound* y *UBound* se utilizan para determinar los límites de las matrices que son utilizadas. Aunque el límite inferior sea uno, se considera más seguro utilizar *Lbound* en caso de que cambie en el futuro.

Sugerencia

La macro que se encuentra en el Listado 5 es cuidadosa y verifica que el argumento sea una matriz o un argumento solo. La macro no verifica que cada valor sea numérico. Puede ser tan cuidadoso como desee. Cuanto más revise, más robusta será la macro, pero también será más lenta.

Pasar un argumento es tan sencillo como pasar dos: agregue otro argumento a la definición de función (vea Listado 6). Cuando llame una función con dos argumentos, separe los argumentos con una coma; por ejemplo, *=TestMax(3, -4)*.

Listado 6. TestMax acepta dos argumentos y regresa el más grande

```
Function TestMax(x, y)
  If x >= y Then
    TestMax = x
  Else
    TestMax = y
  End If
End Function
```

Los argumentos son pasados como valores

Los argumentos que se pasan a una macro desde Calc siempre son valores. No es posible saber qué celdas se utilizan, en el caso de que lo sean. Por ejemplo, *=PositiveSum(A3)* pasa el valor de la celda **A3**, y *PositiveSum* no tiene forma de saber que se utilizó la celda **A3**. Si debe saber qué celdas son referenciadas en vez de conocer los valores en las celdas, pase el rango como una cadena de caracteres, analice la cadena de caracteres y obtenga los valores en las celdas referenciadas.

Cómo escribir macros que actúen como funciones incorporadas

Aunque Calc encuentre y llame a las macros como funciones normales, en realidad no se comportan como funciones ya incorporadas. Por ejemplo, las macros no aparecen en las listas de funciones. Se pueden escribir funciones que se comporten como funciones comunes si se escribe un *Complemento (Add-In)*. Sin embargo, este es un tema avanzado destinado a programadores con experiencia y está más allá del alcance de esta guía.

Cómo acceder a las celdas de manera directa

Se puede acceder directamente a los objetos internos de LibreOffice para manipular un documento Calc. Por ejemplo, la macro en el Listado 7 agrega los valores en la celda **A2** de cada hoja en el documento actual. *ThisComponent* se establece automáticamente para hacer referencia al documento actual cuando se activa la macro. Un documento Calc contiene hojas de cálculo y la macro accede a estas a través de una llamada a *ThisComponent.getSheets()*. Seleccione *getCellByPosition(col, row)* para regresar a una celda en una fila y columna específicas.

Listado 7. SumCellsAllSheets agrega los valores en la celda A2 de cada hoja

```
Function SumCellsAllSheets()
  Dim TheSum As Double
  Dim i As integer
  Dim oSheets
  Dim oSheet
  Dim oCell

  TheSum = 0
  oSheets = ThisComponent.getSheets()
  For i = 0 To oSheets.getCount() - 1
    oSheet = oSheets.getByIndex(i)
    oCell = oSheet.getCellByPosition(0, 1) ' GetCell A2
    TheSum = TheSum + oCell.getValue()
  Next
  SumCellsAllSheets = TheSum
End Function
```

Sugerencia

Un objeto de celda acepta los métodos *getValue(), getString()* y *getFormula()* para obtener el valor numérico, el valor de la cadena o la fórmula utilizada en una celda. Utilice las funciones establecidas correspondientes para establecer los valores adecuados.

Seleccione *oSheet.getCellRangeByName("A2")* para regresar a un rango de celdas por su nombre. Si se referencia una sola celda, entonces se regresa un objeto de celda. Si se da un rango de celdas, entonces se regresa un rango de celdas entero (vea Listado 8). Tenga en cuenta

que un rango de celdas regresa información como una matriz de matrices, que es más complicado que tratarlo como una matriz de dos dimensiones como se muestra en el Listado 5.

