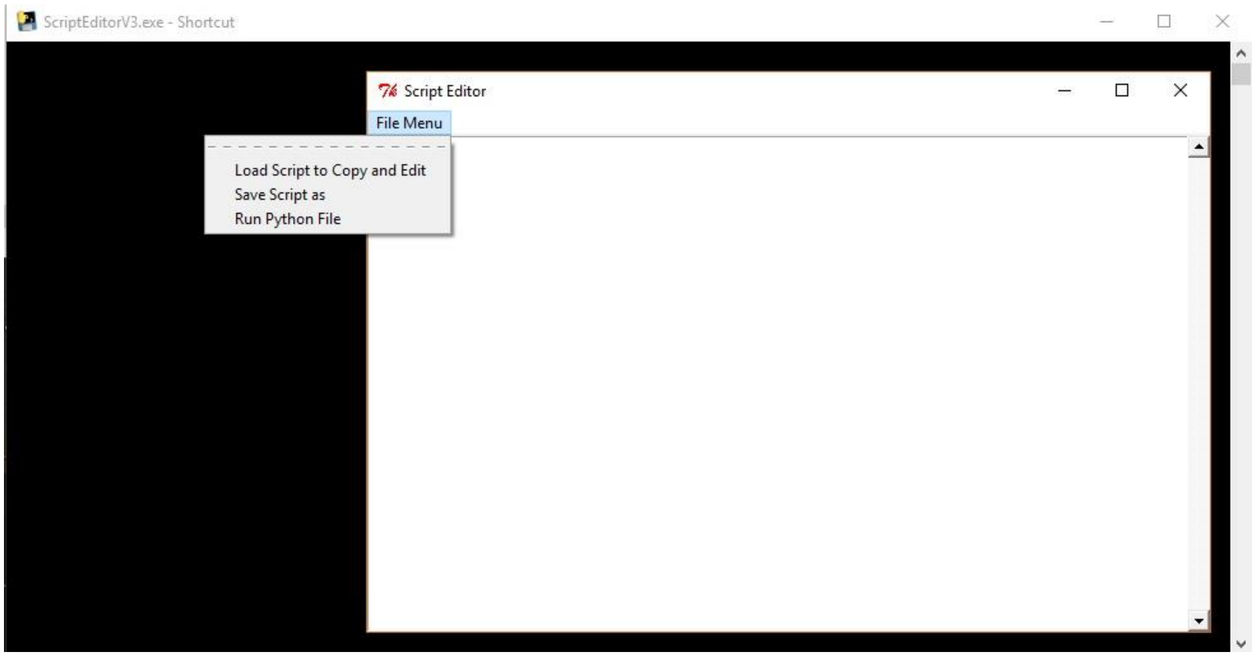
Documentation ScriptEditorV3 Application

Summary. ScriptEditor is designed to assist developers edit their Python scripts during application development. It is not intended to replace Python Integrated Development Environments (IDEs) but to act as a supplemental clipboard from which to access, copy, paste, edit, and run Python scripts. It is very similar but not identical to Notepad applications. It is probably most useful when developers are working between multiple scripts searching for code to merge and modify into a single working application.

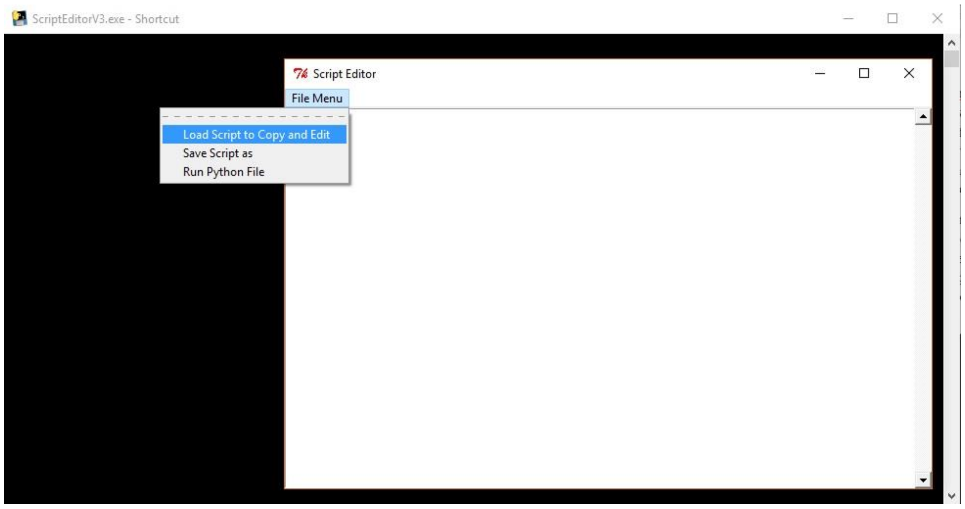
The Tkinter module (Tk interface) is imported to give the developer access to the Tk Graphic User Interface (GUI) structure and controls, thus allowing the user a graphics environment from which to point and click to execute the application's functions. It has been converted into a standalone executable application that can be run in any Windows Operating System after Windows XP. The interface works from a file menu dropdown picklist allowing the user to select a function[1]:

