Politechnika Łódzka
Instytut Elektroniki

Image Processing and Computer Graphics

Laboratory #1:

Introduction to Image Processing and wxPython

M. Kociński, J. Blumenfeld

Medical Electronics Division
Institute of Electronics

In this exercise you will learn how to build simple application with Graphical User Interface (GUI). For this purpose wxPython libraries will be used. Presented examples base on the on-line tutorial prepared by Jan Bodnar.

In the cell below we grouped all imports used in this lab.

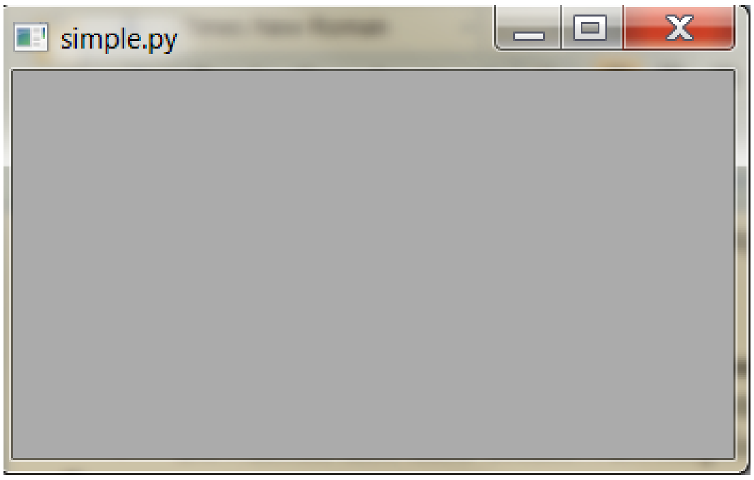
In [22]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy import misc
from IPython.display import Image
```

Empty window

In the first task, based on example create a blank window, as at the figure below:

In [2]:
```
Image(filename="Figures/1_empty_window.png")
```

Out[2]:



The source code is as followes. There is an IPython magic command %reset in the first line. It should be added only wehen you are using IPython notebook as you editor. If case of regular Python script (with .py extention) this line is optional and should be omitted. It is (strongly) reccomended to use regular script while practicing with wxPython library.

In [3]:
```
%reset
# Empty window

#!/usr/bin/python
# simple.py

import wx

app = wx.App()
frame = wx.Frame(None, -1, 'simple.py')
frame.Show()
app.MainLoop()
```

To get help with Keyboard Shortcuts of IPython notebook press "Ctrl+m h". Practice how to add, remove, delete and execute a new cell. You can turn on/off line numbers inside cell with the use of "Ctrl+m l" keys sequence.

Empty window - modifications (background colour, text)

To the existing window add a control to display text (wxStaticText). Place it in the center of the window. Change background color; use different manners of setting couour(use wx.Frame.SetBackgroundColour() function and wx.Colour class and String). Explain each line of the source code.

In [4]:
```
%reset
import wx

app = wx.App()

frame = wx.Frame(None, -1, 'Name :D')
#frame.SetBackgroundColour(wx.Colour(255,255,255))
#frame.SetBackgroundColour((255,100,55))
frame.SetBackgroundColour("Red")
frame.Show()

text = wx.StaticText(frame, -1, "Some text ;-)",(100,50))
text.Center()

