 **AMRITA**

**VIDYALAYAM**

[COMPUTER PROJECT]



NAME- SUBRATA SARKAR

CLASS- XII

SECTION- A

ROLL NO.- 38

**In partial fulfillment of the requirements for the award AISSCE(XI)**

**CERTIFICATE**

** AMRITA**

**VIDYALAYAM**

**Name** SUBRATA SARKAR **Register No**……………….

**Year** 2016-2017 **Class** XII – A – 38

**School** AMRITA VIDYALAYAM

**Subject** COMPUTER SCIENCE

***This is certified to be the bonafide work of the student in this laboratory***

............................ .…….……………………………….. ***Principal Professor/Teacher in charge***

Submitted for the practical examination held in……………………………………... on……………………………………………at…………………………………….

……………………………

...………………………….

Examiners

**CONTENT**

|  |  |  |
| --- | --- | --- |
| **Sl.No.** | **TOPIC** | **PAGE NO.** |
| **1>** | Preface |  |
| **2>** | Acknowledgement |  |
| **3>**  **A)**  **B)**  **C)**  **D)**  **E)**  **F)**  **G)** | Main Report   * Introduction * Objective * Scope * Limitation * Features * Platform * Theoretical Background |  |
| **4>** | Software and Hardware requirement |  |
| **5>** | Literature survey |  |
| **6>** | Input Output screen design |  |
| **7>** | Source Code of the project(summary) |  |
| **8>** | Future Application |  |
| **9>** | Bibliography |  |

**PREFACE**

This project has provided us the opportunity to show the potential of Python. This project has been developed on Python 2.7.9. We are very thankful to our computer sir and the board to provide such an interesting project to work upon. This project has been made in such a way so that even beginners in python can even understand it. The project has been illustrated with screenshots of the program. The interdependency of modules has been illustrated.

**ACKNOWLEDGEMENT**

I Subrata Sarkar of class XII – A, Roll no.:38, wants to acknowledge some precious gems who have contributed greatly in the completion of this project.

First of all I want to thank Mr. Biswajit Dey (our computer science sir) for his able guidance and motivation. Next I want to thanks my group members for helping me. I would also like to thank my parents and friends for giving me advices. Finally, I want to thank the All Mighty God for giving me an opportunity to indulge in such a monumental work.

**MAIN REPORT**

* **Introduction**

Python is an easy to learn yet powerful object oriented very high level programming language. It is a simple language with so many advantages that beginners would learn Python faster than other languages. It uses dynamic type casting which makes it easier for learners to learn. It has high degree of object orientation and refers almost to everything as an object.

This software has been made as a project. The software has been developed in graphical version to make its use easier for the users.

Hope the software helps the users in writing in hindi on PC and makes their everyday work easier .

* **Objective**

The software has been developed to help the everyday users of PC to write in hindi and perform word processing with it, as software of this kind are hard to get in the market free of cost.This software has been made keeping in mind to provide generally needed features for word processing.

* **Scope**

This software has huge scope due to the limited free hindi word processing software which are available free of cost that can be used on PC. The phonetic typepad could be used in many other applications as an input medium. The text-to-speech speech-to-text feature inspire others to develop it more. Also hindi spelling check has been provided.

* **Limitation**

The software has been equipped with as many features as possible but still some limitations remain, as we know even sky is not the limit. Some limitations which could not be addressed for shortage of time include option for changing foreground colour of text and background colour, hyperlinking option, buletting option, indentation option, a better language model for speech recognition and human like voice for speech to text. The users are welcome to send in suggestions for improving the software.

* **Features**

The software has been developed to be run on any device which is supported by Python. The software is designed to work on PCs supporting Python.

Moreover, product feedback has been added so that the user feels like using a real software.

Features of Enigmapad :

* Support changing font,font formatting
* Use of text to speech
* Use of speech to text
* Use of spelling check
* Use of phonetic tyepad
* Support print preview
* Support file save & open
* **Platform**

As we know Python is almost platform independent, so any platform or system able to run Python can run the software.

* **Theoretical Background**

The software upon first run checks the file ‘appenreg.txt’ then writes to the system registry the required information installs ‘vcredist\_x86.exe’, ‘VCForPython27 .msi’ and fonts ‘Siyamrupali.ttf’ , ‘Amarbn\_\_.ttf’. Then aks whether the user wants to take part in the software survey .This gives the sense of using a real software..

**Software and Hardwire requirement**

Software Requirements:

* Operating System: Systems supporting Python(e.g.: Windows>=xp, (Linux(Ubuntu >=12.04),Mac OS>=Xusing windows emulator) etc.)
* Python 2.7.9 or higher[Check [https://docs.python.org](https://docs.python.org/3/) for compatibility issues]
* Pyaudio 0.2.9[Check <http://people.csail.mit.edu/hubert/pyaudio/> for compatibility issues]
* Pocketsphinx 0.0.9[Check <https://github.com/cmusphinx/pocketsphinx-python> for compatibility issues]
* Pyttsx 1.1[Check <https://github.com/parente/pyttsx> for compatibility issues]
* Enchant 1.6[Check <http://www.abisource.com/enchant/> for compatibility issues]
* pyhinavrophonetic 1.0.0[Check <https://bitbucket.org/SubrataSarkar32/pyhinavrophonetic> for compatibility issues]
* pyhinengphonetic 1.0.0[Check <https://bitbucket.org/SubrataSarkar32/pyhinengphonetic> for compatibility issues]
* speech\_recognition 3.4.2[Check [https://github.com/Uberi/speech\_recognition#readme](https://github.com/Uberi/speech_recognition" \l "readme) for compatibility issues]
* PyQt 4 4.11.4[Check <http://www.riverbankcomputing.com/software/pyqt/> for compatibility issues]
* P.I.L 1.1.7 or other [Check <http://effbot.org/media/downloads/> for compatibility issues]

Hardware Requirements:

* Any system able to run Python 2.7.9 release can run this software no other hardware requirement as such.
* For extra features a microphone, a speaker and internet connection(optional) .

**LITERATURE SURVEY**

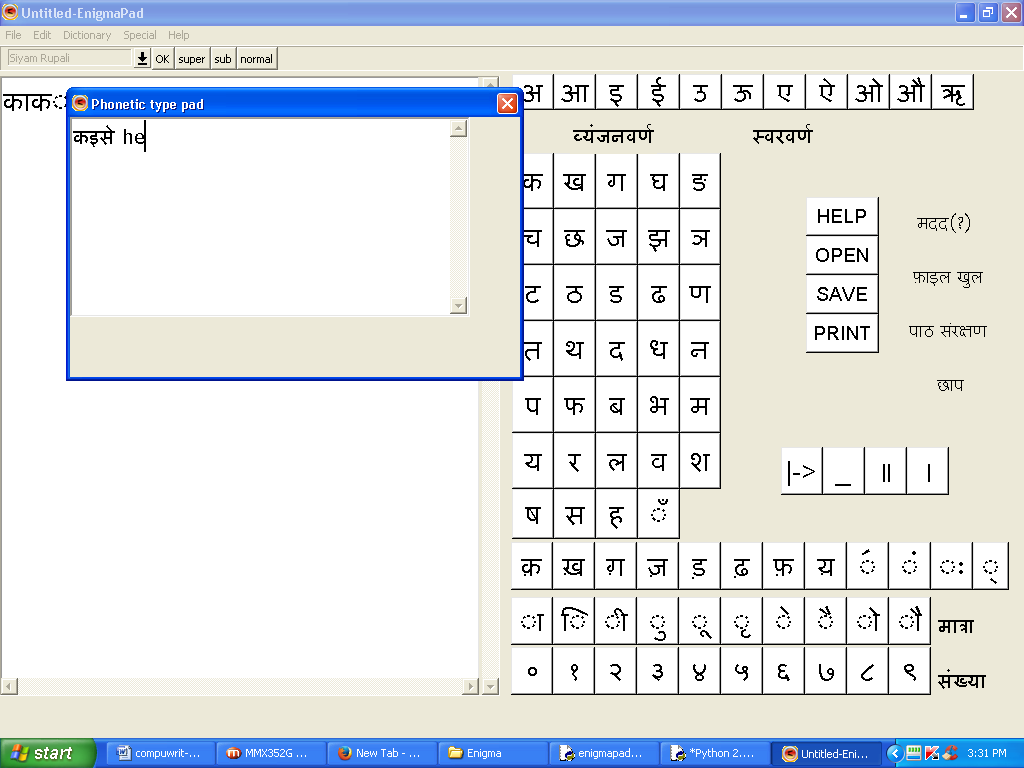
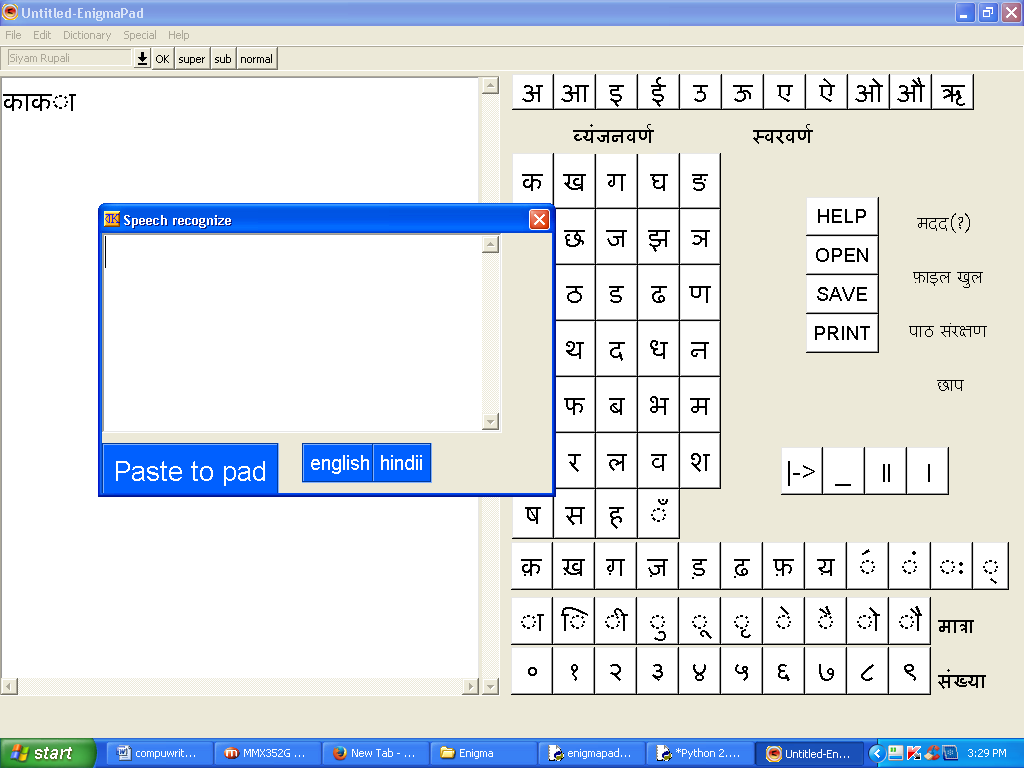
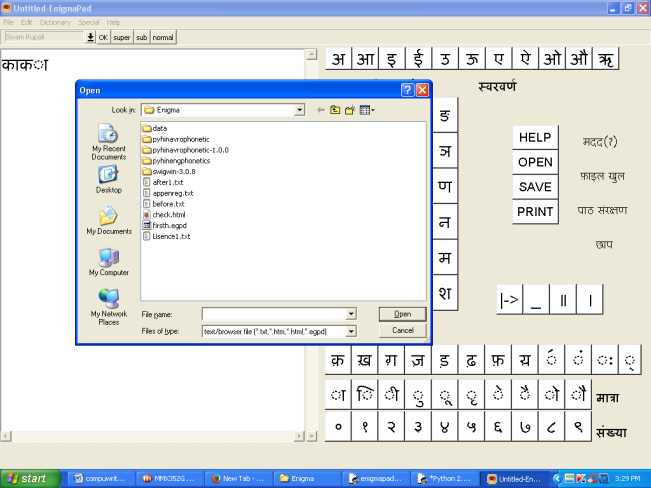
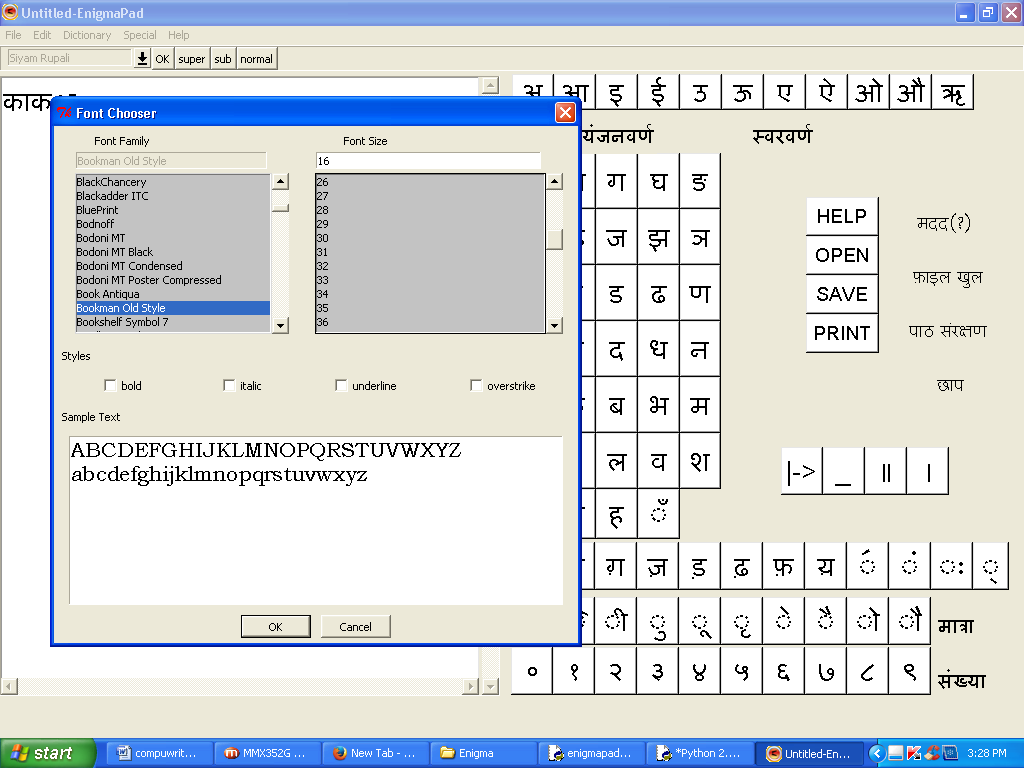
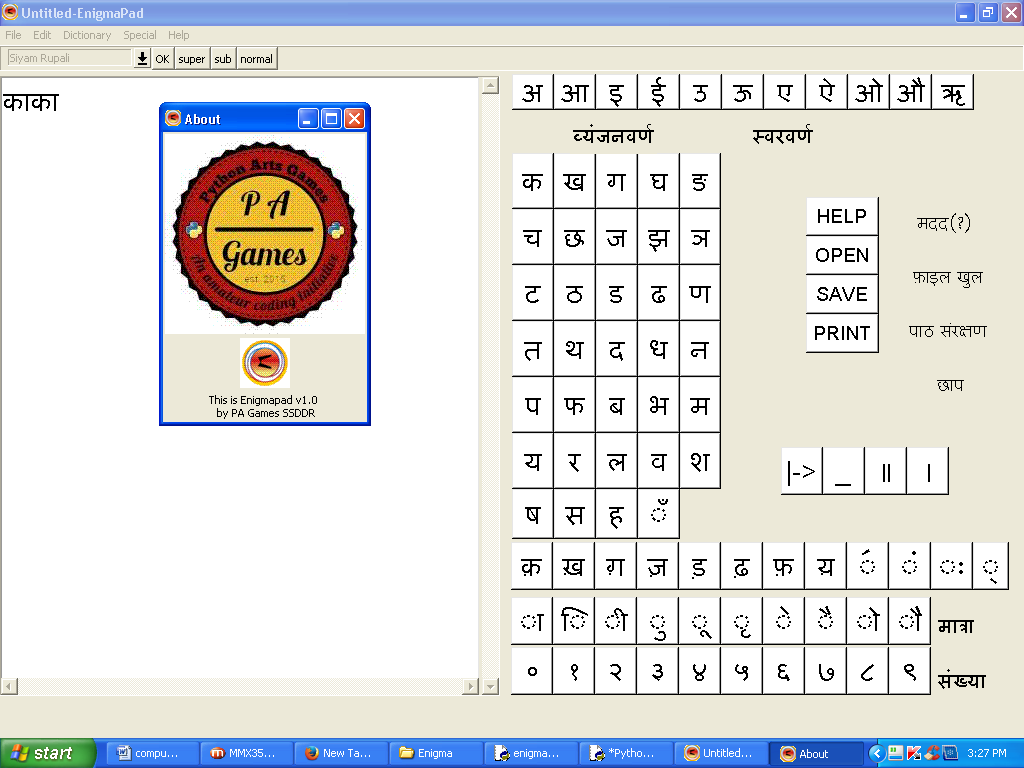
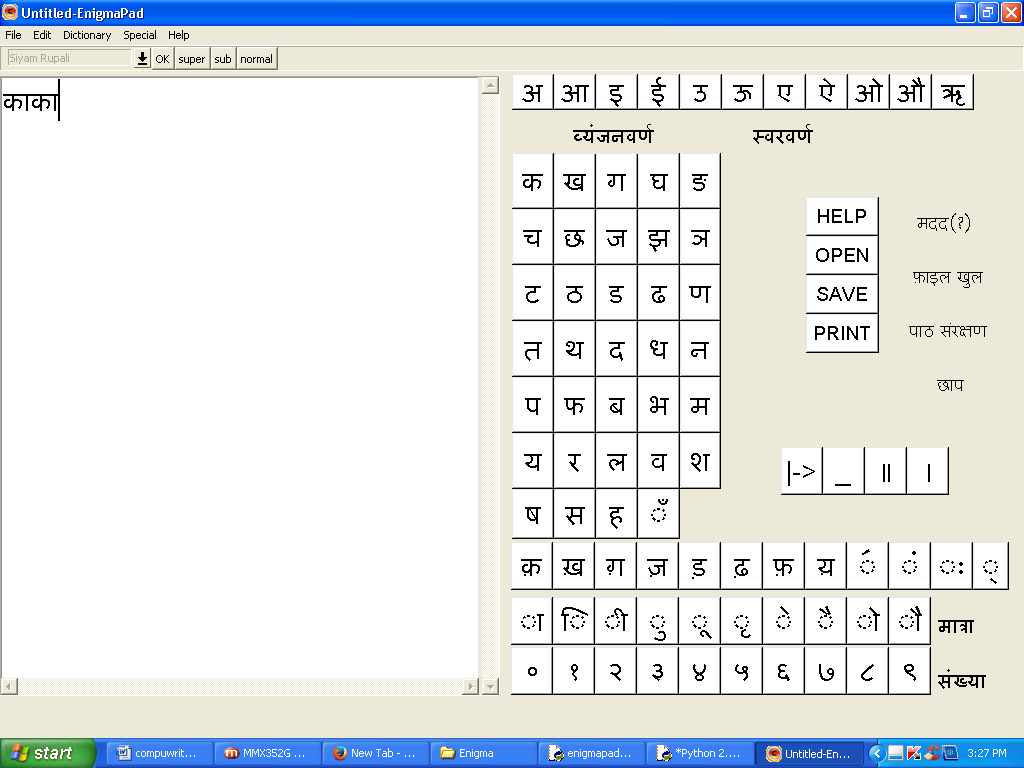
The software has been made in Python programming language. The Python supports this following domains of Operating Systems:

* **Windows:** Microsoft Windows (or simply Windows) is a metafamily of graphical operating systems developed, marketed, and sold by Microsoft. It consists of several families of operating systems, each of which cater to a certain sector of the computing industry. Active Windows families include Windows NT, Windows Embedded and Windows Phone; these may encompass subfamilies, e.g. Windows Embedded Compact (Windows CE) or Windows Server. Defunct Windows families include Windows 9x and Windows Mobile. Microsoft introduced an operating environment named Windows on November 20, 1985, as a graphical operating system shell for MS-DOS in response to the growing interest in graphical user interfaces (GUIs). Microsoft Windows came to dominate the world's personal computer market with over 90% market share, overtaking Mac OS, which had been introduced in 1984. However, since 2012, because of the massive growth of smartphones, Windows sells less than Android, which became the most popular operating system in 2014, when counting all of the computing platforms each operating system runs on; in 2014, the number of Windows devices sold were less than 25% of Android devices sold. However, comparisons across different markets are not fully relevant; and for personal computers, Windows is still the most popular operating system. As of July 2015, the most recent version of Windows for personal computers, tablets and smartphones is Windows 10. The most recent versions for server computers and embedded devices are respectively Windows Server 2012 R2 and Windows Embedded 8. A specialized version of Windows runs on the Xbox One game console.
* **Linux:** Linux is a Unix-like and mostly POSIX-compliant computer operating system (OS) assembled under the model of free and open-source software development and distribution. The defining component of Linux is the Linux kernel, an operating system kernel first released on 5 October 1991 by Linus Torvalds. The Free Software Foundation uses the name GNU/Linux to describe the operating system, which has led to some controversy. Linux was originally developed as a free operating system for personal computers based on the Intel x86 architecture, but has since been ported to more computer hardware platforms than any other operating system. Thanks to its dominance on smartphones, Android, which is built on top of the Linux kernel, has the largest installed base of all general-purpose operating systems. Linux, in its original form, is also the leading operating system on servers and other big iron systems such as mainframe computers and virtually all fastest supercomputers, but is used on only around 1.6% of desktop computers with Linux-based Chrome OS taking about 5% of the overall and nearly 20% of the sub-$300 notebook sales. Linux also runs on embedded systems, which are devices whose operating system is typically built into the firmware and is highly tailored to the system; this includes smartphones and tablet computers running Android and other Linux derivatives, TiVo and similar DVR devices, network routers, facility automation controls, televisions, video game consoles, and smartwatches. The development of Linux is one of the most prominent examples of free and open-source software collaboration. The underlying source code may be used, modified and distributed – commercially or non-commercially – by anyone under the terms of its respective licenses, such as the GNU General Public License. Typically, Linux is packaged in a form known as a Linux distribution, for both desktop and server use. Some of the popular mainstream Linux distributions are Debian, Ubuntu, Linux Mint, Fedora, openSUSE, Arch Linux and Gentoo, together with commercial Red Hat Enterprise Linux and SUSE Linux Enterprise Server distributions. Linux distributions include the Linux kernel, supporting utilities and libraries, and usually a large amount of application software to fulfill the distribution's intended use. Distributions oriented toward desktop use typically include a windowing system, such as X11, Mir or a Wayland implementation, and an accompanying desktop environment, such as GNOME or the KDE Software Compilation; some distributions may also include a less resource-intensive desktop, such as LXDE or Xfce. Distributions intended to run on servers may omit all graphical environments from the standard install, and instead include other software to set up and operate a solution stack such as LAMP. Because Linux is freely redistributable, anyone may create a distribution for any intended use.
* **Mac OS:** Mac OS is a series of graphical user interface–based operating systems developed by Apple Inc. for their Macintosh line of computer systems. The original operating system was first introduced in 1984 as being integral to the original Macintosh, and referred to as the "System". Referred to by its major revision starting with "System 6" and "System 7", Apple rebranded version 7.6 as "Mac OS" as part of their Macintosh clone program in 1996. The Macintosh, specifically its system software, is credited with having popularized the early graphical user interface concept. There are two architectural legacies of Mac OS. Up to major revision 9, from 1984 to 2000, it is historically known as Classic Mac OS. Major revision 10, from 2001 to present, is branded OS X (originally referred to as Mac OS X). Both legacies share a general interface design, and there has been some overlap of application frameworks for compatibility; but the two systems have different origins and use deeply different architectures.

Now some information is being provided on Python and other third-party modules used in our project:

* **Python:** Python is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library. Python interpreters are available for installation on many operating systems, allowing Python code execution on a wide variety of systems. Using third-party tools, such as Py2exe or Pyinstaller, Python code can be packaged into stand-alone executable programs for some of the most popular operating systems, allowing the distribution of Python-based software for use on those environments without requiring the installation of a Python interpreter. CPython, the reference implementation of Python, is free and open-source software and has a community-based development model, as do nearly all of its alternative implementations. CPython is managed by the non-profit Python Software Foundation.
* **P.I.L:** Python Imaging Library (abbreviated as PIL) is a free library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux. The latest version of PIL is 1.1.7, was released in September 2009 and supports Python 1.5.2–2.7, with Python 3 support to be released "later". Development appears to be discontinued with the last commit to the PIL repository coming in 2011. Consequently, a successor project called Pillow has forked the PIL repository and added Python 3.x support. This fork has been adopted as a replacement for the original PIL in Linux distributions including Debian and Ubuntu(since 13.04). Pyaudio 0.2.9[Check <http://people.csail.mit.edu/hubert/pyaudio/> for compatibility issues]
* Pocketsphinx 0.0.9: Helps in speech recognition when offline.
* Pyttsx 1.1: Performs text to speech using service on OS.
* Enchant 1.6: Helps in performing spell-check.
* pyhinavrophonetic 1.0.0: Helps in phonetic conversion of English to hindi
* pyhinengphonetic 1.0.0: Helps in conversion of hindi to English text to be spoken using pyttsx
* speech\_recognition 3.4.2: Helps in speech recognition by providing interface for accessing various speech recognition services
* PyQt 4 4.11.4: Here used for providing print preview window and printing purpose.

**INPUT OUTPUT SCREEN DESIGN**



**SOURCE CODE OF THE PROJECT**

The module interdependency is as follows:

No

**hintextall.py**

If asked for print?’

callreg.py

Yes

httpwidget.py

checkimage.py

Check if first run?

Open ‘appenreg.txt’

No

printinghu.py

Yes

write\_iconreg.py

install ‘vc\_redistx86.exe’

install ‘VCForPython27 .msi’

install font ‘Amarbn\_\_.ttf’

install font ‘Siyamrupali.ttf’

changepathspeech.py

delete changepathspeech.pyc

Ask for survey

* hintextall.py

'''THIS IS A PRODUCT OF PA Games SSDDR.

Enigma Pad1.0 is licensed under the following terms and conditions:

=========================================

1>The projects made by use of this product can be used for commercial purpose

You can use parts of the main code with attribution to PA Games SSDDR.

You may use parts of this program with consent to the makers for free.

The members of PA Games SSDDR include:

1.Subrata Sarkar(<subrotosarkar32@gmail.com>)

2.Sangramjit Chakroborty(<? @gmail.com>)

WEBSITE: pagamesltd.blogspot.com

Other Terms:

--------------

1>This program has been made with Python and is not included under the license of Enigma Pad.

2>Certain modules used are licensed under GPL.Check lisence folder for more detail.'''