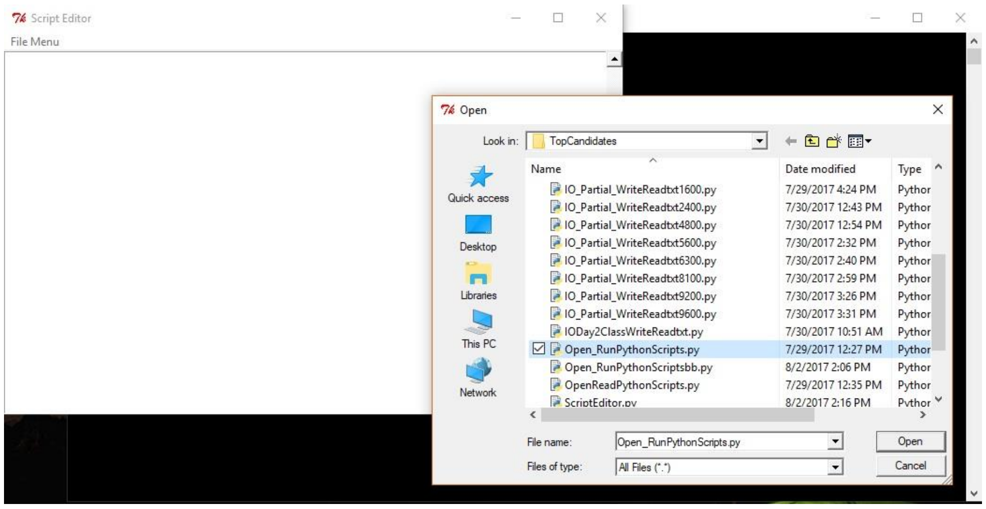


There are three primary functions available: 1) 'Load script to Copy and Edit', 2) 'Save Script as', and 3) 'Run Python File.' Selecting option 1 or 3 will first open a browser to the user's file directory and allow a selection of the Python file they wish to: 1:copy into the ScriptEditor text field or 3:run. Selecting option 2 will allow users to save the text currently displayed in the ScriptEditor text field as either a Python script (filename.py) or a text file (filename.txt) and place it into their file directory.

[1] Note: a Python console is enabled before the Tk interface and stays open until the application is terminated. If script is run that prints to a console, it will print to this console.

Function 1: ‘Load Script to Copy and Edit’



