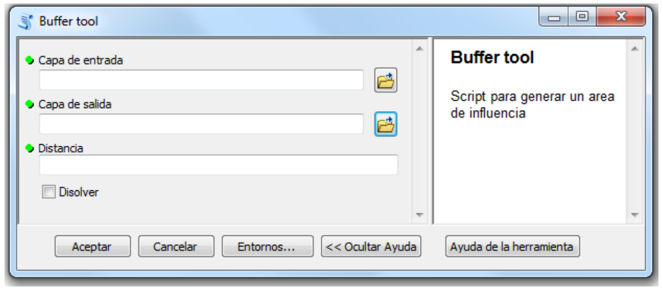
python
ArcGIS

Graphical user interfaces

Alternatives for making GUIs

# When we talk about making GUIs for a script within ArcGIS, we have several options:

- Use ArcToolBox (ATB) to make a GUI.
- Use the “Add-ins” tool.
- Use the library PyQt4 to make an standalone application.



Buffer tool

Script para generar un area de influencia

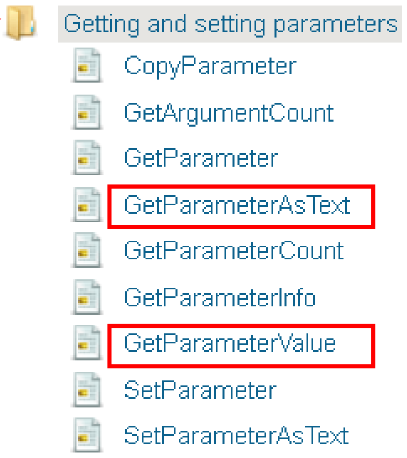
ATB as tool to create GUIs

# Scripts can be linked to a model (Model Builder) or to a tool (ATB).
# First of all, it is necessary to create a new tool and link a script (py file).
# If we include some parameters, these will indicate how the GUI will look like (each parameter will correspond with an specific widget). To get the values of these parameters from a script, we will use the next function:

arcpy.**GetParameterAsText**(index [starts from zero])

Arcpy

Arcpy functions


Getting and setting parameters
CopyParameter
GetArgumentCount
GetParameter
GetParameterAsText
GetParameterCount
GetParameterInfo
GetParameterValue
SetParameter
SetParameterAsText

ATB as tool to create GUIs

# Example 1. Layer loading

```
#import modules
import arcpy
import arcpy.mapping as map
try:
    #current project
    path_mxd = 'CURRENT'
    #get parameter value from tool window
    path_capa = arcpy.GetParameterAsText(0) #shp path
    mxd = map.MapDocument(path_mxd) #map document
    df = map.ListDataFrames(mxd)[0] #firts data frame
    #object layer building from shp path
    layer = map.Layer(path_capa)
    #Adding the layer
    map.AddLayer(df,layer)
    #Updating data view and Table Of Contents
    arcpy.RefreshActiveView()
    arcpy.RefreshTOC()
except:
    print 'An error has occurred'
```

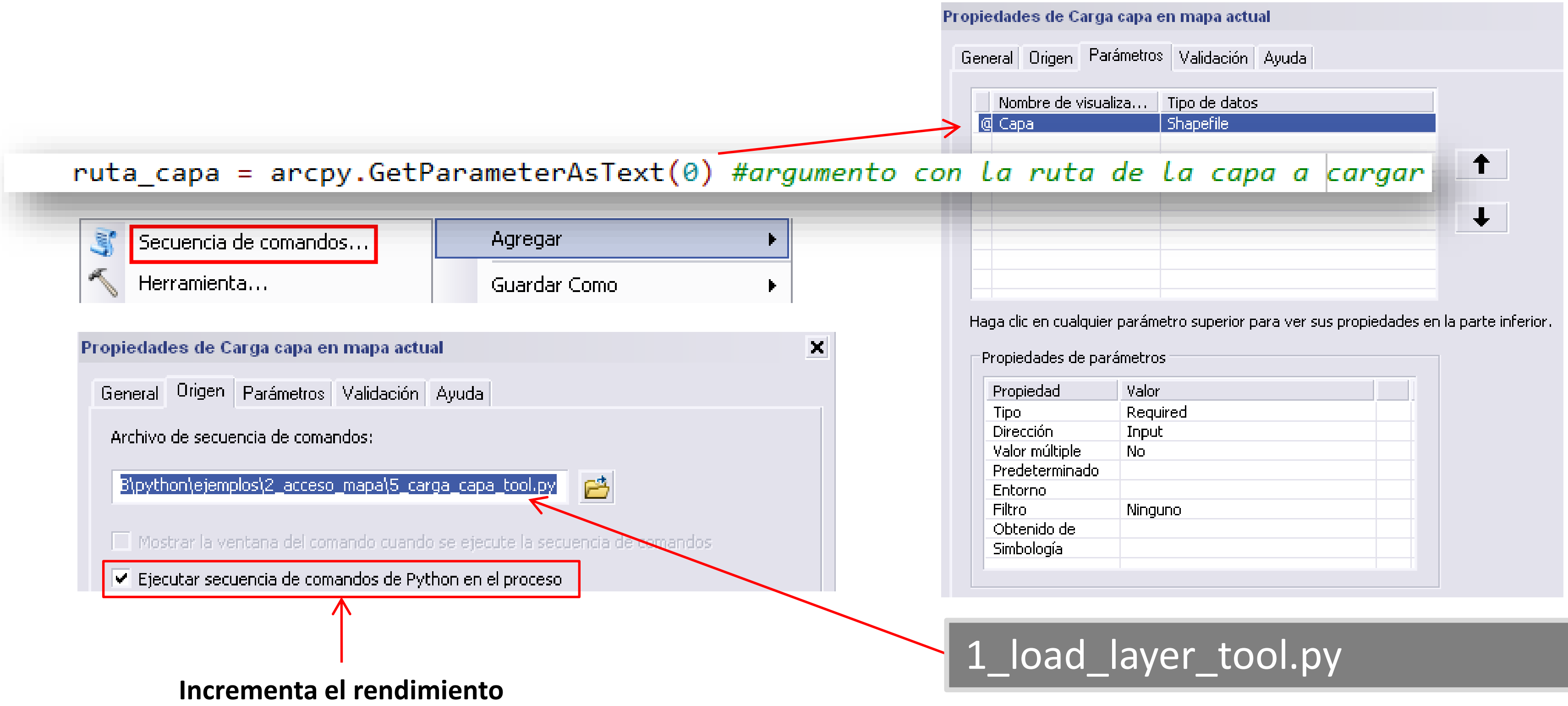
1_load_layer_tool.py

# If we want to add some messages to the processing window of our tool, we have to write the next statement:

Arcpy.AddMessage(message)

ATB as tool to create GUIs

# 1. From a specific tool box right click on "**Add -> Script**".
# 2. Select the tab "Source" and put the path of the linked python file
# 3. Select the tab "Parameters" and add new parameters with its properties. The order of these parameters is important and is directly related with the order in the script.



```
ruta_capa = arcpy.GetParameterAsText(0) #argumento con la ruta de la capa a cargar
```

Propiedades de Carga capa en mapa actual

Archivo de secuencia de comandos:

B\python\ejemplos\2_acceso_mapa\5_carga_capa_tool.py

Mostrar la ventana del comando cuando se ejecute la secuencia de comandos

✓ Ejecutar secuencia de comandos de Python en el proceso

Incrementa el rendimiento

Propiedades de Carga capa en mapa actual

Nombre de visualiza...	Tipo de datos
Capa	Shapefile

Haga clic en cualquier parámetro superior para ver sus propiedades en la parte inferior.