######################################################################

import callreg

#Here begins code of font chooser

import Tix

import tkSimpleDialog

import tkFont

class FontChooser( tkSimpleDialog.Dialog ):

'''This class font chooser derived from tkSimpleDialog.Dialog'''

BASIC = 1

ALL = 2

def \_\_init\_\_( self, parent, defaultfont=None, showstyles=None ):

self.\_family = Tix.StringVar( value='Ariel' )

self.\_sizeString = Tix.StringVar( value='12' )

self.\_weight = Tix.StringVar( value=tkFont.NORMAL )

self.\_slant = Tix.StringVar( value=tkFont.ROMAN )

self.\_isUnderline = Tix.BooleanVar( value=False )

self.\_isOverstrike = Tix.BooleanVar( value=False )

if defaultfont:

self.\_initialize( defaultfont )

self.\_currentFont = tkFont.Font( font=self.getFontTuple() )

self.\_showStyles = showstyles

self.sampleText = None

tkSimpleDialog.Dialog.\_\_init\_\_( self, parent, 'Font Chooser' )

def \_initialize( self, aFont ):

'''initialize font options'''

if not isinstance( aFont, tkFont.Font ):

aFont = tkFont.Font( font=aFont )

fontOpts = aFont.actual( )

self.\_family.set( fontOpts[ 'family' ] )

self.\_sizeString.set( fontOpts[ 'size' ] )

self.\_weight.set( fontOpts[ 'weight' ] )

self.\_slant.set( fontOpts[ 'slant' ] )

self.\_isUnderline.set( fontOpts[ 'underline' ] )

self.\_isOverstrike.set( fontOpts[ 'overstrike' ] )

def body( self, master ):

'''The body of the font chooser'''

theRow = 0

Tix.Label( master, text="Font Family" ).grid( row=theRow, column=0 )

Tix.Label( master, text="Font Size" ).grid( row=theRow, column=2 )

theRow += 1

# Font Families

fontList = Tix.ComboBox( master, command=self.selectionChanged, dropdown=False, editable=False, selectmode=Tix.IMMEDIATE, variable=self.\_family )

fontList.grid( row=theRow, column=0, columnspan=2, sticky=Tix.N+Tix.S+Tix.E+Tix.W, padx=10 )

first = None

familyList = list(tkFont.families( ))

familyList.sort()

for family in familyList:

if family[0] == '@':

continue

if first is None:

first = family

fontList.insert( Tix.END, family )

fontList.configure( value=first )

# Font Sizes

sizeList = Tix.ComboBox( master, command=self.selectionChanged, dropdown=False, editable=True, selectmode=Tix.IMMEDIATE, variable=self.\_sizeString )

sizeList.grid( row=theRow, column=2, columnspan=2, sticky=Tix.N+Tix.S+Tix.E+Tix.W, padx=10 )

for size in xrange( 6,72 ):

sizeList.insert( Tix.END, '%d' % size )

sizeList.configure( value='9' )

# Styles

if self.\_showStyles is not None:

theRow += 1

if self.\_showStyles in ( FontChooser.ALL, FontChooser.BASIC ):

Tix.Label( master, text='Styles', anchor=Tix.W ).grid( row=theRow, column=0, pady=10, sticky=Tix.W )

theRow += 1

Tix.Checkbutton( master, text="bold", command=self.selectionChanged, offvalue='normal', onvalue='bold', variable=self.\_weight ).grid(row=theRow, column=0)

Tix.Checkbutton( master, text="italic", command=self.selectionChanged, offvalue='roman', onvalue='italic', variable=self.\_slant ).grid(row=theRow, column=1)

if self.\_showStyles == FontChooser.ALL:

Tix.Checkbutton( master, text="underline", command=self.selectionChanged, offvalue=False, onvalue=True, variable=self.\_isUnderline ).grid(row=theRow, column=2)

Tix.Checkbutton( master, text="overstrike", command=self.selectionChanged, offvalue=False, onvalue=True, variable=self.\_isOverstrike ).grid(row=theRow, column=3)

# Sample Text

theRow += 1

Tix.Label( master, text='Sample Text', anchor=Tix.W ).grid( row=theRow, column=0, pady=10, sticky=Tix.W )

theRow += 1

self.sampleText = Tix.Text( master, height=11, width=70 )

self.sampleText.insert( Tix.INSERT,

'ABCDEFGHIJKLMNOPQRSTUVWXYZ\nabcdefghijklmnopqrstuvwxyz', 'fontStyle' )

self.sampleText.config( state=Tix.DISABLED )

self.sampleText.tag\_config( 'fontStyle', font=self.\_currentFont )

self.sampleText.grid( row=theRow, column=0, columnspan=4, padx=10 )

def apply( self ):

self.result = self.getFontTuple( )

def selectionChanged( self, something=None ):

self.\_currentFont.configure( family=self.\_family.get(), size=self.\_sizeString.get(),

weight=self.\_weight.get(), slant=self.\_slant.get(),

underline=self.\_isUnderline.get(),

overstrike=self.\_isOverstrike.get() )

if self.sampleText:

self.sampleText.tag\_config( 'fontStyle', font=self.\_currentFont )

def getFontTuple( self ):

family = self.\_family.get()

size = int(self.\_sizeString.get())

styleList = [ ]

if self.\_weight.get() == tkFont.BOLD:

styleList.append( 'bold' )

if self.\_slant.get() == tkFont.ITALIC:

styleList.append( 'italic' )

if self.\_isUnderline.get():

styleList.append( 'underline' )

if self.\_isOverstrike.get():

styleList.append( 'overstrike' )

if len(styleList) == 0:

return family, size

else:

return family, size, ' '.join( styleList )

def askChooseFont( parent, defaultfont=("siyam rupali",20), showstyles=FontChooser.ALL ):

return FontChooser( parent, defaultfont=defaultfont, showstyles=showstyles ).result

import sys

k=sys.argv[1:]

try:

if '.egpd' in k[-1] or '.html' in k[-1] or '.htm' in k[-1] or '.txt' in k[-1]:

f=' '.join(k)

else:

f=' '.join(k)

except IndexError:

f=''

import Tkinter

from Tkconstants import \*

import tkMessageBox

import io

from tkFileDialog import askopenfilename , asksaveasfile ,asksaveasfilename

import \_tkinter # If this fails your Python may not be configured for Tk

tkinter = \_tkinter # b/w compat for export

TclError = \_tkinter.TclError

from pyhinavrophonetic import hinavro

import enchant

d = enchant.Dict("hi\_IN.dic")#currently using hindi dictionary

import checkimage

banglakey=Tix.Tk()

fonti=("Siyam Rupali",20)

banglakey.minsize(600,600)

banglakey.iconbitmap('enigmapad3.ico')

image=[]#stores image data

fontf=''

ioftag=0

tagno=0

def Font():

if text1.tag\_ranges('sel'):

global ioftag

ioftag+=1

fontf=askChooseFont(banglakey)

tags=text1.tag\_names(Tkinter.SEL\_FIRST)

if tags[1][0]=='t':

tag=tags[1]

fonte=return\_font\_list(text1.tag\_cget(tag,'font'))

text1.tag\_configure("tags\_"+str(ioftag), font=fontf)

text1.tag\_remove(tag,Tkinter.SEL\_FIRST, Tkinter.SEL\_LAST)

text1.tag\_add("tags\_"+str(ioftag),Tkinter.SEL\_FIRST, Tkinter.SEL\_LAST)

else:

global fonti

font=askChooseFont(banglakey)

if fonti!=font:

if font!=None:

global tagno

tagno+=1

text1.tag\_configure("ttags\_"+str(tagno), font=font)

fonti=font

fontList.configure( value=fonti[0] )

addword13(font=fonti)

menu = Tkinter.Menu(banglakey)

banglakey.config(menu=menu)

def read\_html(text=''):

#@bama give your code to read html file here. on opening a file it is invoked from function u78

try:

st=text.find('<body>')

ed=text.find('</body>')

text=text[st+6:ed]

text=text.split('>')

text.remove(u'<p')

font=tuple()

global tagno

global ioftag

global image

from copy import deepcopy

for httex in text:

if '''<img alt="Embedded Image" src="data:image/gif;base64,''' in httex:

indeximg=httex.find('''<img alt="Embedded Image" src="data:image/gif;base64,''')

klist=httex.split('''<img alt="Embedded Image" src="data:image/gif;base64,''',1)

httex=klist[0]

kopo=''''''

kopo+=klist[1]

image+=[kopo]

MyPhotoImage = Tkinter.PhotoImage(data=klist[1])

text1.image\_create(Tkinter.INSERT, image=MyPhotoImage)

text1.imglist+=[MyPhotoImage]

if httex!='':

if httex[:5]==u'<font':

httex=httex.split('"')

tagno+=1

font=(str(httex[1]),str(httex[3]))

text1.tag\_configure("ttags\_"+str(tagno), font=font)

elif httex[0]!=u'<':

if httex[-4:]==u'<sub':

text1.insert(INSERT,httex[:-4],"ttags\_"+str(tagno))

ioftag+=1

text1.tag\_configure("tags\_"+str(ioftag), font=font,offset=-6)

elif httex[-5:]==u'</sub':

text1.insert(INSERT,httex[:-5],"tags\_"+str(ioftag))

elif httex[-5:]==u'<font':

text1.insert(INSERT,httex[:-5],"tags\_"+str(ioftag))

httex=httex.split('"')

tagno+=1

font=(str(httex[-4]),str(httex[-2]))

text1.tag\_configure("ttags\_"+str(tagno), font=font)

elif httex[-4:]==u'<sup':

text1.insert(INSERT,httex[:-4],"ttags\_"+str(tagno))

ioftag+=1

text1.tag\_configure("tags\_"+str(ioftag), font=font,offset=+6)

elif httex[-5:]==u'</sup':

text1.insert(INSERT,httex[:-5],"tags\_"+str(ioftag))

elif httex[-3:]==u'</p':

text1.insert(INSERT,httex[:-3],"ttags\_"+str(tagno))

text1.insert(INSERT,u'\n',"ttags\_"+str(tagno))

elif httex[-2:]==u'<i':

text1.insert(INSERT,httex[:-2],"ttags\_"+str(tagno))

tagno+=1

font+=('italic',)

text1.tag\_configure("ttags\_"+str(tagno), font=font)

elif httex[-3:]==u'</i':

text1.insert(INSERT,httex[:-3],"ttags\_"+str(tagno))

uiop=(str(font[0]),str(font[1]))

font=deepcopy(uiop)

tagno+=1

text1.tag\_configure("ttags\_"+str(tagno), font=font)

elif httex[-2:]==u'<b':

text1.insert(INSERT,httex[:-2],"ttags\_"+str(tagno))

tagno+=1

font+=('bold',)

text1.tag\_configure("ttags\_"+str(tagno), font=font)

elif httex[-3:]==u'</b':

text1.insert(INSERT,httex[:-3],"ttags\_"+str(tagno))

uiop=(str(font[0]),str(font[1]))

font=deepcopy(uiop)

tagno+=1

text1.tag\_configure("ttags\_"+str(tagno), font=font)

elif httex[-2:]==u'<u':

text1.insert(INSERT,httex[:-2],"ttags\_"+str(tagno))

tagno+=1

font+=('underline',)

text1.tag\_configure("ttags\_"+str(tagno), font=font)

elif httex[-3:]==u'</u':

text1.insert(INSERT,httex[:-3],"ttags\_"+str(tagno))

uiop=(str(font[0]),str(font[1]))

font=deepcopy(uiop)

tagno+=1

text1.tag\_configure("ttags\_"+str(tagno), font=font)

elif httex[-8:]==u'<strike':

text1.insert(INSERT,httex[:-8],"ttags\_"+str(tagno))

tagno+=1

font+=('overstrike',)

text1.tag\_configure("ttags\_"+str(tagno), font=font)

elif httex[-8:]==u'</strike':

text1.insert(INSERT,httex[:-8],"ttags\_"+str(tagno))

uiop=(str(font[0]),str(font[1]))

font=deepcopy(uiop)

tagno+=1

text1.tag\_configure("ttags\_"+str(tagno), font=font)

elif httex[-6:]==u'</font':

text1.insert(INSERT,httex[:-6],"ttags\_"+str(tagno))

pass

else:

text1.insert(INSERT,httex[:-5],"ttags\_"+str(tagno))

elif httex==u'</p':

text1.insert(INSERT,u'\n',"ttags\_"+str(tagno))

if httex[-4:]==u'<sub':

ioftag+=1

text1.tag\_configure("tags\_"+str(ioftag), font=font,offset=-6)

elif httex[-5:]==u'</sub':

uiop=(str(font[0]),str(font[1]))

font=deepcopy(uiop)

tagno+=1

text1.tag\_configure("ttags\_"+str(tagno), font=font)

elif httex[-4:]==u'<sup':

ioftag+=1

text1.tag\_configure("tags\_"+str(ioftag), font=font,offset=+6)

elif httex[-5:]==u'</sup':

uiop=(str(font[0]),str(font[1]))

font=deepcopy(uiop)

tagno+=1

text1.tag\_configure("ttags\_"+str(tagno), font=font)

elif httex[-2:]==u'<i':

tagno+=1

font+=('italic',)

text1.tag\_configure("ttags\_"+str(tagno), font=font)

elif httex[-3:]==u'</i':

uiop=(str(font[0]),str(font[1]))

font=deepcopy(uiop)

tagno+=1

text1.tag\_configure("ttags\_"+str(tagno), font=font)

elif httex[-2:]==u'<b':

tagno+=1

font+=('bold',)

text1.tag\_configure("ttags\_"+str(tagno), font=font)

elif httex[-3:]==u'</b':

uiop=(str(font[0]),str(font[1]))

font=deepcopy(uiop)

tagno+=1

text1.tag\_configure("ttags\_"+str(tagno), font=font)

elif httex[-2:]==u'<u':

tagno+=1

font+=('underline',)

text1.tag\_configure("ttags\_"+str(tagno), font=font)

elif httex[-3:]==u'</u':

uiop=(str(font[0]),str(font[1]))

font=deepcopy(uiop)

tagno+=1

text1.tag\_configure("ttags\_"+str(tagno), font=font)

elif httex[-8:]==u'<strike':

tagno+=1

font+=('overstrike',)

text1.tag\_configure("ttags\_"+str(tagno), font=font)

elif httex[-8:]==u'</strike':

uiop=(str(font[0]),str(font[1]))

font=deepcopy(uiop)

tagno+=1

text1.tag\_configure("ttags\_"+str(tagno), font=font)

elif httex[-6:]==u'</font':

pass

except:

tkMessageBox.showinfo('Alert!','Invalid .bkpd syntax .Opening with PyQt4')

if f.split('.')[-1]=='html' or f.split('.')[-1]=='htm':

import printinghu

printinghu.call(f)

NewFile()

def u78(event=None):

global f

f= askopenfilename(defaultextension=".egpd",filetypes=[("text/browser file", ("\*.txt","\*.htm","\*.html","\*.egpd")),('all Files','\*.\*')])

import io

global tagno

global ioftag

global images

tagno=0

ioftag=0

images=[]

text1.imglist=[]

if f!='':

if f.split('.')[-1]=='html' or f.split('.')[-1]=='htm' or f.split('.')[-1]=='egpd':

with io.open(f,'r',encoding='utf8') as opf:

text11 = opf.read()

read\_html(text11)

banglakey.title(str(f)+'-EnigmaPad')

elif f.split('.')[-1]=='txt':

with io.open(f,'r',encoding='utf8') as opf:

text11 = opf.read()

text1.insert(INSERT,text11,"ttags\_"+str(tagno))

banglakey.title(str(f)+'-EnigmaPad')

else:

tkMessageBox.showinfo('Error!','Specified format not supported '+f.split('.')[-1])

f=''

banglakey.title('Untitled-EnigmaPad')

def texttohtml(event=None):

textret=''

try:

text1.tag\_remove("sel", "sel.first", "sel.last")

except Exception:

pass

try:

text1.tag\_delete("misspelled")

except Exception:

pass

global image

imgcount1=0

imgcount2=len(image)

