WOLKITE UNIVERSITY

COLLEGE OF COMPUTING AND INFORMATICS

DEPARTMENT OF INFORMATION SYSTEMS

LAB MANUAL

COURSE NAME: INTRODUCTION TO INFORMATION STORAGE AND RETRIEVAL

COURSE CODE: INSY2063

COMPILED BY: ISAYAS WAKGARI KELBESSA **(M.S.c)**

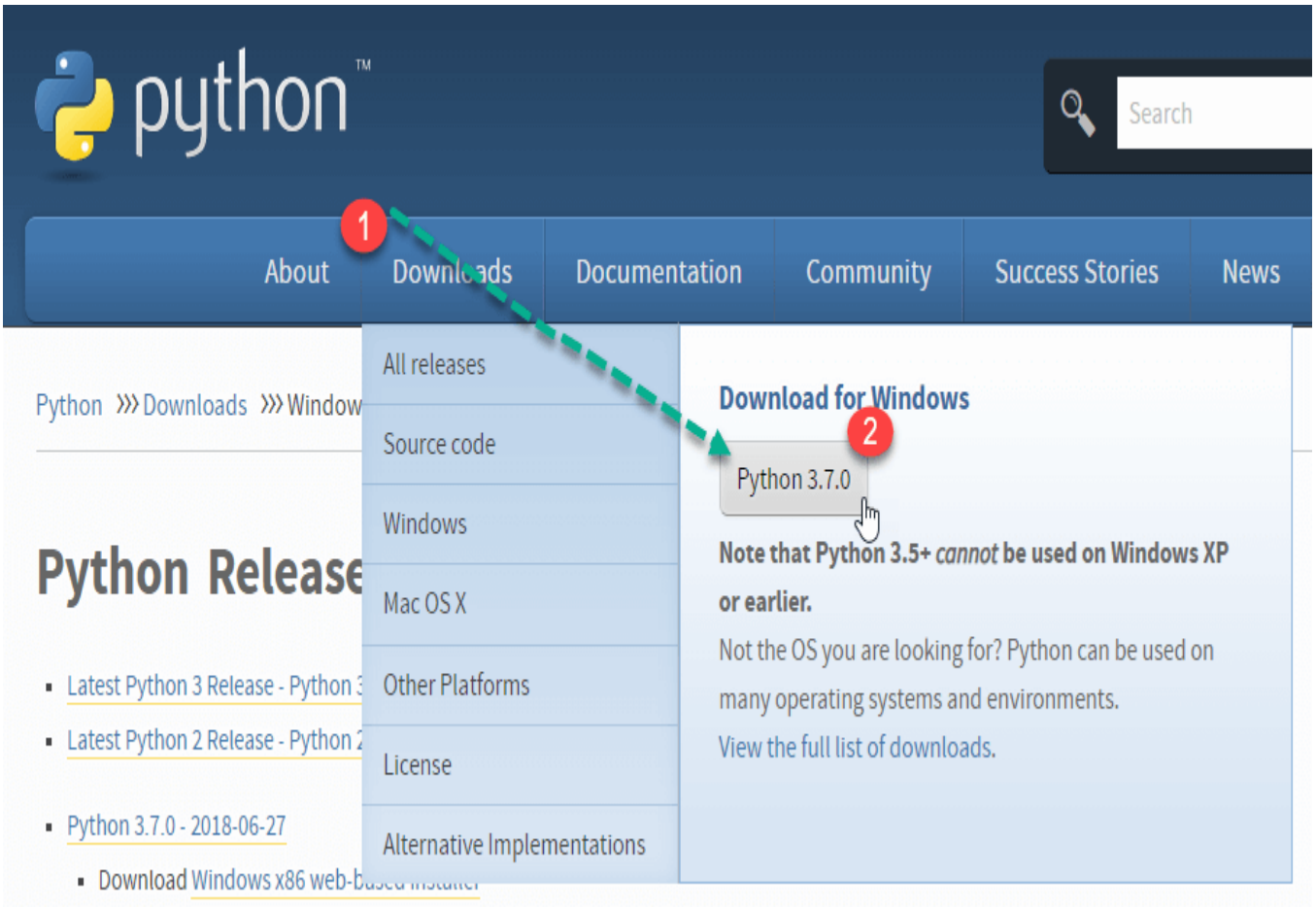
WOLKITE, ETHIOPIA

Installing NLTK in Windows

In this part, we will learn that how to make setup NLTK (Natural Language ToolKit) via terminal (Command prompt in windows). The instructions given below are based on the assumption that you don't have python installed. So, first step is to install python.

Installing Python in Windows:

Step 1) Go to link **https://www.python.org/downloads/,** and select the latest version for windows.



Note: If you don't want to download the latest version, you can visit the download tab and see all releases.

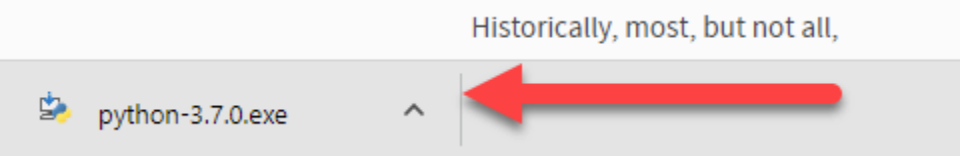
Looking for a specific release?

Python releases by version number:

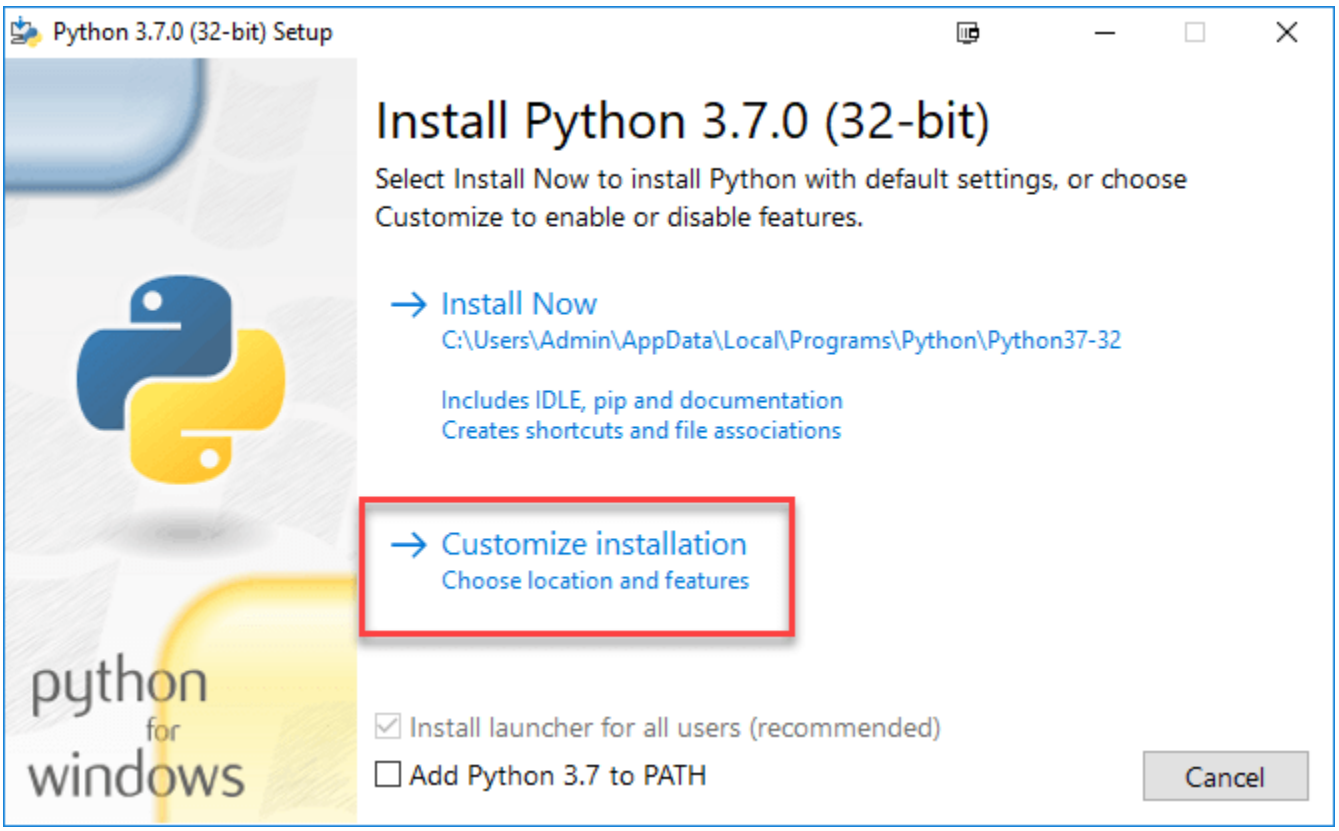
Release version	Release date		Click for more
Python 3.5.6	2018-08-02	Download	Release Notes
Python 3.4.9	2018-08-02	Download	Release Notes
Python 3.7.0	2018-06-27	Download	Release Notes
Python 3.6.6	2018-06-27	Download	Release Notes
Python 2.7.15	2018-05-01	Download	Release Notes
Python 3.6.5	2018-03-28	Download	Release Notes
Python 3.4.8	2018-02-05	Download	Release Notes
Python 3.5.5	2018-02-05	Download	Release Notes

