M.O.B

A 3-level Production

Download.
`https://github.com/3LP/M.O.B`

What is M.O.B? M.O.B in it's original form is an IDE that combines both a computer terminal and a source code editor in one window. We made M.O.B to give scientists with some programming skills, a basic GUI that they can use as a foundation to make their own apps.

M.O.B is written in Python because it can be used to make many different applications, and it is one of the easier programming languages to pick up in order to do computing for science. M.O.B uses Gtk3+ because there is great documentation for GUI design in Python, with this API. Gtk3+ allows M.O.B to be cross-platform between Mac and Linux machines. For Mac users, we highly recommend using homebrew and pypi to manage M.O.B core dependencies.

Dependencies. Vte, Gtk3, GLib, Gdk, GObject, Pango, PangoCairo, numpy, GtkSource, GnomeCommon, Gnome-doc-util, GnomeIconTheme.

Development. M.O.B could not have been built without the guidance provided by the Stack Exchange community. and this tutorial: `http://www.dreamincode.net/forums/topic/150162-a-simple-text-editor-in-pythonpygtk/`

The following link is a valuable resource when you are trying to incorporate your own app into M.O.B: `https://python-gtk-3-tutorial.readthedocs.org/en/latest/`

Why M.O.B? Making apps in Python is not difficult, however figuring out how to make a GUI for the first time will be . First, you want to figure out what platform you want your app to run on. Next, you must research APIs that will satisfy your need's best. Last, you have to write the code for your app. There is a lot of unnecessary groundwork that you must do, M.O.B is here so you don't have to start from scratch. M.O.B is provided with a tutorial so that developers can have an understanding of it's original structure. The tutorials are there to get you up to speed on the layout of code for a GUI in Python, so that you can customize M.O.B to meet your computational desires.

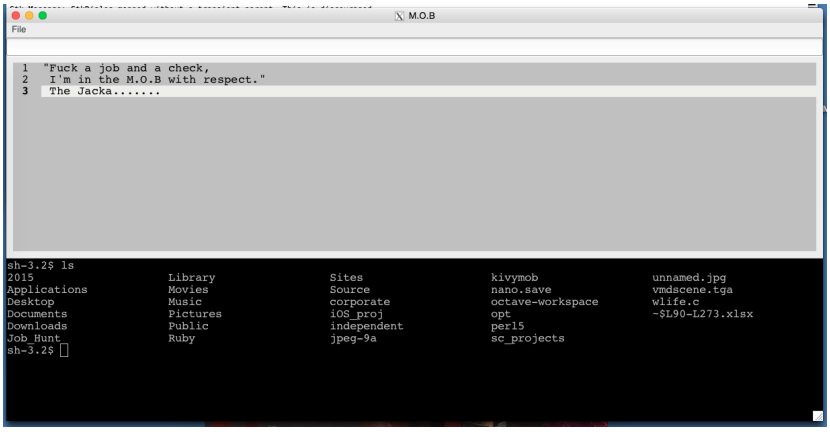

Figure 1. M.O.B screenshot.

IDE M.O.B Tutorial

```
import os,sys
from gi.repository import Gtk, Vte,Pango,PangoCairo
from gi.repository import GLib, Gdk
from gi.repository import GObject
from gi.repository import GtkSource
```

Figure 2. Minimal dependencies for the M.O.B IDE. If you are interested in removing these from M.O.B to replace with your own app, you will not need GtkSource.

Before the class MainWindow is defined, we need to declare any classes that you will need to build your app. For the IDE M.O.B, below are the classes that we need.

```
StatusBar = Gtk.Statusbar()
source = GtkSource.View()
buffer = source.get_buffer()
terminal = Vte.Terminal()
textview = Gtk.TextView()
entry = Gtk.Entry()
menu = Gtk.MenuBar()
filemenu = Gtk.Menu()
scrolledwindow1 = Gtk.ScrolledWindow()
vpaned = Gtk.VPaned()
grid = Gtk.Grid()
```

Figure 3. Declaring instances of required classes

We have set the class MainWindow to inherit functions from Gtk.Window. This class is the foundation of the GUI. It contains the methods necessary to recreate classic GUI functionality; *i.e.* it creates a window with a menubar, that can perform basic Open/Save/SaveAs/Exit functions.

```
class MainWindow(Gtk.Window):
    file_tag = '''
    # Name of File for Open/Save/SaveAs Functions

    def open_file(menuitem, user_param):
        # Code specific to your application
    def save_file(menuitem,user_param):
        # Code specific to your application
    def save_file_as(menuitem,user_param):
        # Code specific to your application
    def entry_go(self,widget):
        # Automatic code generator
        # Convert Latex to formulas into Python and/or C for loops
    def __init__(self):
        Gtk.Window.__init__(self)
        # Window title and Icon
```

```
        self.set_title("M.O.B")
        self.connect("delete-event", Gtk.main_quit)

#Starting Window
window = MainWindow()
Gtk.main()
```

Figure 4. MainWindow Class.