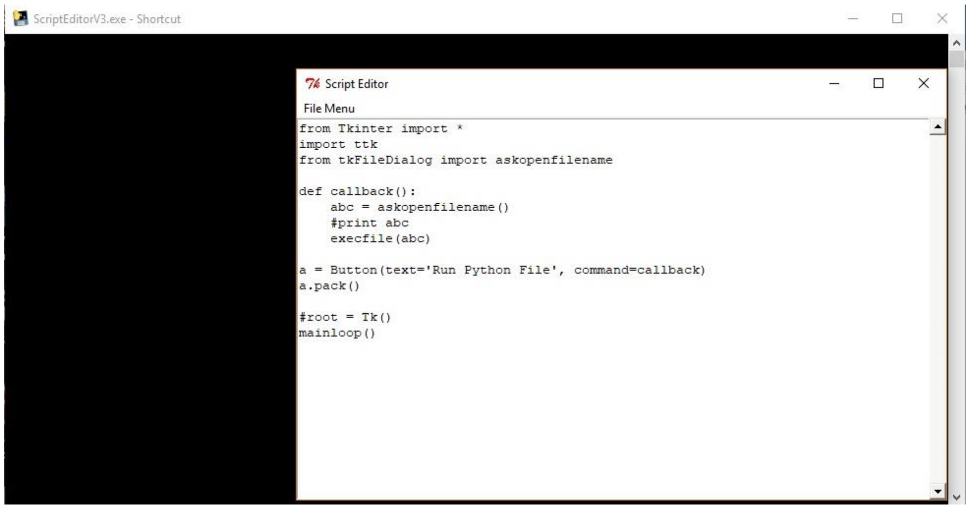


Look in: TopCandidates

Name	Date modified	Type
IO_Partial_WriteReadtxt1600.py	7/29/2017 4:24 PM	Pythor
IO_Partial_WriteReadtxt2400.py	7/30/2017 12:43 PM	Pythor
IO_Partial_WriteReadtxt4800.py	7/30/2017 12:54 PM	Pythor
IO_Partial_WriteReadtxt5600.py	7/30/2017 2:32 PM	Pythor
IO_Partial_WriteReadtxt6300.py	7/30/2017 2:40 PM	Pythor
IO_Partial_WriteReadtxt8100.py	7/30/2017 2:59 PM	Pythor
IO_Partial_WriteReadtxt9200.py	7/30/2017 3:26 PM	Pythor
IO_Partial_WriteReadtxt9600.py	7/30/2017 3:31 PM	Pythor
IODay2ClassWriteReadtxt.py	7/30/2017 10:51 AM	Pythor
Open_RunPythonScripts.py	7/29/2017 12:27 PM	Pythor
Open_RunPythonScriptsbb.py	8/2/2017 2:06 PM	Pythor
OpenReadPythonScripts.py	7/29/2017 12:35 PM	Pythor
ScriptEditor.py	8/2/2017 2:16 PM	Pythor

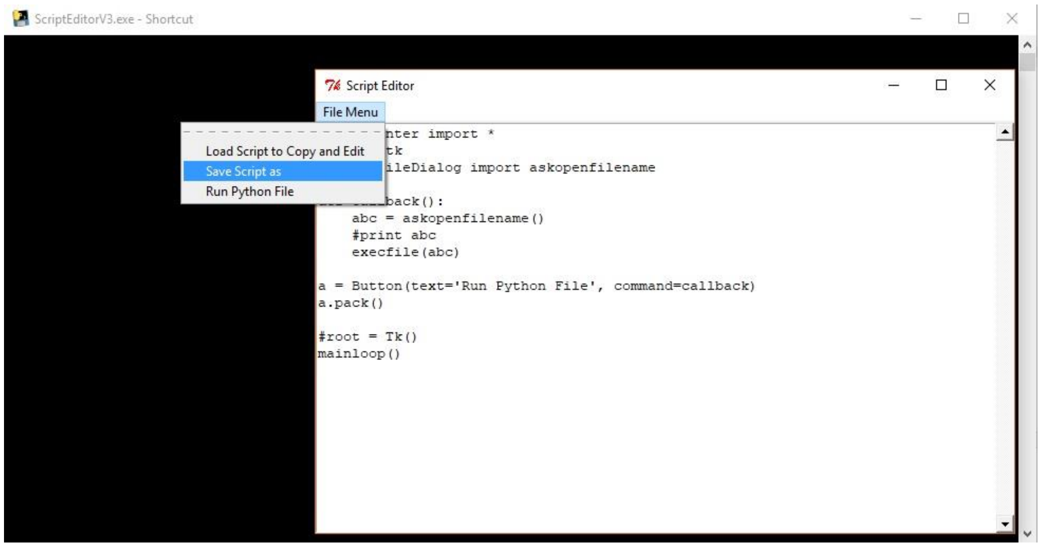
File name: Open_RunPythonScripts.py

Files of type: All Files (*.*)

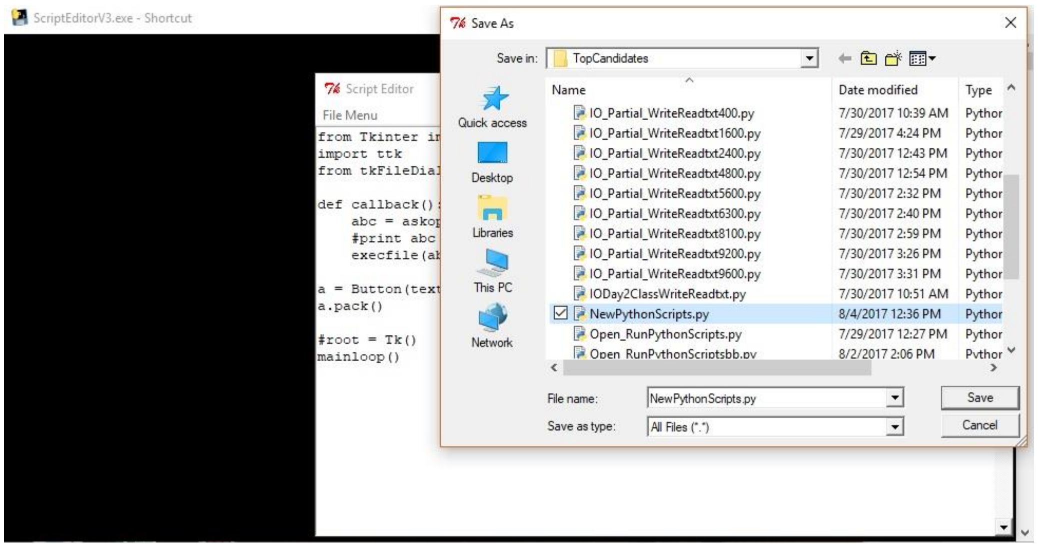


```
from Tkinter import *
import ttk
from tkFileDialog import askopenfilename

def callback():
    abc = askopenfilename()
    #print abc
    execfile(abc)

a = Button(text='Run Python File', command=callback)
a.pack()

#root = Tk()
mainloop()
```

