

PYTHON PROJECT

PYTO MAP

Table of contents:

1. Abstract

2. Introduction

3. Project Description

3.1 Implementation details

3.2 Class diagram

4. Module description

4.1 Go to Map

4.2 Go to Address

4.3 Exit

5. System Analysis

5.1 Hardware requirements

5.2 Software requirements

6. Conclusion

7. Appendix

7.1 Source code

7.2 Screenshot

1. **ABSTRACT:**

Pyto Map is a Python source Based Map Working on a web Browser. From this, we had used Python and its library functions. It gives a GUI for the user where they can input the Starting Place and Destination or they can search the place. Hence, the user can able to search with less effort.

1. **INTRODUCTION:**

### [Tkinter](https://docs.python.org/3/faq/gui.html#id5):

### Standard builds of Python include an object-oriented interface to the Tcl/Tk widget set, called [tkinter](https://docs.python.org/3/library/tk.html#tkinter). This is probably the easiest to install (since it comes included with most [binary distributions](https://www.python.org/downloads/) of Python) and use. For more info about Tk, including pointers to the source, see the [Tcl/Tk home page](https://www.tcl.tk/). Tcl/Tk is fully portable to the Mac OS X, Windows, and UNIX platforms.

### WEBBROWSER:

### Using the web browser in Python. The web browser module provides a high-level interface to allow displaying Web-based documents to users. Under most circumstances, simply calling the open () function from this module will open URL using the default browser. You have to import the module and use open () function.

**3. PROJECT DESCRIPTION:**

**3.1 IMPLEMENTATION DETAILS:**

* (from Tkinter import \* ) – I had imported all the functions from tkinter package (Note: Tkinter for python 2 and tkinter for python 3).
* It is used to raise a new Graphical user interface page where we build the user needed icons and its operations.
* (from webbrowser) I had imported the webbrowser to parse the web page in order to access the google map.

**3.2 CLASS DAIGRAM:**

|  |
| --- |
| Start page |
| Start Page |
| Map Page  Direction Page  Initialize |

|  |
| --- |
| Google Map |
| \_\_init\_\_  Raise page |

|  |
| --- |
| Map page and Direction Page |
| initialize  search |

**4. SYSTEM ANALYSIS:**

**4.1 Hardware Requirements:**

| **Requirement** | **Minimum** | **Recommended** |
| --- | --- | --- |
| Memory | 512 MB | 1 GB |
| Free disk space | 300 MB | 1 GB |

**4.2 Software Requirements:**

* Needs Python interpreter and Eclipse Software with pydev plugin
* Operating System :
* Microsoft Windows 7 or later and Linux
* Python Version : 3.6 needed

**5. MODULE DESCRIPTION:**

**5.1 Google Page:**

An initial frame is created for the Google Map where the start page class is initialized and Raise frame is used for shifting the page. Where looping is used for accessing the page and the frame is packed using the grid option in order to display in the GUI.

* 1. **Start Page:**

A three buttons are created in the frame such as Map\_Button, Direction\_Button, Quit\_Button. Each button is given with command.

* 1. **Direction Page:**

Frame is initialized with buttons and search function is defined which is used to parse the web page for the given address. Back button is used to traverse the pages.

**6. CONCLUSION:**

Explain the **Importance** of **Maps** in Human Geography. **Maps** are crucial to research within Human Geography as they give context to demographic information, and through the **use** of both physical **maps** and Geographical Information Systems, geographers can provide spatial grounding and evidence. Hence I had made the map through the python.