app.MainLoop()
```

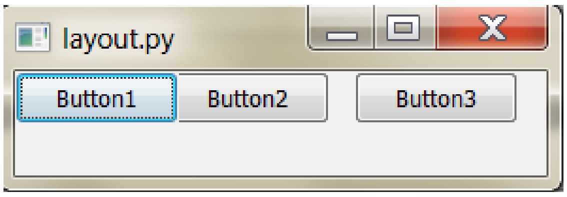
Application with 3 buttons

Based on tutorial write an application that contains three buttons. Use booth methods for layout your widgets:

- manual

In [6]:
```
from IPython.display import Image
Image(filename="Figures/2_buttons_manual.png")
```

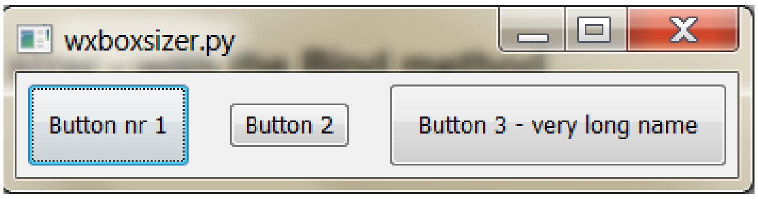
Out[6]:



- with the use of layout managers

In [7]:
```
from IPython.display import Image
Image(filename="Figures/2_buttons_sizers.png")
```

Out[7]:



Remember! To get help use "?" mark

```
wx.Button?
```

Three buttons application without sizers

Use help and explain meaning of each line. What is happend when the window size is changed?

In [8]:
```
%reset
#!/usr/bin/python

# layout.py

import wx

class MyFrame(wx.Frame):
    def __init__(self, parent, id, title):
        wx.Frame.__init__(self, parent, id, title, wx.DefaultPosition)

        panel = wx.Panel(self, -1)
        wx.Button(panel, 1, "Button1", (0,0))
        wx.Button(panel, 2, "Button2", (80,0))
        wx.Button(panel, 3, "Button3", (180,0))
        self.SetSize((300,100))


class MyApp(wx.App):
    def OnInit(self):
        frame = MyFrame(None, -1, 'layout.py')
        frame.Show(True)
        frame.Centre()
        return True

app = MyApp(0)
app.MainLoop()
```

How to connect some action with the buttons?

Based on tutorial explain how to bind any action with your buttons.

Write this piece of code in the Python regular script.

In [9]:
```
%reset
#!/usr/bin/python

# layout.py

import wx

class MyFrame(wx.Frame):
    def __init__(self, parent, id, title):
        wx.Frame.__init__(self, parent, id, title, wx.DefaultPosition)

        panel = wx.Panel(self, -1)
        wx.Button(panel, 1, "Button1", (0,0))
        wx.Button(panel, 2, label="Button2", pos=(80,0))
        wx.Button(panel, id=3, label="Button3", pos=(180,0))
        self.SetSize((300,100))

        self.Bind(wx.EVT_BUTTON, self.OnPrint, id=1)
        self.Bind(wx.EVT_BUTTON, self.OnPrint, id=2)
        self.Bind(wx.EVT_BUTTON, self.OnPrint, id=3)


    def OnPrint(self,event):
            print  'button nr.', event.GetId()

class MyApp(wx.App):
    def OnInit(self):
        frame = MyFrame(None, -1, 'layout.py')
        frame.Show(True)
        frame.Centre()
        return True

app = MyApp(0)
```

Three buttons application with a use of sizer

For volunteers: plaease find a very nice tutorial on sizers.

Use help and explain meaning of each line and every parameter. What is happend when the window size is changed?

In [10]:
```
% reset

#!/usr/bin/python

# wxboxsizer.py

import wx

class MyFrame(wx.Frame):
    def __init__(self, parent, id, title):
        wx.Frame.__init__(self, parent, id, title, (-1, -1), wx.Size(250, 50))
        panel = wx.Panel(self, -1)
        box = wx.BoxSizer(wx.HORIZONTAL)

        self.button1 = wx.Button(panel, id=wx.ID_ANY, label='Button nr 1')
        self.button2 = wx.Button(panel, id=wx.ID_ANY, label='Button 2')
        self.button3 = wx.Button(panel, id=wx.ID_ANY, label='Button 3 - very long r

        box.Add(self.button1, 0, wx.EXPAND | wx.ALL, border=5 )
        box.Add(self.button2, 1, wx.EXPAND | wx.ALL, border=15 )
        box.Add(self.button3, 2, wx.EXPAND | wx.ALL, border=5 )


        panel.SetSizer(box)
        self.Centre()
        self.Fit()
        self.Layout()
        self.SetSize((400,100))

class MyApp(wx.App):
     def OnInit(self):
         frame = MyFrame(None, -1, 'wxboxsizer.py')
         frame.Show(True)
         return True

app = MyApp(0)
app.MainLoop()
```

Different way to bind and identify pressed buttons

Write this piece of code in the Python regular script. Explain how it works.

In [11]:
```
% reset