Function 2: ‘Save Script as‘



```
nter import *
tk
ileDialog import askopenfilename

back():
    abc = askopenfilename()
    #print abc
    execfile(abc)

a = Button(text='Run Python File', command=callback)
a.pack()

#root = Tk()
mainloop()
```



```
from Tkinter i
import ttk
from tkFileDial

def callback()
    abc = askop
    #print abc
    execfile(ab

a = Button(text
a.pack()

#root = Tk()
mainloop()
```

Save in: TopCandidates

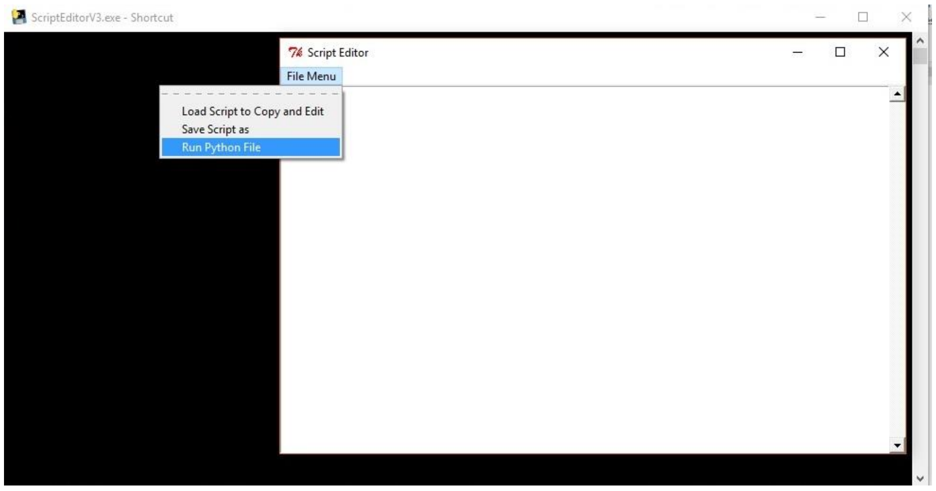
Name	Date modified	Type
IO_Partial_WriteReadtxt400.py	7/30/2017 10:39 AM	Pythor
IO_Partial_WriteReadtxt1600.py	7/29/2017 4:24 PM	Pythor
IO_Partial_WriteReadtxt2400.py	7/30/2017 12:43 PM	Pythor
IO_Partial_WriteReadtxt4800.py	7/30/2017 12:54 PM	Pythor
IO_Partial_WriteReadtxt5600.py	7/30/2017 2:32 PM	Pythor
IO_Partial_WriteReadtxt6300.py	7/30/2017 2:40 PM	Pythor
IO_Partial_WriteReadtxt8100.py	7/30/2017 2:59 PM	Pythor
IO_Partial_WriteReadtxt9200.py	7/30/2017 3:26 PM	Pythor
IO_Partial_WriteReadtxt9600.py	7/30/2017 3:31 PM	Pythor
IODay2ClassWriteReadtxt.py	7/30/2017 10:51 AM	Pythor
✓ NewPythonScripts.py	8/4/2017 12:36 PM	Pythor
Open_RunPythonScripts.py	7/29/2017 12:27 PM	Pythor
Open_RunPythonScriptsbb.py	8/2/2017 2:06 PM	Pythor

