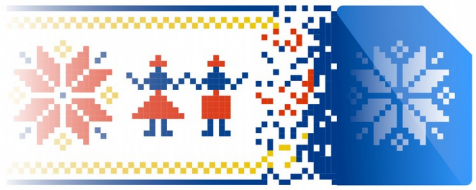
LibreOffice
The Document Foundation

September 20 - 23, 2023



Introduction to LibreOffice Development

LibreOffice automation via Scripting

Rafael Henrique Palma Lima

Why Scripting?

- Automate common and repetitive tasks
- Extend application functionalities
- Interact directly with the API
- Create custom applications tailored for your needs (integrate databases, create dialogs, forms, etc)

Supported Scripting Languages in LibreOffice

- Basic
- Python
- JavaScript
- BeanShell

Outline

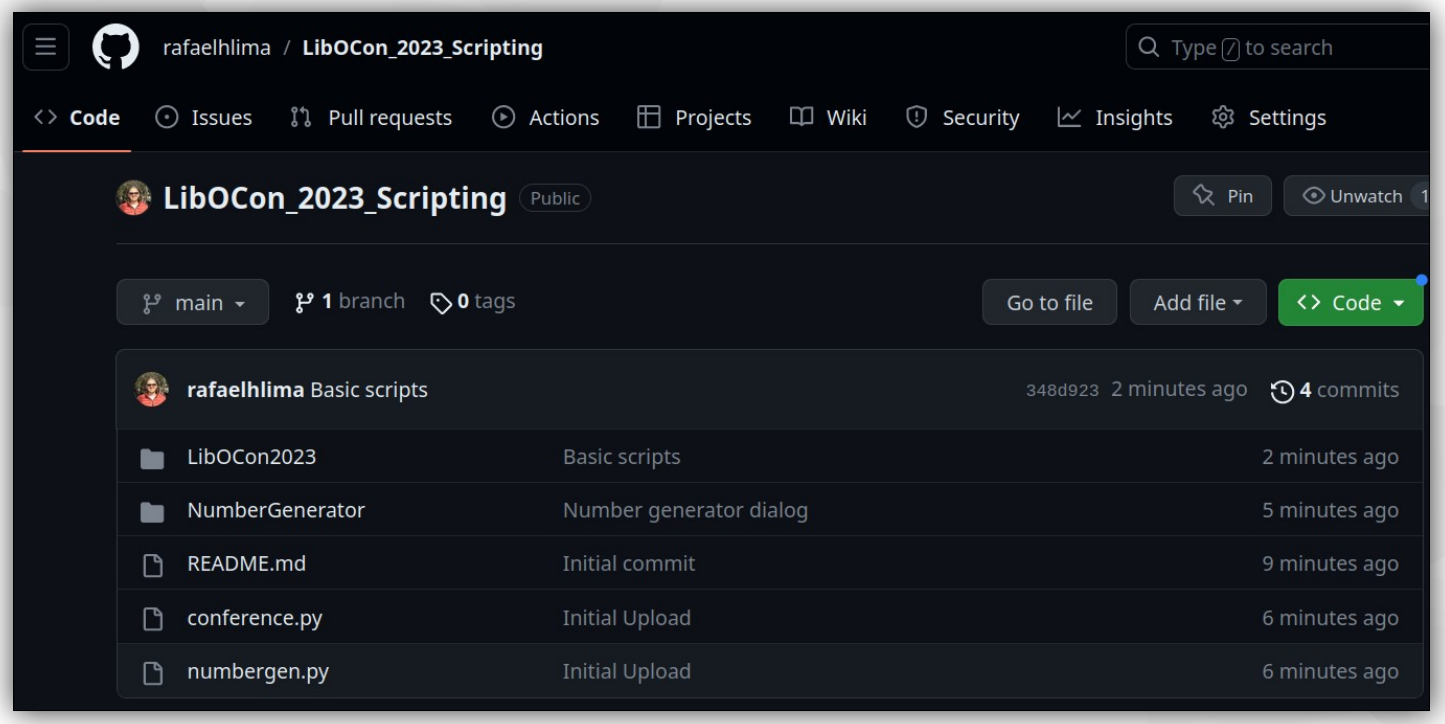
- Creating scripts in Basic
- The ScriptForge library
- Types of scripts in LibreOffice
- Creating Python scripts

python™

BASIC

GitHub repository

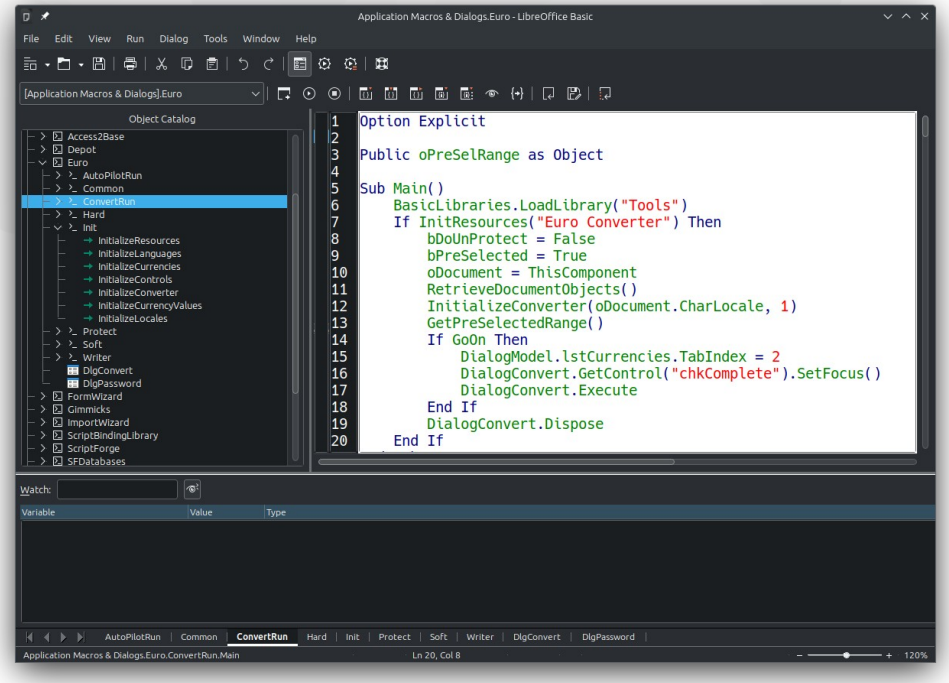
- All examples are available at the following repository

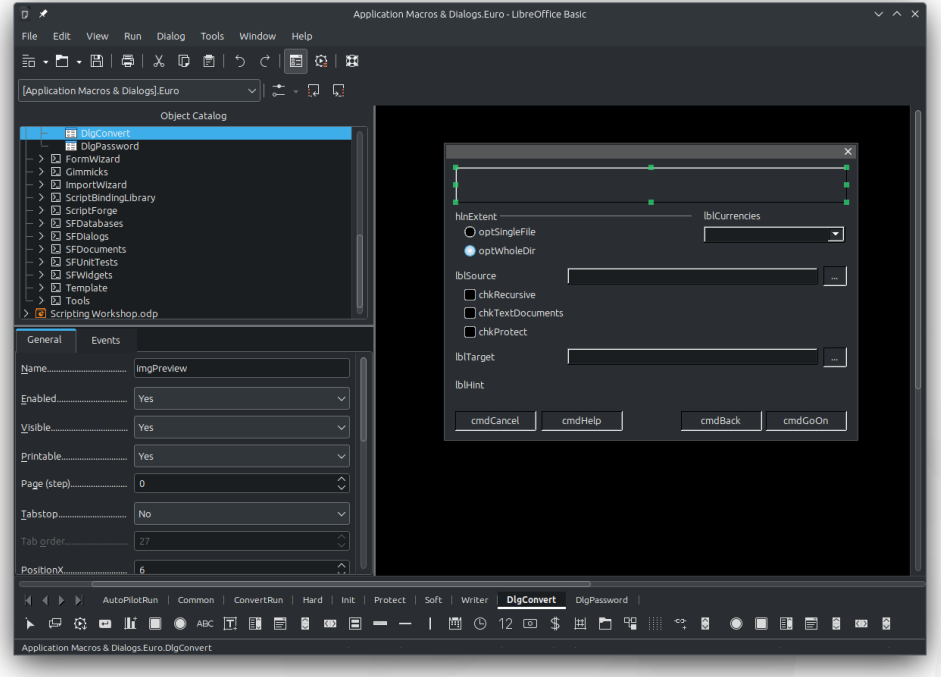


Link: https://github.com/rafaelhlima/LibOCon_2023_Scripting

Starting with the Basics

- LibreOffice has a built-in IDE for creating scripts with the Basic language
- It can also be used to create and edit dialogs