#!/usr/bin/python

# wxboxsizer.py

import wx

class MyFrame(wx.Frame):
    def __init__(self, parent, id, title):
        wx.Frame.__init__(self, parent, id, title, (-1, -1), wx.Size(250, 50))
        panel = wx.Panel(self, -1)
        box = wx.BoxSizer(wx.HORIZONTAL)

        self.button1 = wx.Button(panel, id=wx.ID_ANY, label='Button nr 1')
        self.button2 = wx.Button(panel, id=wx.ID_ANY, label='Button 2')
        self.button3 = wx.Button(panel, id=wx.ID_ANY, label='Button 3 - very long r

        self.Bind(wx.EVT_BUTTON, self.OnPrint, id=self.button1.GetId())
        self.Bind(wx.EVT_BUTTON, self.OnPrint, id=self.button2.GetId())
        self.Bind(wx.EVT_BUTTON, self.OnPrint, id=self.button3.GetId())

        box.Add(self.button1, 0, wx.EXPAND | wx.ALL, border=5 )
        box.Add(self.button2, 1, wx.EXPAND | wx.ALL, border=15 )
        box.Add(self.button3, 2, wx.EXPAND | wx.ALL, border=5 )


        panel.SetSizer(box)
        self.Centre()
        #self.Fit()
        #self.Layout()
        self.SetSize((400,100))

    def OnPrint(self, event):
        if event.GetId() == self.button1.GetId():
            print 'pressed button 1'
        elif event.GetId() == self.button2.GetId():
            print 'pressed button 2'
        elif event.GetId() == self.button3.GetId():
            print 'pressed button 3'
        else:
            print 'pressed some other button'

class MyApp(wx.App):
     def OnInit(self):
         frame = MyFrame(None, -1, 'wxboxsizer.py')
         frame.Show(True)
         return True