Listado 8. SumCellsAllSheets agrega los valores en las celdas A2:C5 de cada hoja

```
Function SumCellsAllSheets()
  Dim TheSum As Double
  Dim iRow As Integer, iCol As Integer, i As Integer
  Dim oSheets, oSheet, oCells
  Dim oRow(), oRows()

  TheSum = 0
  oSheets = ThisComponent.getSheets()
  For i = 0 To oSheets.getCount() - 1
    oSheet = oSheets.getByIndex(i)
    oCells = oSheet.getCellRangeByName("A2:C5")

    REM The getDataArray() method returns strings and numbers
    REM but is not used in this function.

    REM The getData() method returns only numbers and is applicable
    REM to this function.

    oRows() = oCells.getData()
    For iRow = LBound(oRows()) To UBound(oRows())
      oRow() = oRows(iRow)
      For iCol = LBound(oRow()) To UBound(oRow())
        TheSum = TheSum + oRow(iCol)
      Next
    Next
  Next
  SumCellsAllSheets = TheSum
End Function
```

Sugerencia

Cuando una macro se llama como una función de Calc, la macro no puede modificar ningún valor en la hoja desde la cual se llamó la macro, excepto el valor de la celda que contiene la función.

Clasificación

Considere clasificar los datos que se muestran en la *Figura 19*. Primero, clasifíquelos en la columna **B** de forma descendente y luego en la columna **A** de forma ascendente.

Figura 19: Clasifique la columna B descendente y la columna A ascendente

El ejemplo del Listado 9 muestra cómo clasificar en estas dos columnas.

Listado 9. SortRange clasifica las celdas A1:C5 de la Hoja 1

```
Sub SortRange
  Dim oSheet        ' Calc sheet containing data to sort.
  Dim oCellRange    ' Data range to sort.

  REM An array of sort fields determines the columns that are
  REM sorted. This is an array with two elements, 0 and 1.
  REM To sort on only one column, use:
  REM Dim oSortFields(0) As New com.sun.star.util.SortField
  Dim oSortFields(1) As New com.sun.star.util.SortField

  REM The sort descriptor is an array of properties.
  REM The primary property contains the sort fields.
  Dim oSortDesc(0) As New com.sun.star.beans.PropertyValue

  REM Get the sheet named "Sheet1"
  oSheet = ThisComponent.Sheets.getByName("Sheet1")

  REM Get the cell range to sort
  oCellRange = oSheet.getCellRangeByName("A1:C5")

  REM Select the range to sort.
  REM The only purpose would be to emphasize the sorted data.
  'ThisComponent.getCurrentController.select(oCellRange)

  REM The columns are numbered starting with 0, so
  REM column A is 0, column B is 1, etc.
  REM Sort column B (column 1) descending.
  oSortFields(0).Field = 1
  oSortFields(0).SortAscending = FALSE

  REM If column B has two cells with the same value,
  REM then use column A ascending to decide the order.
  oSortFields(1).Field = 0
  oSortFields(1).SortAscending = TRUE

  REM Setup the sort descriptor.
  oSortDesc(0).Name = "SortFields"
  oSortDesc(0).Value = oSortFields()

  REM Sort the range.
  oCellRange.Sort(oSortDesc())
End Sub
```

Resumen de las macros de BeanShell, JavaScript, y Python

Introducción

Es posible que muchos programadores no estén familiarizados con *LibreOffice Basic*, por lo que Calc acepta macros que hayan sido escritas en otros tres lenguajes que les resulten más conocidos. Estos lenguajes son *BeanShell*, *JavaScript* y *Python*.

El lenguaje de programación principal para Calc es *LibreOffice Basic* y en la instalación LibreOffice estándar ofrece un poderoso entorno de desarrollo integrado (IDE) junto con otras opciones para este lenguaje.

Las macros se organizan de la misma manera para los cuatro lenguajes de programación. El contenedor *Macros de LibreOffice* tiene todas las macros que son suministradas en la instalación de LibreOffice. El contenedor de bibliotecas *Mis Macros* tiene todas sus macros que están disponibles para cualquiera de sus documentos LibreOffice. Cada documento puede también contener sus macros que no estén disponibles en algún otro documento.

Cuando se utiliza la función de grabación de macros, Calc crea la macro en *LibreOffice Basic*. Para utilizar los otros lenguajes de programación disponibles, debe escribir el código usted mismo.