Most of the code in figure 4 is missing, however this figure shows the basic structure for a M.O.B app. Below are the main contents of the MainWindow class in the IDE M.O.B.

```
class MainWindow(Gtk.Window):
    file_tag = ""
    # Open File function
    def open_file(menuitem, user_param):
        chooser = Gtk.FileChooserDialog(title="Open a file",action=Gtk.FileChooserAction.OI
        chooser.set_default_response(Gtk.ResponseType.OK)
        filter = Gtk.FileFilter()
        filter2 = Gtk.FileFilter()
        filter2.set_name("All Files")
        filter2.add_pattern("*.*")
        chooser.add_filter(filter2)
        response = chooser.run()
        if response == Gtk.ResponseType.OK:
            filename = chooser.get_filename()
            global file_tag
            file_tag = filename
            textbuffer = source.get_buffer()
            print "Opened File: " + filename
            StatusBar.push(0,"Opened File: " + filename)
            index = filename.replace("\\","/").rfind("/") + 1
            file = open(filename, "r")
            text = file.read()
            textbuffer.set_text(text)
            file.close()
            chooser.destroy()
        elif response == Gtk.ResponseType.CANCEL:
            chooser.destroy()
            chooser.destroy()
    def save_file(menuitem,user_param):
            textbuffer = source.get_buffer()
            StatusBar.push(0,"Saved File: " + file_tag)
            index = file_tag.replace("\\","/").rfind("/") + 1
            text = textbuffer.get_text(textbuffer.get_start_iter() , textbuffer.get_end_ite
            file = open(file_tag , "w+")
            file.write(text)
            file.close()

    def save_file_as(menuitem,user_param):
        chooser = Gtk.FileChooserDialog(title="Save file",action=Gtk.FileChooserAction.SAVI
        chooser.set_default_response(Gtk.ResponseType.OK)
        filter2 = Gtk.FileFilter()
```

```
        filter2.set_name("All Files")
        filter2.add_pattern("*.*")
        chooser.add_filter(filter2)
        response = chooser.run()
        if response == Gtk.ResponseType.OK:
            filename = chooser.get_filename()
            textbuffer = source.get_buffer()
            print "Saved File: " + filename
            StatusBar.push(0,"Saved File: " + filename)
            index = filename.replace("\\","/").rfind("/") + 1
            text = textbuffer.get_text(textbuffer.get_start_iter() , textbuffer.get_end_ite
            file = open(filename, "w")
            file.write(text)
            file.close()
            chooser.destroy()
        elif response == Gtk.ResponseType.CANCEL:
            chooser.destroy()
            chooser.destroy()


    def entry_go(self,widget):
        input = entry.get_text()
        print(input)
#       #Octave or Python Session Instructions
        octave_session = 'octave'
        python_session = 'python'
        command= input+"\n"
        if input == octave_session or input == python_session:
          # send terminal commands
                length = len(command)
                terminal.feed_child(command, length)
                scrolledwindow1.remove(source)
                self.box = Gtk.VBox(homogeneous=False, spacing=0)
                self.add(self.box)
                terminal.menu = Gtk.Menu()
                menu_item = Gtk.ImageMenuItem.new_from_stock("gtk-copy", None)
                menu_item.connect_after("activate", lambda w: self.copy_clipboard())
                terminal.menu.add(menu_item)
                scrolledwindow1.remove(source)
                # Vertical Pane
                vpaned.add1(scrolledwindow1)
                vpaned.add2(terminal)
                # Pack everything in vertical box
                self.box.pack_start(entry, False, False, 0)
                self.box.pack_start(vpaned, True, True, 0)
                self.connect("delete-event", Gtk.main_quit)
                self.show_all()

        length = len(command)
        #This Conditional Reads in Array Info
        if input[length-2]==']':
                for x in range(2,length-2):
                        if input[x]!='[':
                                # Grid Shit
```

```
                                scrolledwindow1.remove(source)
                                scrolledwindow1.add(source)
                                file_tag = ''
                                textbuffer = source.get_buffer()
                                textbuffer.set_text(input)



        terminal.feed_child(command, length)
        entry.set_text(' ')



    def get_text(object, *args):
        term_input = repr(terminal.get_text(lambda *a: True))