Propiedades de parámetros

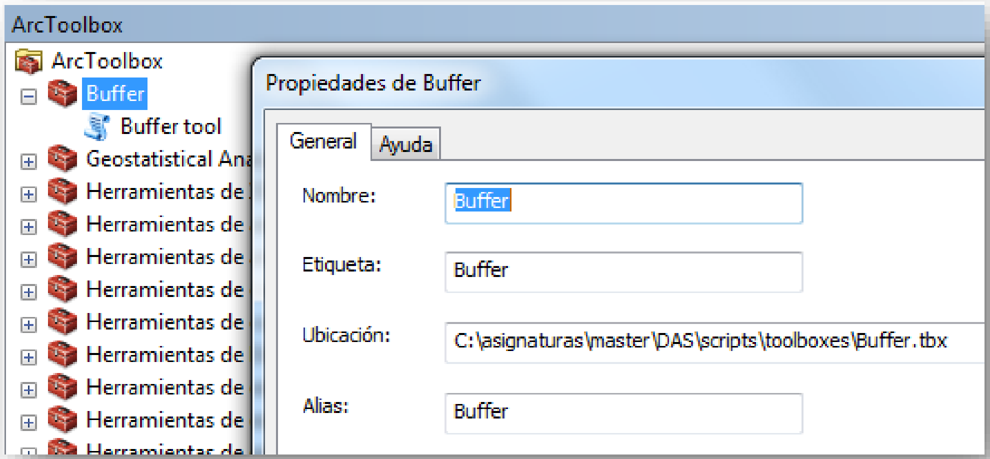
Propiedad	Valor
Tipo	Required
Dirección	Input
Valor múltiple	No
Predeterminado	
Entorno	
Filtro	Ninguno
Obtenido de	
Simbología	

1_load_layer_tool.py

ATB as tool to create GUIs

# Example 2. Buffer

# Firts of all we will make a new tool box ("New Toolbox") in a specific folder. Don't forget to fill the field "Alias". This field can be used to call any tool within this tool box. The reault will be as follow:



ArcToolbox
- ArcToolbox
  - Buffer
    - Buffer tool
  - Geostatistical Ana
  - Herramientas de
  - Herramientas de
  - Herramientas de
  - Herramientas de
  - Herramientas de
  - Herramientas de
  - Herramientas de
  - Herramientas de
  - Herramientas de
  - Herramientas de
  - Herramientas de

Propiedades de Buffer

General | Ayuda

Nombre:	Buffer
Etiqueta:	Buffer
Ubicación:	C:\asignaturas\master\DAS\scripts\toolboxes\Buffer.tbx
Alias:	Buffer

ATB as tool to create GUIs

# Example 2. Buffer

# write this script

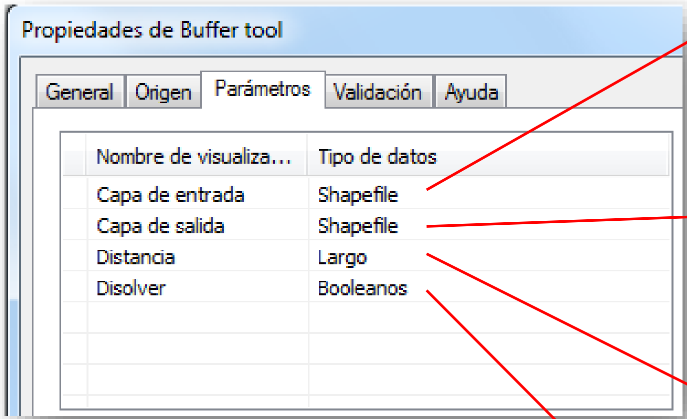
```
#import modules
import arcpy

try:
    #get parameters from tool window
    input_layer = arcpy.GetParameterAsText(0) #input layer
    output_layer = arcpy.GetParameterAsText(1) #output layer
    distance = arcpy.GetParameter(2) #distance
    dissolve = arcpy.GetParameter(2) #dissolve? (boolean)
    #message
    arcpy.AddMessage('Making buffer...')
    if dissolve:
        arcpy.Buffer_analysis(input_layer,output_layer,distance,'FULL','ROUND','ALL')
    else:
        arcpy.Buffer_analysis(input_layer,output_layer,distance)
    arcpy.AddMessage('Process finished.')
except:
    arcpy.AddMessage('An error has occurred.')
    arcpy.AddMessage(arcpy.GetMessages())
```

2_buffer_tool.py

ATB as tool to create GUIs

# Example 2. Buffer

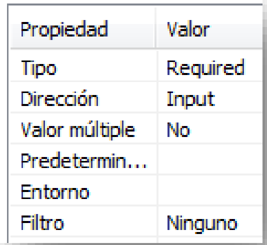
# Add the next parameters setting up its properties



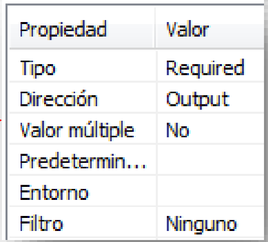
Propiedades de Buffer tool

General | Origen | Parámetros | Validación | Ayuda

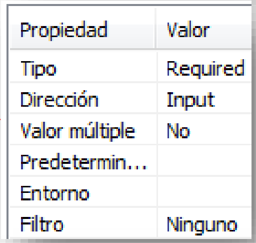
Nombre de visualiza...	Tipo de datos
Capa de entrada	Shapefile
Capa de salida	Shapefile
Distancia	Largo
Disolver	Booleanos



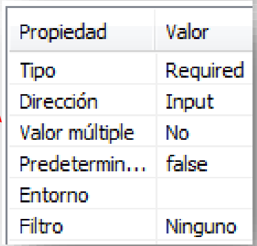
Propiedad	Valor
Tipo	Required
Dirección	Input
Valor múltiple	No
Predetermin...	
Entorno	
Filtro	Ninguno



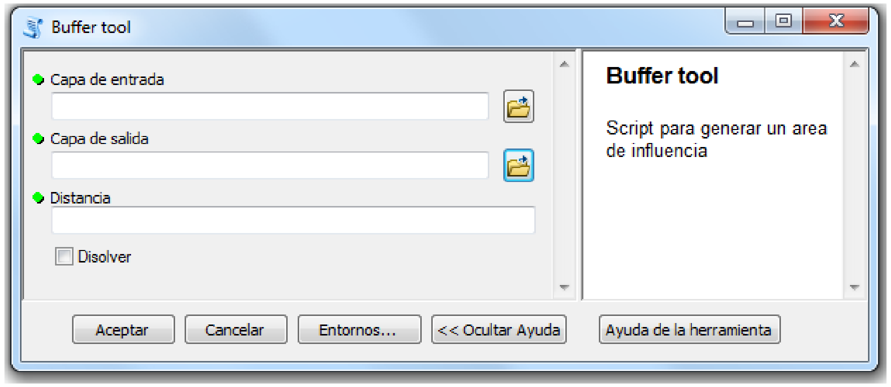
Propiedad	Valor
Tipo	Required
Dirección	Output
Valor múltiple	No
Predetermin...	
Entorno	
Filtro	Ninguno