Cuando ejecute una macro desde **Herramientas > Macros > Ejecutar macro** desde la *barra de menú*, Calc muestra el cuadro de diálogo del *Selector de macros*. El cuadro de diálogo permite seleccionar y ejecutar cualquier macro disponible, escrita en cualquiera de los lenguajes disponibles (*Figura 20*).

Cuando edite una macro desde **Herramientas > Macros > Editar macros** desde la *barra de menú*, Calc muestra el *IDE de LibreOffice Basic*. Este cuadro de diálogo permite seleccionar y editar cualquier macro disponible de *LibreOffice Basic*, pero no macros en otros lenguajes.

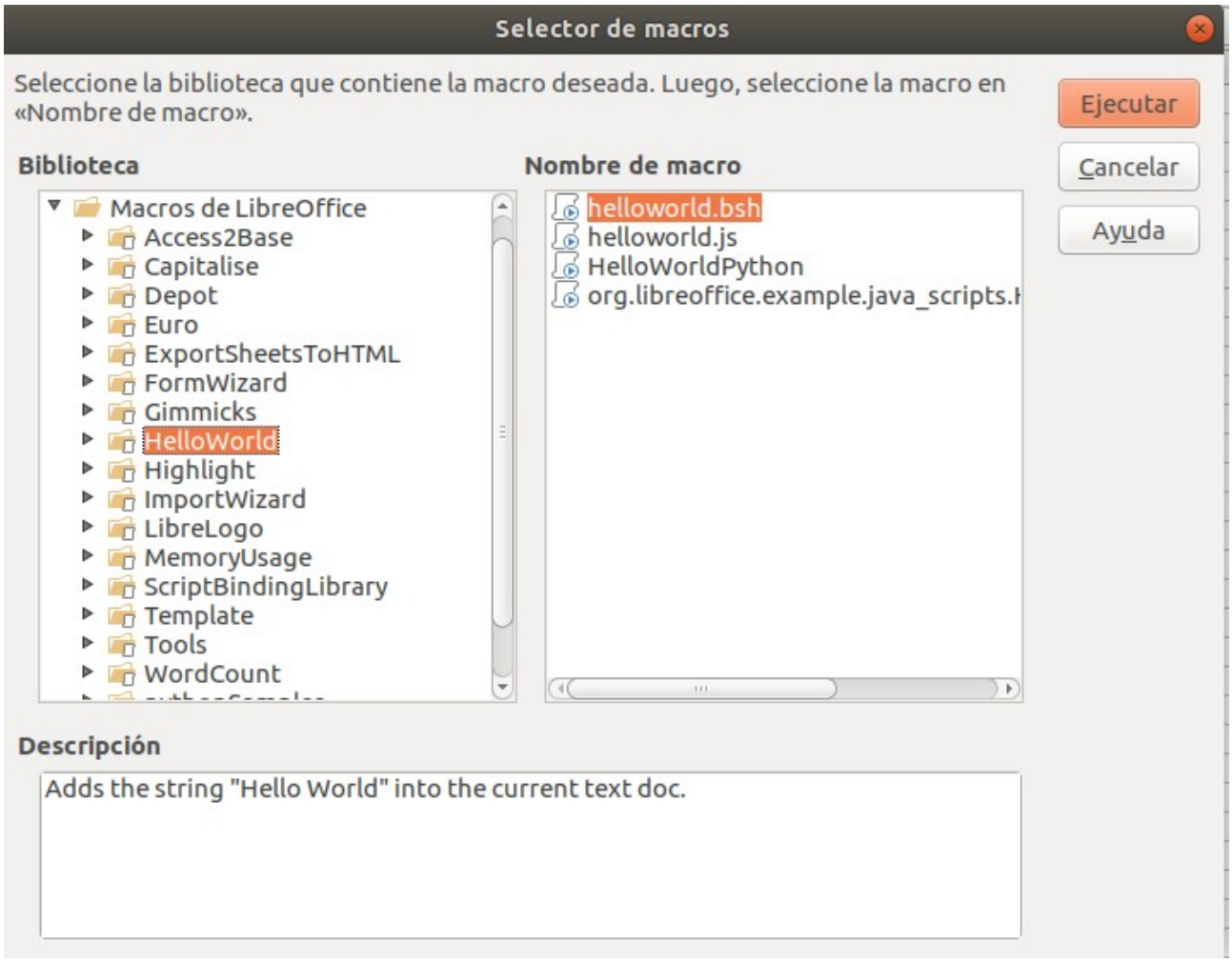


Figura 20: Cuadro de diálogo Selector de macros

El modelo de componente utilizado en LibreOffice se conoce como *Universal Network Objects* o *UNO*. Las macros de LibreOffice en cualquier lenguaje de programación utilizan una interfaz de programación de aplicación *(API)* durante la ejecución de *UNO*. La interfaz **XSCRIPTCONTEXT** se ofrece a las secuencias de comandos para los cuatro lenguajes y proporciona un medio de acceso a las varias interfaces en las que podrían necesitar realizar alguna acción en un documento.

Macros de BeanShell

BeanShell es un lenguaje de programación similar a *Java* que fue lanzado por primera vez en 1999.

Cuando seleccione **Herramientas > Macros > Organizar macros > BeanShell** desde la *barra de menú*, Calc muestra el cuadro de diálogo *Macros en BeanShell* (*Figura 21*).

Figura 21: Cuadro de diálogo Macros en BeanShell

Haga clic en el botón **Editar** del cuadro de diálogo *Macros en BeanShell* para acceder a la *ventana de depuración BeanShell* (*Figura 22*).