app = MyApp(0)
```

How to display a image with the use of wxPython?

In this section it is showed how to glue together numpy, matplotlib and wxPython librares in order to dispaly a image (matrix).

Let's load matric into memory and print some of its properties.

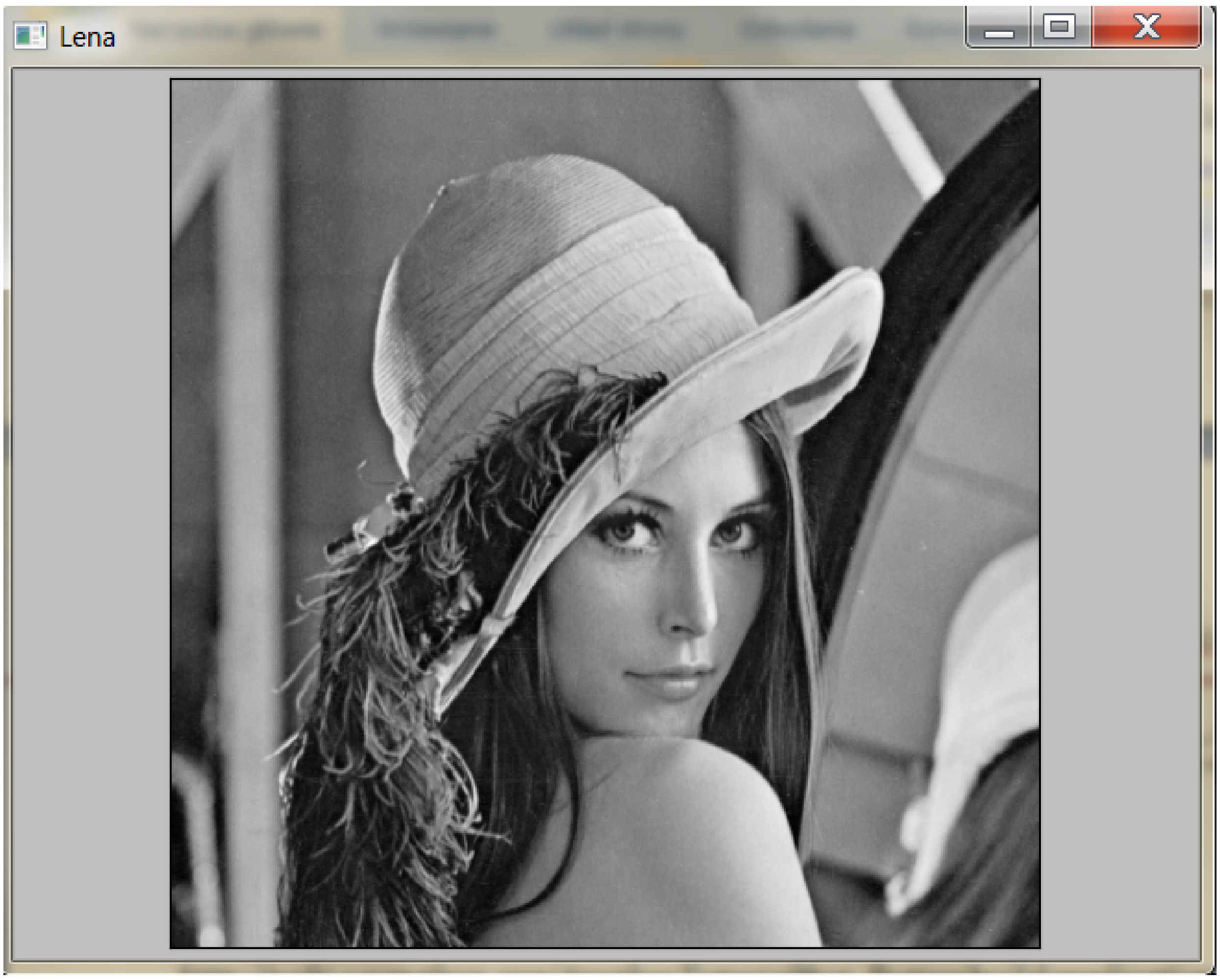
In [20]:
```
from scipy import misc
image = misc.lena()
```

In [14]:
```
print image.min(), image.max(), image.mean()
```