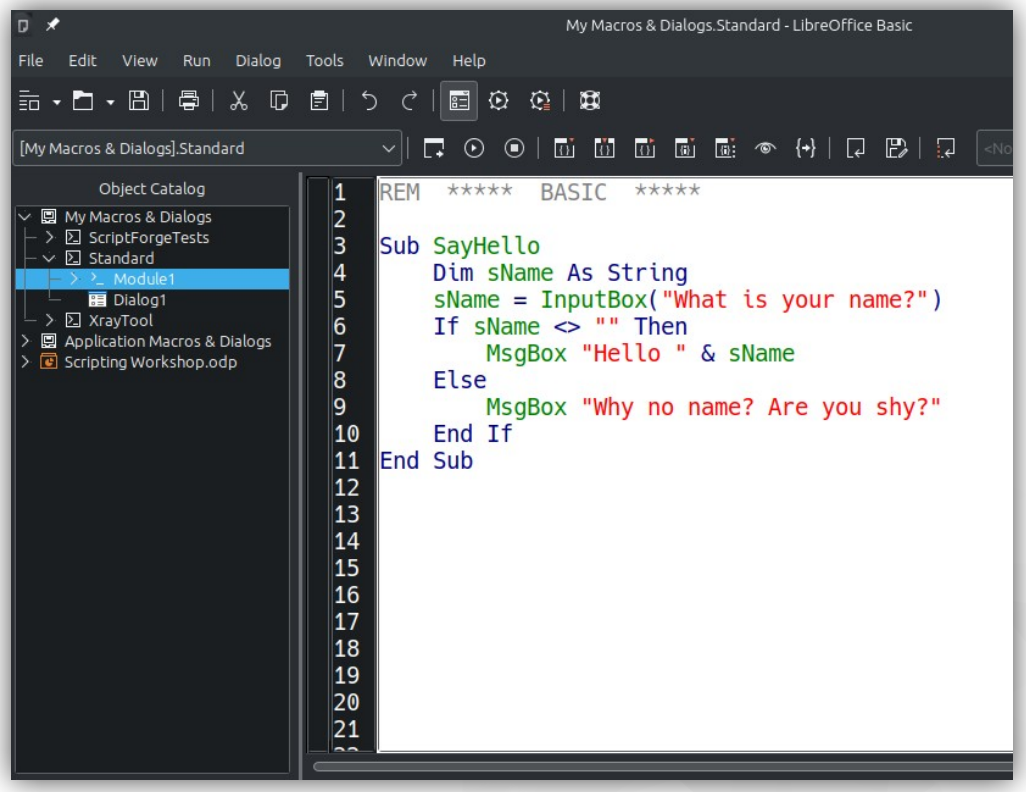




LibreOffice

Starting with the Basics

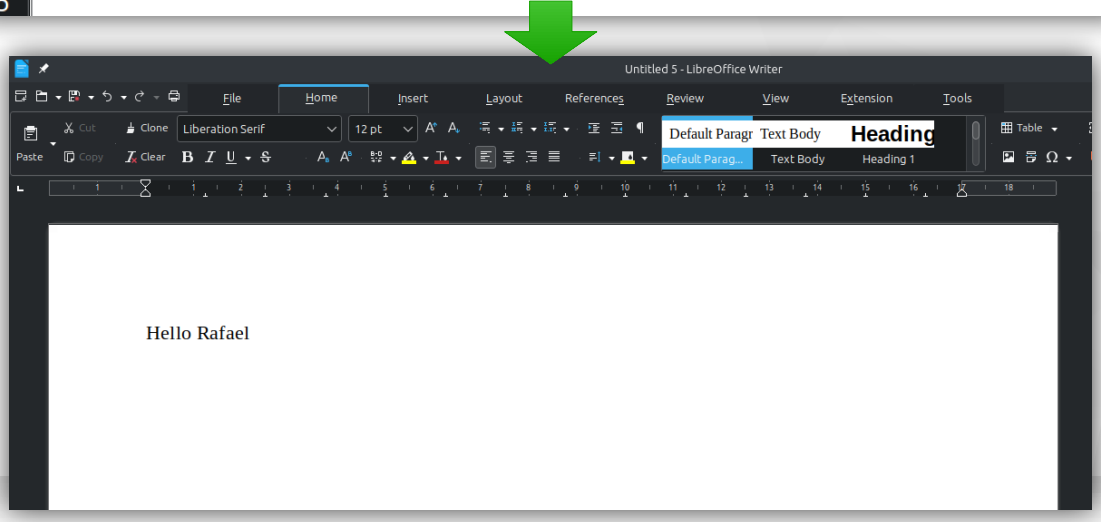
- Open any LibreOffice application (f.i. open Calc)
- Go to Tools – Macros – Edit Macros
- A default module named “Module 1” from the “Standard” library will be selected
- Create the sub SayHello by writing the next code
- Place the cursor anywhere inside the sub
- Click the Run button (or press F5)



Starting with the Basics

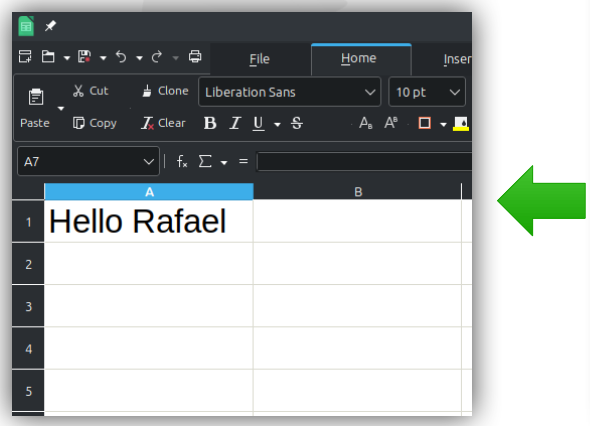
- Open any LibreOffice application
- Add the following subroutine
- This script will create a Writer document with some text in it

```
1  REM  *****  BASIC  *****
2
3  Option Explicit
4
5  Sub SayHelloWriter
6      Dim sName As String
7      sName = InputBox("What is your name?")
8      Dim oDoc As Object, oText As Object
9      oDoc = StarDesktop.loadComponentFromUrl("private:factory/swriter", _
10                                             "_blank", 0, Array())
11     oText = oDoc.getText()
12     oText.insertString(oText.End, "Hello " & sName & CHR$(10), False)
13 End Sub
14
15
```



Starting with the Basics

- Open LibreOffice Calc
- Create the next subroutine and run it
- It will write the provided name in cell “A1”




```
REM  *****  BASIC  *****

Option Explicit

Sub SayHelloCalc
    Dim sName As String
    sName = InputBox("What is your name?")
    Dim oCell As Object
    oCell = ThisComponent.Sheets(0).getCellRangeByName("A1")
    oCell.SetString("Hello " & sName)
End Sub
```

Starting with the Basics

- What are *StarDesktop* and *ThisComponent*?

StarDesktop

Special variable available in Basic to access the desktop component

More info:
https://help.libreoffice.org/latest/en-US/text/sbasic/shared/stardesktop.html

ThisComponent

Special variable that provides access to the current document

More info:
https://help.libreoffice.org/latest/en-US/text/sbasic/shared/03132200.html

Starting with the Basics

- Open LibreOffice Calc
- Create the next subroutine and run it
- It will simply create a table with Z values (standard normal distribution) and cumulative probabilities

```
REM  *****  BASIC  *****

Option Explicit

Sub CreateStdDistrTable
    Dim oSheet As Object, oRange As Object, oCell As Object
    oSheet = ThisComponent.Sheets(0)
    oRange = oSheet.getCellRangeByName("A1:B1")
    oRange.setDataArray(Array(Array("z-Value", "P(Z<z)")))
    Dim zValue As Double, nRow As Integer, sAddress As String
    nRow = 1
    For zValue = -3 To 3 Step 0.5
        oCell = oSheet.getCellByPosition(0, nRow)
        sAddress = oCell.AbsoluteName
        oCell.setValue(zValue)
        oCell = oSheet.getCellByPosition(1, nRow)
        oCell.setFormula("=NORM.S.DIST(" & sAddress & ";1)")
        nRow = nRow + 1
    Next zValue
End Sub

```

	A	B
1	z-Value	P(Z<z)
2	-3	0,001349898
3	-2,5	0,006209665
4	-2	0,022750132
5	-1,5	0,066807201
6	-1	0,158655254
7	-0,5	0,308537539
8	0	0,5
9	0,5	0,691462461
10	1	0,841344746
11	1,5	0,933192799
12	2	0,977249868
13	2,5	0,993790335
14	3	0,998650102

Starting with the Basics

- This code formats the table with a header and borders

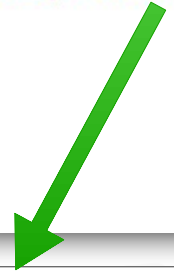
	A	B
1	z-Value	P(Z<z)
2	-3	0,001349898
3	-2,5	0,006209665
4	-2	0,022750132
5	-1,5	0,066807201
6	-1	0,158655254
7	-0,5	0,308537539
8	0	0,5
9	0,5	0,691462461
10	1	0,841344746
11	1,5	0,933192799
12	2	0,977249868
13	2,5	0,993790335
14	3	0,998650102

```
1  REM  *****  BASIC  *****
2
3  Option Explicit
4
5  Sub FormatRange
6      Dim oRange As Object
7      ' Format header
8      oRange = ThisComponent.Sheets(0).getCellRangeByName("A1:B1")
9      oRange.CellBackColor = RGB(200, 200, 200)
10     oRange.CharWeight = com.sun.star.awt.FontWeight.BOLD
11     'Format the entire table
12     oRange = ThisComponent.Sheets(0).getCellRangeByName("A1:B14")
13     oRange.CharFontName = "Arial"
14     ' Align contents at the center
15     Dim eAlign : eAlign = com.sun.star.table.CellHoriJustify
16     oRange.HoriJustify = eAlign.CENTER
17     ' Create a border format object
18     Dim lineFormat As New com.sun.star.table.BorderLine2
19     lineFormat.LineStyle = com.sun.star.table.BorderLineStyle.SOLID
20     lineFormat.LineWidth = 10
21     ' Set borders around all cells
22     oRange.TopBorder = lineFormat
23     oRange.BottomBorder = lineFormat
24     oRange.LeftBorder = lineFormat
25     oRange.RightBorder = lineFormat
26 End Sub
27
```

Starting with the Basics

- The code below calls both Subs and makes sure they are only executed on Calc documents

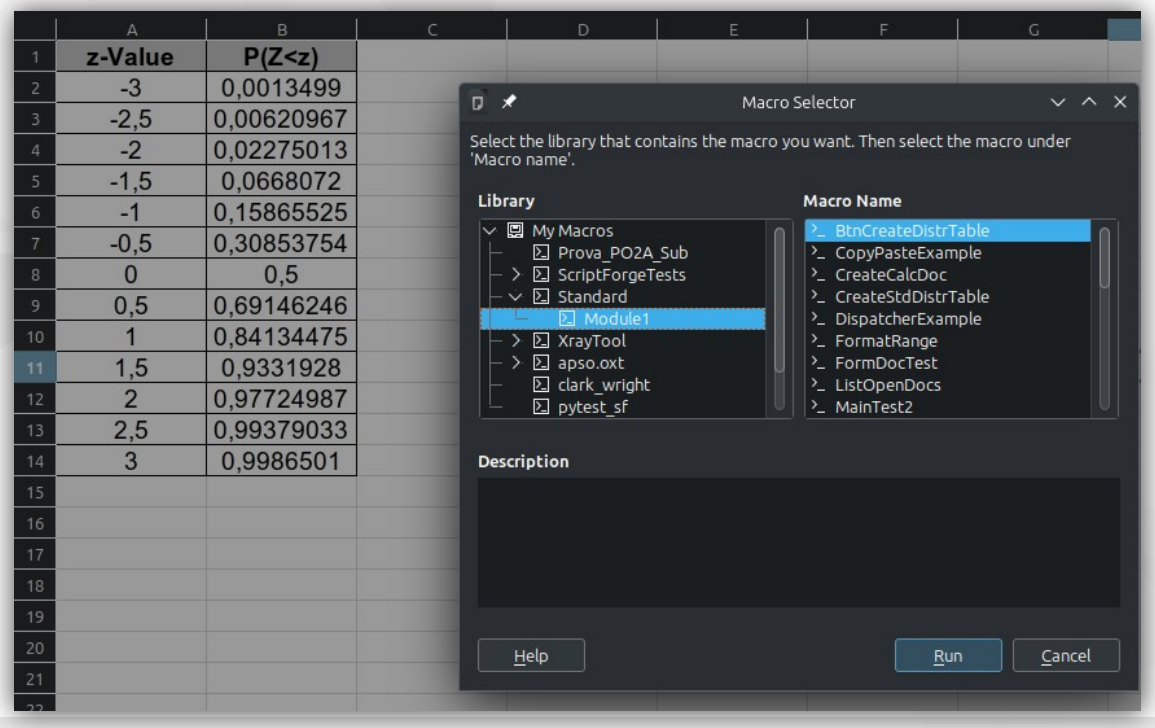
```
98  Sub BtnCreateDistrTable
99      ' Check if we are in a Calc document
100     If Not ThisComponent.SupportsService("com.sun.star.sheet.SpreadsheetDocument") Then
101         Exit Sub
102     End If
103     ' Call both sub routines
104     CreateStdDistrTable
105     FormatRange
106 End Sub
```