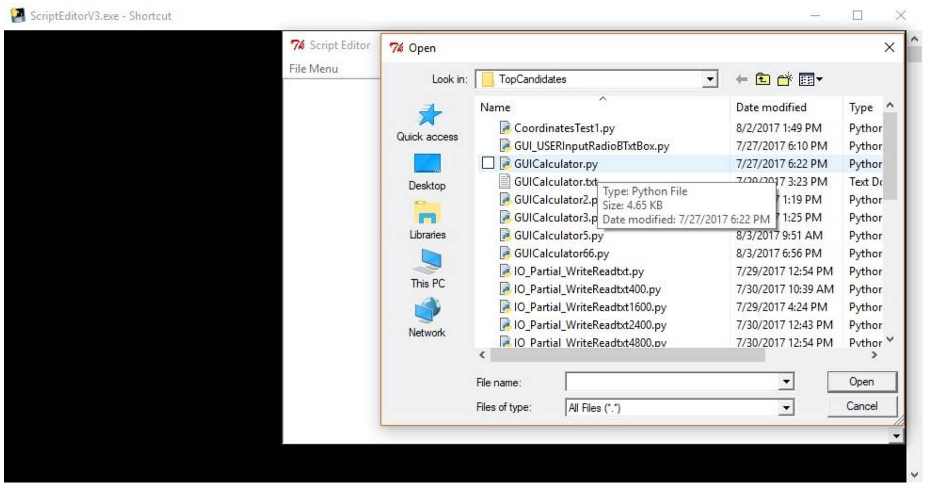
File name: NewPythonScripts.py

Save as type: All Files (*.*)

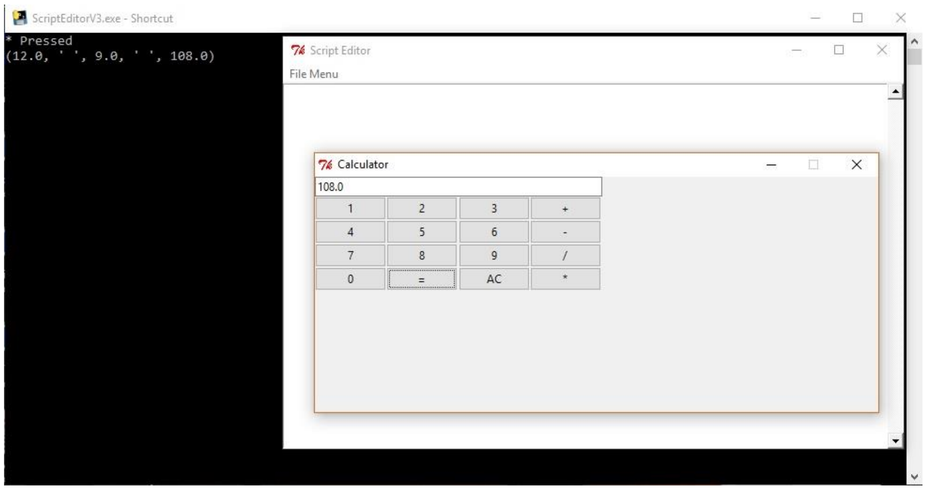
Users can save the script in the ScriptEditor under an existing file name if they want to modify that file or under a completely new file name to differentiate it from other scripts. For users to immediately test run the script, they must save it using the ‘.py’ extension. If the file script is meant to be considered temporary or intermediate data for storage and use in future editing sessions, users have the option to save their scripts as text files.

Function 3: ‘Run Python File’





Name	Date modified	Type
CoordinatesTest1.py	8/2/2017 1:49 PM	Pythor
GUI_USERInputRadioBTxtBox.py	7/27/2017 6:10 PM	Pythor
GUICalculator.py	7/27/2017 6:22 PM	Pythor
GUICalculator.txt	7/29/2017 3:23 PM	Text Do
GUICalculator2.p	1:19 PM	Pythor
GUICalculator3.p	1:25 PM	Pythor
GUICalculator5.py	8/3/2017 9:51 AM	Pythor
GUICalculator66.py	8/3/2017 6:56 PM	Pythor
IO_Partial_WriteReadtxt.py	7/29/2017 12:54 PM	Pythor
IO_Partial_WriteReadtxt400.py	7/30/2017 10:39 AM	Pythor
IO_Partial_WriteReadtxt1600.py	7/29/2017 4:24 PM	Pythor
IO_Partial_WriteReadtxt2400.py	7/30/2017 12:43 PM	Pythor
IO_Partial_WriteReadtxt4800.py	7/30/2017 12:54 PM	Pythor