#displays tags along with associated text and offset

tags=text1.tag\_names(index=None)

pair1=[]

pair2=[]

for tag in tags:

ranges = text1.tag\_ranges(tag)

for i in range(0, len(ranges), 2):

start = ranges[i]

stop = ranges[i+1]

pair1+=[str(start)]

pair2+=[str(stop)]

if "%s-1c" %stop==start:

pass

if text1.get(start)=='':

for i in range(len(text1.imglist)):

pair1+=[text1.index(text1.imglist[i])]

pair2+=[text1.index("%s+1c"%text1.imglist[i])]

pair1.sort(key=lambda x: [x.split('.')[0], int(x.split('.')[1])])

pair2.sort(key=lambda x: [x.split('.')[0], int(x.split('.')[1])])

for pp1 in range(len(pair1)):

tags=text1.tag\_names(pair1[pp1])

if text1.index(pair2[pp1])==text1.index("%s+1c" %pair1[pp1]):

if text1.get(pair1[pp1])==''and imgcount1<imgcount2:

textret+='''<img alt="Embedded Image" src="data:image/gif;base64,'''+image[imgcount1]+'''" />'''

imgcount1+=1

else:

if tags[0][0]=='t':

tag=tags[0]

start=pair1[pp1]

stop=pair2[pp1]

if text1.tag\_cget(tag,'offset')=='':

fontlist=return\_font\_list(text1.tag\_cget(tag,'font'))

if fontlist[0][0]=='{':

fontlist[0]=fontlist[0][1:-1]

textret+='<font face="'+fontlist[0]+'" size="'+fontlist[1]+'">'

for checky in fontlist:

if 'bold' in checky:

textret+='<b>'

elif 'underline' in checky:

textret+='<u>'

elif 'italic' in checky:

textret+='<i>'

elif 'overstrike' in checky:

textret+='<strike>'

else:

pass

for letter in text1.get(start, stop):

if letter=='\n':

textret+='</p><p>'

else:

textret+=letter

for checky in fontlist:

if 'bold' in checky:

textret+='</b>'

elif 'underline' in checky:

textret+='</u>'

elif 'italic' in checky:

textret+='</i>'

elif 'overstrike' in checky:

textret+='</strike>'

else:

pass

textret+='</font>'

elif text1.tag\_cget(tag,'offset')=='6':

fontlist=return\_font\_list(text1.tag\_cget(tag,'font'))

if fontlist[0][0]=='{':

fontlist[0]=fontlist[0][1:-1]

textret+='<font face="'+fontlist[0]+'" size="'+fontlist[1]+'">'

textret+='<sup>'

for letter in text1.get(start, stop):

if letter=='\n':

textret+='</p><p>'

else:

textret+=letter

textret+='</sup></font>'

elif text1.tag\_cget(tag,'offset')=='-6':

fontlist=return\_font\_list(text1.tag\_cget(tag,'font'))

if fontlist[0][0]=='{':

fontlist[0]=fontlist[0][1:-1]

textret+='<font face="'+fontlist[0]+'" size="'+fontlist[1]+'">'

textret+='<sub>'

for letter in text1.get(start, stop):

if letter=='\n':

textret+='</p><p>'

else:

textret+=letter

textret+='</sub></font>'

return textret

def NewFile():

'''Opens up a new file after providing option for saving if file name variable f is blank

else if file name is give it saves and and then open's a new file'''

import io

global f

if f=='':

tkMessageBox.showinfo('Alert','You will now be provided option for saving your current file')

f = asksaveasfilename(defaultextension=".egpd")

if f =='':

f=''

text1.delete(1.0,END)

else:

text2save = text1.get(1.0, END)

prev='''<html>

<head>

<meta http-equiv="Content-Language" content="bn">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>'''+str(f)+'''</title>

</head>

<body><p><font face="Siyam Rupali" size="10">'''

afte='''</font></p>

</body>

</html>'''

with io.open(f,'w',encoding='utf8') as opf:

opf.write(prev+texttohtml()+afte)

opf.close()

f=''

text1.delete(1.0,END)

tagno=0

ioftag=0

images=[]

text1.imglist=[]

else:

prev='''<html>

<head>

<meta http-equiv="Content-Language" content="hn">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>'''+str(f)+'''</title>

</head>

<body><p><font face="Siyam Rupali" size="10">'''

afte='''</font></p>

</body>

</html>'''

with io.open(f,'w',encoding='utf8') as opf:

opf.write(prev+texttohtml()+afte)

opf.close()

f=''

text1.delete(1.0,END)

tagno=0

ioftag=0

images=[]

text1.imglist=[]

def u79(event=None):

'''This function calls if user choose to save from file menu or save button'''

import io

global f

if f=='':

tkMessageBox.showinfo('Alert','You will now be provided option for saving your current file')

f = asksaveasfilename(defaultextension=".html")

if f !='':

prev='''<html>

<head>

<meta http-equiv="Content-Language" content="hn">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>'''+str(f)+'''</title>

</head>

<body><p>'''

afte='''</p>

</body>

</html>'''

with io.open(f,'w',encoding='utf8') as opf:

opf.write(prev+texttohtml()+afte)

opf.close()

banglakey.title(str(f)+'-EnigmaPad')

if f !='':

prev='''<html>

<head>

<meta http-equiv="Content-Language" content="hn">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>'''+str(f)+'''</title>

</head>

<body><p>'''

afte='''</p>

</body>

</html>'''

with io.open(f,'w',encoding='utf8') as opf:

opf.write(prev+texttohtml()+afte)

opf.close()

def helpe(event=None):

'''This functions shows a message box having the below information'''

tkMessageBox.showinfo('Help','1.Enter hindi text using the buttons provided. \

\n2.Enter punctuation and special symbols using keyboard.\

\n3.Ref must be given before enter byanjan barna. \

\n4.Enter english text using keyboard.\

\n5.To copy use Ctrl+c , to cut use Ctrl+x ,to paste use Ctrl+v,to undo use Ctrl+z.\

\n6.Undo works back until it encounters the last key pressed on keyboard.\

\n7.To enter byanjan barna enter first give hasnta then next.\

\n8.Select a word then under edit menu select add word to dict to include the word.\

\n9.Select a word then under edit menu select delete word from dict to exclude the word.\

\n10.To enter list of words to dictionary enter list of words seperated by ","\

Example "প্রকাশ,উন্মুক্ত"\

\n11.Use speech recognition for speech-to-textpeak and using Google(online)\

Sphinx(offline)& pyhinavrophonetic.\

\n12.Use phonetic keyboard to type in english and convert to hindi using pyhinavrophonetic.')

def quitee(event=None):

'''This function is called if the user calls exit from file menu.It asks for saving and then quits'''

import io

global f

if f=='':

tkMessageBox.showinfo('Alert','You will now be provided option for saving your current file')

f = asksaveasfilename(defaultextension=".egpd")

if f !='':

text2save = text1.get(1.0, END)

prev='''<html>

<head>

<meta http-equiv="Content-Language" content="hn">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>'''+str(f)+'''</title>

</head>

<body><p>'''

afte='''</p>

</body>

</html>'''

with io.open(f,'w',encoding='utf8') as opf:

opf.write(prev+texttohtml()+afte)

opf.close()

banglakey.destroy()

elif f !='':

s=tkMessageBox.askyesno( 'Quit?',"Would you like to save the file before quiting?")

if s== True:

prev='''<html>

<head>

<meta http-equiv="Content-Language" content="hn">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>'''+str(f)+'''</title>

</head>

<body><p>'''

afte='''</p>

</body>

</html>'''

with io.open(f,'w',encoding='utf8') as opf:

opf.write(prev+texttohtml()+afte)

opf.close()

else:

opf.close()

banglakey.destroy()

def Spellcheck( event=None):

'''This function is called on selecting speel check.It spell checks the whole document'''

text\_edit=text1.get(1.0,END).split()

index=1.0

for i in range (len(text\_edit)):

word=text\_edit[i]

if d.check(word):

text1.tag\_remove("misspelled", index, "%s+%dc" % (index, len(word)))

else:

text1.tag\_add("misspelled", index, "%s+%dc" % (index, len(word)))

if i<len(text\_edit)-1:

index=text1.search(text\_edit[i+1],index,forwards=True,regexp=None)

def Spellchecksing( event=None):

'''This function is called on selecting speel check.It spell checks the the selected text'''

text\_edit=text1.get(Tkinter.SEL\_FIRST, Tkinter.SEL\_LAST).split()

index=Tkinter.SEL\_FIRST

for i in range (len(text\_edit)):

word=text\_edit[i]

if d.check(word):

text1.tag\_remove("misspelled", index, "%s+%dc" % (index, len(word)))

else:

text1.tag\_add("misspelled", index, "%s+%dc" % (index, len(word)))

if i<len(text\_edit)-1:

index=text1.search(text\_edit[i+1],index,forwards=True,regexp=None)

def addword3(event=None):

'''This function opens the window for phonetic typepad .Dibyendu's code to be added'''

banglakey.adlayer1=Tkinter.Toplevel()

banglakey.adlayer1.title('Phonetic type pad')

banglakey.adlayer1.iconbitmap('enigmapad3.ico')

banglakey.adlayer1.transient(banglakey)

banglakey.adlayer1.minsize(width=450,height=260)

from pyhinavrophonetic import hinavro

banglakey.adlayer1.text\_frame13=Tkinter.Frame(banglakey.adlayer1,borderwidth=1, relief="sunken")

banglakey.adlayer1.text13 =Tkinter.Text(banglakey.adlayer1,wrap="word",background="white",undo=True,font=("siyam rupali",15)) banglakey.adlayer1.scroll3=Tkinter.Scrollbar(banglakey.adlayer1,orient="vertical",relief=FLAT,width=20,command=banglakey.adlayer1.text13.yview)

banglakey.adlayer1.text13.config(yscrollcommand=banglakey.adlayer1.scroll3.set)

banglakey.adlayer1.scroll3.pack(in\_=banglakey.adlayer1.text13,side="right", fill="y", expand=False)

banglakey.adlayer1.text13.pack(in\_=banglakey.adlayer1.text\_frame13, side="left", fill="both", expand=True)

banglakey.adlayer1.text13.focus()

banglakey.adlayer1.text\_frame13.place(x=0,y=0,height=200, width=400)

def addword13(g,event=None):

v1=banglakey.adlayer1.text13.get(1.0, INSERT)

if g==' ':

banglakey.adlayer1.text13.delete(1.0,INSERT)

banglakey.adlayer1.text13.insert(INSERT, hinavro.parse(v1))

banglakey.adlayer1.text13.bind('<Key>',lambda event: addword13(event.char))

banglakey.adlayer1.mainloop()

def suggest(event=None):

'''This function opens the window and returns the suggested words in label'''

try:

word1=text1.get(Tkinter.SEL\_FIRST, Tkinter.SEL\_LAST)

indexingwrd=text1.index(Tkinter.SEL\_FIRST)

from copy import deepcopy

word2nd=deepcopy(word1)

if d.check(word1)==False:

k=d.suggest(word1)

banglakey.gh=Tkinter.Toplevel()

banglakey.gh.title('Suggestion')

banglakey.gh.iconbitmap('enigmapad3.ico')

def word\_change(wrrdgot):

'''inserts the word'''

text1.delete(Tkinter.SEL\_FIRST, Tkinter.SEL\_LAST)

text1.insert(indexingwrd, wrrdgot,"ttags\_"+str(tagno))

banglakey.gh.destroy()

#font chooser in mainwindow

banglakey.gh.suggestList = Tix.ComboBox( banglakey.gh, command= word\_change, dropdown=True, editable=False, selectmode=Tix.IMMEDIATE ,variable=word1)

wordList = k

wordList.sort()

for juinum in range(len(k)):

banglakey.gh.suggestList.insert(Tix.END,k[juinum])

banglakey.gh.suggestList.pack()

banglakey.gh.mainloop()

except Exception as e:

pass

def About():

'''This says aout our grourp sowing logo'''

banglakey.about=Tkinter.Toplevel()

banglakey.about.title('About')

banglakey.about.iconbitmap('enigmapad3.ico')

label18=Tkinter.Label(banglakey.about,text='This is Enigmapad v1.0 \n by PA Games SSDDR')

img0 = Tkinter.PhotoImage(file='data/pagames.GIF')

img9 = Tkinter.PhotoImage(file='data/enigma1.GIF')

Label6 = Tkinter.Label(banglakey.about, image = img0)

Label6.pack(side=Tkinter.TOP)

L6 = Tkinter.Label(banglakey.about, image = img9)

L6.pack(side=Tkinter.TOP)

label18.pack(side=Tkinter.TOP)

banglakey.about.mainloop()

def recognize():

banglakey.speech=Tkinter.Toplevel()

banglakey.speech.title('Speech recognize')

banglakey.speech.iconbitmap('data/banglakey1.ico')

banglakey.speech.transient(banglakey)

banglakey.speech.minsize(width=450,height=260)

banglakey.speech.text\_frame12=Tkinter.Frame(banglakey.speech,borderwidth=1, relief="sunken")

banglakey.speech.text12 =Tkinter.Text(banglakey.speech,wrap="word",background="white",undo=True,font=("siyam rupali",15))

banglakey.speech.scroll2=Tkinter.Scrollbar(banglakey.speech,orient="vertical",relief=FLAT,width=20,command=banglakey.speech.text12.yview)

banglakey.speech.text12.config(yscrollcommand=banglakey.speech.scroll2.set)

banglakey.speech.scroll2.pack(in\_=banglakey.speech.text12,side="right", fill="y", expand=False)

banglakey.speech.text12.pack(in\_=banglakey.speech.text\_frame12, side="left", fill="both", expand=True)

banglakey.speech.text12.focus()

banglakey.speech.text\_frame12.place(x=0,y=0,height=200, width=400)

v=''

def text\_to\_pad(event=None):

text1.insert(INSERT,banglakey.speech.text12.get(),"ttags\_"+str(tagno))

def addword895(event=None):

import speech\_recognition as sr

# obtain audio from the microphone

r = sr.Recognizer()

with sr.Microphone() as source:

tkMessageBox.showinfo('Attention',"Say something! After clicking ok")

audio = r.listen(source)

# recognize speech using Google Speech Recognition

try:

# for testing purposes, we're just using the default API key

# to use another API key, use `r.recognize\_google(audio, key="GOOGLE\_SPEECH\_RECOGNITION\_API\_KEY")`

# instead of `r.recognize\_google(audio)`

v='lekha'

v = r.recognize\_google(audio,language='hn')

except sr.UnknownValueError:

tkMessageBox.showinfo('Attention',"Google could not understand audio")

v='Kichu shoona jayni'

except sr.RequestError as e:

tkMessageBox.showinfo('Attention',"Could not request results from Google service.Reccognized using sphinx")

try:

# for testing purposes, we're just using the default API key

# to use another API key, use `r.recognize\_google(audio, key="GOOGLE\_SPEECH\_RECOGNITION\_API\_KEY")`

# instead of `r.recognize\_google(audio)`

v='lekha'

v = r.recognize\_sphinx(audio,language = "hn")

except sr.UnknownValueError:

tkMessageBox.showinfo('Attention',"Sphinx could not understand audio")

v='Kichu shoona jayni'

except sr.RequestError as e:

pass

banglakey.speech.text12.insert(INSERT, v)

def addword12(event=None):

import speech\_recognition as sr

# obtain audio from the microphone

r = sr.Recognizer()

with sr.Microphone() as source:

tkMessageBox.showinfo('Attention',"Say something! After clicking ok")

audio = r.listen(source)