Propiedad	Valor
Tipo	Required
Dirección	Input
Valor múltiple	No
Predetermin...	
Entorno	
Filtro	Ninguno



Propiedad	Valor
Tipo	Required
Dirección	Input
Valor múltiple	No
Predetermin...	false
Entorno	
Filtro	Ninguno



Buffer tool

Capa de entrada

Capa de salida

Distancia

Disolver

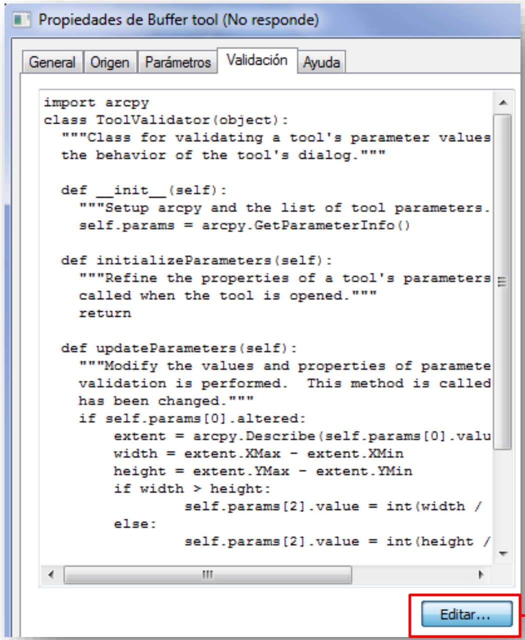
Buffer tool

Script para generar un area de influencia

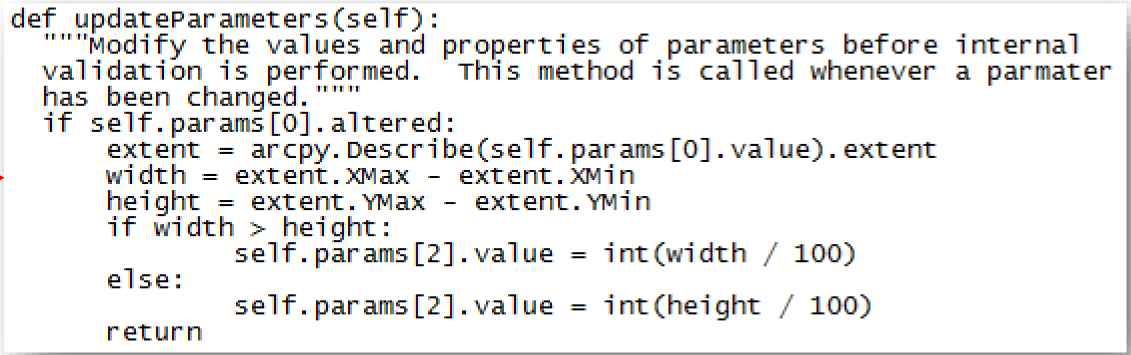
ATB as tool to create GUIs

# Example 2. Buffer

# Validation script. This kind of script allow us to perform some important tasks: to initialize values, to validate user inputs or to update fields according to user actions. The associated code of this script will allways run before click on the "Ok" button in the tool box. We can see an example here:



```
import arcpy
class ToolValidator(object):
  """Class for validating a tool's parameter values
  the behavior of the tool's dialog."""

  def __init__(self):
    """Setup arcpy and the list of tool parameters.
    self.params = arcpy.GetParameterInfo()

  def initializeParameters(self):
    """Refine the properties of a tool's parameters
    called when the tool is opened."""
    return

  def updateParameters(self):
    """Modify the values and properties of paramete
    validation is performed.  This method is called
    has been changed."""
    if self.params[0].altered:
        extent = arcpy.Describe(self.params[0].valu
        width = extent.XMax - extent.XMin
        height = extent.YMax - extent.YMin
        if width > height:
                self.params[2].value = int(width /
        else:
                self.params[2].value = int(height /
```



```
def updateParameters(self):
  """Modify the values and properties of parameters before internal
  validation is performed.  This method is called whenever a parmater
  has been changed."""
  if self.params[0].altered:
      extent = arcpy.Describe(self.params[0].value).extent
      width = extent.XMax - extent.XMin
      height = extent.YMax - extent.YMin
      if width > height:
              self.params[2].value = int(width / 100)
      else:
              self.params[2].value = int(height / 100)
      return
```

# 1. Edit the text file.
# 2. Save and close.
# 3. Click on “Apply” button.

Add-ins as tool to create GUIs

# Add-ins provide a way to add widges to the ArcGIS GUI (buttons, tools, menus, combobox, etc.)..

# In the past, you only could make add-ins using .NET or java, but now, from ArcGIS 10.1, we can also use python.

# Add-ins can be shared easily among users and don't need any kind of installation.

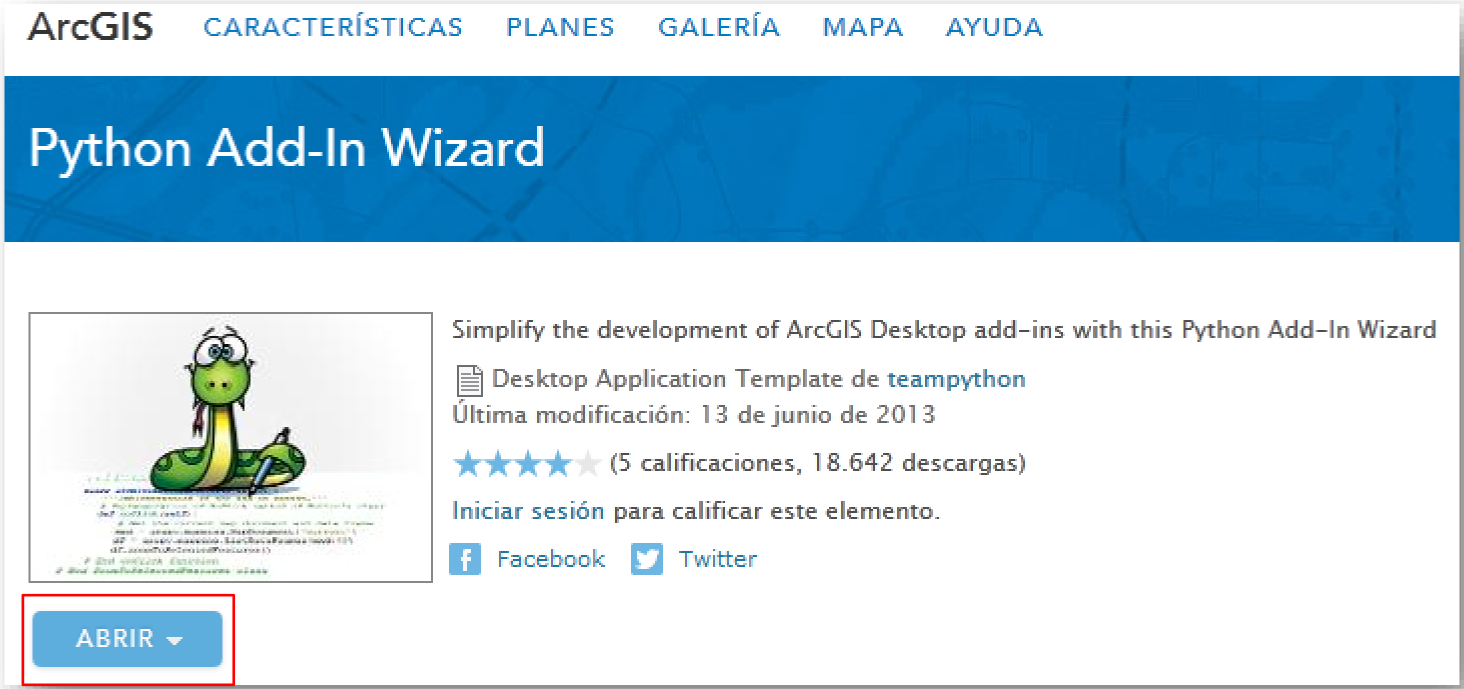
# In order to speed-up the process ArcGIS provides a wizard call "Python Add-in". Follow these links to get the wizard and some help:

http://www.arcgis.com/home/item.html?id=5f3aefe77f6b4f61ad3e4c62f30bff3b

http://resources.arcgis.com/en/help/main/10.1/index.html#/na/014p00000025000000

Add-ins as tool to create GUIs

# First step: download the wizard. We will get a zip file that we can uncompress in any folder.



ArcGIS CARACTERÍSTICAS PLANES GALERÍA MAPA AYUDA

Python Add-In Wizard

Simplify the development of ArcGIS Desktop add-ins with this Python Add-In Wizard

Desktop Application Template de teampython

Última modificación: 13 de junio de 2013

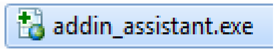
★★★★☆ (5 calificaciones, 18.642 descargas)

Iniciar sesión para calificar este elemento.

Facebook Twitter

ABRIR

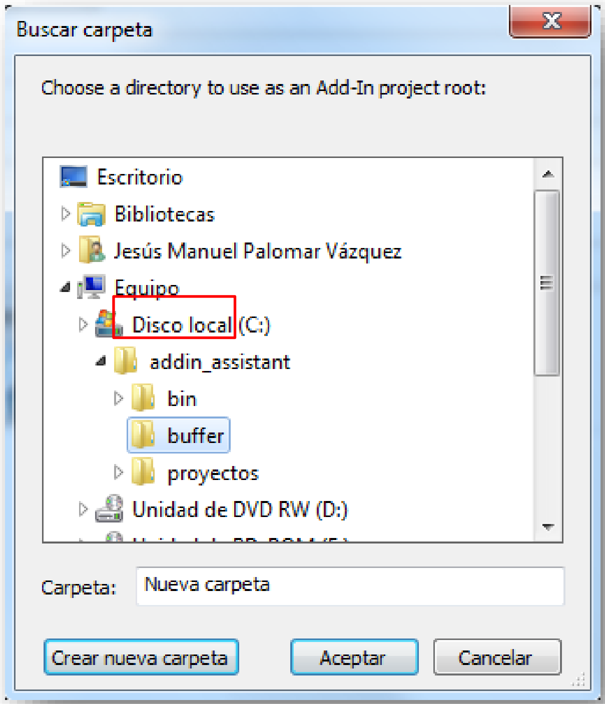
# Open the "\bin" folder and run the file addin_assistant.exe.



Add-ins as tool to create GUIs

# En el asistente lo primero será elegir la carpeta de proyecto.

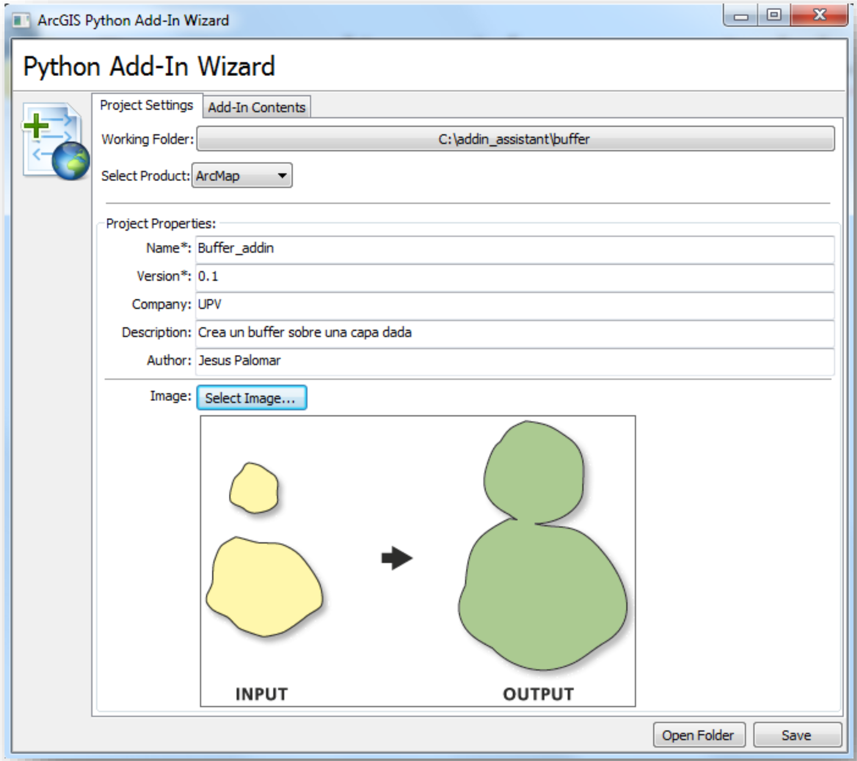
# First of all, we must select the project folder.
After that, we will write some basic data: add-in name (mandatory), version, company, description, author and thumbnail image.