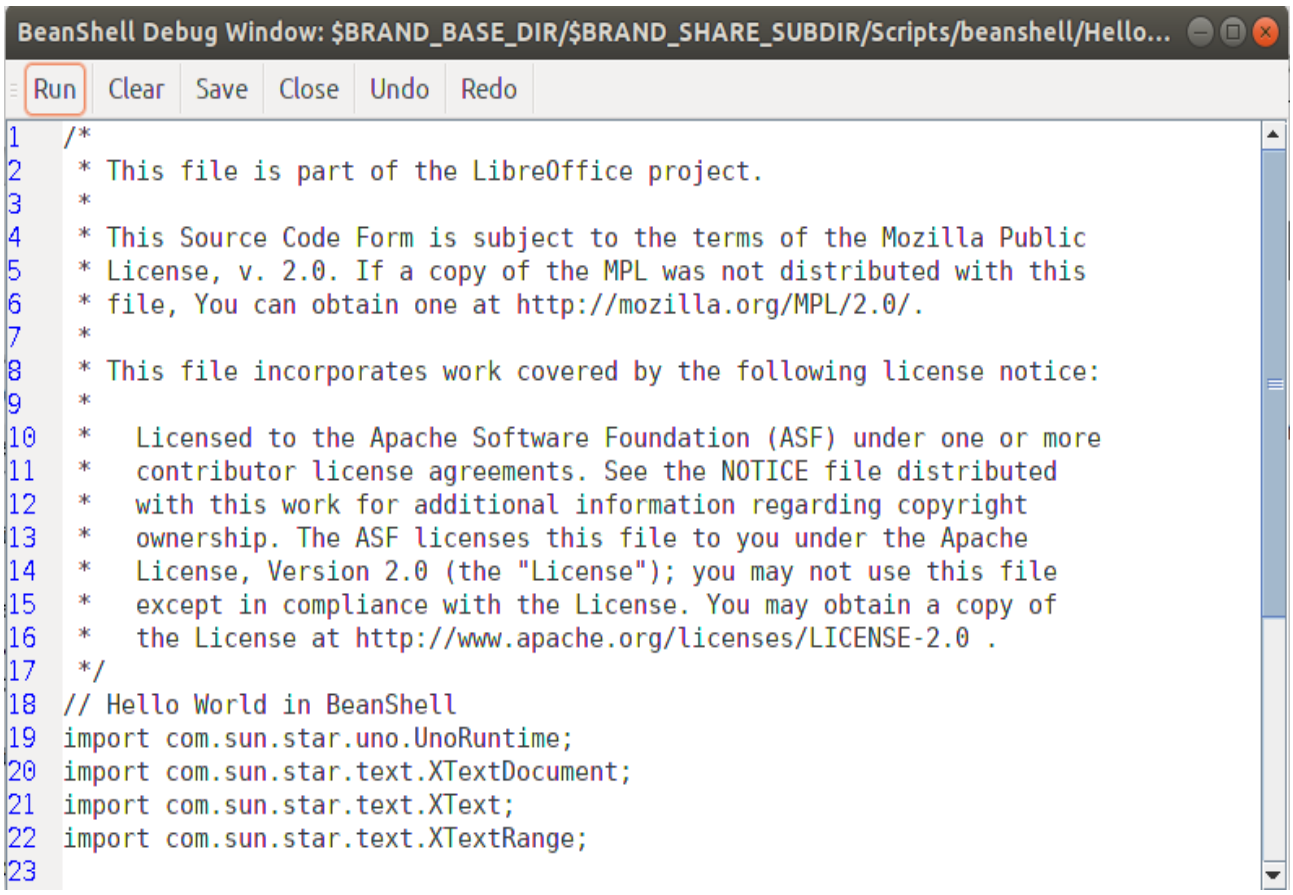


Figura 22: Ventana de depuración BeanShell Debug

El Listado 10 es un ejemplo de una macro *BeanShell* que inserta el texto *"Hello World from BeanShell"* en la celda **A1** de la hoja de cálculo Calc activa.

Listado 10. Muestra de macro BeanShell

```
import com.sun.star.uno.UnoRuntime;
import com.sun.star.sheet.XSpreadsheetView;
import com.sun.star.text.XText;
model = XSCRIPTCONTEXT.getDocument();

controller = model.getCurrentController();

view = UnoRuntime.queryInterface(XSpreadsheetView.class, controller);

sheet = view.getActiveSheet();

cell = sheet.getCellByPosition(0, 0);

cellText = UnoRuntime.queryInterface(XText.class, cell);