View older releases

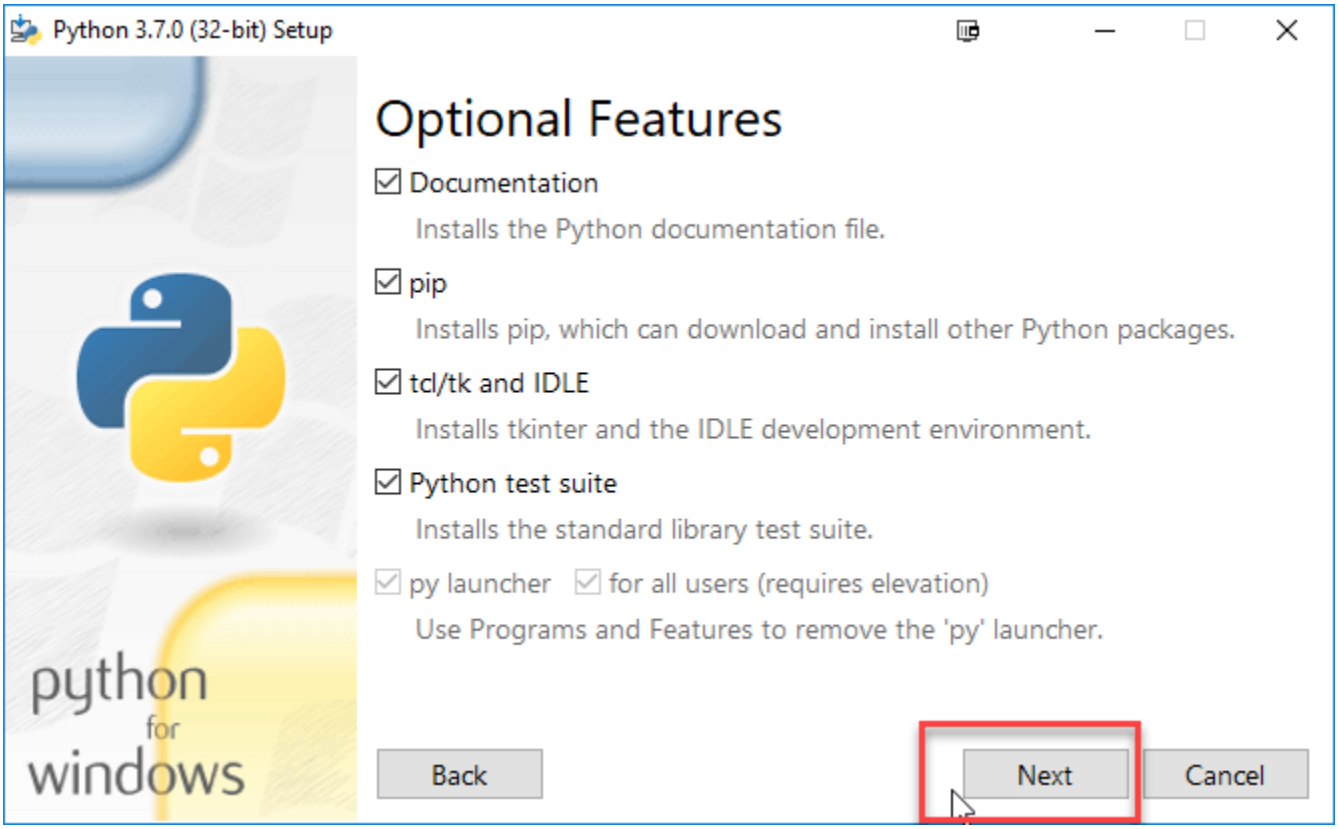
Step 2) Click on the Downloaded File



Step 3) Select Customize Installation

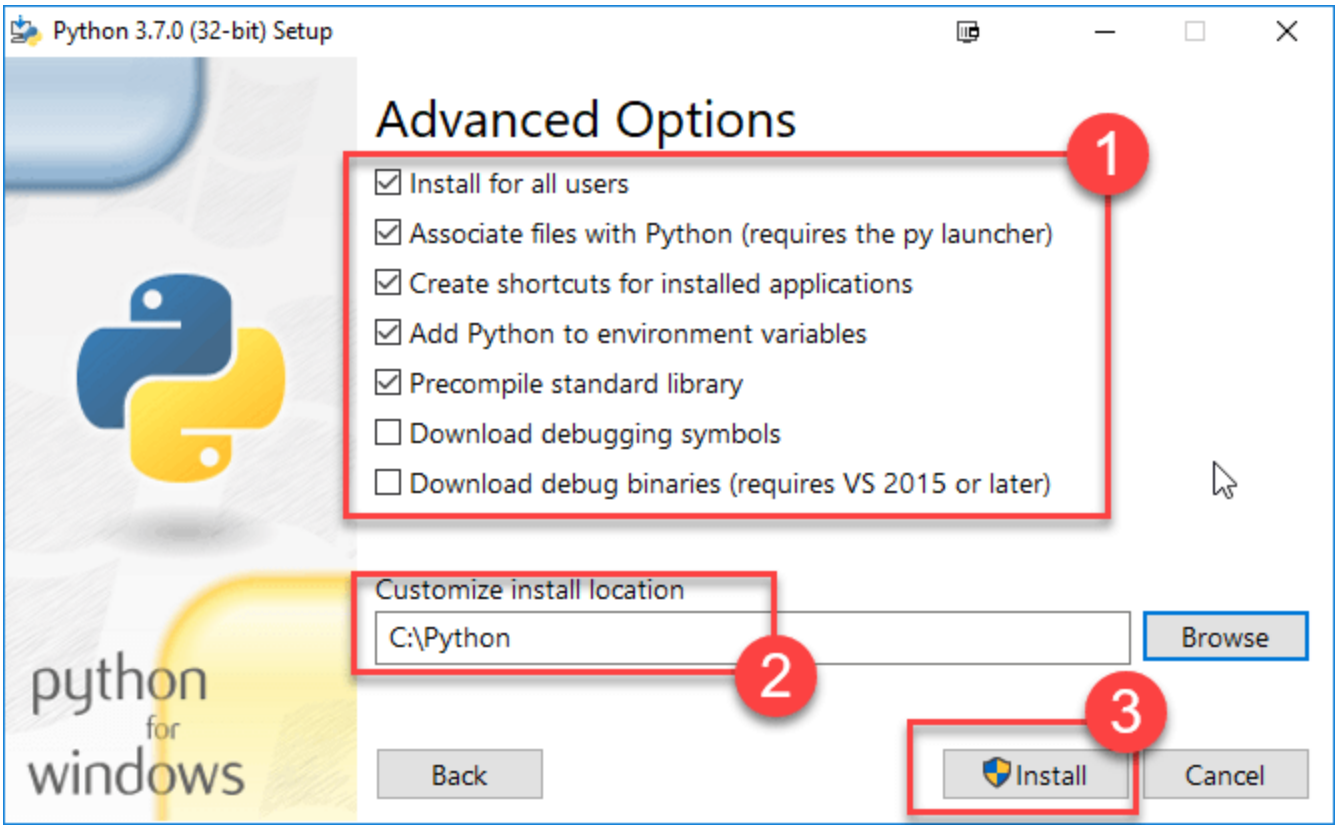


Step 4) Click NEXT

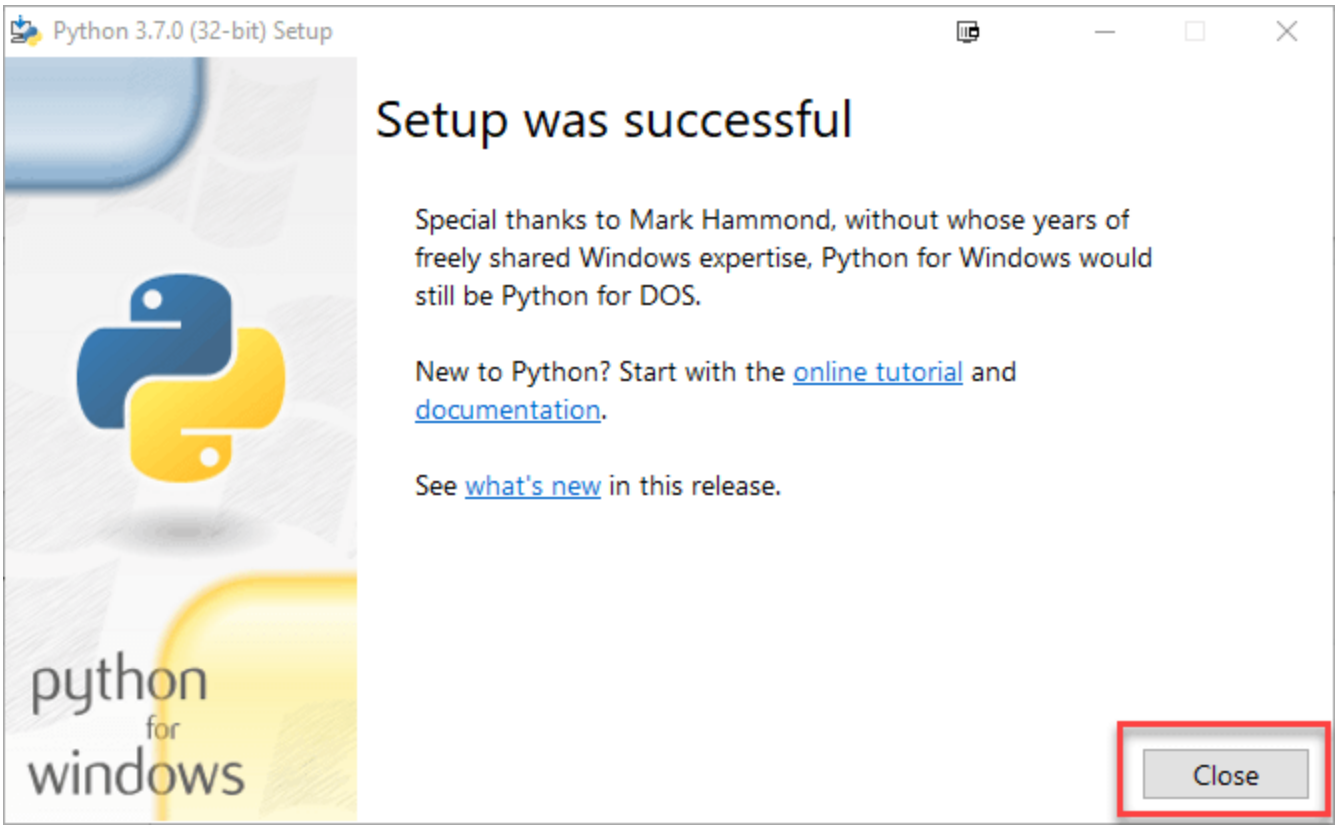


Step 5) In next screen

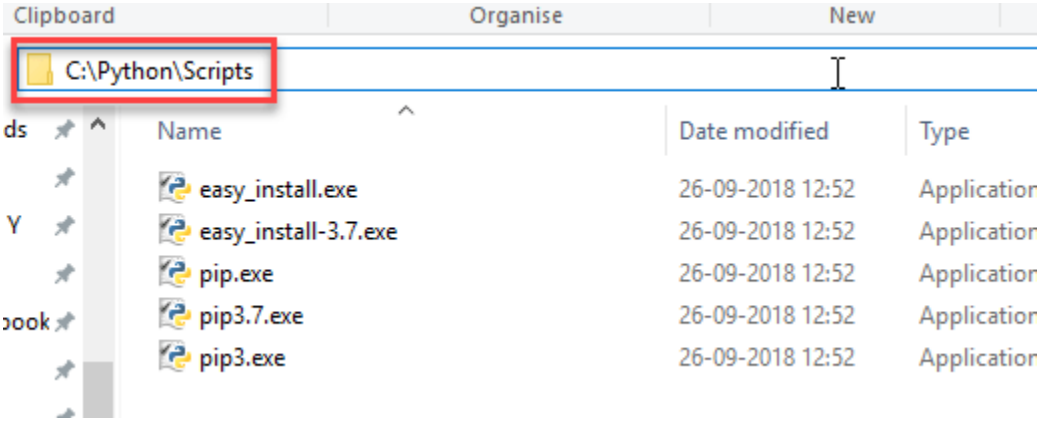
1. Select the advanced options
2. Give a Custom install location. In my case, a folder on C drive is chosen for ease in operation
3. Click Install



Step 6) Click Close button once install is done.



Step 7) Copy the path of your Scripts folder.



Step 8) In windows command prompt

- Navigate to the location of the pip folder
- Enter command to install NLTK

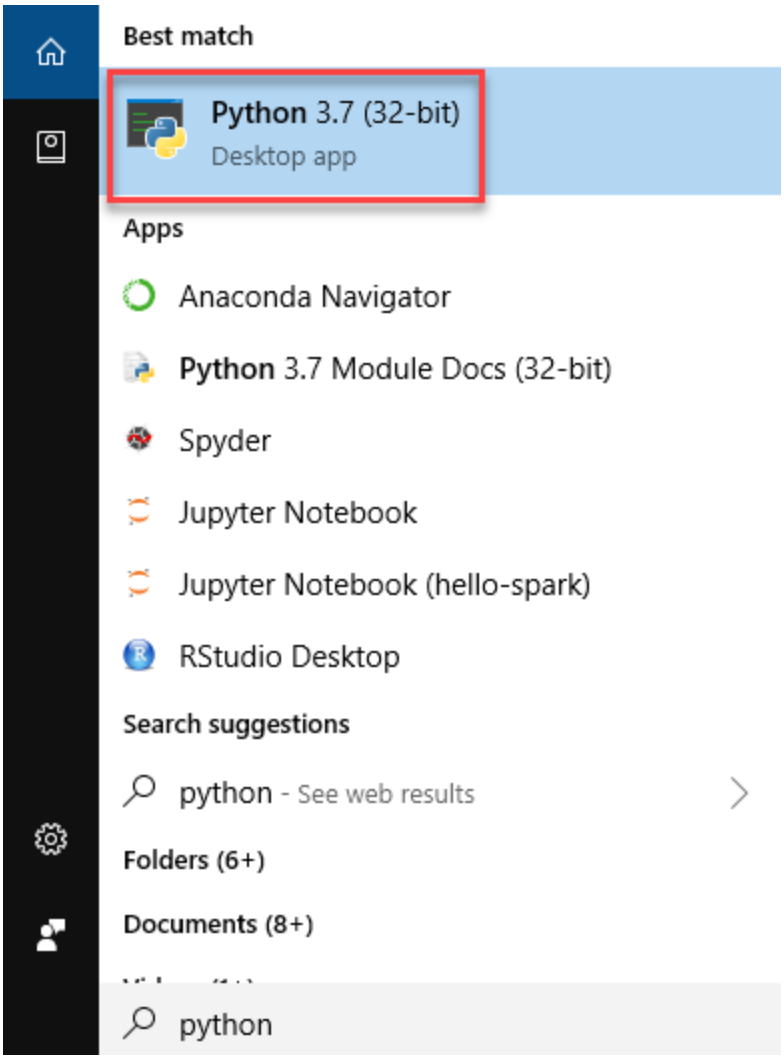
  pip3 install nltk

- Installation should be done successfully

```
Command Prompt

C:\Users\Admin>cd C:\Python\Scripts 1

C:\Python\Scripts>pip3 install nltk 2
Collecting nltk
  Downloading https://files.pythonhosted.org/packages/50/09/3b1755d528ad9156ee7243d52aa5cd2b809e
f053a0f31b53d92853dd653a/nltk-3.3.0.zip (1.4MB)