Buscar carpeta

Choose a directory to use as an Add-In project root:

- Escritorio
- Bibliotecas
- Jesús Manuel Palomar Vázquez
- Equipo
  - Disco local (C:)
    - addin_assistant
      - bin
      - buffer
      - proyectos
  - Unidad de DVD RW (D:)

Carpeta: Nueva carpeta


ArcGIS Python Add-In Wizard

Python Add-In Wizard

Working Folder: C:\addin_assistant\buffer

Select Product: ArcMap

Project Properties:

Name*: Buffer_addin

Version*: 0.1

Company: UPV

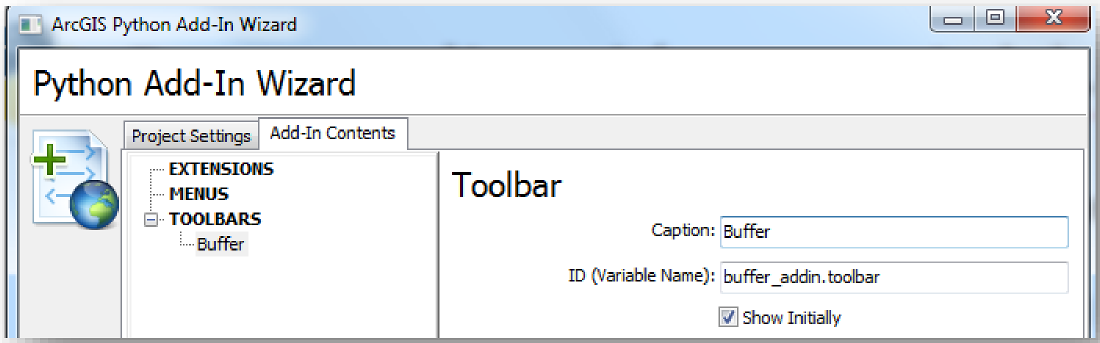
Description: Crea un buffer sobre una capa dada

Author: Jesus Palomar

Image: Select Image...

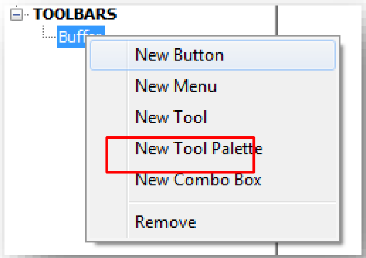
Add-ins as tool to create GUIs

# In the tab "Add-In Contents", we will add a new toolbar (right click on the "TOOLBARS" section and select "New Toolbar"). This toolbar will be the container for any button that we will need. Fill out some properties (caption).



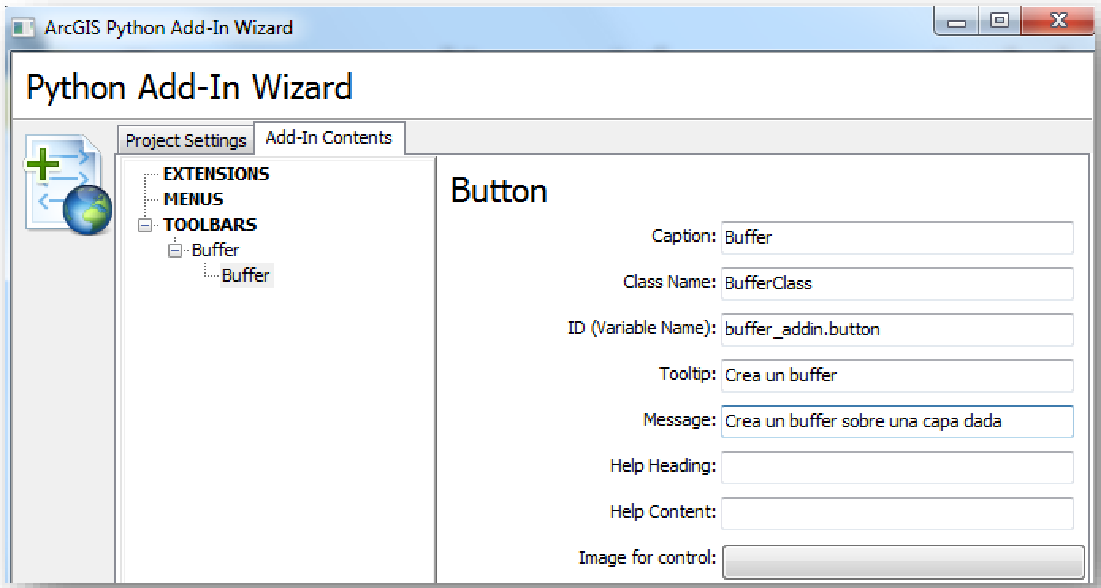
Caption: Buffer
ID (Variable Name): buffer_addin.toolbar
☑ Show Initially

# In the same way, right click on the new toolbar and select "New button".



Add-ins as tool to create GUIs

# Fill out some basic data, like the text of the button ("caption" property), the class name (without white spaces) that will appear in the script written by the wizard, the text for the "Tooltip" or "Message" properties, or an image (16x16 pixels).



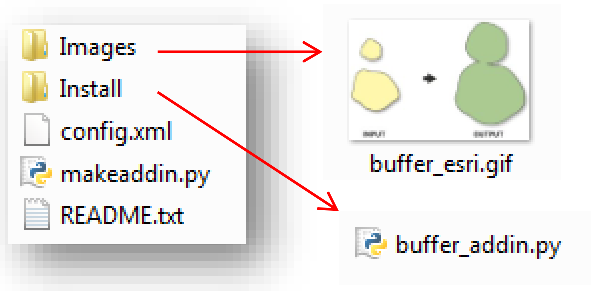
Button

Caption:	Buffer
Class Name:	BufferClass
ID (Variable Name):	buffer_addin.button
Tooltip:	Crea un buffer
Message:	Crea un buffer sobre una capa dada
Help Heading:	
Help Content:	
Image for control:	

# Click on the "Save" button. This will create a folder structure within the project folder.

Add-ins as tool to create GUIs

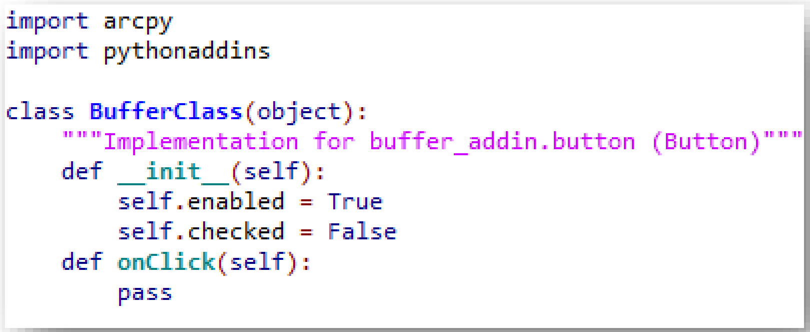
# This structure contains all files we need to make the Add-In.



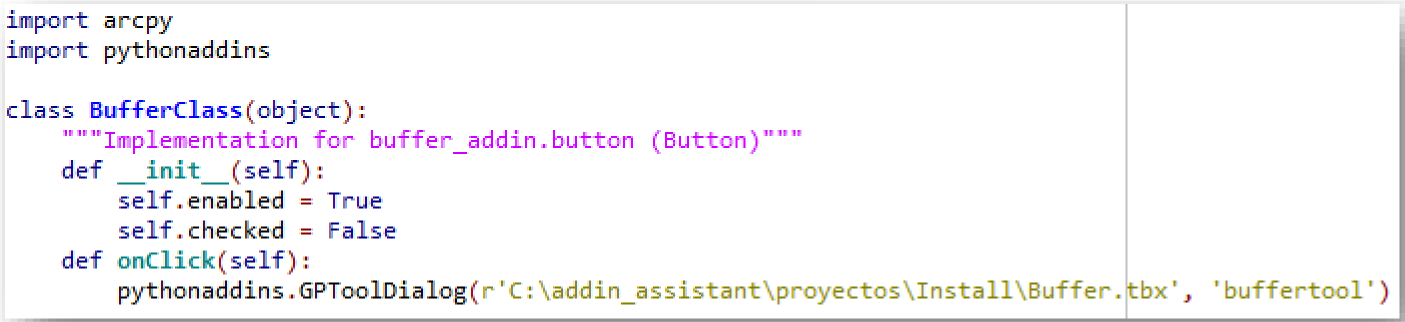
# Next step: we have to customize the script "Buffer_addin.py", located in the folder "Install". In this case, we add some code in order to open the tool box that we created previously (slides 6 to 8).