**7. Appendix:**

7.1 Source code

from tkinter import \*

import webbrowser

class GoogleMap(Tk):

def \_\_init\_\_(self):

Tk.\_\_init\_\_(self)

Tk.title(self, 'PYTO MAP')

container = Frame(self)

container.grid()

self.frames = {}

for page in (StartPage, MapPage, DirectionPage):

frame = page(container, self)

self.frames[page] = frame

frame.grid(row=0, column=0, sticky=N+S+E+W)

self.raise\_frame(StartPage)

def raise\_frame(self, frame\_title):

frame = self.frames[frame\_title]

frame.tkraise()

class StartPage(Frame):

def \_\_init\_\_(self, parent, controller):

Frame.\_\_init\_\_(self, parent)

Map\_Button = Button(self, text='Go to Map',

command=lambda: controller.raise\_frame(MapPage))

Map\_Button.grid(row=0, column=0)

Direction\_Button = Button(self, text='Go to Direction',

command=lambda: controller.raise\_frame(DirectionPage))

Direction\_Button.grid(row=0, column=1)

Quit\_Button = Button(self, text='Quit', command=quit)

Quit\_Button.grid(row=0, column=2)

class MapPage(Frame):

def \_\_init\_\_(self, parent, controller):

Frame.\_\_init\_\_(self, parent)

Address\_Label = Label(self, text='Street address:')

Address\_Label.grid(row=0, column=0)

String\_Entry = Entry(self)

String\_Entry.grid(row=0, column=1)

def search():

url = 'https://www.google.com/maps/place/' + String\_Entry.get()

webbrowser.open(url)

Search\_Button = Button(self, text='Search', command=search)

Search\_Button.grid(row=0, column=2)

Back\_Button = Button(self, text='Go back',

command=lambda: controller.raise\_frame(StartPage))

Back\_Button.grid(row=1, columnspan=3)

class DirectionPage(Frame):

def \_\_init\_\_(self, parent, controller):

Frame.\_\_init\_\_(self, parent)

Start\_Label = Label(self, text='Start address:')

Start\_Label.grid(row=0, column=0)

Start\_Entry = Entry(self)

Start\_Entry.grid(row=0, column=1)

Finish\_Label = Label(self, text='Finish address:')

Finish\_Label.grid(row=1, column=0)

Finish\_Entry = Entry(self)

Finish\_Entry.grid(row=1, column=1)

Back\_Button = Button(self, text='Go back',

command=lambda: controller.raise\_frame(StartPage))

Back\_Button.grid(row=0, column=2)

def search():

url = ('https://www.google.com/maps/dir/' + Start\_Entry.get() +

'/' + Finish\_Entry.get())

webbrowser.open(url)

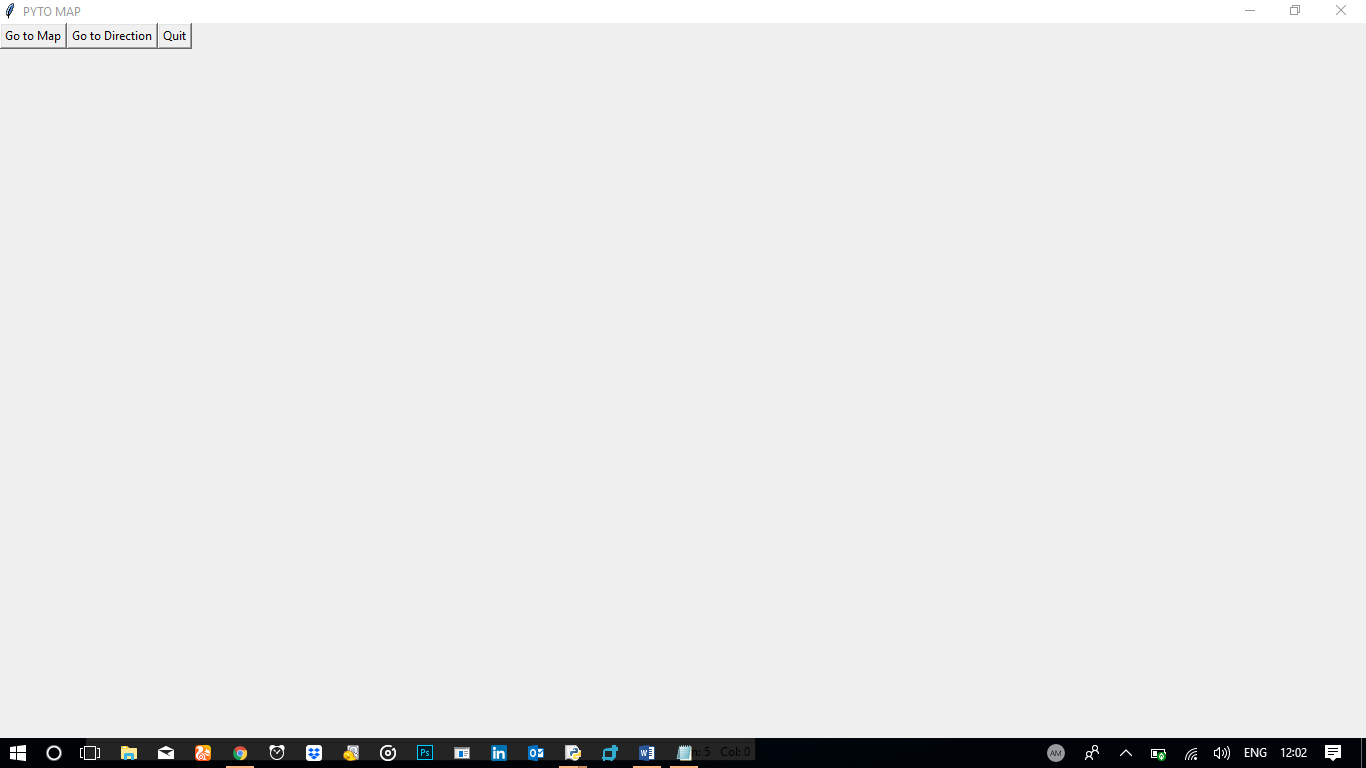
Search\_Button = Button(self, text='Search', command=search)