    100% |                                | 1.4MB 3.9MB/s
Collecting six (from nltk)
  Using cached https://files.pythonhosted.org/packages/67/4b/141a581104b1f6397bfa78ac9d43d8ad29a
7ca43ea90a2d863fe3056e86a/six-1.11.0-py2.py3-none-any.whl
Installing collected packages: six, nltk
  Running setup.py install for nltk ... done
Successfully installed nltk-3.3 six-1.11.0 3
You are using pip version 10.0.1, however version 18.0 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Python\Scripts>
```

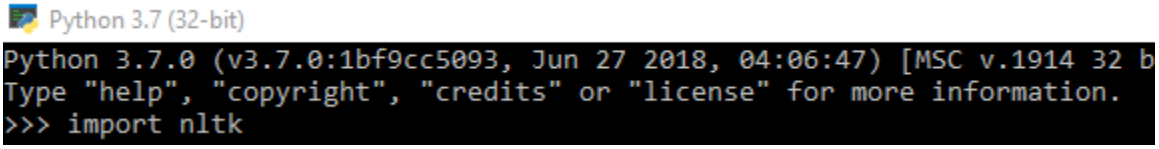
NOTE: For Python2 use the command pip2 install nltk

Step 9) In Windows Start Menu, search and open PythonShell



Step 10) You can verify whether the installation is accurate supplying the below command

import nltk



If you see no error, Installation is complete.

NLTK Dataset

NLTK module has many datasets available that you need to download to use. More technically it is called **corpus**. Some of the examples are **stopwords**, **gutenberg**, **framenet_v15**, **large_grammars**and so on.