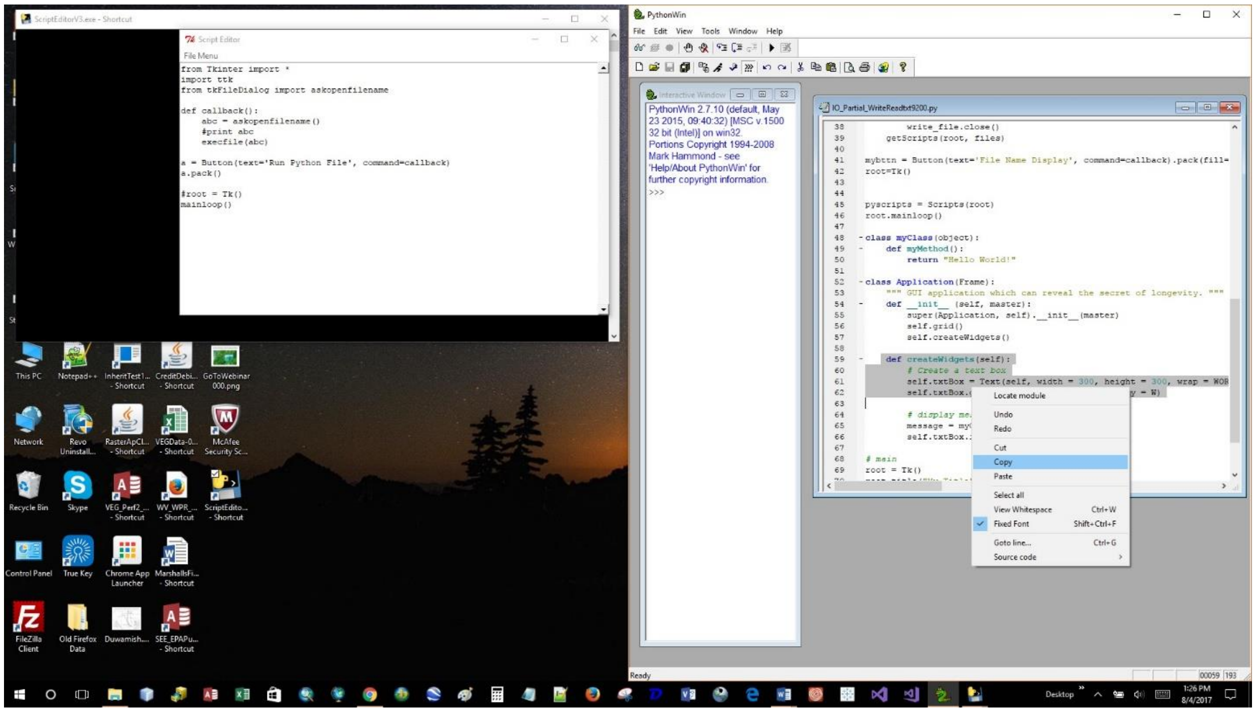


```
* Pressed
(12.0, ' ', 9.0, ' ', 108.0)
```

108.0

Notice in the run Python script function example above, since the application being run prints to a console when in use, the print results display in the ScriptEditer console.

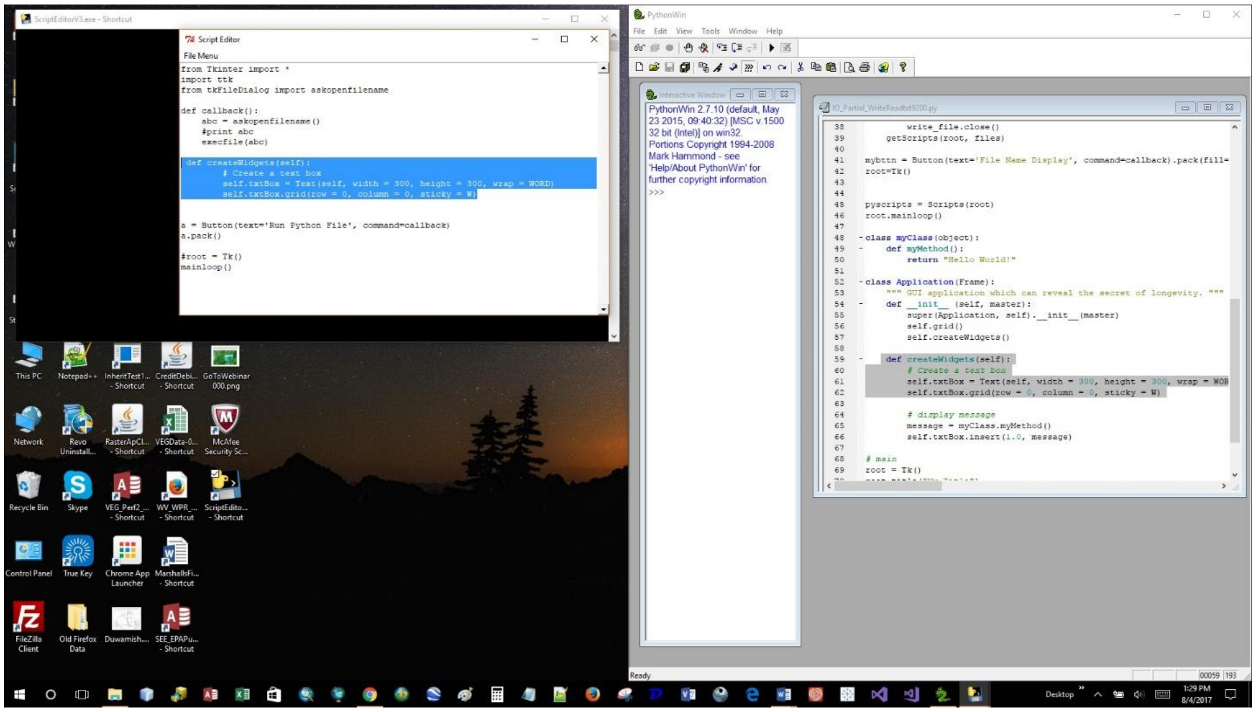
Interacting with Python IDE



```
from Tkinter import *
import ttk
from tkFileDialog import askopenfilename

def callback():
    abc = askopenfilename()
    #print abc
    execfile(abc)

a = Button(text='Run Python File', command=callback)
a.pack()

#root = Tk()
mainloop()
```

```
PythonWin 2.7.10 (default, May 23 2015, 09:40:32) [MSC v.1500 32 bit (Intel)] on win32.
Portions Copyright 1994-2008 Mark Hammond - see 'Help/About PythonWin' for further copyright information.
>>>
```

```
            write_file.close()
        getScripts(root, files)

mybttn = Button(text='File Name Display', command=callback).pack(fill=
root=Tk()


pyscripts = Scripts(root)
root.mainloop()

class myClass(object):
    def myMethod():
        return "Hello World!"

class Application(Frame):
    """ GUI application which can reveal the secret of longevity. """
    def __init__ (self, master):
        super(Application, self).__init__(master)
        self.grid()
        self.createWidgets()

    def createWidgets(self):
        # Create a text box
        self.txtBox = Text(self, width = 300, height = 300, wrap = WOR
        self.txtBox.(                                      y = W)

        # display me
        message = my
        self.txtBox.

# main
root = Tk()
```