# We will delete the "pass" statement in the "Click" function. Instead this statement, we will use the method GPtoolDialog from pythonaddins module. This method allows to open whatever arctoolbox tool.

Add-ins as tool to create GUIs



```
import arcpy
import pythonaddins

class BufferClass(object):
    """Implementation for buffer_addin.button (Button)"""
    def __init__(self):
        self.enabled = True
        self.checked = False
    def onClick(self):
        pass
```

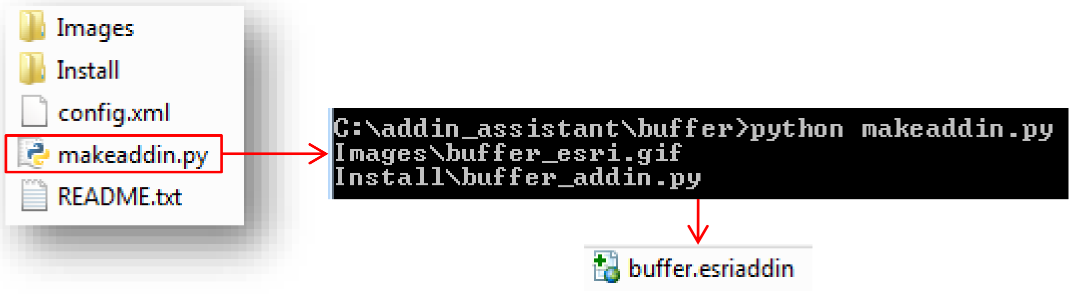


```
import arcpy
import pythonaddins

class BufferClass(object):
    """Implementation for buffer_addin.button (Button)"""
    def __init__(self):
        self.enabled = True
        self.checked = False
    def onClick(self):
        pythonaddins.GPToolDialog(r'C:\addin_assistant\proyectos\Install\Buffer.tbx', 'buffertool')
```

# Save the script. Now we are going to install the Add-In. We have to follow some easy steps:

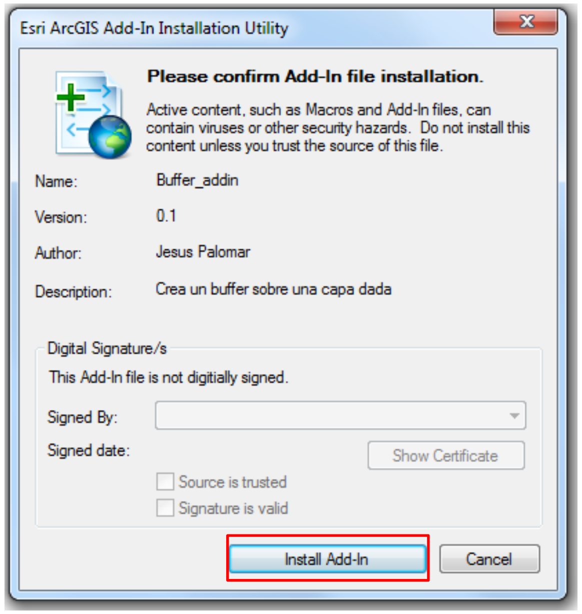
Add-ins as tool to create GUIs

# 1. Open and run the script "makeaddin.py" using pyscripter. That will create a compress file call "buffer.esriaddin"



```
C:\addin_assistant\buffer>python makeaddin.py
Images\buffer_esri.gif
Install\buffer_addin.py
```

# 2. Run this file. The installation wizard will be displayed. Finally, Click on the button "Install Add-In".



Esri ArcGIS Add-In Installation Utility

Please confirm Add-In file installation.

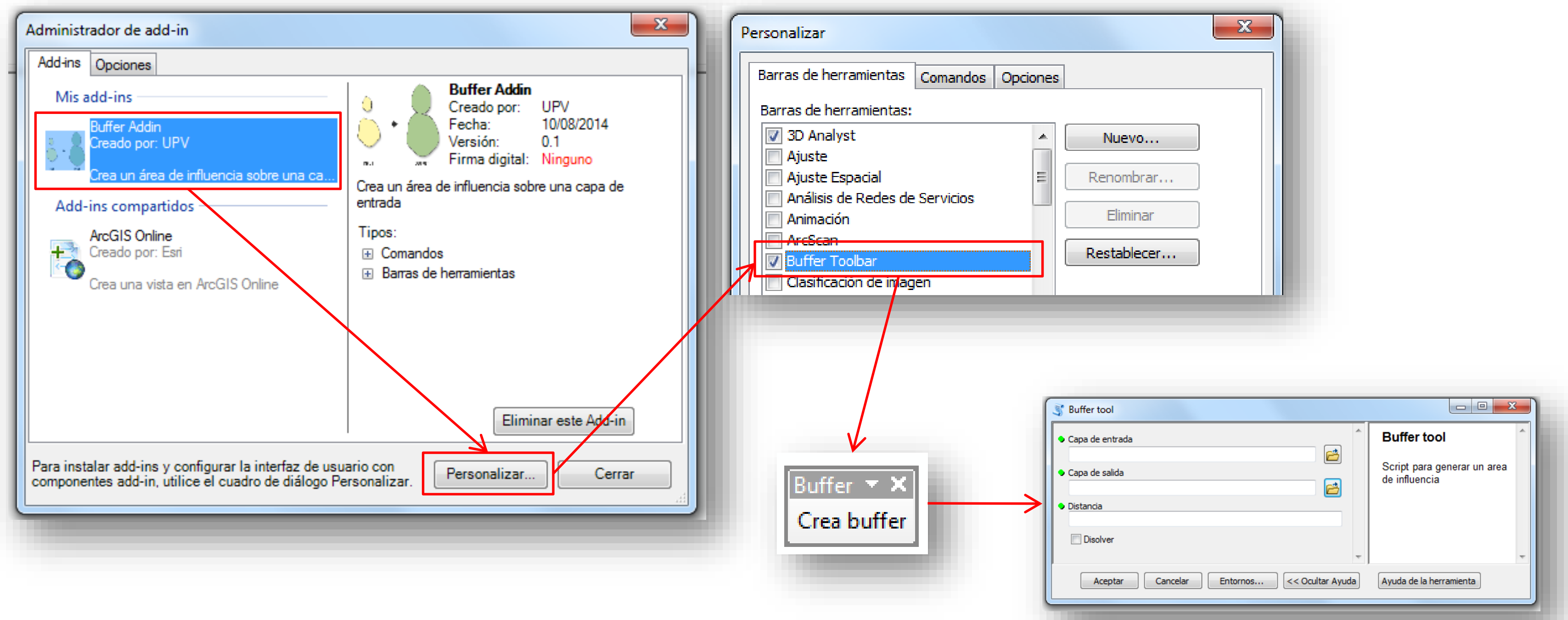
Active content, such as Macros and Add-In files, can contain viruses or other security hazards. Do not install this content unless you trust the source of this file.