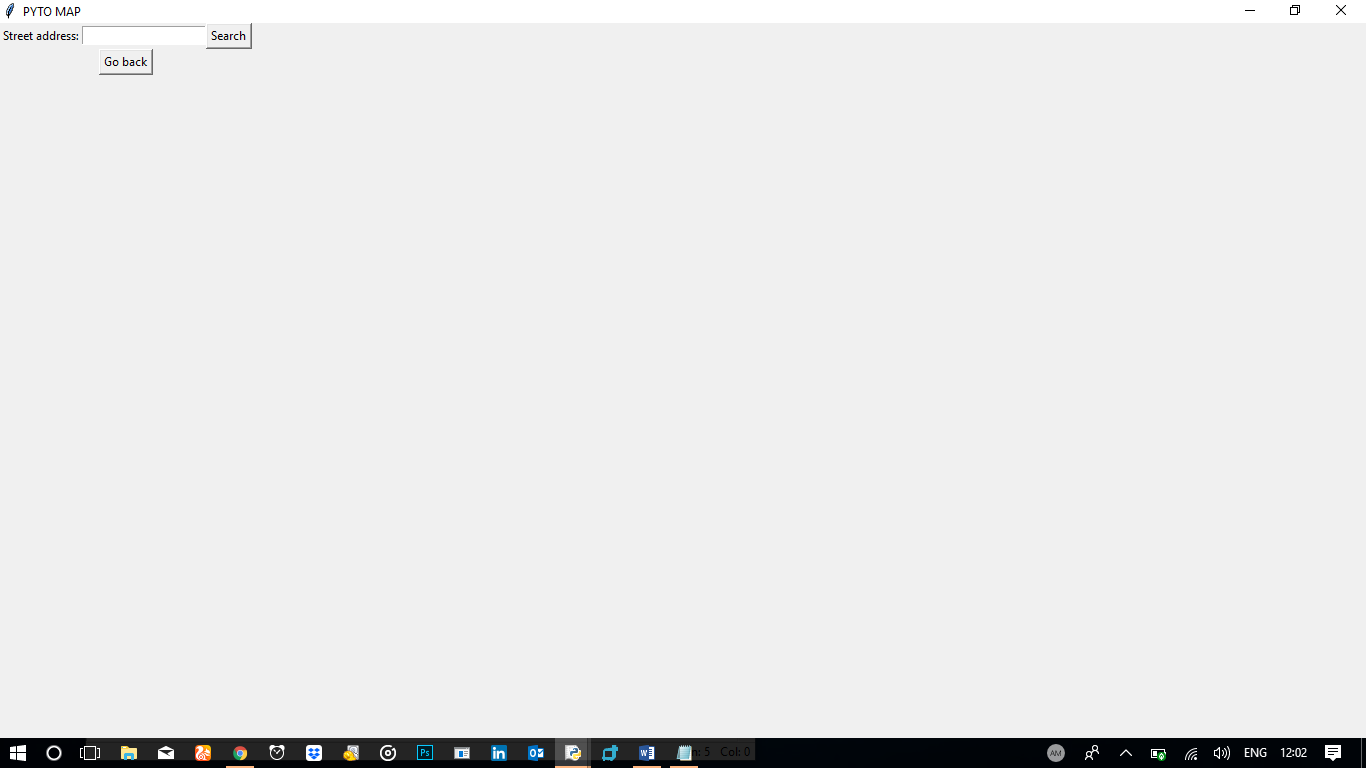
Search\_Button.grid(row=1, column=2)