ScriptEditorV3.exe - Shortcut
Script Editor
File Menu
PythonWin
File Edit View Tools Window Help
Interactive Window
IO_Partial_WriteReadtxt9200.py
Ready

```
from Tkinter import *
import ttk
from tkFileDialog import askopenfilename

def callback():
    abc = askopenfilename()
    #print abc
    execfile(abc)

 def createWidgets(self):
        # Create a text box
        self.txtBox = Text(self, width = 300, height = 300, wrap = WORD)
        self.txtBox.grid(row = 0, column = 0, sticky = W)


a = Button(text='Run Python File', command=callback)
a.pack()

#root = Tk()
mainloop()
```

```
PythonWin 2.7.10 (default, May 23 2015, 09:40:32) [MSC v.1500 32 bit (Intel)] on win32.
Portions Copyright 1994-2008 Mark Hammond - see 'Help/About PythonWin' for further copyright information.
>>>
```

```
            write_file.close()
        getScripts(root, files)

mybttn = Button(text='File Name Display', command=callback).pack(fill=
root=Tk()


pyscripts = Scripts(root)
root.mainloop()

class myClass(object):
    def myMethod():
        return "Hello World!"

class Application(Frame):
    """ GUI application which can reveal the secret of longevity. """
    def __init__ (self, master):
        super(Application, self).__init__(master)
        self.grid()
        self.createWidgets()

    def createWidgets(self):
        # Create a text box
        self.txtBox = Text(self, width = 300, height = 300, wrap = WOR
        self.txtBox.grid(row = 0, column = 0, sticky = W)

        # display message
        message = myClass.myMethod()
        self.txtBox.insert(1.0, message)

# main
root = Tk()
```

Script can be copied and pasted between the ScriptEditor and the Python IDE by using Windows, control <C> for copy and control <V> for paste, commands. I am currently working on adding a ‘Parse Text’ function to ScriptEditorV4.

A ‘Cat’, ‘Bat’ ‘Mat’ Philosophy to Code Development.

Following my early programming training, I started this effort by drafting an outline of my goals and then looking for an existing application with functions as closely matched to my target goals as I could find. One of my code instructors referred to this method as the “Cat, Bat, Mat” strategy to code development. In other words, you start with code that is as close to what you want in a final product that you can find and then, very carefully and methodically, slowly begin tweaking, testing, and changing it until it becomes the program you want it to be.

Initially all I came across were a few relatively small programs, each with some subset of the needed functions as compared to an ideally fully functional ScriptEditor. Over a period of a several weeks, I managed to stitch together two or three slightly larger programs that collectively managed most of the needed functions but that individually still fell far short. Much of the challenge was trying to unite them inside a Tkinter structure. Finally, I found a You Tube tutorial for developing a text editor program inside Tkinter and, with a few minor variations, used the tutorial to write a script that did about 75 to 80-percent of what I wanted. Later I inserted a run Python code method and call that I found in another script, resulting in the ScriptEditor application meeting 85 to 90-percent of its original project goals.