Calc	com.sun.star.sheet.SpreadsheetDocument
Draw	com.sun.star.drawing.DrawDocument
Impress	com.sun.star.presentation.PresentationDocument
Writer	com.sun.star.text.TextDocument
Base	com.sun.star.sdb.OfficeDatabaseDocument

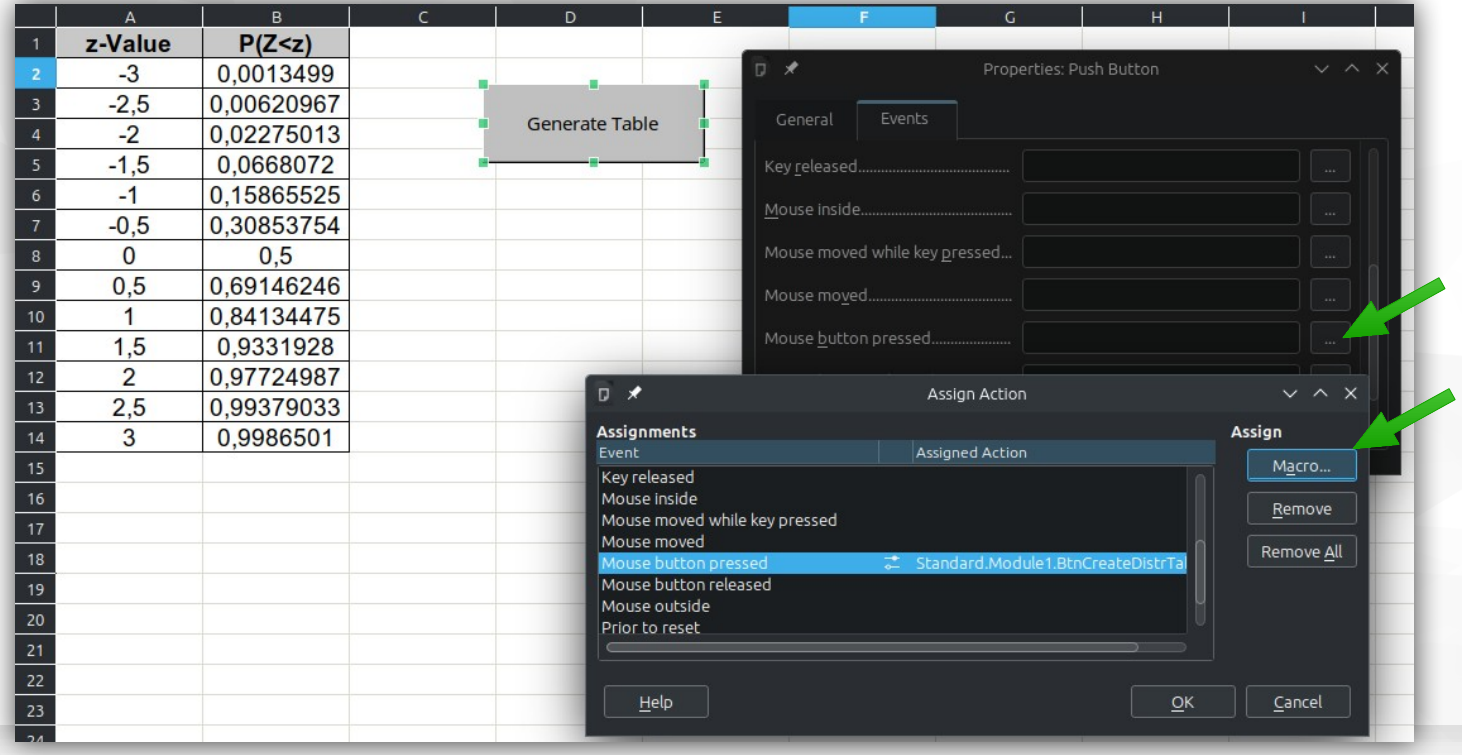
Running Scripts

- The easiest way to run a script is to go to **Tools – Macros – Run Macro** and the use the Macro Selector to choose the desired macro



Running Scripts

- You can also add controls to the document and associate macros with control events. For instance, create a button and associate it with our macro:



How do I learn all of that?

API references for the previous example

- **(service) com.sun.star.table.CellRange**

https://api.libreoffice.org/docs/idl/ref/servicecom_1_1sun_1_1star_1_1table_1_1CellRange.html

- **(service) com.sun.star.text.CellRange**

https://api.libreoffice.org/docs/idl/ref/servicecom_1_1sun_1_1star_1_1text_1_1CellRange.html

- **(service) com.sun.star.style.CharacterProperties**

https://api.libreoffice.org/docs/idl/ref/servicecom_1_1sun_1_1star_1_1style_1_1CharacterProperties.html

- **(struct) com.sun.star.table.BorderLine2**

https://api.libreoffice.org/docs/idl/ref/structcom_1_1sun_1_1star_1_1table_1_1BorderLine2.html

- **(constant group) com.sun.star.table.BorderLineStyle**

https://api.libreoffice.org/docs/idl/ref/namespacecom_1_1sun_1_1star_1_1table_1_1BorderLineStyle.html

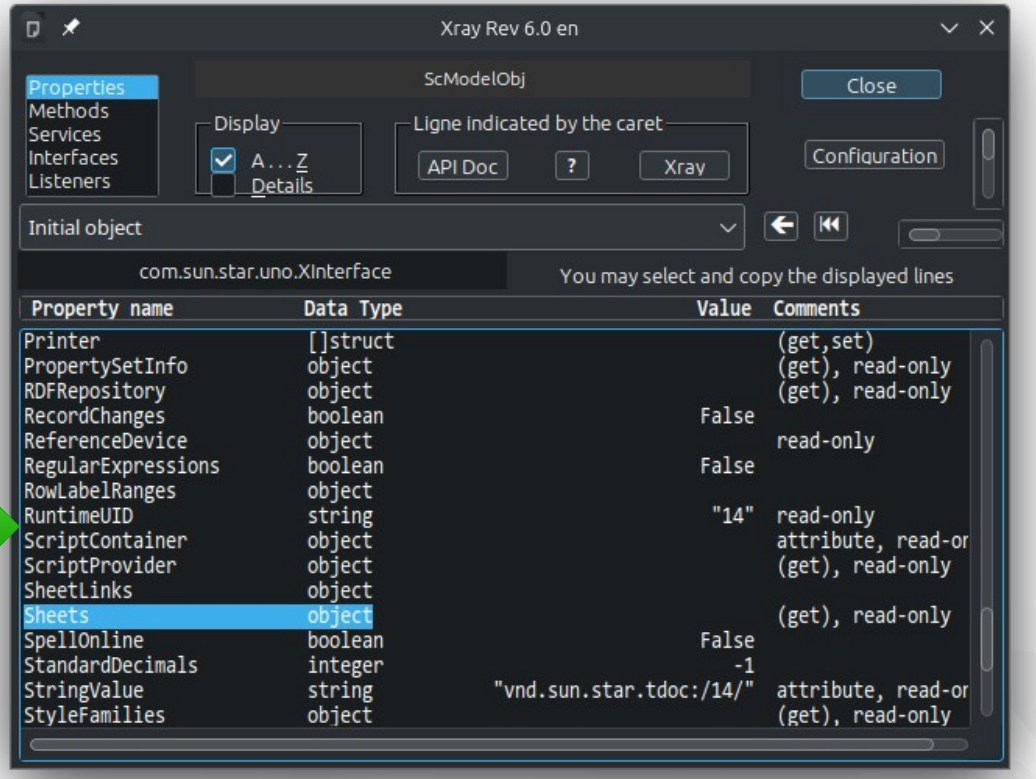
- **(constant group) com.sun.star.awt.FontWeight**

https://api.libreoffice.org/docs/idl/ref/namespacecom_1_1sun_1_1star_1_1awt_1_1FontWeight.html

How do I learn all of that?