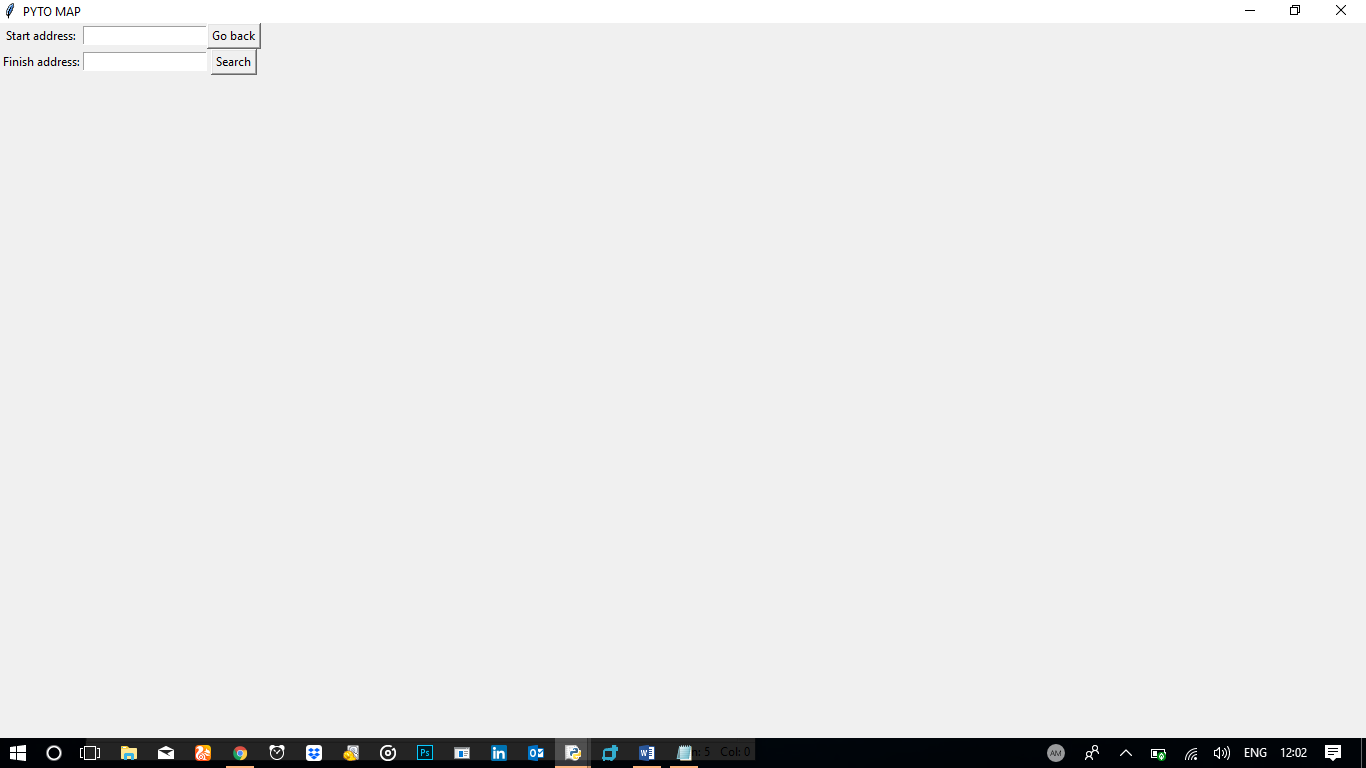
GoogleMap()