Tkinter to Create a Graphic User Interface (GUI)

Tkinter is an acronym for Tk interface and was originally developed as a GUI extension for ‘Td’ scripting in the early 1990’s. It was later adopted into several programming languages, including Python. The interface is useful in GUI development because it provides many common GUI elements such as various entry and display areas, labels, buttons, etc. Depending on the needs of the application, various Tkinter imports collectively assist the developer in creating a GUI structure that users can easily interact with. From the developer’s perspective, the most challenging aspect of Tkinter may be adapting the internal class / method script syntax to correctly relate to and subsequently run inside the Tkinter structure. The following is a typical Tkinter syntax (note there are many variations in specific terms but the elements themselves appear to be reasonably consistent):[2]

```
from Tkinter import *

root = Tk()

myApp = myClassConstructor

root.mainloop()
```

Making Python Scripts into Executable Applications.

To make the functionality of this program accessible to a broad spectrum of users it had to be easily executable on a wide range of computer operating systems. To achieve this, I used the following procedure to convert the Python script to an independent executable application:

[2] See ScriptEditorV3.py Python script for more specific Tkinter syntax associated with this application.

1. Reviewed documentation for ‘PyInstaller’
2. Found Python installation folder on my ‘C’ drive and navigated to the ‘Scripts’ folder (e.g., C:\Python27\ArcGIS10.4\Scripts).
3. Opened my command prompt in computers with Windows 10 OS and right clicked on windows icon, selected Run, and entered ‘cmd’ into open textbox.
4. In command line, entered ‘cd’ to change directory then copied and pasted (e.g., <control> [3]C, <control> V) the directory path to the ‘Scripts’ folder after ‘cd’ and selected ‘Enter’ (e.g., cd C:\Python27\ArcGIS10.4\ Scripts <Enter>).
5. In new directory, I entered ‘pip install pyinstaller’ (e.g., C:\Python27\ ArcGIS10.4\Scripts> pip install pyinstaller<Enter>). I made sure I was connected to the internet to enable pip to find and install pyinstaller. It took a few seconds for the installation to complete. Once completed I observed the installed new files in my ‘Scripts’ folder.
6. I navigated back to my command window and made sure it was still using the ‘Scripts’ directory (e.g., ‘C:\Python27\ArcGIS10.4\Scripts>’). Then I entered: pyinstaller.exe - -[4]onefile and the directory path to my Python script (ScriptEditorV3.py) to create a standalone executable application (e.g., C:\Python27\ ArcGIS10.4\Scripts> pyinstaller.exe –onefile C:\EsriPress\Python\ Data\ Exercise13\ ScriptEditorV3.py<Enter>).
7. Once the procedure completed running, the last line of the run output indicated where my execution file had been placed (e.g., C:\Python27\ArcGIS10.4 \ Scripts\dist\ScriptEditorV3.exe).
8. I double clicked on my executable file in the ‘dist’ folder to test run the application. I then created a shortcut on my desktop and uploaded a copy of the executable file along with the Python script and supporting documentation to my University of Washington Google Drive Geography 565 Class Final Project Folder.

[3] <> denotes key on computer keyboard.
[4] - - denotes parameter entered into the pyinstaller.exe function. Look to pyinstaller documentation for more information about the available parameters.

Proposed Solution Design Goals

The final project will have four primary phases of development:

1. Develop python file nomenclature and classification system conducive to timely and efficient file search, retrieval, review, and opening:

   Status: Not Completed

2. Write the Python script that implements python file search, find, review, and open methods:

   Status: Partially Completed

3. Develop Tool's Graphic User Interface (GUI):

   Status: Partially Completed

4. Write all tool documentation and help procedures:

   Status: Partially Completed

Preliminary Proposal for Tool Workflow.

1. Allow User to Select File Retrieval Workspace.

   Status: Completed

2. Allow User to Select Pre-established Search Criteria for File Object(s).

   Status: Not Completed

3. Allow User to Initiate File Find Method(s).

   Status: Partially Completed

4. Allow User to Select Candidate Files for Preliminary Script Display

   Status: Completed

5. Allow User to Open Selected Files in IDE.

   Status: Completed through Workaround

Final Project Deliverables.

1. Tool box:

   Status: Completed using Tk Interface to create a GUI

2. Script:

   Status: Partially Completed - versioned scripts submitted

3. Test data:

   Status: Partially Completed – test scripts submitted

4. Documentation:

   Status: Completed – Submitted Documentation

5. Project Summary:

   Status: Partially Completed – Submitted Partial Summary