Name: Buffer_addin
Version: 0.1
Author: Jesus Palomar
Description: Crea un buffer sobre una capa dada

Digital Signature/s
This Add-In file is not digitially signed.
Signed By:
Signed date:

Add-ins as tool to create GUIs

# After install the Add-In, we will open ArcMap and click on "**Customize -> Add-In Manager**". Select your Add_in and click on "**Customize**". Then we can activate the tool bar to make it visible. Finally, push on the tool button to see the result.



Buffer Addin
Creado por: UPV
Fecha: 10/08/2014
Versión: 0.1
Firma digital: Ninguno

Crea un área de influencia sobre una capa de entrada

Tipos:
⊞ Comandos
⊞ Barras de herramientas

Para instalar add-ins y configurar la interfaz de usuario con componentes add-in, utilice el cuadro de diálogo Personalizar.

Barras de herramientas: ✓ 3D Analyst, ✓ Buffer Toolbar

Buffer tool

Script para generar un area de influencia

Building GUIs with PyQt4

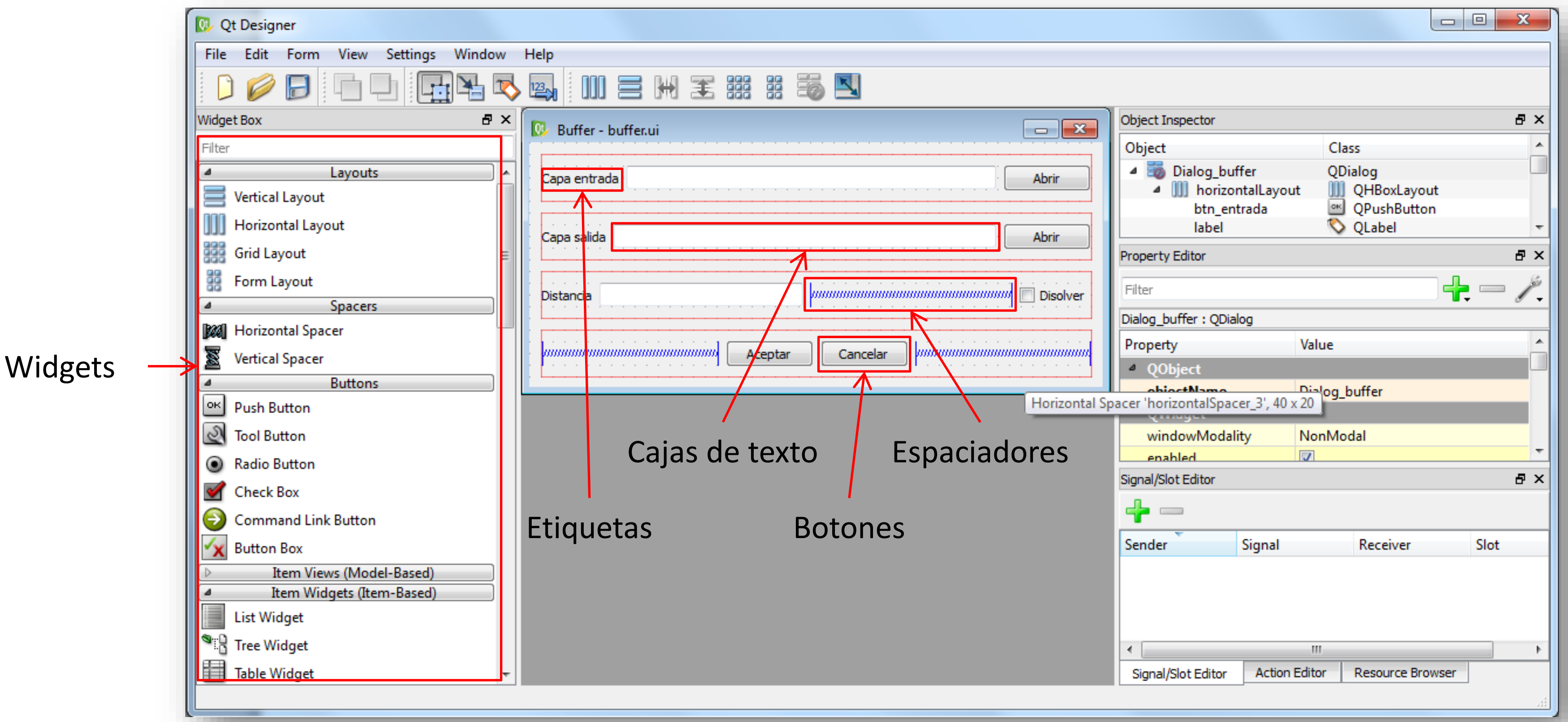
# PyQt4 is a library that make easier to develop GUIs using python.

# We can build both standalone applications or customized GUIs associated to Arctoolbox tools or Add-Ins.

# Once the library has been installed, we can build a new GUI writing code or, easier, using the application PyQT Designer.

# PyQT Designer allows the development of GUI inside a visual IDE. This way, we can add forms and widgets to make easier the communication between the program and the user.

Building GUIs with PyQt4



Object Inspector:

Object	Class
Dialog_buffer	QDialog
horizontalLayout	QHBoxLayout
btn_entrada	QPushButton
label	QLabel

Property Editor: Dialog_buffer : QDialog

Property	Value
QObject	
objectName	Dialog_buffer
windowModality	NonModal
enabled	✓

Horizontal Spacer 'horizontalSpacer_3', 40 x 20

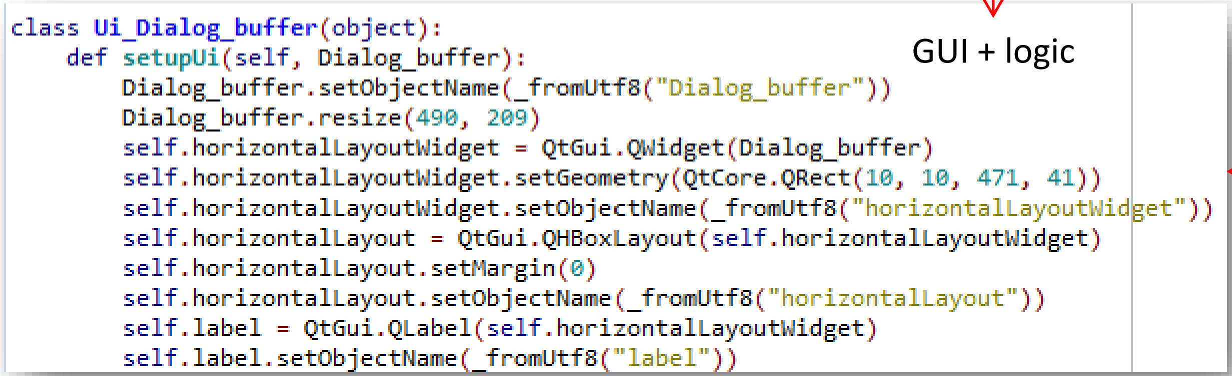
Sender	Signal	Receiver	Slot

# PyQT Designer build a .ui file (user interface).

Building GUIs with PyQt4

# There are two ways to work with the file .ui: translate the .ui file to a .py file or load directly the .ui file inside the .py file. That way we maintain both files (interface and logic) separated.