- The learning curve may seem steep at first, but there are tools and learning resources to help smooth it
- **XRay tool:** used to inspect objects and helps to discover their properties and methods

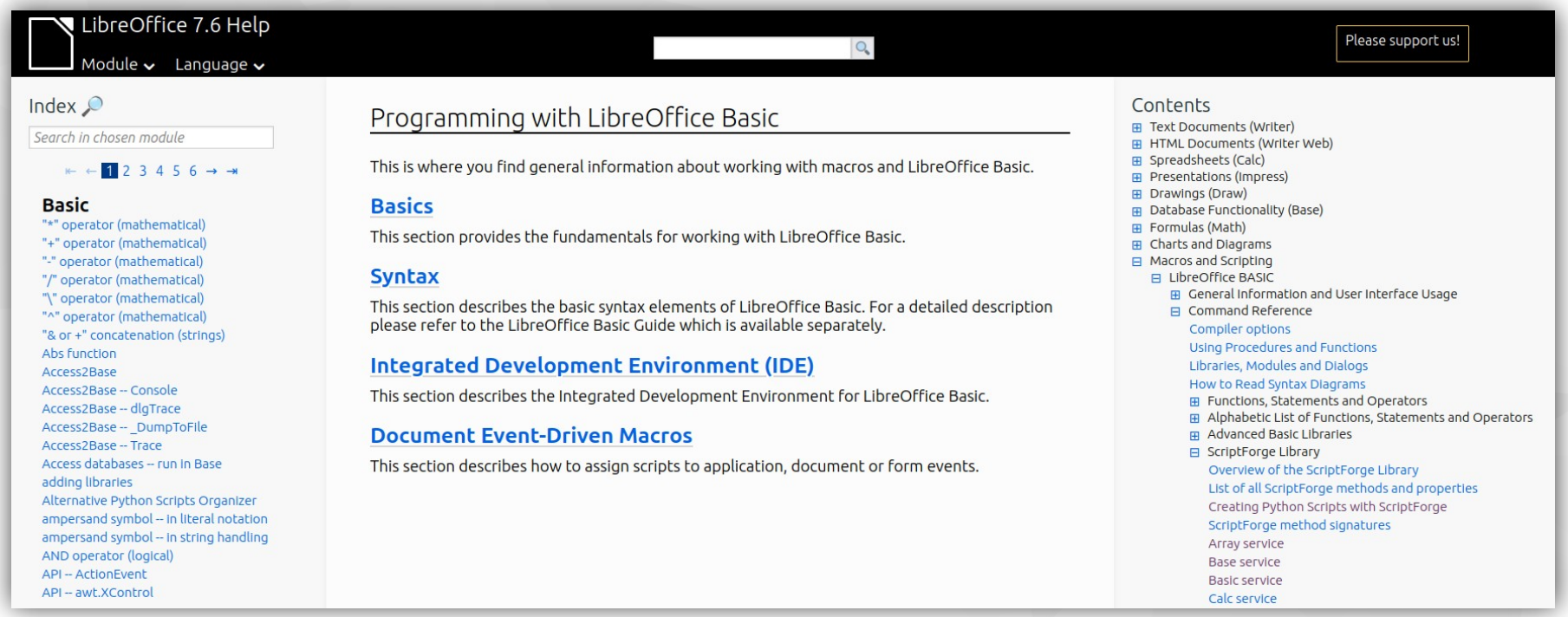
```
1  REM  *****  BASIC  *****
2
3  Option Explicit
4
5  Sub XrayExample
6      ' Inspect the object ThisComponent
7      XRay ThisComponent
8  End Sub
9
```



Download link:
https://berma.pagesperso-orange.fr/Files_en/XrayTool60_en.odt

How do I learn all of that?

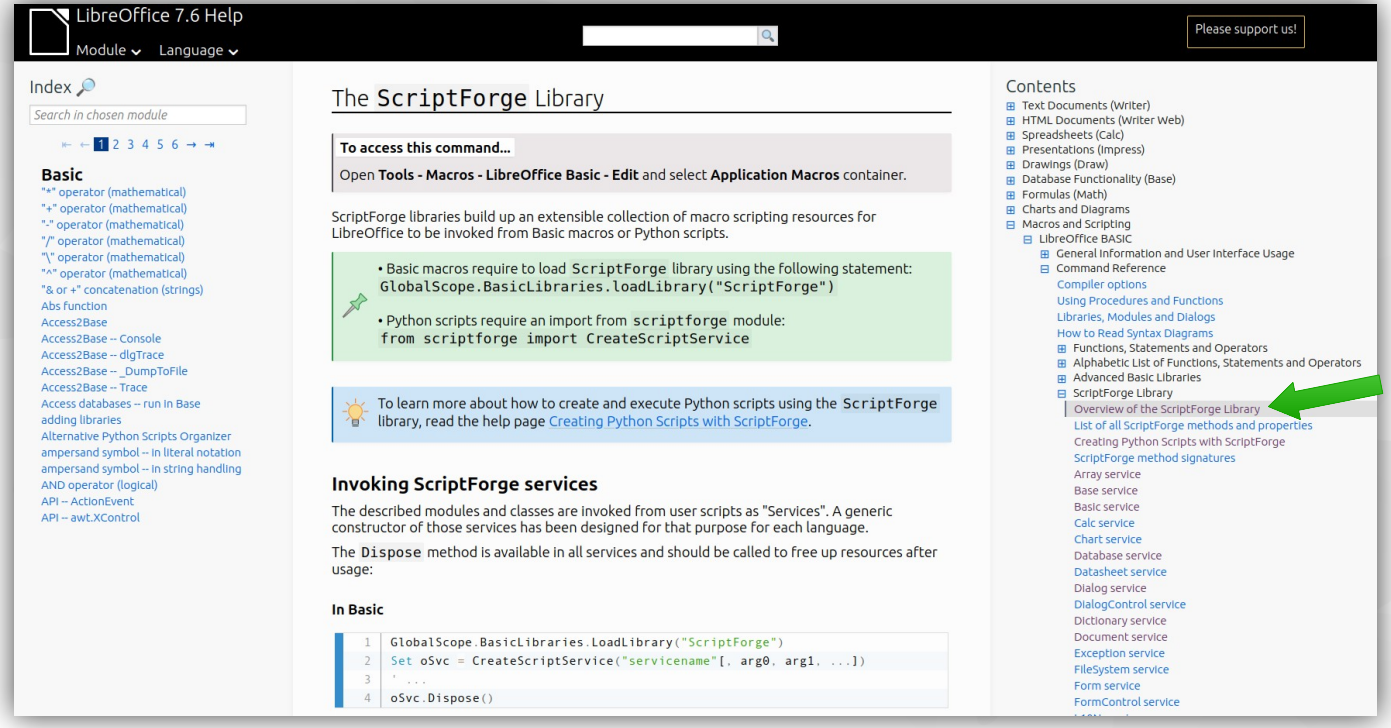
LibreOffice Basic official help pages



Link: https://help.libreoffice.org/latest/en-US/text/sbasic/shared/01000000.html

How do I learn all of that?

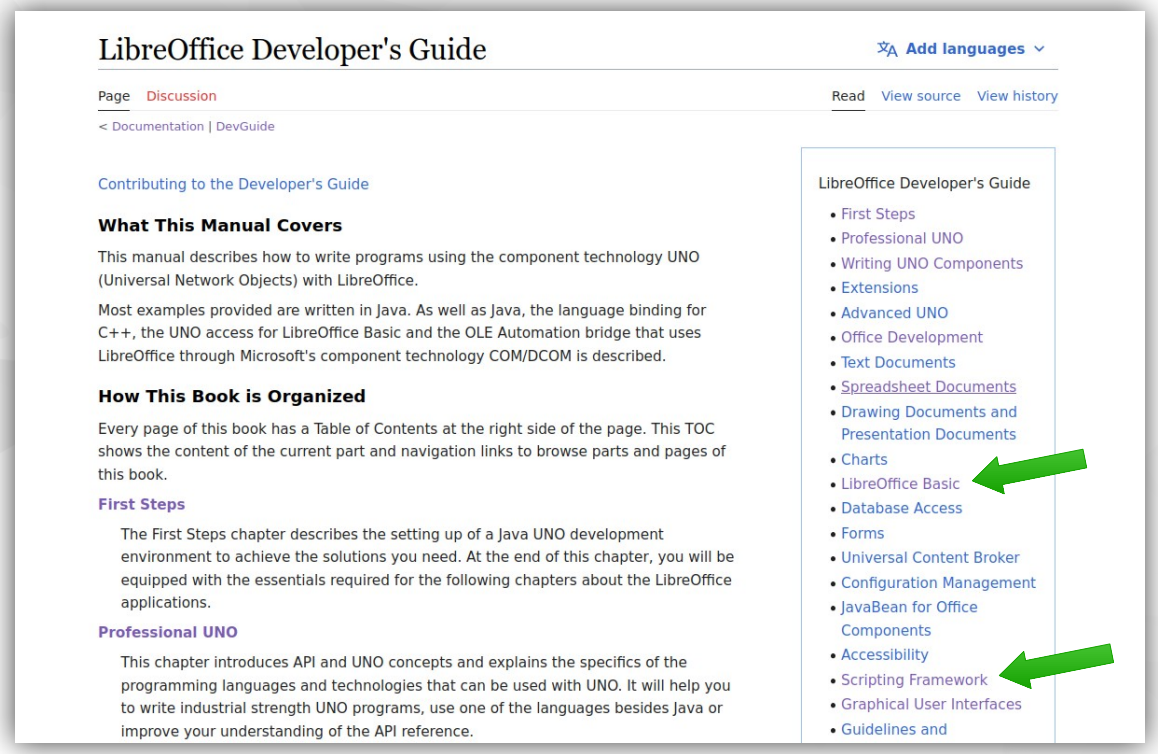
ScriptForge Library help pages



Link: https://help.libreoffice.org/latest/en-US/text/sbasic/shared/03/lib_ScriptForge.html

How do I learn all of that?

LibreOffice Developer's Guide



Link: https://wiki.documentfoundation.org/Documentation/DevGuide/LibreOffice_Developers_Guide

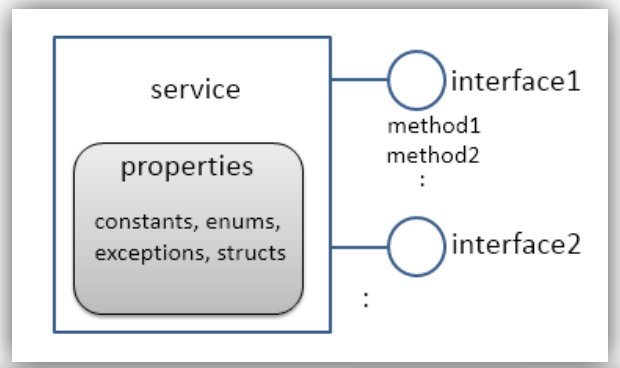
How do I learn all of that?