How to Download all packages of NLTK

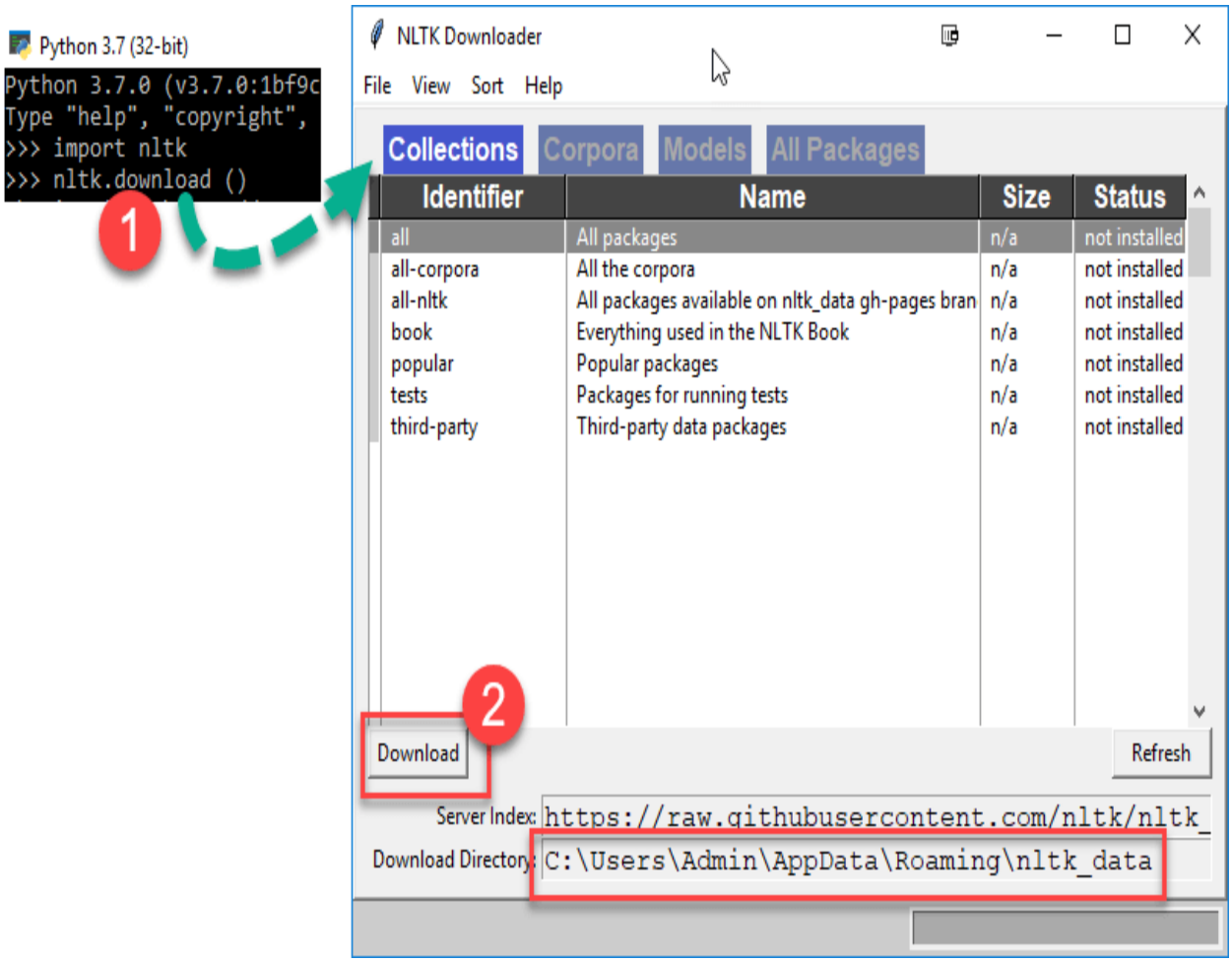
Step 1) Run the Python interpreter in Windows or Linux

Step 2)

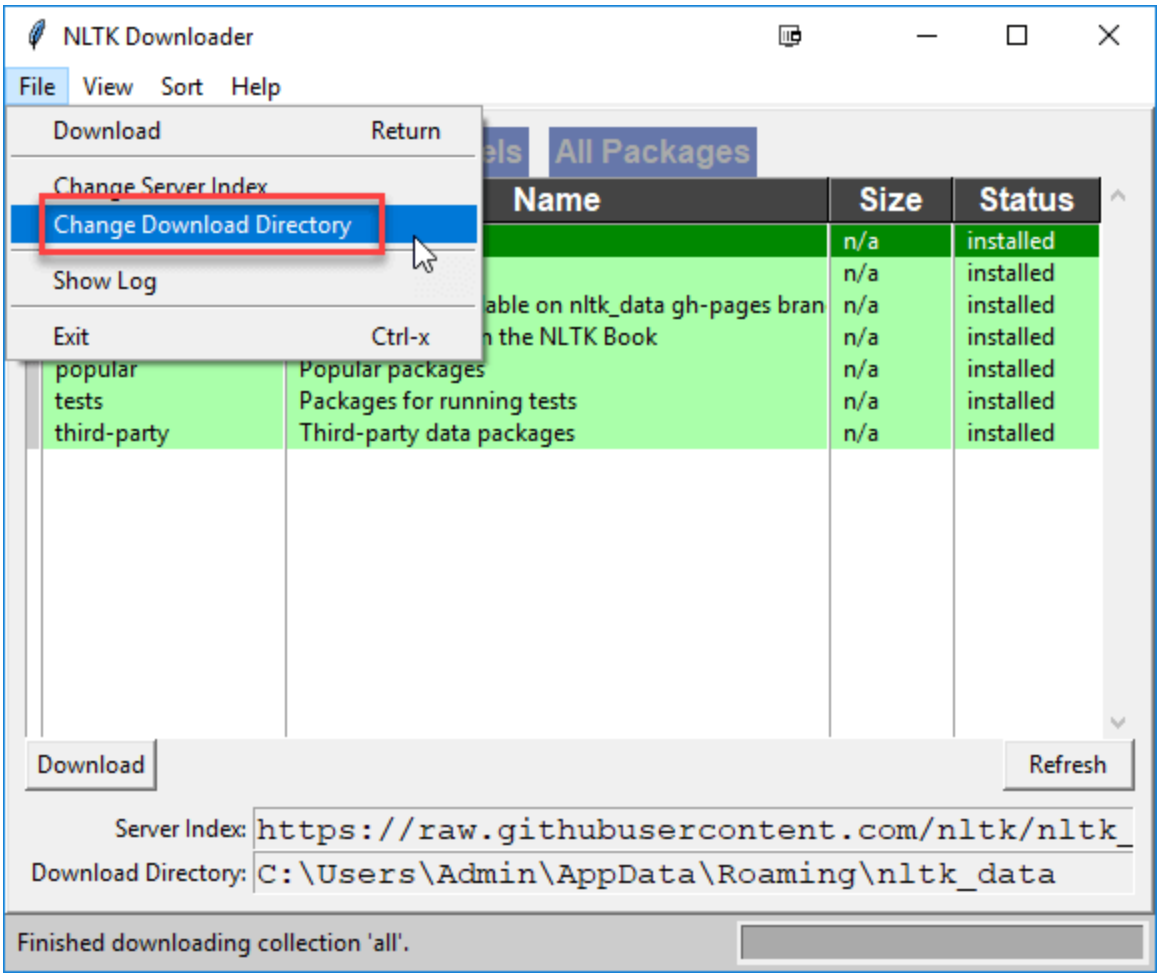
1. Enter the commands

```
import nltk
nltk.download ()
```