```
25 245 124.046783447
```

In [15]:
```
from IPython.display import Image
Image(filename="Figures/3_display_lena.png")
```

Out[15]:

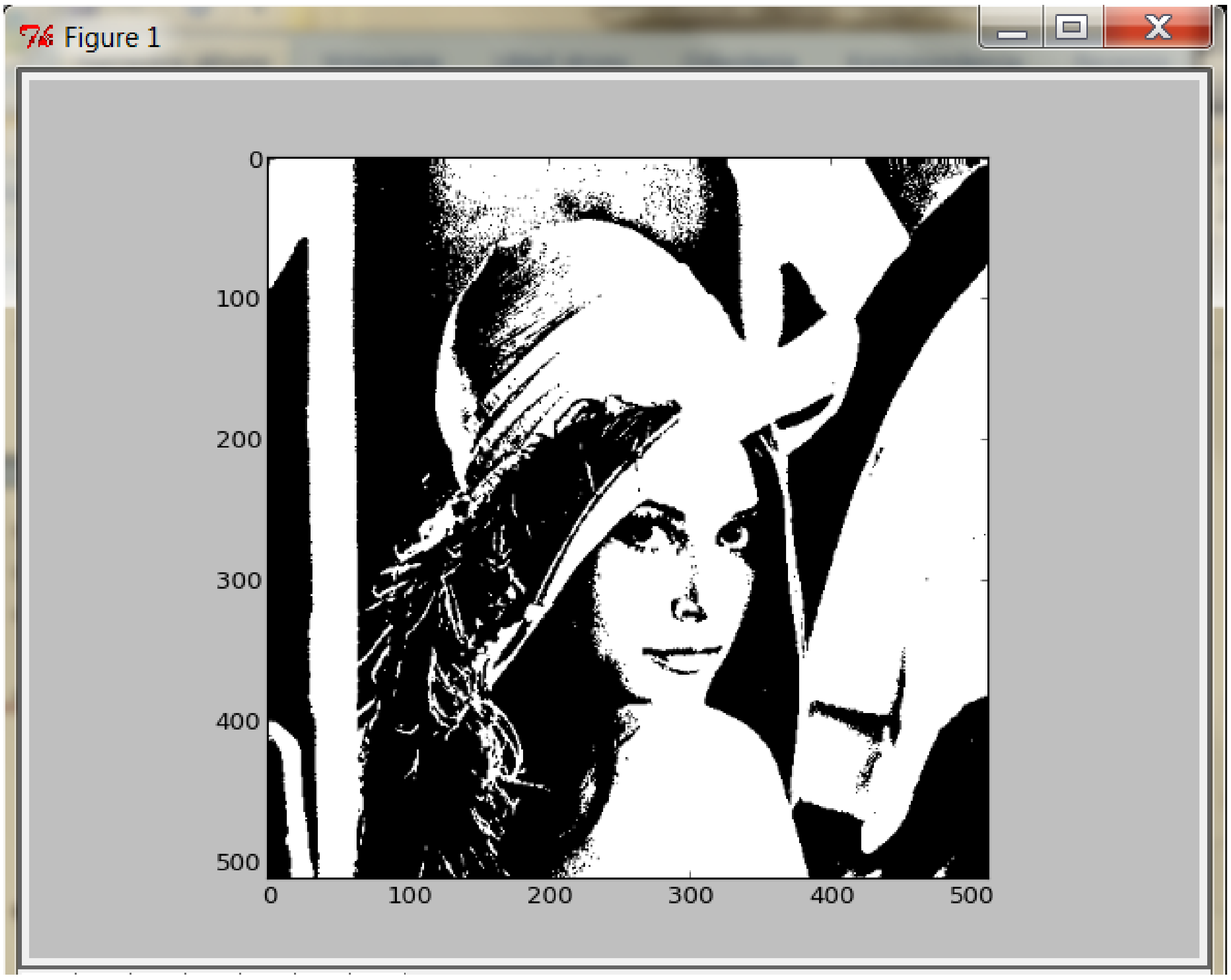


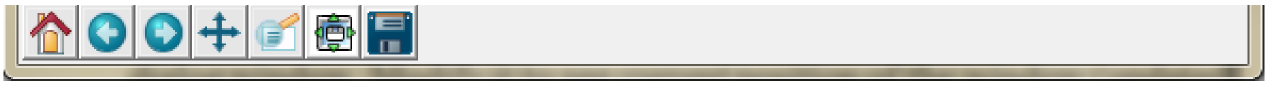
In [16]:
```
%reset

import wx
import numpy as np

from scipy import misc
from matplotlib.figure import Figure
from matplotlib.backends.backend_wxagg import FigureCanvasWxAgg as FigCanvas


class imageShow(wx.Frame):
    def __init__(self,parent):
        self.img = misc.lena()
        wx.Frame.__init__(self, parent, title="Lena", size=(wx.GetClientDisplayRect
        self.CreatePanel()
        self.DrawFigure()