# recognize speech using Google Speech Recognition

try:

# for testing purposes, we're just using the default API key

# to use another API key, use `r.recognize\_google(audio, key="GOOGLE\_SPEECH\_RECOGNITION\_API\_KEY")`

# instead of `r.recognize\_google(audio)`

v='lekha'

v = r.recognize\_google(audio,language='hn')

except sr.UnknownValueError:

tkMessageBox.showinfo('Attention',"Google could not understand audio")

v='Kichu shoona jayni'

except sr.RequestError as e:

tkMessageBox.showinfo('Attention',"Could not request results from Google service.Reccognized using sphinx")

try:

# for testing purposes, we're just using the default API key

# to use another API key, use `r.recognize\_google(audio, key="GOOGLE\_SPEECH\_RECOGNITION\_API\_KEY")`

# instead of `r.recognize\_google(audio)`

v='lekha'

v = r.recognize\_sphinx(audio,language = "hn")

except sr.UnknownValueError:

tkMessageBox.showinfo('Attention',"Sphinx could not understand audio")

v='Kichu shoona jayni'

except sr.RequestError as e:

pass

from pyhinavrophonetic import hinavro

banglakey.speech.text12.insert(INSERT, hinavro.parse(v))

banglakey.speech.button2=Tkinter.Button(banglakey.speech, text =u"Paste to pad", command = text\_to\_pad,relief="groove",bg="#0060FF",fg="white",font=("amar bangla",20))

banglakey.speech.button2.place(x=0,y=210)

banglakey.speech.button3=Tkinter.Button(banglakey.speech, text =u"english", command = addword895,relief="groove",bg="#0060FF",fg="white",font=("amar bangla",15))

banglakey.speech.button3.place(x=200,y=210)

banglakey.speech.button1=Tkinter.Button(banglakey.speech, text =u"hindii", command = addword12,relief="groove",bg="#0060FF",fg="white",font=("amar bangla",15))

banglakey.speech.button1.place(x=270,y=210)

banglakey.speech.mainloop()

def addword(event=None):

'''adds single selected words to dictionary'''

try:

word1=text1.get(Tkinter.SEL\_FIRST, Tkinter.SEL\_LAST)

if d.check(word1)==False:

d.add(word1)

except TclError:

pass

def addword1(event=None):

'''This function opens the window to add multipe words.Words must be seperated by (,)comma'''

banglakey.adlayer=Tkinter.Toplevel()

banglakey.adlayer.title('Add list of word')

banglakey.adlayer.iconbitmap('enigmapad3.ico')

banglakey.adlayer.transient(banglakey)

banglakey.adlayer.minsize(width=450,height=260)

text\_frame11=Tkinter.Frame(banglakey.adlayer,borderwidth=1, relief="sunken")

text11 =Tkinter.Text(banglakey.adlayer,wrap="word",background="white",undo=True,font=("siyam rupali",15)) scroll1=Tkinter.Scrollbar(banglakey.adlayer,orient="vertical",relief=FLAT,width=20,command=text11.yview)

text11.config(yscrollcommand=scroll1.set)

scroll1.pack(in\_=text11,side="right", fill="y", expand=False)

text11.pack(in\_=text\_frame11, side="left", fill="both", expand=True)

text11.focus()

text\_frame11.place(x=0,y=0,height=200, width=400)

def addword11(event=None):

textp=text11.get(1.0,END)

jk=textp.split(',')

for poie in jk:

if d.check(poie)==False:

d.add(poie)

tkMessageBox.showinfo('Attention','Added words to dictionary')

button=Tkinter.Button(banglakey.adlayer, text =u"Add", command = addword11,relief="groove",bg="#0060FF",fg="white",font=("amar bangla",20))

button.place(x=210,y=210)

banglakey.adlayer.mainloop()

def speak(event=None):

gitre=text1.get('1.0','%s-1c'%Tkinter.END)

from pyhinengphonetic import conparse

conparse.speak(gitre)

def delword(event=None):

try:

word1=text1.get(Tkinter.SEL\_FIRST, Tkinter.SEL\_LAST)

if d.check(word1)==True:

d.remove(word1)

except TclError:

pass

def checgreateqindex(index1,index2):

pictyu=[]

pictyu+=[index1]

pictyu+=[index2]

pictyu.sort(key=lambda x: [x.split('.')[0], int(x.split('.')[1])])

if pictyu[0]==pictyu[1]:

return True

if pictyu[0]==index1:

return False

else:

return True

def copy\_to\_clipboard():

#copy to clipboard function

banglakey.clipboard\_clear()

try:

#text in text1 marked, so copy that

banglakey.clipboard\_append(text1.get("sel.first", "sel.last"))

text1.tag\_remove("sel", "sel.first", "sel.last") #NEW LINE

except:

pass

def paste\_from\_clipboard():

global image

if text1.tag\_ranges(SEL):

pic1=[]

for i in range(len(text1.imglist)):

pic1+=[text1.index(text1.imglist[i])]

for j in range(len(pic1)):

if checgreateqindex(pic1[j],text1.index("sel.first")):

break

for k in range(len(pic1)):

if checgreateqindex(pic1[k],text1.index("sel.last")):

break

try:

pic1=pic1[j:k-1]

for nopt in range(len(pic1)):

poiky=text1.dump( pic1[nopt])

for indexim in range(len(poiky)):

if 'image' in poiky[indexim]:

tist=indexim

break

gi=poiky[tist][1]

for ingu in range(len(text1.imglist)):

if gi==str(text1.imglist[ingu]):

text1.imglist.pop(ingu)

joe=ingu

break

image.pop(joe)

text1.delete(pic1[nopt])

text1.delete('sel.first','sel.last')

text1.see('sel.first')

except:

text1.delete('sel.first','sel.last')

text1.see('sel.first')

try:

#text in clipboard marked, so paste that

tex=banglakey.clipboard\_get()

try:

tags=text1.tag\_names(Tkinter.SEL\_FIRST)

except Exception:

text1.insert("insert", tex,"ttags\_"+str(tagno)) #NEW LINE

except Exception as e:

from PIL import ImageGrab, Image,ImageTk

import cStringIO, base64

im= ImageGrab.grabclipboard()

fp = cStringIO.StringIO()

im.save(fp,'GIF')

kopo=''''''

kopo+=base64.encodestring(fp.getvalue())

image+=[kopo]

MyPhotoImage = Tkinter.PhotoImage(data=base64.encodestring(fp.getvalue()))

text1.image\_create(Tkinter.INSERT, image=MyPhotoImage)

text1.imglist+=[MyPhotoImage]

pass

def onCut():

#copy to clipboard function after cut

if text1.tag\_ranges(SEL):

pic1=[]

for i in range(len(text1.imglist)):

pic1+=[text1.index(text1.imglist[i])]

for j in range(len(pic1)):

if checgreateqindex(pic1[j],text1.index("sel.first")):

break

for k in range(len(pic1)):

if checgreateqindex(pic1[k],text1.index("sel.last")):

break

try:

pic1=pic1[j:k-1]

for nopt in range(len(pic1)):

poiky=text1.dump( pic1[nopt])

for indexim in range(len(poiky)):

if 'image' in poiky[indexim]:

tist=indexim

break

gi=poiky[tist][1]

for ingu in range(len(text1.imglist)):

if gi==str(text1.imglist[ingu]):

text1.imglist.pop(ingu)

joe=ingu

break

image.pop(joe)

text1.delete(pic1[nopt])

banglakey.clipboard\_clear()

banglakey.clipboard\_append(text1.get("sel.first", "sel.last"))

text1.delete('sel.first','sel.last')

text1.see('sel.first')

except:

banglakey.clipboard\_clear()

banglakey.clipboard\_append(text1.get("sel.first", "sel.last"))

text1.delete('sel.first','sel.last')

text1.see('sel.first')

filemenu = Tkinter.Menu(menu)

menu.add\_cascade(label="File", menu=filemenu)

filemenu.add\_command(label="New", command=NewFile)

filemenu.add\_command(label="Open...", command=u78)

filemenu.add\_command(label="Save", command=u79)

filemenu.add\_separator()

filemenu.add\_command(label="Exit",activebackground='red', command=quitee)

edit = Tkinter.Menu(menu)

menu.add\_cascade(label="Edit", menu=edit)

edit.add\_command(label="Copy",activebackground='blue', command=copy\_to\_clipboard)

edit.add\_command(label="Cut",activebackground='blue', command=onCut)

edit.add\_command(label="Paste",activebackground='blue', command=paste\_from\_clipboard)

edit.add\_command(label="Font",activebackground='blue', command=Font)

diction = Tkinter.Menu(menu)

menu.add\_cascade(label="Dictionary", menu=diction)

diction.add\_command(label="Spell Check",activebackground='blue', command=Spellcheck)

diction.add\_command(label="Suggest word",activebackground='blue', command=suggest)

diction.add\_command(label="Add word to dict",activebackground='blue', command=addword)

diction.add\_command(label="Add listed word to dict",activebackground='blue', command=addword1)

diction.add\_command(label="Delete word from dict",activebackground='blue', command=delword)

special = Tkinter.Menu(menu)

menu.add\_cascade(label="Special", menu=special)

special.add\_command(label="Recognize speech",activebackground='magenta', command=recognize)

special.add\_command(label="Use phonetic typepad",activebackground='magenta', command=addword3)

special.add\_command(label="Text to spech",activebackground='magenta', command=speak)

helpmenu = Tkinter.Menu(menu)

menu.add\_cascade(label="Help", menu=helpmenu)

helpmenu.add\_command(label="About...", command=About)

helpmenu.add\_command(label="Help file",activebackground='purple', command=helpe)

toolbar=Tkinter.Frame(banglakey,borderwidth=1, relief="sunken")

text\_frame=Tkinter.Frame(banglakey,borderwidth=1, relief="sunken")

text1 =Tkinter.Text(banglakey,wrap="word",background="white",undo=True,font=("siyam rupali",20))

scroll=Tkinter.Scrollbar(banglakey,orient="vertical",relief=FLAT,width=20,command=text1.yview)

scroll1=Tkinter.Scrollbar(banglakey,orient="horizontal",relief=FLAT,width=10,command=text1.xview)

text1.config(xscrollcommand=scroll1.set)

text1.config(yscrollcommand=scroll.set)

scroll.pack(in\_=text\_frame,side="right", fill="y", expand=False)

scroll1.pack(in\_=text\_frame,side="bottom", fill="x", expand=False)

text1.pack(in\_=text\_frame, side="left", fill="both", expand=True)

text1.focus()

toolbar.pack(side='top',fill='x' )

text\_frame.place(bordermode=OUTSIDE,height=620,width=500,y=30 )

text1.imglist=[]

text1.tag\_configure("ttags\_"+str(0), font=fonti)

text1.tag\_configure("misspelled", foreground="red", underline=True)

def addword13(font,g='',event=None):

try:

global image

if event.keysym == 'BackSpace':

if text1.tag\_ranges(SEL):

pic1=[]

for i in range(len(text1.imglist)):

pic1+=[text1.index(text1.imglist[i])]

for j in range(len(pic1)):

if checgreateqindex(pic1[j],text1.index("sel.first")):

break

for k in range(len(pic1)):

if checgreateqindex(pic1[k],text1.index("sel.last")):

break

try:

pic1=pic1[j:k-1]

for nopt in range(len(pic1)):

poiky=text1.dump( pic1[nopt])

for indexim in range(len(poiky)):

if 'image' in poiky[indexim]:

tist=indexim

break

gi=poiky[tist][1]

for ingu in range(len(text1.imglist)):

if gi==str(text1.imglist[ingu]):

text1.imglist.pop(ingu)

joe=ingu

break

image.pop(joe)

text1.delete(pic1[nopt])

text1.delete('sel.first','sel.last')

text1.see('sel.first')

except:

text1.delete('sel.first','sel.last')

text1.see('sel.first')

elif text1.get("%s-1c" % Tkinter.INSERT, Tkinter.INSERT)=='':

poiky=text1.dump("%s-1c" % Tkinter.INSERT)

for indexim in range(len(poiky)):

if 'image' in poiky[indexim]:

tist=indexim

break

gi=poiky[tist][1]

for ingu in range(len(text1.imglist)):

if gi==str(text1.imglist[ingu]):

text1.imglist.pop(ingu)

joe=ingu

break

image.pop(joe)

text1.delete("%s-1c" % Tkinter.INSERT, Tkinter.INSERT)

text1.see(Tkinter.INSERT)

else:

text1.delete("%s-1c" % Tkinter.INSERT, Tkinter.INSERT)

text1.see(Tkinter.INSERT)

return 'break'

elif event.keysym == 'Delete':

if text1.tag\_ranges(SEL):

pic1=[]

for i in range(len(text1.imglist)):

pic1+=[text1.index(text1.imglist[i])]

for j in range(len(pic1)):

if checgreateqindex(pic1[j],text1.index("sel.first")):

break

for k in range(len(pic1)):

if checgreateqindex(pic1[k],text1.index("sel.last")):

break

try:

pic1=pic1[j:k-1]

for nopt in range(len(pic1)):

poiky=text1.dump( pic1[nopt])

for indexim in range(len(poiky)):

if 'image' in poiky[indexim]:

tist=indexim

break

gi=poiky[tist][1]

for ingu in range(len(text1.imglist)):

if gi==str(text1.imglist[ingu]):

text1.imglist.pop(ingu)

joe=ingu

break

image.pop(joe)

text1.delete(pic1[nopt])

text1.delete('sel.first','sel.last')

except:

text1.delete('sel.first','sel.last')

elif text1.get("%s" % Tkinter.INSERT)=='':

poiky=text1.dump("%s" % Tkinter.INSERT)

for indexim in range(len(poiky)):

if 'image' in poiky[indexim]:

tist=indexim

break

gi=poiky[tist][1]

for ingu in range(len(text1.imglist)):

if gi==str(text1.imglist[ingu]):

text1.imglist.pop(ingu)

joe=ingu

break

image.pop(joe)

text1.delete("%s" % Tkinter.INSERT)

else:

text1.delete("%s" % Tkinter.INSERT)

text1.see(Tkinter.INSERT)

return 'break'

elif event.keysym == 'Return':

text1.insert(Tkinter.INSERT,'\n',"ttags\_"+str(tagno))

text1.see(Tkinter.INSERT)

return 'break'

elif event.keysym == 'space':

text1.insert(Tkinter.INSERT,u' ',"ttags\_"+str(tagno))

text1.see(Tkinter.INSERT)

return 'break'

elif event.keysym == 'Control':

return 'break'

elif event.keysym == 'Control\_L':

return 'break'

elif event.keysym == 'Control+V':

return 'break'

elif event.keysym == 'Prior':

pass

elif event.keysym == 'Next':

pass

elif event.keysym == 'Home':

pass

elif event.keysym == 'End':

pass

elif event.keysym == 'Up':

pass

elif event.keysym == 'Down':

pass

elif event.keysym == 'Left':

pass

elif event.keysym == 'Right':

pass

elif event.keysym == 'Shift':

pass

elif event.keysym == 'Alt':

pass

elif len(event.keysym)==1:

text1.insert(Tkinter.INSERT,unicode(g),"ttags\_"+str(tagno))

text1.see(Tkinter.INSERT)

return 'break'

else:

text1.insert(Tkinter.INSERT,g,"ttags\_"+str(tagno))

text1.see(Tkinter.INSERT)

return 'break'

except AttributeError:

pass

def onSelectAll():