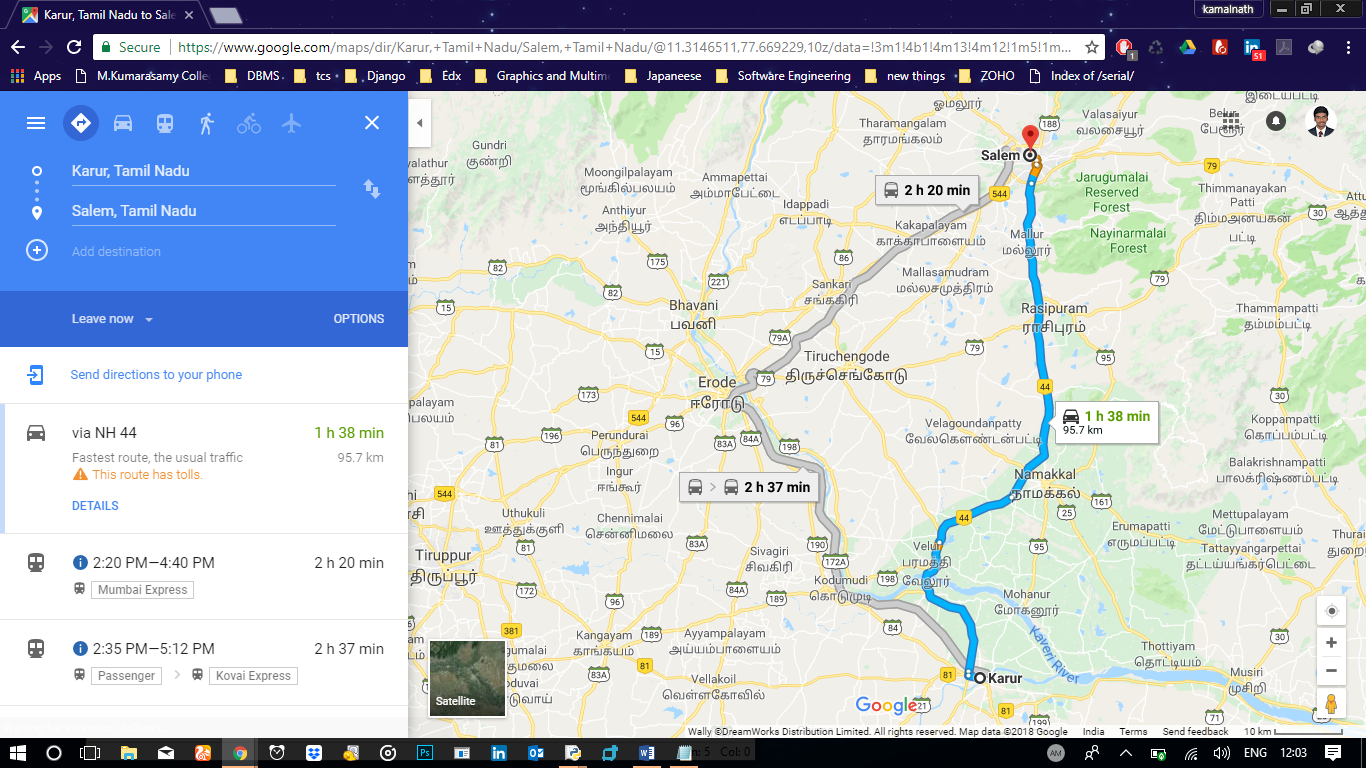
mainloop()

**7.2 Screen shot:**

****

****

****

****