2. NLTK Downloaded Window Opens. Click the Download Button to download the dataset. This process will take time, based on your internet connection



NOTE: You can change the download location by Clicking File> Change Download Directory



Step 3) To test the installed data use the following code

>>> from nltk.corpus import brown

>>>brown.words()

['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]

```
>>> from nltk.corpus import brown
>>> brown.words()
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
>>>
```

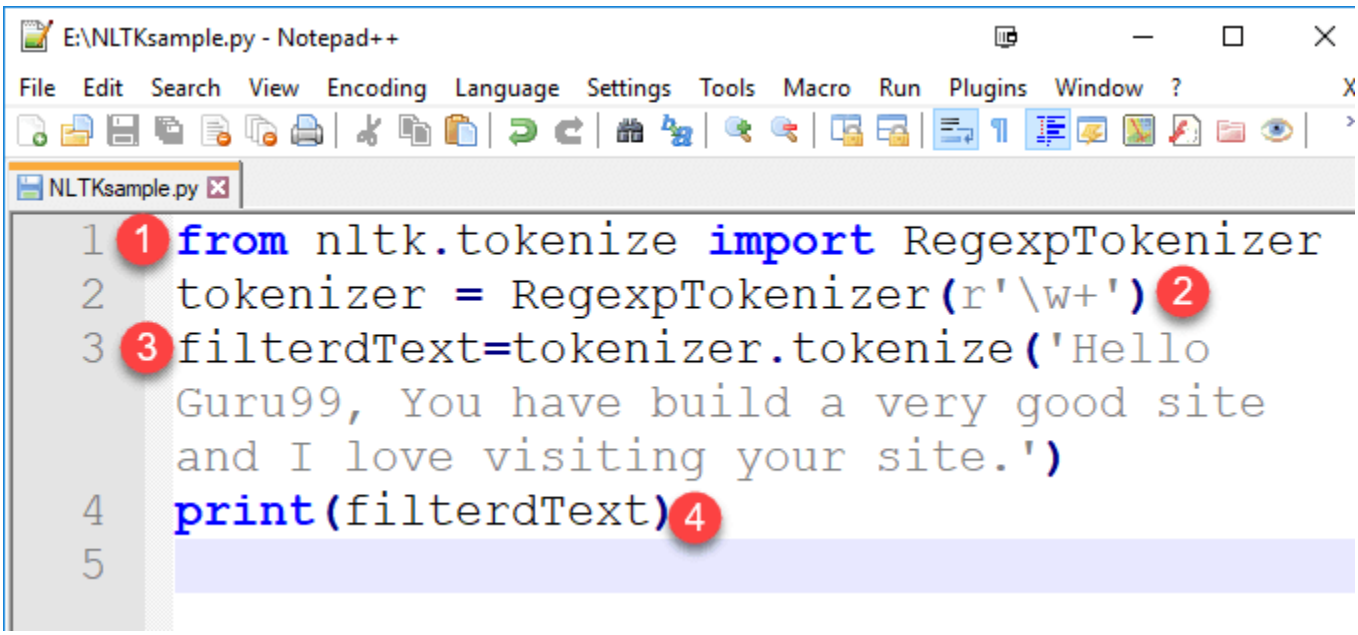
Running the NLP Script

We are going to discuss how NLP script will be executed on our local PC. There are many libraries for Natural Language Processing present in the market. So choosing a library depends on fitting your requirements. Here is the list of NLP libraries

How to Run NLTK Script

Step1) In your favorite code editor, copy the code and save the file as "NLTKsample.py"

```
from nltk.tokenize import RegexpTokenizer
tokenizer = RegexpTokenizer(r'\w+')
filterdText=tokenizer.tokenize('Hello Guru99, You have build a very good site and I love visiting your site.')
print(filterdText)
```



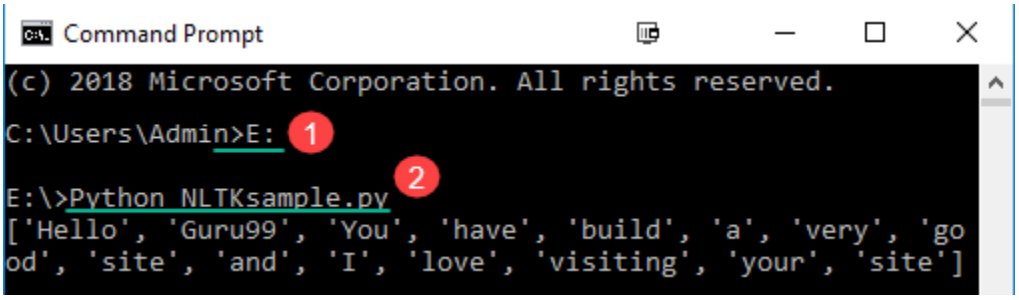
Code Explanation:

1. In this program, the objective was to remove all type of punctuations from given text. We imported "RegexpTokenizer" which is a module of NLTK. It removes all the expression, symbol, character, numeric or any things whatever you want.

2. You just have passed the regular Expression to the "RegexpTokenizer" module.
3. Further, we tokenized the word using "tokenize" module. The output is stored in the "filterdText" variable.
4. And printed them using "print()."

Step2) In the command prompt

- Navigate to the location where you have saved the file
- Run the command Python NLTKsample.py



This will show output as :

['Hello', 'Guru99', 'You', 'have', 'build', 'a', 'very', 'good', 'site', 'and', 'I', 'love', 'visiting', 'your', 'site']

TEXT PREPROCESSING

Preprocessing is the process of *controlling the size of the vocabulary* or the number of distinct words used as index terms. Preprocessing will lead to an improvement in the information retrieval performance.

What is Tokenization?

Tokenization is the process by which big quantity of text is divided into smaller parts called **tokens**.

Tokenization: Tokenization is the process (task) of chopping it up into pieces, which is called tokens, and also at the same time it is the process of throwing away some characters, illuminating punctuation and special characters in a given character sequence and in a document unit

If we have the original document *'Oromiyaan qabeenya uumamaa bareedaa qabdi*', then the tokens are: *'Oromiyaan','qabeenya','uumamaa', 'bareedaa', 'qabdi'.* However, there are challenges related to tokenization in every language. The first challenge is differentiating single word from compound word. For example, if we take the word '*harka-qalleessa*' to mean '*poor*'; if it is tokenized as '*harka*' and '*qalleessa*', the meaning of the word will be changed. Because,we cannot separate such like words from one another in Afaan Oromo. Such like challenge is not only in Afaan Oromo, but also in other languages such as English. Example, words like*: San Francisco, Hewlett-Packard Versus Hewlett and Packard, Addis Ababa Versus Addis Ababa* are common in English.