#select text

text1.tag\_add('sel', "1.0", Tkinter.END)

text1.mark\_set(Tkinter.INSERT, "1.0")

text1.see(Tkinter.INSERT)

return 'break'

def disp\_menu(event=None):

menul = Tkinter.Menu(text1)

try:

tags=text1.tag\_names(Tkinter.SEL\_FIRST)

menul.add\_command(label="Copy",activebackground='blue', command=copy\_to\_clipboard)

menul.add\_command(label="Cut",activebackground='blue', command=onCut)

menul.add\_command(label="Paste",activebackground='blue', command=paste\_from\_clipboard)

menul.add\_command(label="Font",activebackground='magenta', command=Font)

menul.add\_command(label="Insert word \nto dict",activebackground='magenta', command=addword)

menul.add\_command(label="Delete word \nfrom dict",activebackground='magenta', command=delword)

menul.add\_command(label="Spell Check",activebackground='magenta', command=Spellchecksing)

except TclError:

menul.add\_command(label="Font",activebackground='magenta', command=Font)

menul.add\_command(label="Paste",activebackground='blue', command=paste\_from\_clipboard)

menul.add\_command(label="Spell Check",activebackground='magenta', command=Spellcheck)

menul.post(event.x,event.y)

pass

text1.bind('<KeyPress>',lambda event: addword13(fonti,event.char,event))

text1.bind('<Control-c>',lambda event: copy\_to\_clipboard())

text1.bind('<Control-v>',lambda event: paste\_from\_clipboard())

text1.bind('<Control-a>',lambda event: onSelectAll())

text1.bind('<Control-x>',lambda event: onCut())

text1.bind('<Control-Key>',lambda event: None)

text1.bind('<Button-3>',lambda event: disp\_menu(event))

def return\_font\_list(string=''):

#converts tag\_cget(tagname,'font') to a list

l=[]

add=''

index=0

countspace=0

font=''

i1=1

i2=1

li=string.split()

if string[0]=='{':

for i in range(len(string)):

if string[i]!='}':

add+=string[i]

elif string[i]=='}':

add+=string[i]

for j in range(len(add)):

if add[j]==' ':

countspace+=1

break

while countspace>0:

li.pop(countspace)

countspace-=1

li[0]=add

else:

pass

for t in range(len(li)):

font=font+li[t]+' '

return li

def font\_main\_change(fontgot):

'''changes the main font i.e. ttags<number>'''

from copy import deepcopy

global fonti

global tagno

kii=tuple()

kii+=(fontgot,)

for iop in range(1,len(fonti)):

kii+=(fonti[iop],)

fonti=deepcopy(kii)

tagno+=1

text1.tag\_configure("ttags\_"+str(tagno), font=fonti)

fontList = Tix.ComboBox( banglakey, command= font\_main\_change, dropdown=True, editable=False, selectmode=Tix.IMMEDIATE ,variable=fonti[0])

fontList.pack(in\_=toolbar,side='left')

familyList = list(tkFont.families( ))

familyList.sort()

for family in familyList:

if family[0] == '@':

continue

if fonti[0] is None:

fonti[0] = family

fontList.insert( Tix.END, family )

fontList.configure( value=fonti[0] )

def textin(event=None):

#prints tags along with associated text, image's name and offset

tags=text1.tag\_names(index=None)

pair1=[]

pair2=[]

for tag in tags:

ranges = text1.tag\_ranges(tag)

for i in range(0, len(ranges), 2):

start = ranges[i]

stop = ranges[i+1]

pair1+=[str(start)]

pair2+=[str(stop)]

if "%s-1c" %stop==start:

if text1.get(start)=='':

pass

for i in range(len(text1.imglist)):

pair1+=[text1.index(text1.imglist[i])]

pair2+=[text1.index("%s+1c"%text1.imglist[i])]

pair1.sort(key=lambda x: [x.split('.')[0], int(x.split('.')[1])])

pair2.sort(key=lambda x: [x.split('.')[0], int(x.split('.')[1])])

def fchange1():

#changes main font or font of selected text to superscript

try:

tags=text1.tag\_names(Tkinter.SEL\_FIRST)

if tags[1][0]=='t':

tag=tags[1]

fonte=return\_font\_list(text1.tag\_cget(tag,'font'))

try:

global ioftag

if text1.tag\_cget(tag,'offset')=='0'or text1.tag\_cget(tag,'offset')=='':

k=int(fonte[1])/2

fonte[1]=str(k)

ioftag+=1

kondaa=''

for konda in fonte:

kondaa=kondaa+konda+' '

fonte=kondaa

text1.tag\_configure("tags\_"+str(ioftag), font=fonte,offset=+6)

text1.tag\_remove(tag,Tkinter.SEL\_FIRST, Tkinter.SEL\_LAST)

text1.tag\_add("tags\_"+str(ioftag),Tkinter.SEL\_FIRST, Tkinter.SEL\_LAST)

elif text1.tag\_cget(tag,'offset')=='-6':

ioftag+=1

kondaa=''

for konda in fonte:

kondaa=kondaa+konda+' '

fonte=kondaa

text1.tag\_configure("tags\_"+str(ioftag), font=fonte,offset=+6)

text1.tag\_remove(tag,Tkinter.SEL\_FIRST, Tkinter.SEL\_LAST)

text1.tag\_add("tags\_"+str(ioftag),Tkinter.SEL\_FIRST, Tkinter.SEL\_LAST)

except Exception as e:

#remove exception when sure

pass

except TclError:

pass

def fchange2():

#changes main font or font of selected text to subscript

try:

tags=text1.tag\_names(Tkinter.SEL\_FIRST)

if tags[1][0]=='t':

tag=tags[1]

fonte=return\_font\_list(text1.tag\_cget(tag,'font'))

try:

global ioftag

if text1.tag\_cget(tag,'offset')=='0'or text1.tag\_cget(tag,'offset')=='':

k=int(fonte[1])/2

fonte[1]=str(k)

ioftag+=1

kondaa=''

for konda in fonte:

kondaa=kondaa+konda+' '

fonte=kondaa

text1.tag\_configure("tags\_"+str(ioftag), font=fonte,offset=-6)

text1.tag\_remove(tag,Tkinter.SEL\_FIRST, Tkinter.SEL\_LAST)

text1.tag\_add("tags\_"+str(ioftag),Tkinter.SEL\_FIRST, Tkinter.SEL\_LAST)

elif text1.tag\_cget(tag,'offset')=='6':

ioftag+=1

kondaa=''

for konda in fonte:

kondaa=kondaa+konda+' '

fonte=kondaa

text1.tag\_configure("tags\_"+str(ioftag), font=fonte,offset=-6)

text1.tag\_remove(tag,Tkinter.SEL\_FIRST, Tkinter.SEL\_LAST)

text1.tag\_add("tags\_"+str(ioftag),Tkinter.SEL\_FIRST, Tkinter.SEL\_LAST)

except Exception as e:

#remove exception when sure

pass

except TclError:

pass

def fchange3():

#changes main font or font of selected text to normal i.e. removes superscript and subscript and enlages font size

try:

tags=text1.tag\_names(Tkinter.SEL\_FIRST)

if tags[1][0]=='t':

tag=tags[1]

fonte=return\_font\_list(text1.tag\_cget(tag,'font'))

try:

global ioftag

if text1.tag\_cget(tag,'offset')=='6'or text1.tag\_cget(tag,'offset')=='-6':

k=int(fonte[1])\*2

fonte[1]=str(k)

ioftag+=1

kondaa=''

for konda in fonte:

kondaa=kondaa+konda+' '

fonte=kondaa

text1.tag\_configure("tags\_"+str(ioftag), font=fonte,offset=0)

text1.tag\_remove(tag,Tkinter.SEL\_FIRST, Tkinter.SEL\_LAST)

text1.tag\_add("tags\_"+str(ioftag),Tkinter.SEL\_FIRST, Tkinter.SEL\_LAST)

elif text1.tag\_cget(tag,'offset')=='0'or text1.tag\_cget(tag,'offset')=='':

pass

except Exception as e:

pass

except TclError:

pass

button=Tkinter.Button(banglakey,text='OK',command=textin)

button.pack(in\_=toolbar,side='left')

button1=Tkinter.Button(banglakey,text='super',command=fchange1)

button1.pack(in\_=toolbar,side='left')

button2=Tkinter.Button(banglakey,text='sub',command=fchange2)

button2.pack(in\_=toolbar,side='left')

button2=Tkinter.Button(banglakey,text='normal',command=fchange3)

button2.pack(in\_=toolbar,side='left')

#@changed from here

##HINDI KEYBOARD----------------------------------------------------------------------------------------

#SBarna--------------------------------------------------

sbarna\_frame=Tkinter.Frame(banglakey,borderwidth=2, relief='flat')

def u1(event=None):

text1.insert(INSERT,u'\u0905',"ttags\_"+str(tagno))

text1.see(INSERT)

b5=Tkinter.Button(banglakey, text =u"\u0905",width=2,bg="white",fg="black", command = u1, font=("arial",20))

b5.pack(in\_=sbarna\_frame,side="left", expand=False)

def u2(event=None):

text1.insert(INSERT,u'\u0906',"ttags\_"+str(tagno))

text1.see(INSERT)

b51=Tkinter.Button(banglakey, text =u"\u0906", command = u2,width=2,bg="white",fg="black",font=("amar bangla",20))

b51.pack(in\_=sbarna\_frame,side="left", expand=False)

def u3(event=None):

text1.insert(INSERT,u'\u0907',"ttags\_"+str(tagno))

text1.see(INSERT)

b52=Tkinter.Button(banglakey, text =u"\u0907", command = u3,width=2,bg="white",fg="black",font=("amar bangla",20))

b52.pack(in\_=sbarna\_frame,side="left", expand=False)

def u4(event=None):

text1.insert(INSERT,u'\u0908',"ttags\_"+str(tagno))

text1.see(INSERT)

b53=Tkinter.Button(banglakey, text =u"\u0908", command = u4,width=2,bg="white",fg="black",font=("amar bangla",20))

b53.pack(in\_=sbarna\_frame,side="left", expand=False)

def u5(event=None):

text1.insert(INSERT,u'\u0909',"ttags\_"+str(tagno))

text1.see(INSERT)

b54=Tkinter.Button(banglakey, text =u"\u0909", command = u5,width=2,bg="white",fg="black",font=("amar bangla",20))

b54.pack(in\_=sbarna\_frame,side="left", expand=False)

def u6(event=None):

text1.insert(INSERT,u'\u090a',"ttags\_"+str(tagno))

text1.see(INSERT)

b55=Tkinter.Button(banglakey, text =u"\u090a", command = u6,width=2,bg="white",fg="black",font=("amar bangla",20))

b55.pack(in\_=sbarna\_frame,side="left", expand=False)

def u7(event=None):

text1.insert(INSERT,u'\u090f',"ttags\_"+str(tagno))

text1.see(INSERT)

b56=Tkinter.Button(banglakey, text =u"\u090f", command = u7,width=2,bg="white",fg="black",font=("amar bangla",20))

b56.pack(in\_=sbarna\_frame,side="left", expand=False)

def u8(event=None):

text1.insert(INSERT,u'\u0910',"ttags\_"+str(tagno))

text1.see(INSERT)

b57=Tkinter.Button(banglakey, text =u"\u0910", command = u8,width=2,bg="white",fg="black",font=("amar bangla",20))

b57.pack(in\_=sbarna\_frame,side="left", expand=False)

def u9(event=None):

text1.insert(INSERT,u'\u0913',"ttags\_"+str(tagno))

text1.see(INSERT)

b58=Tkinter.Button(banglakey, text =u"\u0913", command = u9,width=2,bg="white",fg="black",font=("amar bangla",20))

b58.pack(in\_=sbarna\_frame,side="left", expand=False)

def u10(event=None):

text1.insert(INSERT,u'\u0914',"ttags\_"+str(tagno))

text1.see(INSERT)

b59=Tkinter.Button(banglakey, text =u"\u0914", command = u10,width=2,bg="white",fg="black",font=("amar bangla",20))

b59.pack(in\_=sbarna\_frame,side="left", expand=False)

def u11(event=None):

text1.insert(INSERT,u'\u090b',"ttags\_"+str(tagno))

text1.see(INSERT)

b510=Tkinter.Button(banglakey, text =u"\u090b", command = u11,width=2,bg="white",fg="black",font=("amar bangla",20))

b510.pack(in\_=sbarna\_frame,side="left", expand=False)

sbarna\_frame.place(bordermode=OUTSIDE,x=510,y=26 , height=40, width=480)

label2=Tkinter.Label(banglakey,text=u'स्वरवर्ण',font=('siyam rupali',15))

label2.place(x=750,y=70)

bbarna\_frame1=Tkinter.Frame(banglakey,borderwidth=2)

bbarna\_frame=Tkinter.Frame(banglakey,borderwidth=0, relief="sunken",)

def u13(event=None):

text1.insert(INSERT,u'\u0915',"ttags\_"+str(tagno))

text1.see(INSERT)

b512=Tkinter.Button(banglakey, text =u"\u0915", command = u13,width=2,bg="white",fg="black",font=("amar bangla",20))

b512.pack(in\_=bbarna\_frame,side="left", expand=False)

def u14(event=None):

text1.insert(INSERT,u'\u0916',"ttags\_"+str(tagno))

text1.see(INSERT)

b513=Tkinter.Button(banglakey, text =u"\u0916", command = u14,width=2,bg="white",fg="black",font=("amar bangla",20))

b513.pack(in\_=bbarna\_frame,side="left", expand=False)

def u15(event=None):

text1.insert(INSERT,u'\u0917',"ttags\_"+str(tagno))

text1.see(INSERT)

b514=Tkinter.Button(banglakey, text =u"\u0917", command = u15,width=2,bg="white",fg="black",font=("amar bangla",20))