textCursor = cellText.createTextCursor();

cellText.insertString(textCursor, "Hello World from BeanShell", true);

return 0;
```

Macros de JavaScript

JavaScript es un lenguaje de programación de alto nivel que fue lanzado por primera vez en 1995.

Cuando seleccione **Herramientas > Macros > Organizar Macros > JavaScript** en la *barra de menú*, Calc muestra la caja de diálogo de *Macros en JavaScript* (*Figura 23*).

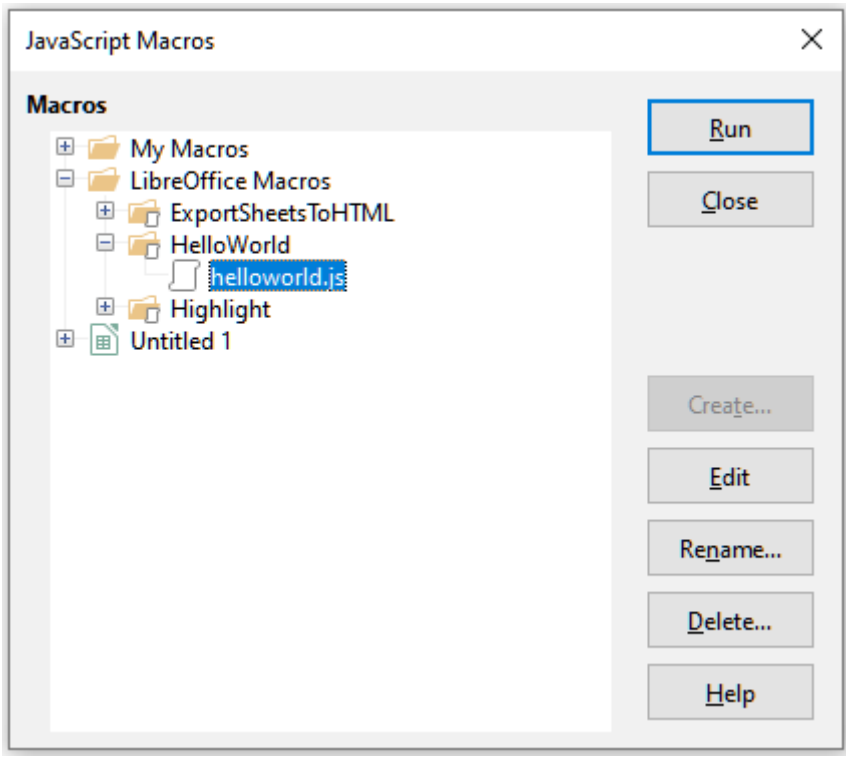


Figura 23: Caja de diálogo Macros en JavaScript

Haga clic en el botón **Editar** en la caja de diálogo *Macros en JavaScript* para acceder al *depurador Rhino de JavaScript* (*Figura 24*).