Natural language processing is used for building applications such as Text classification, intelligent chatbot, sentimental analysis, language translation, etc. It becomes vital to understand the pattern in the text to achieve the above-stated purpose. **This token are very useful for finding such patterns as well as is considered as a base step for stemming and lemmatization.**

For the time being, don't worry about stemming and lemmatization but treat them as steps for textual data cleaning using NLP (Natural language processing). We will discuss stemming and

lemmatization later in the tutorial. Tasks such as **Text classification or spam filtering** makes use of NLP along with deep learning libraries such as Keras and Tensorflow.

Natural Language toolkit has very important module **tokenize** which further compromises of sub-modules

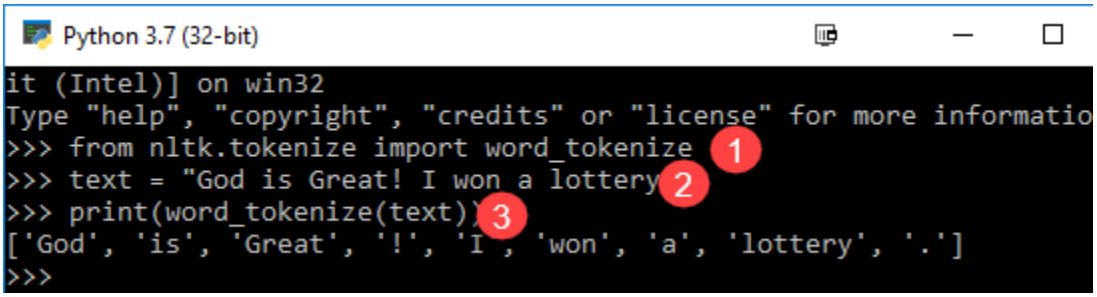
1. word tokenize
2. sentence tokenize

Tokenization of words

We use the method **word_tokenize()** to split a sentence into words. The output of word tokenization can be converted to Data Frame for better text understanding in machine learning applications. It can also be provided as input for further text cleaning steps such as punctuation removal, numeric character removal or stemming. Machine learning models need numeric data to be trained and make a prediction. Word tokenization becomes a crucial part of the text (string) to numeric data conversion. Please read about Bag of Words or CountVectorizer. Please refer to below example to understand the theory better.

```
from nltk.tokenize import word_tokenize
text = "God is Great! I won a lottery."
print(word_tokenize(text))
```

Output: ['God', 'is', 'Great', '!', 'I', 'won', 'a', 'lottery', '.']



Code Explanation

1. word_tokenize module is imported from the NLTK library.
2. A variable "text" is initialized with two sentences.
3. Text variable is passed in word_tokenize module and printed the result. This module breaks each word with punctuation which you can see in the output.

Tokenization of Sentences

Sub-module available for the above is sent_tokenize. An obvious question in your mind would be **why sentence tokenization is needed when we have the option of word tokenization**. Imagine you need to count average words per sentence, how you will calculate? For accomplishing such a task, you need both sentence tokenization as well as words to calculate the ratio. Such output serves as an important feature for machine training as the answer would be numeric.

Check the below example to learn how sentence tokenization is different from words tokenization.

```
from nltk.tokenize import sent_tokenize
text = "God is Great! I won a lottery."
print(sent_tokenize(text))
```

Output: ['God is Great!', 'I won a lottery ']

We have **12** words and **two sentences** for the same input.

```
>>> from nltk.tokenize import sent_tokenize 1
>>> text = "God is Great! I won a lottery." 2
>>> print(sent_tokenize(text)) 3
['God is Great!', 'I won a lottery.']
>>>
```

Explanation of the program:

1. In a line like the previous program, imported the sent_tokenize module.

2. We have taken the same sentence. Further sent module parsed that sentences and show output. It is clear that this function breaks each sentence.

Above examples are good settings stones to understand the mechanics of the word and sentence tokenization.

Using immediate mode

- **Removing punctuation**

```
>>> p="string. With. Punctuation?"
>>> import string
>>> for c in string.punctuation:
...     p=p.replace(c,"")
```

Tokenization example 1:

```
>>> import nltk
>>> m="my favorite color is blue"
>>> nltk.word_tokenize(m)
```

The output will be: ➔['my', 'favorite', 'color', 'is', 'blue']

My file name

and ‘ r’ shows that to read the file

Example of tokenization 2

```
with open('tokenizationexample.txt','r')  as myfile:
    file=myfile.read()
    myList=[]
    myList.extend(file.split())
for i in myList:
    print(i,end='\n')
    print('____________________')
```

Remove the punctuation and Tokenize

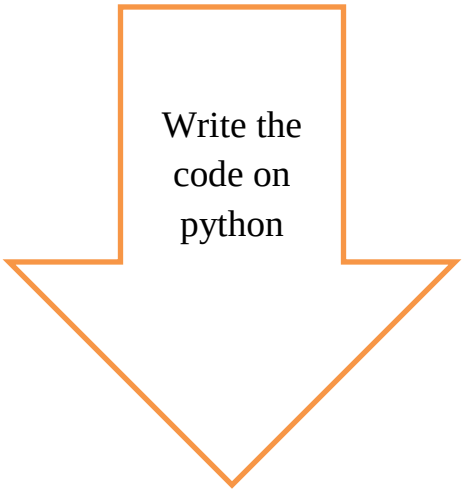
```
with open('one.txt','r')as myFile:
    str1=myFile.read()
    print('===The given Statement is:====')
    print(str1)
    punctuation = ['(', ')', '?', ':', ';', ',', '.', '!', '/', '"', '"']
for i in punctuation:
        str1 = str1.replace(i," ")
        myList=[]
        myList.extend(str1.split(" "))
print (str1)
for i in myList:
    print(i,end='\n')
    print ("_____")
```

Stop word removal

Sometimes, some *extremely common words* which would appear to be of **little value** in helping select documents matching a user need are excluded from the vocabulary entirely. These words are called **stop words**. **However dropping stop words isn't always useful**. The phrase query '**President of the United States**' which contain two stopwords 'of' and 'the' is more useful than search query '**President United States**'. Some common quotes ('**to be or not to be**'), phrases, poems, song lyrics (**Let It Be**) contain a lot of stop words and excluding them will make *information retrieval more difficult*.

- The general trend in IR systems over time has been from standard use of quite large stop lists (200–300 terms) to very small stop lists (7–12 terms) to no stop list whatsoever.
- Web search engines generally do not use stop lists.
- Ultimately using stop word lists depends on IR tasks and the kind of documents.