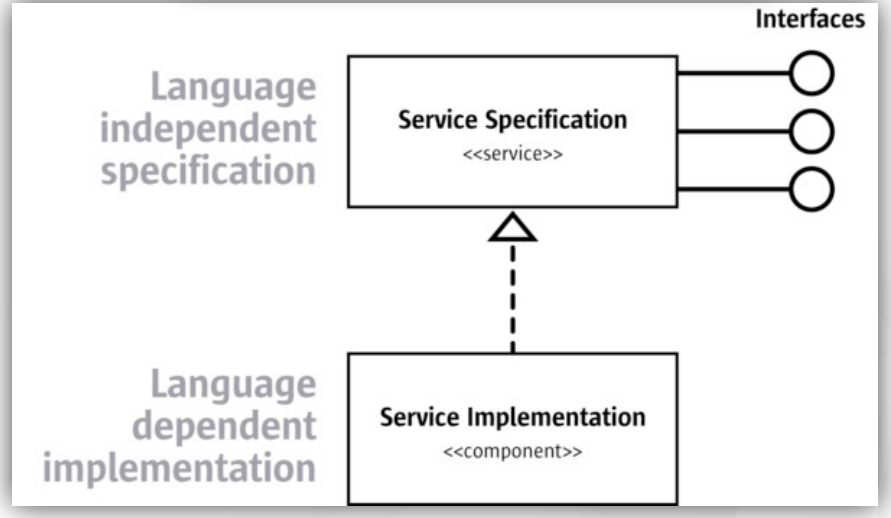
Python LibreOffice Programming

Link: https://python-ooo-dev-tools.readthedocs.io/en/latest/odev/index.html

The UNO API

- UNO stands for Universal Network Objects and is the base component technology for LibreOffice
- It allows to write components using multiple languages (C++, Java, Basic and Python)
- The Scripting framework allows to create scripts using Basic, Python, JavaScript and BeanShell
- Terminology in the API: interfaces, services, properties, constants and components





The UNO API

Service Manager

- Allows to instantiate UNO services and use them in scripts

```
REM  *****  BASIC  *****

Option Explicit

Sub SpellCheckerExample
    Dim oSpellChecker As Object, bReturn As Boolean
    Dim aLocale As New com.sun.star.lang.Locale
    aLocale.Language = "en"
    aLocale.Country = "US"
    ' Create an instance of the SpellChecker service
    oSpellChecker = CreateUnoService("com.sun.star.linguistic2.SpellChecker")
    ' Use the service to test if a word is valid
    Dim sWord As String
    sWord = InputBox("Type a word")
    bReturn = oSpellChecker.isValid(sWord, aLocale, Array())
    If bReturn Then
        MsgBox "The word '" & sWord & "' is spelled correctly"
    Else
        MsgBox "The word '" & sWord & "' is not spelled correctly"
    End If
End Sub
```

Service documentation:
https://api.libreoffice.org/docs/idl/ref/servicecom_1_1sun_1_1star_1_1linguistic2_1_1SpellChecker.html

UNO Commands

Command Dispatcher: you can also dispatch UNO commands in your scripts using the DispatchHelper service

```
5  Sub DispatcherExample
6      Dim oFrame As Object, oDispatcher As Object
7      oFrame = ThisComponent.CurrentController.Frame
8      oDispatcher = CreateUnoService("com.sun.star.frame.DispatchHelper")
9      ' Dispatch the .uno:Save command (equivalent to File - Save)
10     oDispatcher.executeDispatch(oFrame, ".uno:Save", "", , Array())
11 End Sub
```

There are various commands that can be dispatched, for a full list, see https://wiki.documentfoundation.org/Development/DispatchCommands

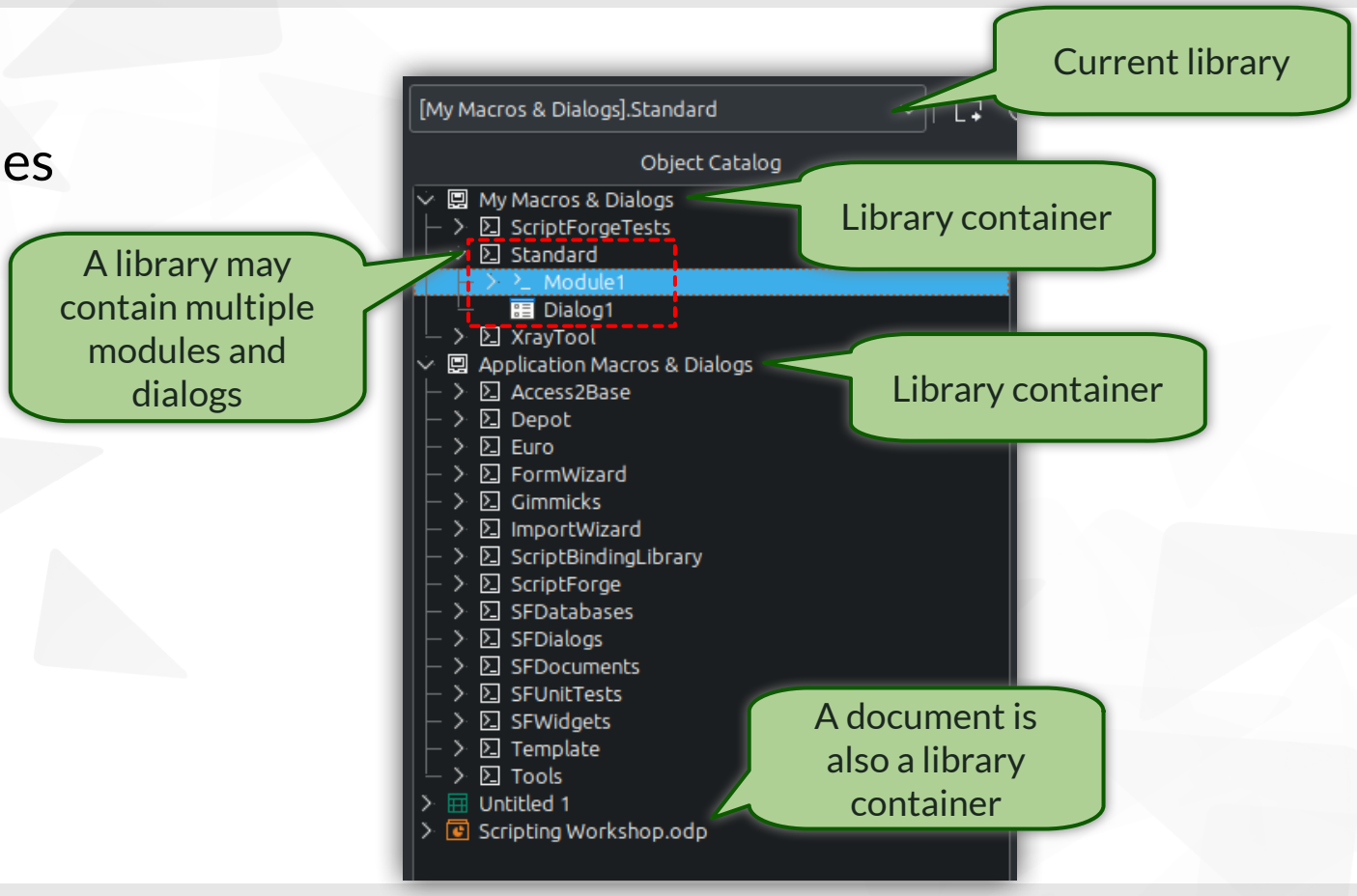
UNO Commands

- The next example uses UNO commands to copy cell A1 and paste it into cell A2

```
REM  *****  BASIC  *****

Option Explicit

Sub CopyPasteExample
    ' Instantiate the dispatcher
    Dim oFrame As Object, oDispatcher As Object
    oFrame = ThisComponent.CurrentController.Frame
    oDispatcher = CreateUnoService("com.sun.star.frame.DispatchHelper")
    ' Moves to cell A1
    Dim args1(0) As New com.sun.star.beans.PropertyValue
    args1(0).Name = "ToPoint"
    args1(0).Value = "$A$1"
    oDispatcher.executeDispatch(oFrame, ".uno:GoToCell", "", , args1())
    ' Copy its contents (equivalent to Edit - Copy)
    oDispatcher.executeDispatch(oFrame, ".uno:Copy", "", , Array())
    ' Moves to cell A2
    args1(0).Value = "$A$2"
    oDispatcher.executeDispatch(oFrame, ".uno:GoToCell", "", , args1())
    ' Paste contentx (equivalent to Edit - Paste)
    oDispatcher.executeDispatch(oFrame, ".uno:Paste", "", , Array())
End Sub

```

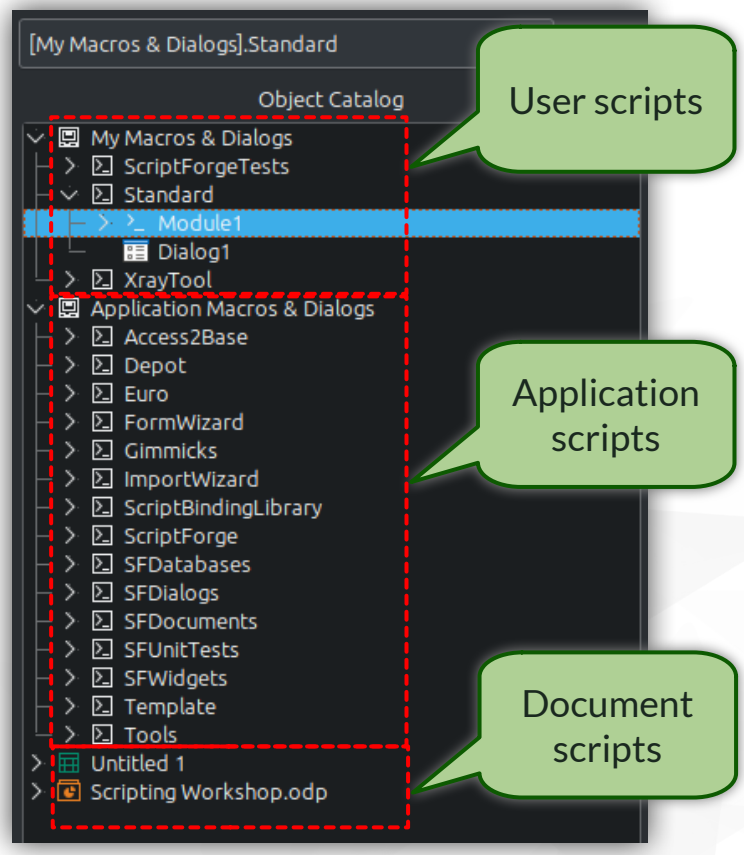
Organizing Scripts

- Containers, libraries, modules and dialogs



Organizing Scripts

- **My Macros & Dialogs:** available to all documents for the current user
- **Application Macros & Dialogs:** available to all users and documents (read-only)
- **Document Macros & Dialogs:** used to embed macros in a document