Encuentre instrucciones más detalladas para utilizar esta herramienta en el sitio web de *Mozilla* en https://developer.mozilla.org/en-US/docs/Mozilla/Projects/Rhino/Debugger.

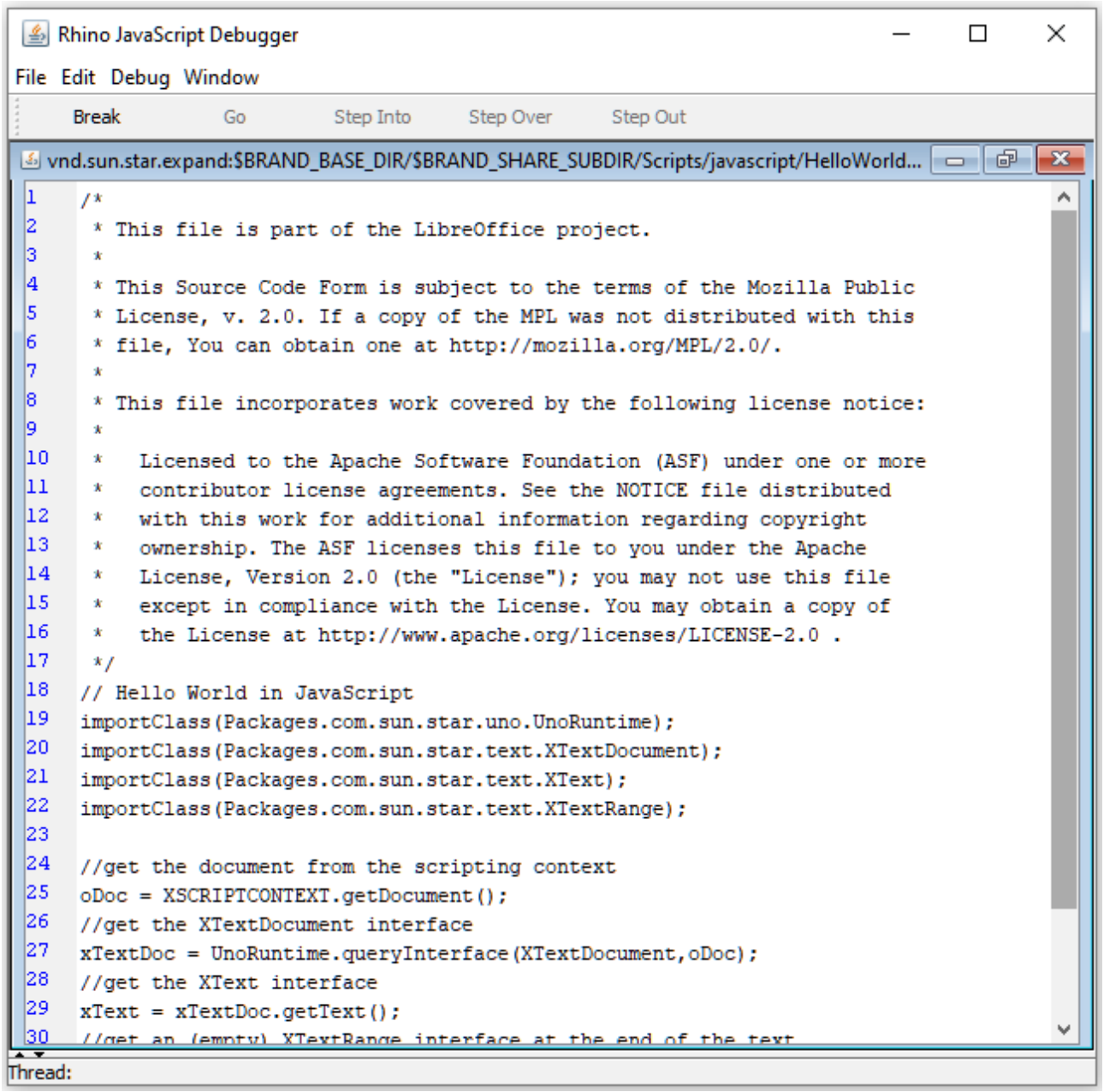


Figura 24: Depurador Rhino de JavaScript

El Listado 11 es un ejemplo de una macro *JavaScript* que inserta el texto *"Hello World from JavaScript"* en la celda **A1** de la primera hoja en la hoja de cálculo Calc.

Listado 11. Muestra de macro JavaScript

```
importClass(Packages.com.sun.star.uno.UnoRuntime);
importClass(Packages.com.sun.star.sheet.XSpreadsheetDocument);
importClass(Packages.com.sun.star.container.XIndexAccess);
importClass(Packages.com.sun.star.table.XCellRange);
importClass(Packages.com.sun.star.table.XCell);

documentRef = XSCRIPTCONTEXT.getDocument();

spreadsheetInterface = UnoRuntime.queryInterface(XSpreadsheetDocument,
documentRef);

allSheets = UnoRuntime.queryInterface(XIndexAccess,
spreadsheetInterface.getSheets());

theSheet = allSheets.getByIndex(0);

Cells = UnoRuntime.queryInterface(XCellRange,theSheet);

cellA1 = Cells.getCellByPosition(0,0);

theCell = UnoRuntime.queryInterface(XCell,cellA1);

theCell.setFormula("Hello World from JavaScript");
```

Macros de Python

Python es un lenguaje de programación general de alto nivel que fue lanzado por primera vez en 1991.

Cuando seleccione **Herramientas > Macros > Organizar Macros > Python** de la *barra de menú*, Calc muestra el cuadro de diálogo *Macros en Python* (*Figura 25*).

Figura 25: Cuadro de diálogo Macros en Python