b514.pack(in\_=bbarna\_frame,side="left", expand=False)

def u16(event=None):

text1.insert(INSERT,u'\u0918',"ttags\_"+str(tagno))

text1.see(INSERT)

b515=Tkinter.Button(banglakey, text =u"\u0918", command = u16,width=2,bg="white",fg="black",font=("amar bangla",20))

b515.pack(in\_=bbarna\_frame,side="left", expand=False)

def u17(event=None):

text1.insert(INSERT,u'\u0919',"ttags\_"+str(tagno))

text1.see(INSERT)

b516=Tkinter.Button(banglakey, text =u"\u0919", command = u17,width=2,bg="white",fg="black",font=("amar bangla",20))

b516.pack(in\_=bbarna\_frame,side="left", expand=False)

bbarna\_frame.pack(in\_=bbarna\_frame1,side="top", expand=False,fill=BOTH)

bbarna\_frame2=Tkinter.Frame(banglakey,borderwidth=0, relief="sunken")

def u18(event=None):

text1.insert(INSERT,u'\u091A',"ttags\_"+str(tagno))

text1.see(INSERT)

b517=Tkinter.Button(banglakey, text =u"\u091A", command = u18,width=2,bg="white",fg="black",font=("amar bangla",20))

b517.pack(in\_=bbarna\_frame2,side="left", expand=False)

def u19(event=None):

text1.insert(INSERT,u'\u091B',"ttags\_"+str(tagno))

text1.see(INSERT)

b518=Tkinter.Button(banglakey, text =u"\u091B", command = u19,width=2,bg="white",fg="black",font=("amar bangla",20))

b518.pack(in\_=bbarna\_frame2,side="left", expand=False)

def u20(event=None):

text1.insert(INSERT,u'\u091C',"ttags\_"+str(tagno))

text1.see(INSERT)

b519=Tkinter.Button(banglakey, text =u"\u091C", command = u20,width=2,bg="white",fg="black",font=("amar bangla",20))

b519.pack(in\_=bbarna\_frame2,side="left", expand=False)

def u21(event=None):

text1.insert(INSERT,u'\u091D',"ttags\_"+str(tagno))

text1.see(INSERT)

b520=Tkinter.Button(banglakey, text =u"\u091D", command = u21,width=2,bg="white",fg="black",font=("amar bangla",20))

b520.pack(in\_=bbarna\_frame2,side="left", expand=False)

def u22(event=None):

text1.insert(INSERT,u'\u091E',"ttags\_"+str(tagno))

text1.see(INSERT)

b521=Tkinter.Button(banglakey, text =u"\u091E", command = u22,width=2,bg="white",fg="black",font=("amar bangla",20))

b521.pack(in\_=bbarna\_frame2,side="left", expand=False)

bbarna\_frame2.pack(in\_=bbarna\_frame1,side="top", expand=False,fill=BOTH)

bbarna\_frame3=Tkinter.Frame(banglakey,borderwidth=0, relief="sunken")

def u23(event=None):

text1.insert(INSERT,u'\u091F',"ttags\_"+str(tagno))

text1.see(INSERT)

b522=Tkinter.Button(banglakey, text =u"\u091F", command = u23,width=2,bg="white",fg="black",font=("amar bangla",20))

b522.pack(in\_=bbarna\_frame3,side="left", expand=False)

def u24(event=None):

text1.insert(INSERT,u'\u0920',"ttags\_"+str(tagno))

text1.see(INSERT)

b523=Tkinter.Button(banglakey, text =u"\u0920", command = u24,width=2,bg="white",fg="black",font=("amar bangla",20))

b523.pack(in\_=bbarna\_frame3,side="left", expand=False)

def u26(event=None):

text1.insert(INSERT,u'\u0921',"ttags\_"+str(tagno))

text1.see(INSERT)

b525=Tkinter.Button(banglakey, text =u"\u0921", command = u26,width=2,bg="white",fg="black",font=("amar bangla",20))

b525.pack(in\_=bbarna\_frame3,side="left", expand=False)

def u27(event=None):

text1.insert(INSERT,u'\u0922',"ttags\_"+str(tagno))

text1.see(INSERT)

b526=Tkinter.Button(banglakey, text =u"\u0922", command = u27,width=2,bg="white",fg="black",font=("amar bangla",20))

b526.pack(in\_=bbarna\_frame3,side="left", expand=False)

def u28(event=None):

text1.insert(INSERT,u'\u0923',"ttags\_"+str(tagno))

text1.see(INSERT)

b526=Tkinter.Button(banglakey, text =u"\u0923", command = u28,width=2,bg="white",fg="black",font=("amar bangla",20))

b526.pack(in\_=bbarna\_frame3,side="left", expand=False)

bbarna\_frame3.pack(in\_=bbarna\_frame1,side="top", expand=False,fill=BOTH)

bbarna\_frame4=Tkinter.Frame(banglakey,borderwidth=0, relief="sunken")

def u29(event=None):

text1.insert(INSERT,u'\u0924',"ttags\_"+str(tagno))

text1.see(INSERT)

b527=Tkinter.Button(banglakey, text =u"\u0924", command = u29,width=2,bg="white",fg="black",font=("amar bangla",20))

b527.pack(in\_=bbarna\_frame4,side="left", expand=False)

def u30(event=None):

text1.insert(INSERT,u'\u0925',"ttags\_"+str(tagno))

text1.see(INSERT)

b528=Tkinter.Button(banglakey, text =u"\u0925", command = u30,width=2,bg="white",fg="black",font=("amar bangla",20))

b528.pack(in\_=bbarna\_frame4,side="left", expand=False)

def u31(event=None):

text1.insert(INSERT,u'\u0926',"ttags\_"+str(tagno))

text1.see(INSERT)

b529=Tkinter.Button(banglakey, text =u"\u0926", command = u31,width=2,bg="white",fg="black",font=("amar bangla",20))

b529.pack(in\_=bbarna\_frame4,side="left", expand=False)

def u32(event=None):

text1.insert(INSERT,u'\u0927',"ttags\_"+str(tagno))

text1.see(INSERT)

b530=Tkinter.Button(banglakey, text =u"\u0927", command = u32,width=2,bg="white",fg="black",font=("amar bangla",20))

b530.pack(in\_=bbarna\_frame4,side="left", expand=False)

def u33(event=None):

text1.insert(INSERT,u'\u0928',"ttags\_"+str(tagno))

text1.see(INSERT)

b531=Tkinter.Button(banglakey, text =u"\u0928", command = u33,width=2,bg="white",fg="black",font=("amar bangla",20))

b531.pack(in\_=bbarna\_frame4,side="left", expand=False)

bbarna\_frame4.pack(in\_=bbarna\_frame1,side="top", expand=False,fill=BOTH)

bbarna\_frame5=Tkinter.Frame(banglakey,borderwidth=0, relief="sunken")

def u34(event=None):

text1.insert(INSERT,u'\u092A',"ttags\_"+str(tagno))

text1.see(INSERT)

b532=Tkinter.Button(banglakey, text =u"\u092A", command = u34,width=2,bg="white",fg="black",font=("amar bangla",20))

b532.pack(in\_=bbarna\_frame5,side="left", expand=False)

def u35(event=None):

text1.insert(INSERT,u'\u092B',"ttags\_"+str(tagno))

text1.see(INSERT)

b533=Tkinter.Button(banglakey, text =u"\u092B", command = u35,width=2,bg="white",fg="black",font=("amar bangla",20))

b533.pack(in\_=bbarna\_frame5,side="left", expand=False)

def u36(event=None):

text1.insert(INSERT,u'\u092C',"ttags\_"+str(tagno))

text1.see(INSERT)

b534=Tkinter.Button(banglakey, text =u"\u092C", command = u36,width=2,bg="white",fg="black",font=("amar bangla",20))

b534.pack(in\_=bbarna\_frame5,side="left", expand=False)

def u37(event=None):

text1.insert(INSERT,u'\u092D',"ttags\_"+str(tagno))

text1.see(INSERT)

b535=Tkinter.Button(banglakey, text =u"\u092D", command = u37,width=2,bg="white",fg="black",font=("amar bangla",20))

b535.pack(in\_=bbarna\_frame5,side="left", expand=False)

def u38(event=None):

text1.insert(INSERT,u'\u092E',"ttags\_"+str(tagno))

text1.see(INSERT)

b536=Tkinter.Button(banglakey, text =u"\u092E", command = u38,width=2,bg="white",fg="black",font=("amar bangla",20))

b536.pack(in\_=bbarna\_frame5,side="left", expand=False)

bbarna\_frame5.pack(in\_=bbarna\_frame1,side="top", expand=False,fill=BOTH)

bbarna\_frame6=Tkinter.Frame(banglakey,borderwidth=0, relief="sunken")

def u44(event=None):

text1.insert(INSERT,u'\u092F',"ttags\_"+str(tagno))

text1.see(INSERT)

b542=Tkinter.Button(banglakey, text =u"\u092F", command = u44,width=2,bg="white",fg="black",font=("amar bangla",20))

b542.pack(in\_=bbarna\_frame6,side="left", expand=False)

def u45(event=None):

text1.insert(INSERT,u'\u0930',"ttags\_"+str(tagno))

text1.see(INSERT)

b543=Tkinter.Button(banglakey, text =u"\u0930", command = u45,width=2,bg="white",fg="black",font=("amar bangla",20))

b543.pack(in\_=bbarna\_frame6,side="left", expand=False)

def u46(event=None):

text1.insert(INSERT,u'\u0932',"ttags\_"+str(tagno))

text1.see(INSERT)

b544=Tkinter.Button(banglakey, text =u"\u0932", command = u46,width=2,bg="white",fg="black",font=("amar bangla",20))

b544.pack(in\_=bbarna\_frame6,side="left", expand=False)

def u47(event=None):

text1.insert(INSERT,u'\u0935',"ttags\_"+str(tagno))

text1.see(INSERT)

b545=Tkinter.Button(banglakey, text =u"\u0935", command = u47,width=2,bg="white",fg="black",font=("amar bangla",20))

b545.pack(in\_=bbarna\_frame6,side="left", expand=False)

def u48(event=None):

text1.insert(INSERT,u'\u0936',"ttags\_"+str(tagno))

text1.see(INSERT)

b546=Tkinter.Button(banglakey, text =u"\u0936", command = u48,width=2,bg="white",fg="black",font=("amar bangla",20))

b546.pack(in\_=bbarna\_frame6,side="left", expand=False)

bbarna\_frame6.pack(in\_=bbarna\_frame1,side="top", expand=False,fill=BOTH)

bbarna\_frame51=Tkinter.Frame(banglakey,borderwidth=0, relief="sunken")

def u39(event=None):

text1.insert(INSERT,u'\u0937',"ttags\_"+str(tagno))

text1.see(INSERT)

b537=Tkinter.Button(banglakey, text =u"\u0937", command = u39,width=2,bg="white",fg="black",font=("amar bangla",20))

b537.pack(in\_=bbarna\_frame51,side="left", expand=False)

def u40(event=None):

text1.insert(INSERT,u'\u0938',"ttags\_"+str(tagno))

text1.see(INSERT)

b538=Tkinter.Button(banglakey, text =u"\u0938", command = u40,width=2,bg="white",fg="black",font=("amar bangla",20))

b538.pack(in\_=bbarna\_frame51,side="left", expand=False)

def u41(event=None):

text1.insert(INSERT,u'\u0939',"ttags\_"+str(tagno))

text1.see(INSERT)

b539=Tkinter.Button(banglakey, text =u"\u0939", command = u41,width=2,bg="white",fg="black",font=("amar bangla",20))

b539.pack(in\_=bbarna\_frame51,side="left", expand=False)

def u411(event=None):

text1.insert(INSERT,u'\u0901',"ttags\_"+str(tagno))

text1.see(INSERT)

b539=Tkinter.Button(banglakey, text =u"\u0901", command = u411,width=2,bg="white",fg="black",font=("amar bangla",20))

b539.pack(in\_=bbarna\_frame51,side="left", expand=False)

bbarna\_frame51.pack(in\_=bbarna\_frame1,side="top", expand=False,fill=BOTH)

bbarna\_frame1.place(bordermode=OUTSIDE,x=510,y=105 , height=390, width=213)

label1=Tkinter.Label(banglakey,text=u'व्यंजनवर्ण',font=('siyam rupali',15))

label1.place(x=570,y=70)

sp\_frame=Tkinter.Frame(banglakey,borderwidth=1, relief="flat")

def u49(event=None):

text1.insert(INSERT,u'\u0964',"ttags\_"+str(tagno))

text1.see(INSERT)

b547=Tkinter.Button(banglakey, text =u"\u0964", command = u49,width=2,bg="white",fg="black",font=("amar bangla",20))

b547.pack(in\_=sp\_frame,side="right", expand=False)

def u491(event=None):

text1.insert(INSERT,u'\u0965',"ttags\_"+str(tagno))

text1.see(INSERT)

b547=Tkinter.Button(banglakey, text =u"\u0965", command = u491,width=2,bg="white",fg="black",font=("amar bangla",20))

b547.pack(in\_=sp\_frame,side="right", expand=False)

def u50(event=None):

text1.insert(INSERT,' ',"ttags\_"+str(tagno))

text1.see(INSERT)

b548=Tkinter.Button(banglakey, text ='\_', command = u50,width=2,bg="white",fg="black",font=("amar bangla",20))

b548.pack(in\_=sp\_frame,side="right", expand=False)

def u25(event=None):

text1.insert(INSERT,'\n',"ttags\_"+str(tagno))

text1.see(INSERT)

b524=Tkinter.Button(banglakey, text ='|->', command = u25,width=2,bg="white",fg="black",font=("amar bangla",20))

b524.pack(in\_=sp\_frame,side="right", expand=False)

sp\_frame.place(bordermode=OUTSIDE,x=780,y=400 , height=50, width=170)

kbarna\_frame=Tkinter.Frame(banglakey,borderwidth=1, relief="flat")

def u51(event=None):

text1.insert(INSERT,u'\u0958',"ttags\_"+str(tagno))

text1.see(INSERT)

b549=Tkinter.Button(banglakey, text =u'\u0958', command = u51,width=2,bg="white",fg="black",font=("amar bangla",20))

b549.pack(in\_=kbarna\_frame,side="left", expand=False)

def u52(event=None):

text1.insert(INSERT,u'\u0959',"ttags\_"+str(tagno))

text1.see(INSERT)

b550=Tkinter.Button(banglakey, text =u'\u0959', command = u52,width=2,bg="white",fg="black",font=("amar bangla",20))

b550.pack(in\_=kbarna\_frame,side="left", expand=False)