ScriptForge Library

- Offers a variety of services and methods to simplify the creation of Basic and Python scripts by hiding the complexity of the UNO API

Category	Services		
LibreOffice Basic	Array Dictionary	Exception FileSystem	String TextStream
Document Content	Base Calc Chart	Database Datasheet	Document Writer
User Interface	Dialog DialogControl Form	FormControl Menu	PopupMenu UI
Utilities	Basic L10N Platform	Region Services Session	Timer UnitTest

Extend the Basic API

Access document contents

Access dialogs and dialog controls, as well as UI elements

Utilities to simplify common tasks performed by macros

ScriptForge Library

- We need to load the ScriptForge library before using it

```
5  Sub ListOpenDocs
6      ' We need to make sure the ScriptForge library is loaded
7      GlobalScope.BasicLibraries.LoadLibrary("ScriptForge")
8      '...
9  End Sub
```

- We can create a library loader do make it easier

```
5  Sub RequiresSF
6      If Not GlobalScope.BasicLibraries.IsLibraryLoaded("ScriptForge") Then
7          GlobalScope.BasicLibraries.LoadLibrary("ScriptForge")
8      End If
9  End Sub
10
11 Sub ListOpenDocs
12     RequiresSF
13     Dim svcUI As Object
14     svcUI = CreateScriptService("UI")
15     '...
16 End Sub
```

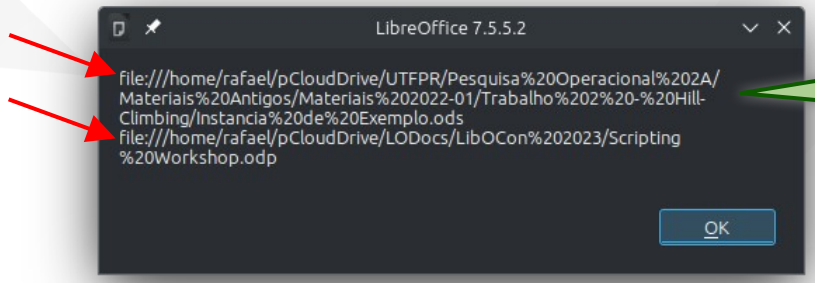
We just need to load SF just once in the same session; use this only if you're unsure SF will have already been loaded

ScriptForge Library

- **Example:** Check all open document windows

```
11  Sub ListOpenDocs
12      RequiresSF
13      Dim svcUI As Object, arrDocs As Object
14      svcUI = CreateScriptService("UI")
15      ' Array with the list of open documents
16      arrDocs = svcUI.Documents
17      ' Concatenate the names in the array
18      Dim sOpenDocs As String
19      sOpenDocs = Join(arrDocs, CHR$(13))
20      MsgBox sOpenDocs
21  End Sub
```

The UI service provides access to all open windows



The documents here are shown using URL notation

ScriptForge Library

- **Example:** Open an existing Calc document and maximize its window

```
5   Sub OpenCalcDoc
6       Dim sDocPath As String, svcUI As Object, oDoc As Object
7       sDocPath = "file:///home/rafael/Documents/mydoc.ods"
8       svcUI = CreateScriptService("UI")
9       oDoc = svcUI.OpenDocument(sDocPath)
10      svcUI.Maximize(sDocPath)
11      MsgBox "This is a " & oDoc.DocumentType & " file"
12  End Sub
```

In this macro "oDoc" is an instance of the "Calc" service

ScriptForge Library

- **Example:** Creates an empty Calc document, adds some content and saves it without showing the window to the user

```
5   Sub CreateCalcDoc
6       Dim sDocPath As String, svcUI As Object, oDoc As Object
7       sDocPath = "file:///home/rafael/Documents/newdoc.ods"
8       svcUI = CreateScriptService("UI")
9       oDoc = svcUI.CreateDocument(DocumentType := "Calc", Hidden := True)
10      oDoc.SetValue("A1", "Hello")
11      oDoc.SaveAs(sDocPath, Overwrite := True)
12      oDoc.CloseDocument()
13  End Sub
```

ScriptForge Library

- Services are composed of properties and methods (see their documentation in the corresponding help pages); Below is the “Document” service:

Properties

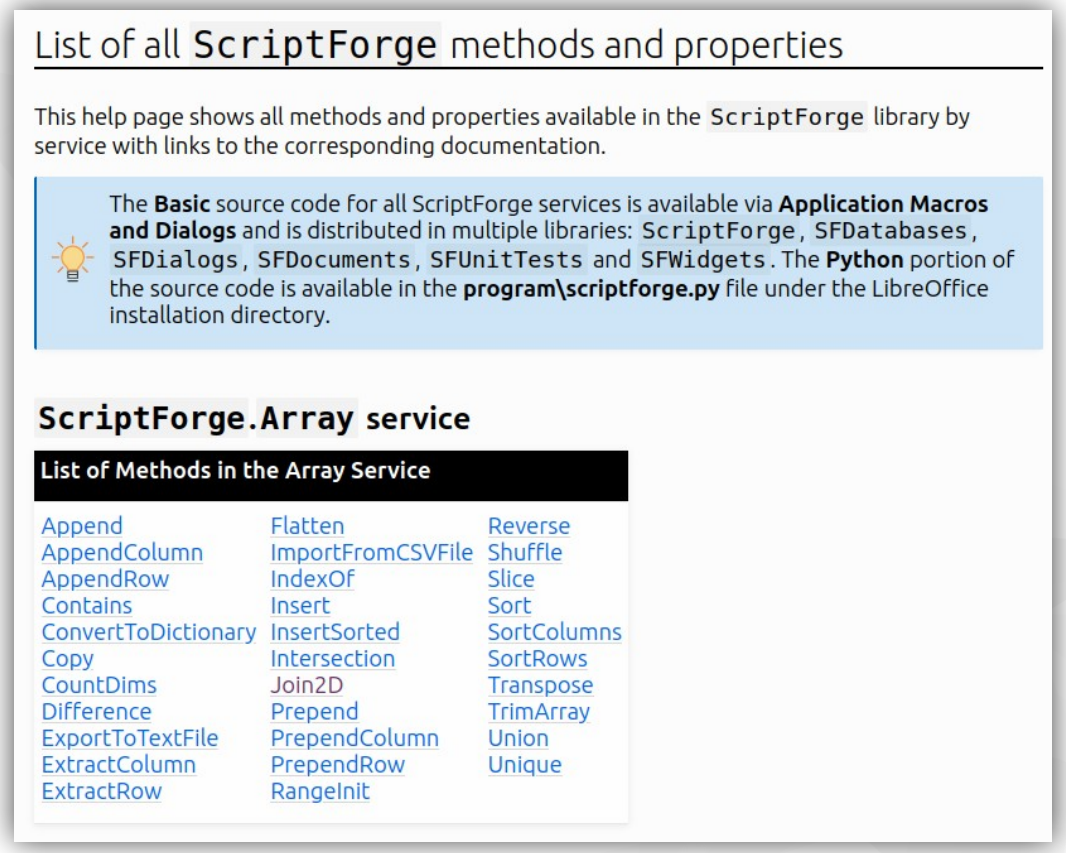
Name	Readonly	Type	Description
CustomProperties (*)	No	Dictionary service	Returns a `ScriptForge.Dictionary` object instance. After update, can be passed again to the property for updating the document. Individual items of the dictionary may be either strings, numbers, (Basic) dates or `com.sun.star.util.Duration` items.
Description (*)	No	String	Gives access to the Description property of the document (also known as "Comments")
DocumentProperties (*)	Yes	Dictionary service	Returns a `ScriptForge.Dictionary` object containing all the entries. Document statistics are included. Note that they are specific to the type of document. As an example, a Calc document includes a "CellCount" entry. Other documents do not.
DocumentType	Yes	String	String value with the document type ("Base", "Calc", "Writer", etc)
ExportFilters	Yes	String array	Returns a list with the export filter names applicable to the current document as a zero-based array of strings. Filters used for both import/export are also returned.

List of Methods in the Document Service

Activate	PrintOut	SaveAs
CloseDocument	RemoveMenu	SaveCopyAs
CreateMenu	RunCommand	SetPrinter
ExportAsPDF	Save	

ScriptForge Library

- This help page has a list of all properties and methods provided by all ScriptForge services



List of all `ScriptForge` methods and properties

This help page shows all methods and properties available in the `ScriptForge` library by service with links to the corresponding documentation.

The **Basic** source code for all ScriptForge services is available via **Application Macros and Dialogs** and is distributed in multiple libraries: `ScriptForge`, `SFDatabases`, `SFDialogs`, `SFDocuments`, `SFUnitTests` and `SFWidgets`. The **Python** portion of the source code is available in the **program\scriptforge.py** file under the LibreOffice installation directory.

`ScriptForge.Array` service

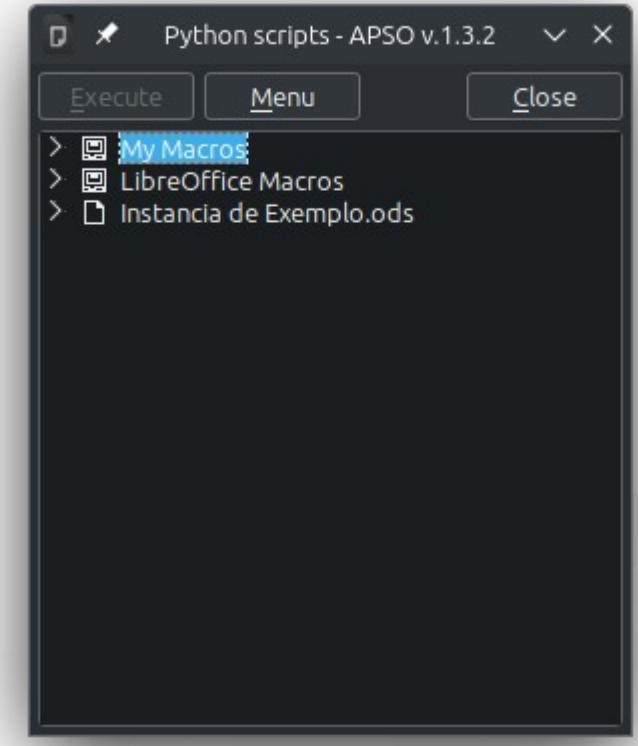
List of Methods in the Array Service

Append	Flatten	Reverse
AppendColumn	ImportFromCSVFile	Shuffle
AppendRow	IndexOf	Slice
Contains	Insert	Sort
ConvertToDictionary	InsertSorted	SortColumns
Copy	Intersection	SortRows
CountDims	Join2D	Transpose
Difference	Prepend	TrimArray
ExportToTextFile	PrependColumn	Union
ExtractColumn	PrependRow	Unique
ExtractRow	RangeInit	