Las funciones para editar y depurar las secuencias de comandos *Python* no están actualmente incorporadas en la interfaz de usuario estándar de LibreOffice. No obstante, puede editar las secuencias de comandos *Python* con el editor de texto que prefiera o en un *IDE* externo. La extensión *Alternative Python Script Organizer (APSO)* facilita la edición de las secuencias de comandos *Python*, especialmente cuando están incrustadas en un documento. Al utilizar *APSO*, puede configurar el editor de código fuente que desee, ejecutar el *shell de Python* integrado y depurar las secuencias de comandos *Python*.

Para obtener más información, busque *Python* en el sistema *Ayuda de LibreOffice* y visite la sección *Diseño y desarrollo de aplicaciones en Python* de la *wiki* de *The Document Foundation* (https://wiki.documentfoundation.org/Macros/Python_Design_Guide).

El Listado 12 es un ejemplo de una macro *Python* que establece la celda **A1** de la primera hoja en una hoja de cálculo Calc para el texto *"Hello World from Python"*.

Listado 12. Muestra de la macro Python

```
import uno

def HelloWorld():
    doc = XSCRIPTCONTEXT.getDocument()
    cell = doc.Sheets[0]['A1']
    cell.setString('Hello World from Python')
    return
```

Conclusión

Este capítulo proporciona un resumen de cómo crear bibliotecas y módulos, cómo utilizar la grabadora de macros, cómo utilizar las macros como funciones de Calc y cómo escribir sus propias macros sin la grabadora de macros. Cada tema merece al menos un capítulo y aprender a

escribir sus propias macros para Calc podría fácilmente requerir un libro entero. En resumen, esto es solo el comienzo de todo lo que puede aprender.

Más información sobre las características de las macros de Calc puede obtenerse desde el sistema *Ayuda*, las páginas de *wiki* de *The Document Foundation* (por ejemplo, https://wiki.documentfoundation.org/Macros) y otras fuentes de Internet (por ejemplo, el sitio de preguntas y respuestas http://ask.libreoffice.org/).