    def CreatePanel(self):
        self.fig = Figure()
        self.fig.subplots_adjust(left=0.01, right=0.99, top=0.99, bottom=0.01)
        self.canvas = FigCanvas(self, -1, self.fig)
        self.axes = self.fig.add_subplot(111)
        self.axes.get_xaxis().set_visible(False)
        self.axes.get_yaxis().set_visible(False)

        self.hbox = wx.BoxSizer(wx.VERTICAL) #main siezer
        self.hbox.Add(self.canvas, 1, wx.EXPAND | wx.ALL,1)

        self.SetSizer(self.hbox)
        self.SetAutoLayout(True)
        self.Fit()


    def DrawFigure(self,  minn = 0, maxx = 255):
        self.axes.clear()
        self.imshow = self.axes.imshow(self.img.clip(minn,maxx), cmap="gray")
        self.canvas.draw()


app = wx.App(False)
img = imageShow(None)
img.Show()
```

Write a function to threshold a image

In [17]:
```
from IPython.display import Image
Image(filename="Figures/4_threshold_lena.png")
```

Out[17]:





With the use of for loops

```
In [18]: def thresh1(im, th=125):
             img = im.copy()
             size = img.shape
             for y in range(size[0]):
                 for x in range(size[1]):
                     if img[y,x] >= th:
                         img[y,x]= 255
                     else:
                         img[y,x] = 0
             return img
```

```
In [23]: img_th1 = thresh1(image, th=125)
         plt.imshow(img_th1, plt.cm.gray)

Out[23]: <matplotlib.image.AxesImage at 0x8fa27f0>
```

Matrix convention (1)

```
In [24]: def thresh2(im, th=125):
             img = im.copy()
             img[img>=th] = 255
             img[img<th] = 0
             return img
```

```
In [25]: img_th2 = thresh2(image, th=150)
         plt.imshow(img_th2, plt.cm.gray)