$$\begin{tikzcd}
 & & & \text{form.py GUI} \\
\text{.ui} \arrow[r] & \text{pyuic4.py} \arrow[r] & \text{.py} \arrow[ur, dashed] \arrow[dr, dashed] \arrow[d] & \\
 & & \text{GUI + logic} & \text{main.py Logic}
\end{tikzcd}$$



```
class Ui_Dialog_buffer(object):
    def setupUi(self, Dialog_buffer):
        Dialog_buffer.setObjectName(_fromUtf8("Dialog_buffer"))
        Dialog_buffer.resize(490, 209)
        self.horizontalLayoutWidget = QtGui.QWidget(Dialog_buffer)
        self.horizontalLayoutWidget.setGeometry(QtCore.QRect(10, 10, 471, 41))
        self.horizontalLayoutWidget.setObjectName(_fromUtf8("horizontalLayoutWidget"))
        self.horizontalLayout = QtGui.QHBoxLayout(self.horizontalLayoutWidget)
        self.horizontalLayout.setMargin(0)
        self.horizontalLayout.setObjectName(_fromUtf8("horizontalLayout"))
        self.label = QtGui.QLabel(self.horizontalLayoutWidget)
        self.label.setObjectName(_fromUtf8("label"))
```

GUI code

```
self.btn_entrada = QtGui.QPushButton(self.horizontalLayoutWidget)
self.btn_entrada.setObjectName(_fromUtf8("btn_entrada"))
self.horizontalLayout.addWidget(self.btn_entrada)
```

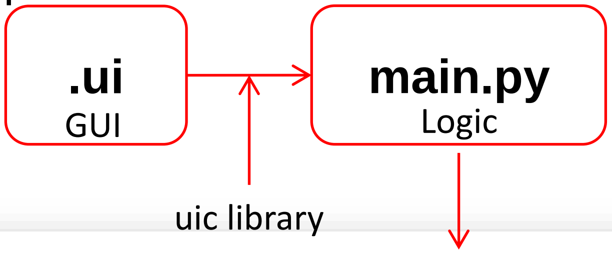
Building a button

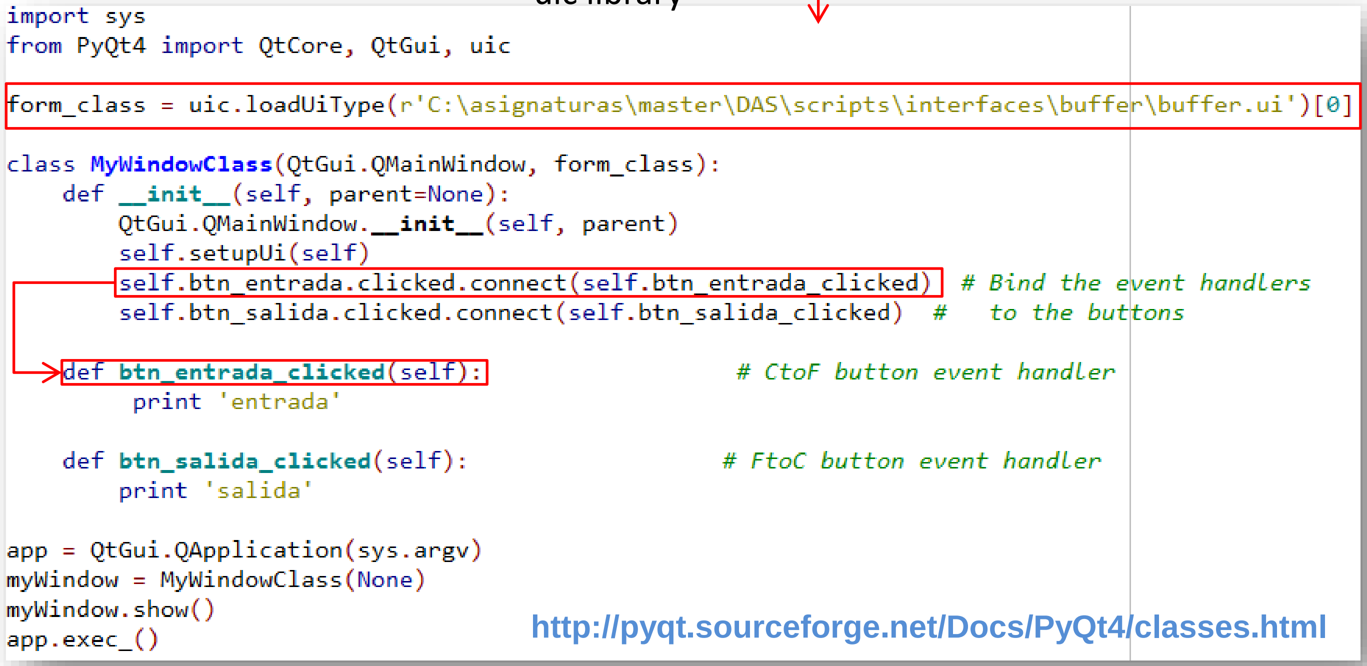
```
self.connect(self.ui.pb_sumar, QtCore.SIGNAL("clicked()"),self.sumar)

 def sumar(self):
        a = float(self.ui.lineEdit_1.text())
        b = float(self.ui.lineEdit_2.text())
        self.ui.lineEdit_resultado.setText(str(a+b))
```

Linking a "click" event to a function

Building GUIs with PyQt4

# There are two ways to work with the file .ui: translate the .ui file to a .py file or load directly the .ui file inside the .py file. That way we maintain both files (interface and logic) separated.





```
import sys
from PyQt4 import QtCore, QtGui, uic

form_class = uic.loadUiType(r'C:\asignaturas\master\DAS\scripts\interfaces\buffer\buffer.ui')[0]

class MyWindowClass(QtGui.QMainWindow, form_class):
    def __init__(self, parent=None):
        QtGui.QMainWindow.__init__(self, parent)
        self.setupUi(self)
        self.btn_entrada.clicked.connect(self.btn_entrada_clicked)  # Bind the event handlers
        self.btn_salida.clicked.connect(self.btn_salida_clicked)  #    to the buttons

    def btn_entrada_clicked(self):                      # CtoF button event handler
         print 'entrada'

    def btn_salida_clicked(self):                      # FtoC button event handler
        print 'salida'

app = QtGui.QApplication(sys.argv)
myWindow = MyWindowClass(None)
myWindow.show()
app.exec_()
```

http://pyqt.sourceforge.net/Docs/PyQt4/classes.html