        file = open('term_input.txt','w+')
        file.write(term_input)
        file.close()

    def Trap1(self,widget):
        scrolledwindow1.remove(source)
        scrolledwindow1.add(source)
        file_tag = ''
        textbuffer = source.get_buffer()
        textbuffer.set_text('')



    def Trap2(self,widget):
        self.box = Gtk.VBox(homogeneous=False, spacing=0)
        self.add(self.box)
        terminal.menu = Gtk.Menu()
        menu_item = Gtk.ImageMenuItem.new_from_stock("gtk-copy", None)
        menu_item.connect_after("activate", lambda w: self.copy_clipboard())
        terminal.menu.add(menu_item)
        scrolledwindow1.remove(source)
        # Grid Shit
        table_entry1 = Gtk.Entry()
        table_entry2 = Gtk.Entry()
        self.add(grid)
        grid.add(table_entry1)
        grid.add(table_entry2)
        scrolledwindow1.add(grid)
        # Vertical Pane
        vpaned.add1(scrolledwindow1)
        vpaned.add2(terminal)
        # Pack everything in vertical box
        #self.box.pack_start(menu, False, False, 0)
        self.box.pack_start(entry, False, False, 0)
        self.box.pack_start(vpaned, True, True, 0)
        self.connect("delete-event", Gtk.main_quit)
        self.show_all()

    def __init__(self):
```

```
        Gtk.Window.__init__(self)
        # Window title and Icon
        self.set_title("M.O.B")
        # Vertical Box
        self.box = Gtk.VBox(homogeneous=False, spacing=0)
        self.add(self.box)
        # Menu
        filem = Gtk.MenuItem("File")
        #Popup Menu
        terminal.menu = Gtk.Menu()
        menu_item = Gtk.ImageMenuItem.new_from_stock("gtk-copy", None)
        menu_item.connect_after("activate", lambda w: self.copy_clipboard())
        terminal.menu.add(menu_item)
        # Import
        imenu = Gtk.Menu()
        importm = Gtk.MenuItem("Options and Modes")
        importm.set_submenu(imenu)
        itrap1 = Gtk.MenuItem("New Source")
        itrap2 = Gtk.MenuItem("Table Mode")
        imenu.append(itrap1)
        imenu.append(itrap2)
        #TRAP Signals
        itrap1.connect("activate",self.Trap1)
        itrap2.connect("activate",self.Trap2)
        #Signals for other menu items
        openm = Gtk.MenuItem("Open")
        savem = Gtk.MenuItem("Save")
        saveasm = Gtk.MenuItem("Save As")
        exit = Gtk.MenuItem("Exit")
        openm.connect("activate",self.open_file)
        saveasm.connect("activate",self.save_file_as)
        savem.connect("activate",self.save_file)
        exit.connect("activate", Gtk.main_quit)
        filemenu.append(importm)
        filemenu.append(openm)
        filemenu.append(savem)
        filemenu.append(saveasm)
        filemenu.append(exit)
        filem.set_submenu(filemenu)
        menu.append(filem)
        # Source View
        source.set_show_line_numbers(True)
        source.set_show_line_marks(True)
        source.set_highlight_current_line(True)
        source.modify_bg(Gtk.StateType.NORMAL, Gdk.color_parse("silver"))
        source.modify_fg(Gtk.StateType.NORMAL, Gdk.color_parse("black"))
        # textbuffer = source.get_buffer()
        # textbuffer.set_text(input)
        # Highlighting Functionality
        # textbuffer = source.get_buffer()
        # Terminal settings
        terminal.set_scroll_on_output(True)
        terminal.set_scroll_on_keystroke(True)
        #terminal.set_visible(True)
```

```
        terminal.set_scrollback_lines(-1)
        terminal.set_font_from_string("monospace 10")
        terminal.set_color_background(Gdk.color_parse("black"))
        terminal.set_color_foreground(Gdk.color_parse("silver"))
        terminal.set_cursor_blink_mode(True)
        terminal.connect("child-exited", lambda w: gtk.main_quit())
        terminal.set_encoding("UTF-8")
        terminal.fork_command_full(Vte.PtyFlags.DEFAULT,os.environ['HOME'],["/bin/sh"],[],C
        # send terminal commands
        # Through Text Entry Box
        entry.connect("activate",self.entry_go)
        input = entry.get_text()
        print(input)
        # Scrolled Text Window
        scrolledwindow1.set_hexpand(True)
        scrolledwindow1.set_vexpand(True)
        scrolledwindow1.set_border_width(10)
        scrolledwindow1.add(source)
        # Vertical Pane
        vpaned.add1(scrolledwindow1)
        vpaned.add2(terminal)
        # Pack everything in vertical box
        self.box.pack_start(menu, False, False, 0)
        self.box.pack_start(entry, False, False, 0)
        self.box.pack_start(vpaned, True, True, 0)
        self.connect("delete-event", Gtk.main_quit)
        self.show_all()
```