Link: https://help.libreoffice.org/latest/en-US/text/sbasic/shared/03/sf_toc.html

Creating Python Scripts

- LibreOffice has supported Python scripts since its first release in 2010
- The default LibreOffice installation comes with a bundled Python interpreter
- However, LibreOffice does not have an integrated Python IDE (yet ☺)
- To make it easier to create and run Python scripts, it is recommended to use the APSO (Alternative Script Organizer for Python) extension



Download link:
https://extensions.libreoffice.org/en/extensions/show/apso-alternative-script-organizer-for-python

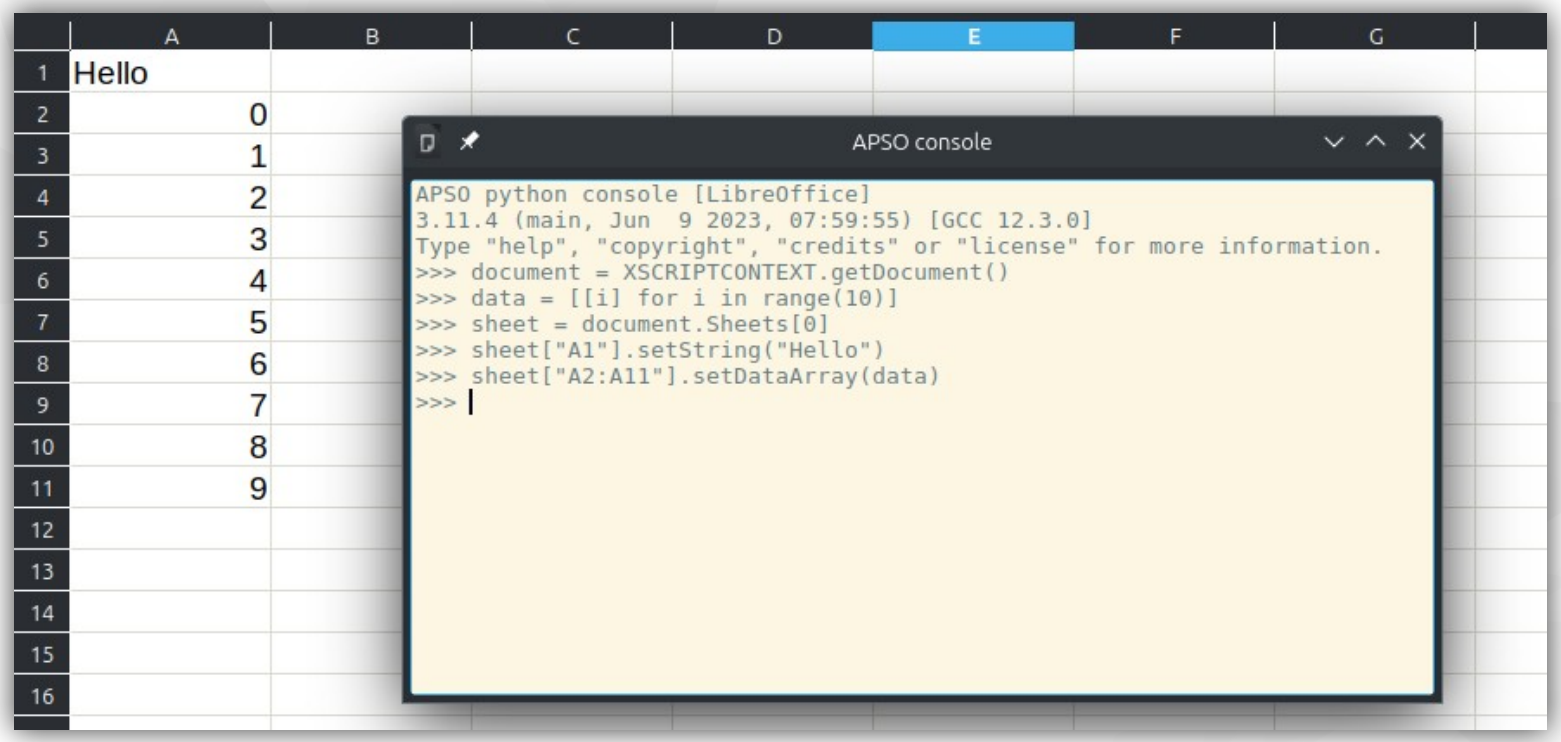
Creating Python Scripts

- You can use APSO's Python console to execute Python commands

XSCRIPTCONTEXT is documented here:
https://help.libreoffice.org/latest/en-US/text/sbasic/python/python_programming.htm

Creating Python Scripts

- In Python scripts we can use a more “Pythonic” syntax:



Creating Python Scripts

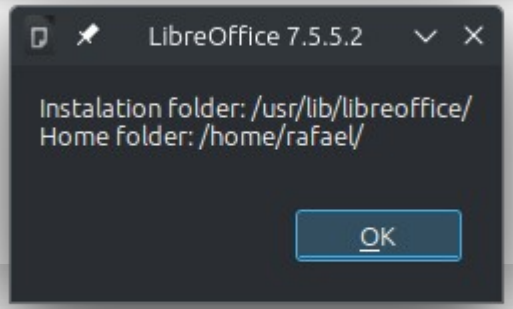
Where can we create user scripts that can be used across documents?

- Application Macros (all users)
  - **For Windows:** {Installation}\share\Scripts\python
  - **For Linux and macOS:** {Installation}/share/Scripts/python
- User Macros (current user only)
  - **For Windows:** %APPDATA%\LibreOffice\4\user\Scripts\python
  - **For Linux and macOS:** $HOME/.config/libreoffice/4/user/Scripts/python

Creating Python Scripts

- Run the following Basic script to locate where your installation and home folder are (note that the script uses the FileSystem service from the ScriptForge library):

```
5   Sub GetFolders
6       Dim fs as Object, sMessage As String
7       fs = CreateScriptService("FileSystem")
8       ' Use native OS file naming notation
9       fs.FileNaming = "SYS"
10      sMessage = "Instalation folder: " & fs.InstallFolder & CHR$(13) & _
11                 "Home folder: " & fs.HomeFolder
12      MsgBox sMessage
13  End Sub
```



Creating Python Scripts

- Let's convert to Python the script that creates a Writer document with a hello message in it (using ScriptForge)

```
1  from scriptforge import CreateScriptService
2
3  bas = CreateScriptService("Basic")
4  ui = CreateScriptService("UI")
5
6  def create_writer_doc(args=None):
7      name = bas.InputBox("What is your name?")
8      doc = ui.createDocument("Writer")
9      doc_component = doc.XComponent
10     text = doc_component.getText()
11     text.insertString(text.End, f"Hello {name}\n", False)
12
13 g_exportedScripts = (create_writer_doc, )
14
```

Tells which scripts will be available via Tools – Macros – Run Macro dialog

Creating Python Scripts

- Now let's convert the "normal distribution" example using Python and ScriptForge

```python
from scriptforge import CreateScriptService

bas = CreateScriptService("Basic")

def create_normdist_table(args=None):
    # Get the Document service instance for the current component
    doc = CreateScriptService("Document", bas.ThisComponent)
    # If it is not a Calc document, do nothing
    if not doc.isCalc:
        return
    # Insert the data in the sheet
    doc.setValue("A1:B1", ["z-Value", "P(Z<z)"])
    z_values = [-3 + z * 0.5 for z in range(13)]
    z_range = doc.Offset("A2", height=len(z_values))
    doc.setValue(z_range, z_values)
    # Insert the formulas
    formula_range = doc.Offset("B2", height=len(z_values))
    base_formula = "=NORM.S.DIST(A2;1)"
    doc.setFormula(formula_range, base_formula)

g_exportedScripts = (create_normdist_table, )

```

Creating Python Scripts

- Now let's add the table formatting code
- Note the new imports of the UNO API

```
1  from scriptforge import CreateScriptService
2  from com.sun.star.awt import FontWeight
3  from com.sun.star.table import BorderLineStyle
4  from uno import createUnoStruct, Enum
```

Continuation of create_normdist_table

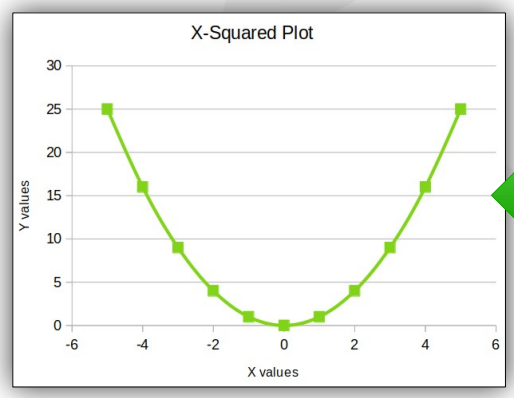
```
31      # Format the header of the table
32      range_obj = doc.XCellRange("A1:B1")
33      range_obj.CellBackColor = bas.RGB(200, 200, 200)
34      range_obj.CharWeight = FontWeight.BOLD
35      # Format the remainder of the table
36      table_range = doc.Offset("A1", width=2, height=len(z_values)+1)
37      range_obj = doc.XCellRange(table_range)
38      range_obj.CharFontName = "Arial"
39      justify_center = Enum("com.sun.star.table.CellHoriJustify", "CENTER")
40      range_obj.HoriJustify = justify_center
41      # Struct that defines the line format
42      line_format = createUnoStruct("com.sun.star.table.BorderLine2")
43      line_format.LineStyle = BorderLineStyle.SOLID
44      line_format.LineWidth = 10
45      range_obj.TopBorder = line_format
46      range_obj.BottomBorder = line_format
47      range_obj.LeftBorder = line_format
48      range_obj.RightBorder = line_format
```

Creating Python Scripts

- This example plots a chart of the X-squared function to illustrate charting capabilities