def u53(event=None):

text1.insert(INSERT,u'\u095a',"ttags\_"+str(tagno))

text1.see(INSERT)

b551=Tkinter.Button(banglakey, text =u'\u095a', command = u53,width=2,bg="white",fg="black",font=("amar bangla",20))

b551.pack(in\_=kbarna\_frame,side="left", expand=False)

def u54(event=None):

text1.insert(INSERT,u'\u095b',"ttags\_"+str(tagno))

text1.see(INSERT)

b552=Tkinter.Button(banglakey, text =u'\u095b', command = u54,width=2,bg="white",fg="black",font=("amar bangla",20))

b552.pack(in\_=kbarna\_frame,side="left", expand=False)

def u75(event=None):

text1.insert(INSERT,u'\u095c',"ttags\_"+str(tagno))

text1.see(INSERT)

b573=Tkinter.Button(banglakey, text =u'\u095c', command = u75,width=2,bg="white",fg="black",font=("amar bangla",20))

b573.pack(in\_=kbarna\_frame,side="left", expand=False)

def u751(event=None):

text1.insert(INSERT,u'\u095d',"ttags\_"+str(tagno))

text1.see(INSERT)

b573=Tkinter.Button(banglakey, text =u'\u095d', command = u751,width=2,bg="white",fg="black",font=("amar bangla",20))

b573.pack(in\_=kbarna\_frame,side="left", expand=False)

def u752(event=None):

text1.insert(INSERT,u'\u095e',"ttags\_"+str(tagno))

text1.see(INSERT)

b573=Tkinter.Button(banglakey, text =u'\u095e', command = u752,width=2,bg="white",fg="black",font=("amar bangla",20))

b573.pack(in\_=kbarna\_frame,side="left", expand=False)

def u753(event=None):

text1.insert(INSERT,u'\u095f',"ttags\_"+str(tagno))

text1.see(INSERT)

b573=Tkinter.Button(banglakey, text =u'\u095f', command = u753,width=2,bg="white",fg="black",font=("amar bangla",20))

b573.pack(in\_=kbarna\_frame,side="left", expand=False)

def u754(event=None):

text1.insert(INSERT,u'\u0954',"ttags\_"+str(tagno))

text1.see(INSERT)

b573=Tkinter.Button(banglakey, text =u'\u0954', command = u754,width=2,bg="white",fg="black",font=("amar bangla",20))

b573.pack(in\_=kbarna\_frame,side="left", expand=False)

def u755(event=None):

text1.insert(INSERT,u'\u0902',"ttags\_"+str(tagno))

text1.see(INSERT)

b573=Tkinter.Button(banglakey, text =u'\u0902', command = u755,width=2,bg="white",fg="black",font=("amar bangla",20))

b573.pack(in\_=kbarna\_frame,side="left", expand=False)

def u756(event=None):

text1.insert(INSERT,u'\u0903',"ttags\_"+str(tagno))

text1.see(INSERT)

b573=Tkinter.Button(banglakey, text =u'\u0903', command = u756,width=2,bg="white",fg="black",font=("amar bangla",20))

b573.pack(in\_=kbarna\_frame,side="left", expand=False)

def u757(event=None):

text1.insert(INSERT,u'\u094d',"ttags\_"+str(tagno))

text1.see(INSERT)

b573=Tkinter.Button(banglakey, text =u'\u094d', command = u757,width=2,bg="white",fg="black",font=("amar bangla",20))

b573.pack(in\_=kbarna\_frame,side="left", expand=False)

kbarna\_frame.place(bordermode=OUTSIDE,x=510,y=495 , height=50, width=500)

kar\_frame=Tkinter.Frame(banglakey,borderwidth=1, relief="flat")

def u55(event=None):

text1.insert(INSERT,u'\u093E',"ttags\_"+str(tagno))

text1.see(INSERT)

b553=Tkinter.Button(banglakey, text =u'\u093E', command = u55,width=2,bg="white",fg="black",font=("amar bangla",20))

b553.pack(in\_=kar\_frame,side="left", expand=False)

def u56(event=None):

text1.insert(INSERT,u'\u093F',"ttags\_"+str(tagno))

text1.see(INSERT)

b554=Tkinter.Button(banglakey, text =u'\u093F', command = u56,width=2,bg="white",fg="black",font=("amar bangla",20))

b554.pack(in\_=kar\_frame,side="left", expand=False)

def u57(event=None):

text1.insert(INSERT,u'\u0940',"ttags\_"+str(tagno))

text1.see(INSERT)

b555=Tkinter.Button(banglakey, text =u'\u0940', command = u57,width=2,bg="white",fg="black",font=("amar bangla",20))

b555.pack(in\_=kar\_frame,side="left", expand=False)

def u58(event=None):

text1.insert(INSERT,u'\u0941',"ttags\_"+str(tagno))

text1.see(INSERT)

b556=Tkinter.Button(banglakey, text =u'\u0941', command = u58,width=2,bg="white",fg="black",font=("amar bangla",20))

b556.pack(in\_=kar\_frame,side="left", expand=False)

def u59(event=None):

text1.insert(INSERT,u'\u0942',"ttags\_"+str(tagno))

text1.see(INSERT)

b557=Tkinter.Button(banglakey, text =u'\u0942', command = u59,width=2,bg="white",fg="black",font=("amar bangla",20))

b557.pack(in\_=kar\_frame,side="left", expand=False)

def u60(event=None):

text1.insert(INSERT,u'\u0943',"ttags\_"+str(tagno))

text1.see(INSERT)

b558=Tkinter.Button(banglakey, text =u'\u0943', command = u60,width=2,bg="white",fg="black",font=("amar bangla",20))

b558.pack(in\_=kar\_frame,side="left", expand=False)

def u61(event=None):

text1.insert(INSERT,u'\u0947',"ttags\_"+str(tagno))

text1.see(INSERT)

b559=Tkinter.Button(banglakey, text =u'\u0947', command = u61,width=2,bg="white",fg="black",font=("amar bangla",20))

b559.pack(in\_=kar\_frame,side="left", expand=False)

def u62(event=None):

text1.insert(INSERT,u'\u0948',"ttags\_"+str(tagno))

text1.see(INSERT)

b560=Tkinter.Button(banglakey, text =u'\u0948', command = u62,width=2,bg="white",fg="black",font=("amar bangla",20))

b560.pack(in\_=kar\_frame,side="left", expand=False)

def u63(event=None):

text1.insert(INSERT,u'\u094B',"ttags\_"+str(tagno))

text1.see(INSERT)

b561=Tkinter.Button(banglakey, text =u'\u094B', command = u63,width=2,bg="white",fg="black",font=("amar bangla",20))

b561.pack(in\_=kar\_frame,side="left", expand=False)

def u64(event=None):

text1.insert(INSERT,u'\u094C',"ttags\_"+str(tagno))

text1.see(INSERT)

b562=Tkinter.Button(banglakey, text =u'\u094C', command = u64,width=2,bg="white",fg="black",font=("amar bangla",20))

b562.pack(in\_=kar\_frame,side="left", expand=False)

kar\_frame.place(bordermode=OUTSIDE,x=510,y=550 , height=50, width=450)

label5=Tkinter.Label(banglakey,text=u'मात्रा',font=('siyam rupali',15))

label5.place(x=935,y=560)

num\_frame=Tkinter.Frame(banglakey,borderwidth=1, relief="flat")

def u65(event=None):

text1.insert(INSERT,u'\u0966',"ttags\_"+str(tagno))

text1.see(INSERT)

b563=Tkinter.Button(banglakey, text =u'\u0966', command = u65,width=2,bg="white",fg="black",font=("amar bangla",20))

b563.pack(in\_=num\_frame,side="left", expand=False)

def u66(event=None):

text1.insert(INSERT,u'\u0967',"ttags\_"+str(tagno))

text1.see(INSERT)

b564=Tkinter.Button(banglakey, text =u'\u0967', command = u66,width=2,bg="white",fg="black",font=("amar bangla",20))

b564.pack(in\_=num\_frame,side="left", expand=False)

def u67(event=None):

text1.insert(INSERT,u'\u0968',"ttags\_"+str(tagno))

text1.see(INSERT)

b565=Tkinter.Button(banglakey, text =u'\u0968', command = u67,width=2,bg="white",fg="black",font=("amar bangla",20))

b565.pack(in\_=num\_frame,side="left", expand=False)

def u68(event=None):

text1.insert(INSERT,u'\u0969',"ttags\_"+str(tagno))

text1.see(INSERT)

b566=Tkinter.Button(banglakey, text =u'\u0969', command = u68,width=2,bg="white",fg="black",font=("amar bangla",20))

b566.pack(in\_=num\_frame,side="left", expand=False)

def u69(event=None):

text1.insert(INSERT,u'\u096A',"ttags\_"+str(tagno))

text1.see(INSERT)

b567=Tkinter.Button(banglakey, text =u'\u096A', command = u69,width=2,bg="white",fg="black",font=("amar bangla",20))

b567.pack(in\_=num\_frame,side="left", expand=False)

def u70(event=None):

text1.insert(INSERT,u'\u096B',"ttags\_"+str(tagno))

text1.see(INSERT)

b568=Tkinter.Button(banglakey, text =u'\u096B', command = u70,width=2,bg="white",fg="black",font=("amar bangla",20))

b568.pack(in\_=num\_frame,side="left", expand=False)

def u71(event=None):

text1.insert(INSERT,u'\u096C',"ttags\_"+str(tagno))

text1.see(INSERT)

b569=Tkinter.Button(banglakey, text =u'\u096C', command = u71,width=2,bg="white",fg="black",font=("amar bangla",20))

b569.pack(in\_=num\_frame,side="left", expand=False)

def u72(event=None):

text1.insert(INSERT,u'\u096D',"ttags\_"+str(tagno))

text1.see(INSERT)

b570=Tkinter.Button(banglakey, text =u'\u096D', command = u72,width=2,bg="white",fg="black",font=("amar bangla",20))

b570.pack(in\_=num\_frame,side="left", expand=False)

def u73(event=None):

text1.insert(INSERT,u'\u096E',"ttags\_"+str(tagno))

text1.see(INSERT)

b571=Tkinter.Button(banglakey, text =u'\u096E', command = u73,width=2,bg="white",fg="black",font=("amar bangla",20))

b571.pack(in\_=num\_frame,side="left", expand=False)

def u74(event=None):

text1.insert(INSERT,u'\u096F',"ttags\_"+str(tagno))

text1.see(INSERT)

b572=Tkinter.Button(banglakey, text =u'\u096F', command = u74,width=2,bg="white",fg="black",font=("amar bangla",20))

b572.pack(in\_=num\_frame,side="left", expand=False)

num\_frame.place(bordermode=OUTSIDE,x=510,y=600 , height=50, width=450)

label3=Tkinter.Label(banglakey,text=u'संख्या',font=('siyam rupali',15))

label3.place(x=935,y=615)

menu\_frame=Tkinter.Frame(banglakey,borderwidth=1, relief="flat")

b577=Tkinter.Button(banglakey, text =u'HELP', command = None,width= 7,bg="white",fg="black",font=("arial",14))

b577.pack(in\_=menu\_frame,side="top", expand=False)

b578=Tkinter.Button(banglakey, text =u'OPEN', command = u78,width= 7,bg="white",fg="black",font=("arial",14))

b578.pack(in\_=menu\_frame,side="top", expand=False)

b579=Tkinter.Button(banglakey, text =u'SAVE', command = u79,width= 7,bg="white",fg="black",font=("arial",14))

b579.pack(in\_=menu\_frame,side="top", expand=False)

def u80(event=None):

'''This function first saves the file and then uses PyQt to pdisplay and print'''

import os

global f

if os.name=='nt' or os.name=='poisx':

if f=='':

tkMessageBox.showinfo('Alert','You will now be provided option for saving your current file')

f = asksaveasfilename(defaultextension=".egpd")

if f !='':

prev='''<html>

<head>

<meta http-equiv="Content-Language" content="bn">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>'''+str(f)+'''</title>

</head>

<body><p>'''

afte='''</p>

</body>

</html>'''

with io.open(f,'w',encoding='utf8') as opf:

opf.write(prev+texttohtml()+afte)

opf.close()

else:

prev='''<html>

<head>

<meta http-equiv="Content-Language" content="bn">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<title>'''+str(f)+'''</title>

</head>

<body><p>'''

afte='''</p>

</body>

</html>'''

with io.open(f,'w',encoding='utf8') as opf:

opf.write(prev+texttohtml()+afte)

opf.close()

import os

if f is not None:

import printinghu

printinghu.call(f)

else:

tkMessageBox.showinfo('Error','Printer option not supported on this system')

b580=Tkinter.Button(banglakey, text =u'PRINT', command = u80,width= 7,bg="white",fg="black",font=("arial",14))

b580.pack(in\_=menu\_frame,side="top", expand=False)

menu\_frame.place(bordermode=OUTSIDE,x=805,y=150 , height=160, width=75)

label5=Tkinter.Label(banglakey,text=u'मदद(?) \n\n फ़ाइल खुल \n\n पाठ संरक्षण \n\n छाप',font=('siyam rupali',13))

label5.place(x=900,y=160)

#code below here checks for file name change and changes title

if f=='':

banglakey.title('Untitled-EnigmaPad')

else:

banglakey.title(str(f)+'-EnigmaPad')

if f.split('.')[-1]=='html' or f.split('.')[-1]=='htm' or f.split('.')[-1]=='egpd':

with io.open(f,'r',encoding='utf8') as opf:

text11 = opf.read()

read\_html(text11)

banglakey.title(str(f)+'-EnigmaPad')

elif f.split('.')[-1]=='txt':

with io.open(f,'r',encoding='utf8') as opf:

text11 = opf.read()

text1.insert(INSERT,text11,"ttags\_"+str(tagno))

banglakey.title(str(f)+'-EnigmaPad')

else:

tkMessageBox.showinfo('Error!','Specified format not supported '+f.split('.')[-1])

f=''

banglakey.title('Untitled-EnigmaPad')

banglakey.mainloop()

**FUTURE APPLICATION**

The project has vast future application. The write in registry feature can be used in other applications. This software could be more developed and can be used for more applications using hindi as a language. Other than that the knowledge of making GUI applications can be used for making various other projects.

**BIBLIOGRAPHY**

Python itself is such a easy to learn language that it motivates any one to learn it. The documents provided with Python are of great help for reference. Apart from this the following websites have proven themselves to be of great help for me. The websites are as follows:

* http://www.tutorialspoint.com
* http://effbot.org
* http://stackoverflow.com
* http://zetcode.com
* http://tkinter.unpythonic.net
* http://pythoncentral.io
* https://docs.python.org