Out[25]: <matplotlib.image.AxesImage at 0x91dd130>
```

Matrix convention (2)

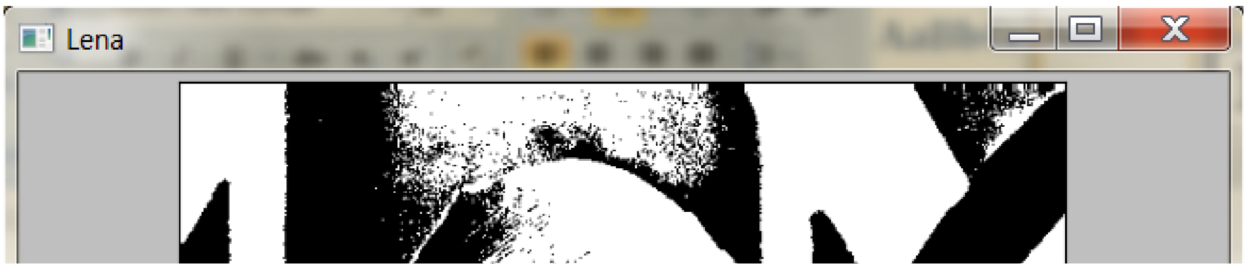
```
In [26]: th = 200
         img_th3 = np.where(image>th, 255, 0)
```

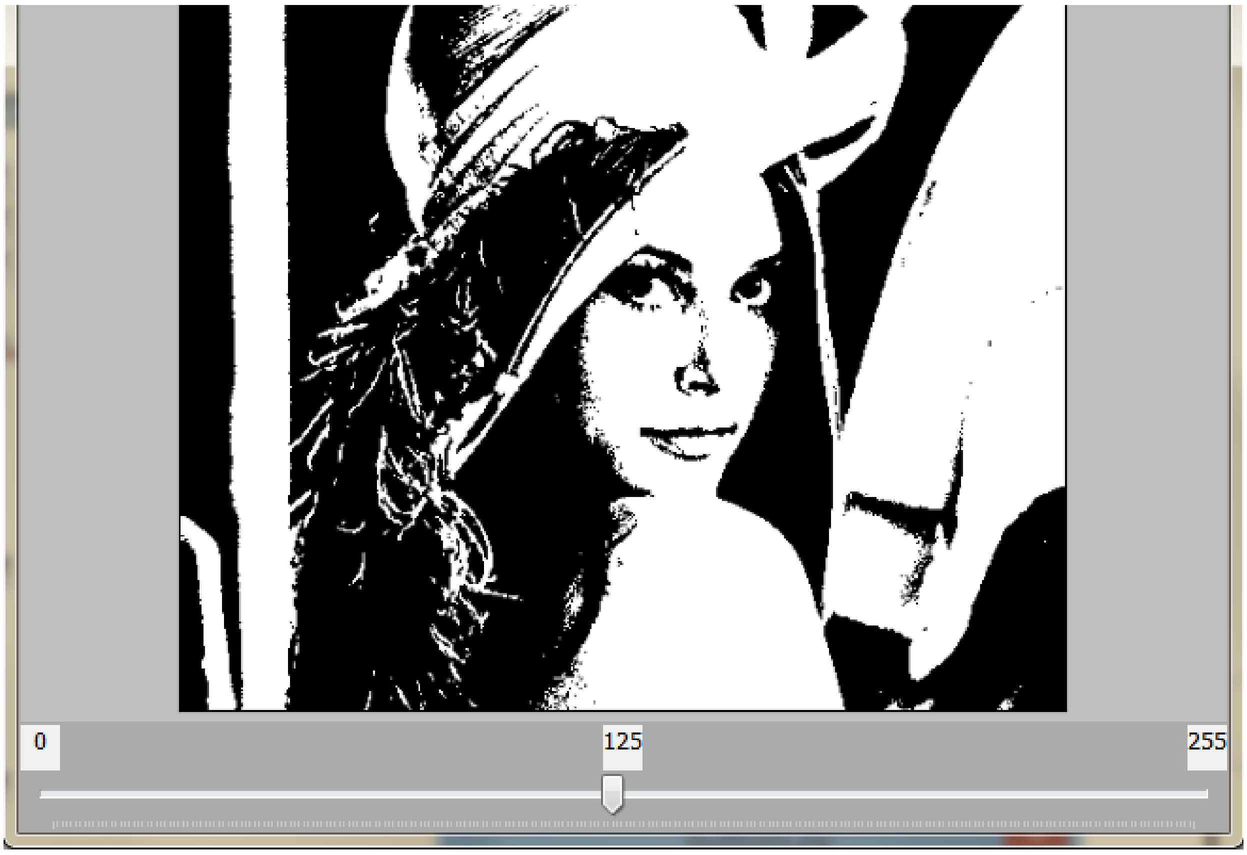
```
In [27]: plt.imshow(img_th3, plt.cm.gray)

Out[27]: <matplotlib.image.AxesImage at 0x93406f0>
```

with additional GUI elemetns

```
In [28]: from IPython.display import Image
         Image(filename="Figures/4_threshold_lena_sizer.png")
```

Out[28]:





In [29]:
```python
%reset

import wx
import numpy as np

from scipy import misc
from matplotlib.figure import Figure
from matplotlib.backends.backend_wxagg import FigureCanvasWxAgg as FigCanvas


class imageShow(wx.Frame):
    def __init__(self,parent):
        self.img = misc.lena()
        wx.Frame.__init__(self, parent, title="Lena", size=(wx.GetClientDisplayRect
        self.CreatePanel()
        self.DrawFigure()

    def CreatePanel(self):
        self.fig = Figure()
        self.fig.subplots_adjust(left=0.01, right=0.99, top=0.99, bottom=0.01)
        self.canvas = FigCanvas(self, -1, self.fig)
        self.axes = self.fig.add_subplot(111)
        self.axes.get_xaxis().set_visible(False)
        self.axes.get_yaxis().set_visible(False)

        self.slider1 = wx.Slider(self,id=wx.ID_ANY, value=125,minValue=0,maxValue=2

        self.Bind(wx.EVT_SLIDER, self.OnSlider)