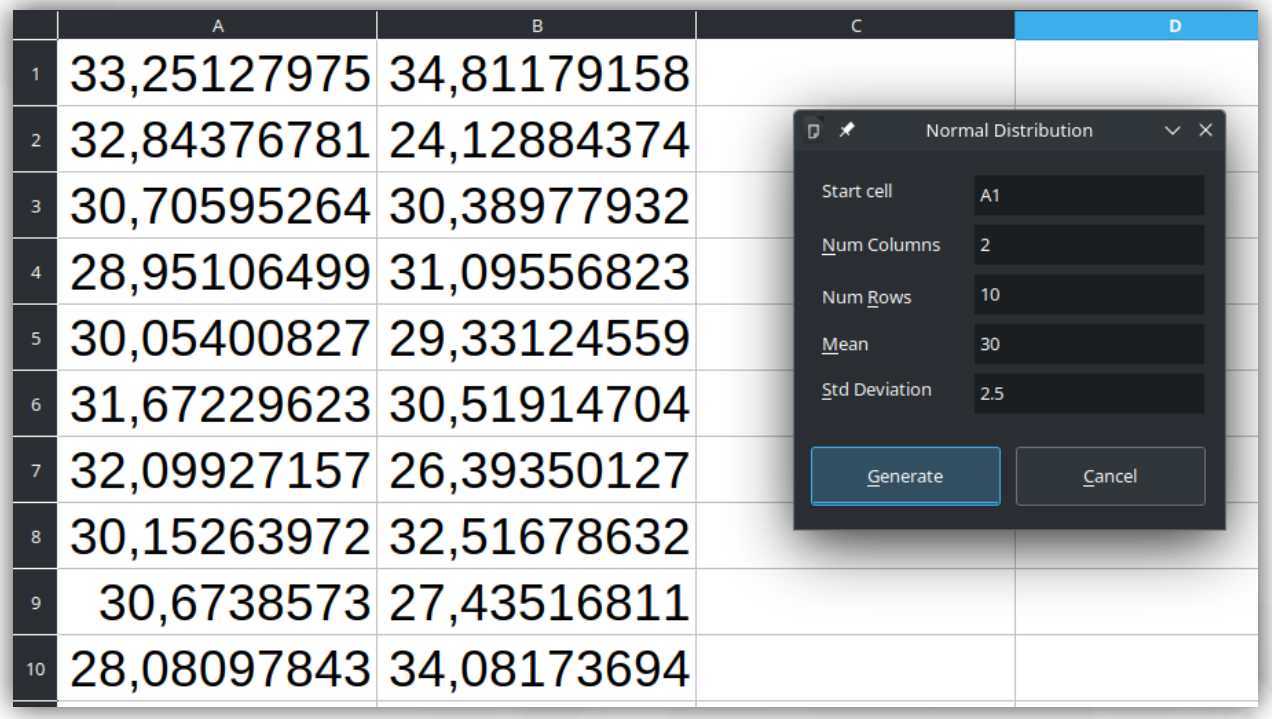
Instance of the "Chart" service

```
51  def plot_function(args=None):
52      # Plot X and Y values for X² function
53      data = [[x, math.pow(x, 2)] for x in range(-5, 6)]
54      doc = CreateScriptService("Calc", bas.ThisComponent)
55      doc.setValue("A1:B1", [["X", "Y"]])
56      data_range = doc.Offset("A2", width=2, height=len(data))
57      doc.setValue(data_range, data)
58      # Select the entire table
59      table_range = doc.Region("A1")
60      # Insert the chart
61      cur_sheet = doc.SheetName(doc.CurrentSelection)
62      chart = doc.CreateChart("X-Squared", cur_sheet, table_range)
63      chart.ChartType = "XY"
64      chart.Legend = False
65      chart.Title = "X-Squared Plot"
66      chart.XTitle = "X values"
67      chart.YTitle = "Y values"
68      # Set the line tipe to "smooth" (using cubic splines)
69      diagram = chart.XDiagram
70      diagram.SplineType = Enum("com.sun.star.chart2.CurveStyle", "CUBIC_SPLINES")
71      # Place the Y-axis at -6 for better visualization
72      y_axis = diagram.getYAxis()
73      y_axis.CrossoverPosition = Enum("com.sun.star.chart.ChartAxisPosition", "VALUE")
74      y_axis.CrossoverValue = -6
```



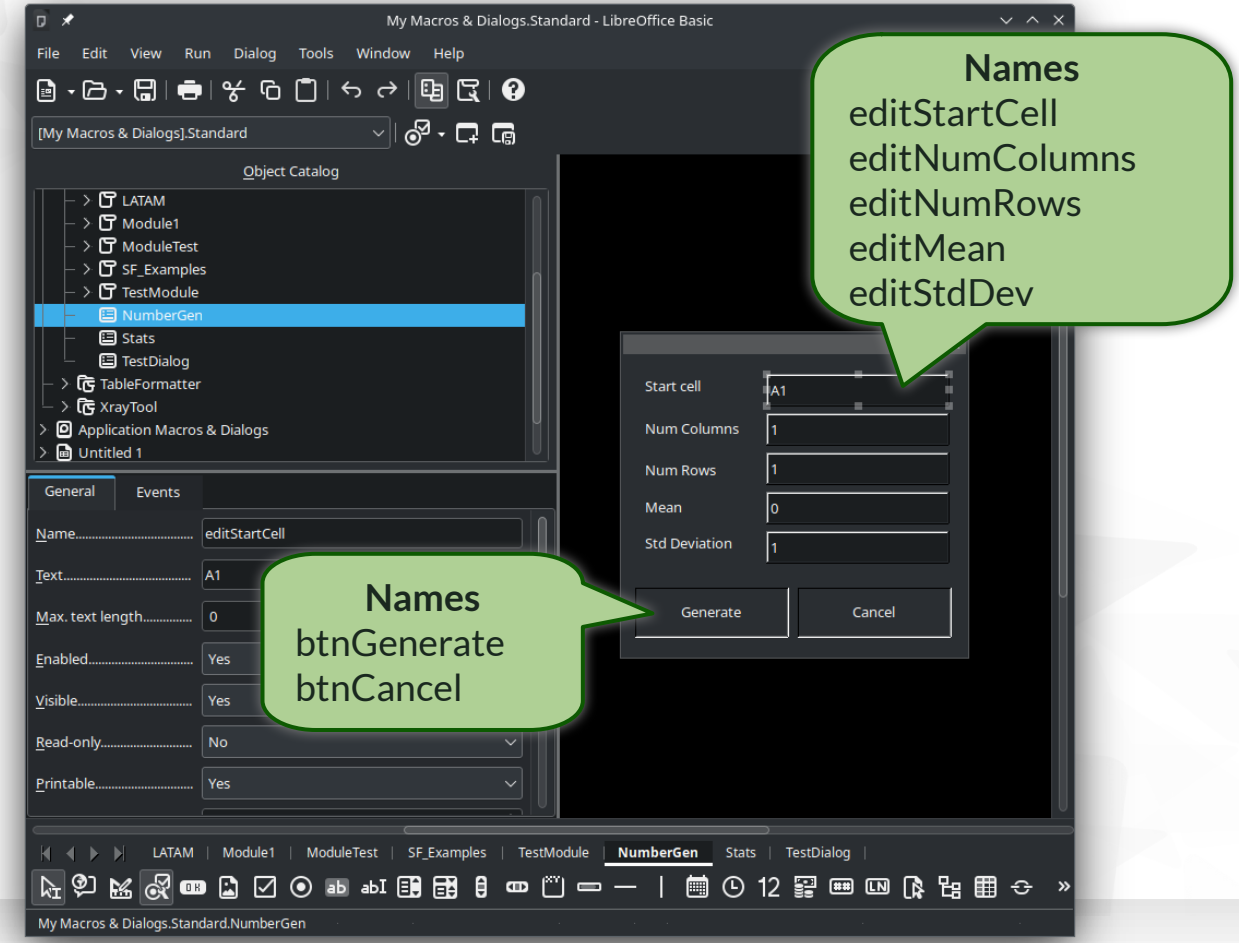
Creating Python Scripts

- Now we will create a script that uses a dialog
- It will be a simple random number generator that uses the Gaussian implementation in the “random” Python module



Creating Python Scripts

- First we need to create the dialog using the Basic IDE
- The dialog name is NumberGenDlg
- Let's create it in a new library called NumberGenerator



Creating Python Scripts

- Now let's create a Python file *numbergen.py* with the macros that open the dialog and close it when "Cancel" is pressed

```
1   from scriptforge import CreateScriptService
2   import random as rnd
3
4   bas = CreateScriptService("Basic")
5
6   def open_dialog(args=None):
7       dlg = CreateScriptService("Dialog", "GlobalScope", "NumberGenerator", "NumberGenDlg")
8       dlg.Execute()
9
10  def btn_cancel_click(event=None):
11      # Get the control that was clicked
12      control = CreateScriptService("DialogEvent", event)
13      # Get the parent dialog and terminate it
14      dlg = control.Parent
15      dlg.EndExecute(bas.IDCANCEL)
16
```

Creating Python Scripts

- This macro needs to be associated with the “Generate” button

```
17  def btn_generate_click(event=None):
18      # Get the control that was clicked
19      control = CreateScriptService("DialogEvent", event)
20      dlg = control.Parent
21      # Get the parameters from the dialog
22      start_cell = dlg.Controls("editStartCell").Value
23      num_cols = int(dlg.Controls("editNumColumns").Value)
24      num_rows = int(dlg.Controls("editNumRows").Value)
25      mean = float(dlg.Controls("editMean").Value)
26      std_dev = float(dlg.Controls("editStdDev").Value)
27      # Generate the numbers
28      numbers = [[rnd.gauss(mean, std_dev) for _ in range(num_cols)] for _ in range(num_rows)]
29      # Get the document and add the data
30      doc = CreateScriptService("Document", bas.ThisComponent)
31      num_range = doc.Offset(start_cell, width=num_cols, height=num_rows)
32      doc.setValue(num_range, numbers)
```

Thank you ...

LibreOffice
The Document Foundation

- The ScriptForge team
  - Jean-Pierre Ledure, Rafael Lima, Alain Romedenne
- Documentation
  - https://help.libreoffice.org/latest/en-US/text/sbasic/shared/03/lib_ScriptForge.html
- ScriptForge Sources
  - https://gitlab.com/LibreOfficiant/scriptforge
- Type Support (typings) for ScriptForge, by PauL Moss.
  - https://github.com/Amourspirit/python-types-scriptforge
- Telegram groups
  - ScriptForge
  - LibreOffice Macros & Scripting