        self.hbox = wx.BoxSizer(wx.VERTICAL) #main siezer
        self.hbox.Add(self.canvas, 1, wx.EXPAND | wx.ALL,1)
        self.hbox.Add(self.slider1,0.5, wx.EXPAND | wx.ADJUST_MINSIZE |wx.ALL, 1)
        self.SetSizer(self.hbox)
        self.SetAutoLayout(True)
        self.Fit()
    def OnSlider(self,event):
        self.DrawFigure()

    def DrawFigure(self,  minn = 0, maxx = 255):

        self.axes.clear()
        self.imshow = self.axes.imshow(np.where(self.img>self.slider1.GetValue(),25
        self.canvas.draw()


app = wx.App(False)
img = imageShow(None)
img.Show()
```

For volunteers:

Create new script that allows the manipulation of the text. There should be a special place to enter a text (wx.TextCtrl) and function keys shown below:

In [32]:
```
from IPython.display import Image
Image(filename="Figures/5_text_manipulation.png")
```

Out[32]:

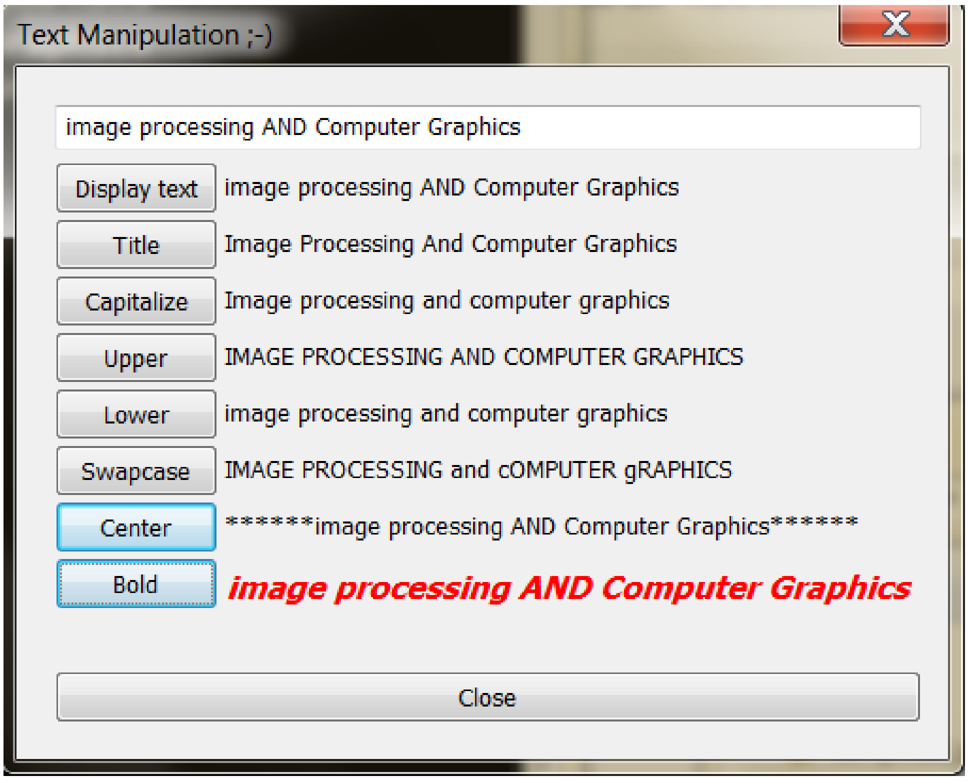


image processing AND Computer Graphics
Image Processing And Computer Graphics
Image processing and computer graphics
IMAGE PROCESSING AND COMPUTER GRAPHICS
image processing and computer graphics
IMAGE PROCESSING and cOMPUTER gRAPHICS
******image processing AND Computer Graphics******
image processing AND Computer Graphics