To see lists of stopwords of English language use the following code




```
from nltk.corpus import stopwords

stop_word=set(stopwords.words("english"))

print(stop_word)
```

The output will be:

```
Python 3.5 (32-bit)
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:16:59) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from nltk.corpus import stopwords
>>> stop_word=set(stopwords.words("english"))
>>> print(stop_word)
{'have', "aren't", 'those', 'why', 'some', 'couldn', 'our', 'there', 'you', 'hadn', 'so', 'at', "wouldn't", 'doesn', 'ab
out', 'mightn', 'should', 'too', 'haven', 'your', 'as', 'them', "shan't", 'shan', 'both', 'nor', 'won', 'having', "you're
", 'she', 'hers', 'after', 'or', 'once', 'of', 'above', 'o', 'while', 'did', 'isn', 'will', 'don', "mustn't", 'aren', '
few', 'whom', "didn't", "don't", 'from', 'down', 'and', 'such', 'if', 'his', 'under', 'the', 'herself', 'by', 'what', "d
oesn't", 'where', 'on', 're', 'himself', 'more', 'myself', 'because', "hadn't", 'has', 'up', 's', "that'll", 'other', 'm
', 'off', 'wasn', 'than', 'i', 'her', 'are', 'ma', 'its', 'll', 'my', 'same', 'out', 'for', 'that', 'each', "you'd", "yo
u've", 'wouldn', 'he', 'weren', 'who', 'doing', 'shouldn', 'we', 'most', 'to', 'before', 'no', 'yourself', 'is', 'throug
h', 'can', 'further', 'were', 'had', 'now', "couldn't", 'ourselves', "shouldn't", 'do', 'being', 'during', 'does', 'your
selves', 'here', 'was', 'but', 'not', 'me', 'in', 'how', 'their', 'only', 'any', 'didn', 'him', 'again', 'ours', 'd', 'a
gainst', "weren't", "haven't", "you'll", 've', 'needn', "isn't", "she's", 'very', 'an', 'between', 'until', 'below', 'it
self', 'be', 'it', "mightn't", 'these', 't', 'this', 'all', 'they', 'a', 'ain', 'been', 'themselves', 'just', 'own', 'th
en', 'y', 'which', "hasn't", 'hasn', 'am', 'with', 'when', 'mustn', 'theirs', 'yours', "wasn't", "it's", 'over', 'into',
 "needn't", "should've", "won't"}
>>>
```

Example

print('==========This is stop word removal=========')

with open('one.txt','r')as myFile:

    str1=myFile.read()

    stop_words ="not", "is", "it", "By","between","This","By","A","when","And","up","Then","was","by","It","If","can","an","he","This","or","And","a","i","it","am","at","on","in","of","to","is","so","too","my","the","and","but","are","very","here","even","from","them","then","than","this","that","though","be","But","these"

    myList=[]

    myList.extend(str1.split(" "))

for i in myList:

```
if i not in stop_words:
    print ("_____")
    print(i,end='\n')
```

Normalization

Normalization—After breaking up documents (and also our query) into tokens, the easy case is if tokens in the query just match tokens in the token list of the document.

However, there are many cases when two character sequences are not quite the same but you would like a match to occur. **For instance**, if you search for USA, you might hope to also match documents containing U.S.A.

Normalizing is also very subjective in case of English language. For example we would want to normalize all U.S.A, USA and US to a single token, but not in case of C.A.T & cat or WHO & who. Also we would want to normalize window and windows into a single token but search query for 'Windows OS' should not return window related results

Example

```
with open('tokenizationexample.txt','r') as myfile:
    file=myfile.read()
    myList=[]
    myList.extend(file.split(" "))
    print(file.upper())

print("To convert to lowercase")
print("========================")

with open('tokenizationexample.txt','r') as myfile:
    file=myfile.read()
    myList=[]
    myList.extend(file.split(" "))
    print(file.lower())
```

What is Stemming?

Stemming is a kind of normalization for words. Normalization is a technique where a set of words in a sentence are converted into a sequence to shorten its lookup. The words which have the same meaning but have some variation according to the context or sentence are normalized.

In another word, there is one root word, but there are many variations of the same words. For example, the root word is "eat" and it's variations are "eats, eating, eaten and like so". In the same way, with the help of Stemming, we can find the root word of any variations.

For example

He was riding.
He was taking the ride.

In the above two sentences, the meaning is the same, i.e., riding activity in the past. A human can easily understand that both meanings are the same. But for machines, both sentences are different. Thus it became hard to convert it into the same data row. In case we do not provide the same data-set, then machine fails to predict. So it is necessary to differentiate the meaning of each word to prepare the dataset for machine learning. And here stemming is used to categorize the same type of data by getting its root word.

Let's implement this with a Python program.NLTK has an algorithm named as "PorterStemmer". This algorithm accepts the list of tokenized word and stems it into root word.

Program for understanding Stemming

```
from nltk.stem import PorterStemmer
e_words= ["wait", "waiting", "waited", "waits"]
ps =PorterStemmer()
for w in e_words:
    rootWord=ps.stem(w)
    print(rootWord)
```

Output:

wait

wait

wait

wait

```
from nltk.stem import PorterStemmer
e_words= ["wait", "waiting", "waited", "waits"
ps =PorterStemmer()
for w in e_words:
    rootWord=ps.stem(w)
    print(rootWord)
```

Code Explanation:

- There is a stem module in NLTk which is imported. If ifyou import the complete module, then the program becomes heavy as it contains thousands of lines of codes. So from the entire stem module, we only imported "PorterStemmer."
- We prepared a dummy list of variation data of the same word.
- An object is created which belongs to class nltk.stem.porter.PorterStemmer.
- Further, we passed it to PorterStemmer one by one using "for" loop. Finally, we got output root word of each word mentioned in the list.

From the above explanation, it can also be concluded that stemming is considered as an important preprocessing step because it removed redundancy in the data and variations in the same word. As a result, data is filtered which will help in better machine training.

Now we pass a complete sentence and check for its behavior as an output.

Program:

```
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize
sentence="Hello Guru99, You have to build a very good site and I love visiting your site."
words = word_tokenize(sentence)
ps = PorterStemmer()
for w in words:
        rootWord=ps.stem(w)
        print(rootWord)
```

Output:

```
hello
guru99
,
you
have
build
a
veri
good
site
and
I
love
visit
your
site
```

```
1  from nltk.stem import PorterStemmer
2  from nltk.tokenize import sent_tokenize,
   word_tokenize
3  sentence="Hello Guru99, You have to
   build a very good site and I love
   visiting your site."
4  words = word_tokenize(sentence)
5  ps = PorterStemmer()
6  for w in words:
7          rootWord=ps.stem(w)
8          print(rootWord)
```

Code Explanation

- Package PorterStemer is imported from module stem
- Packages for tokenization of sentence as well as words are imported
- A sentence is written which is to be tokenized in the next step.
- Word tokenization is implemented in this step.
- An object for PorterStemmer is created here.
- Loop is run and stemming of each word is done using the object created in the code line 5

Conclusion:

Stemming is a data-preprocessing module. The English language has many variations of a single word. These variations create ambiguity in machine learning training and prediction. To create a successful model, it's vital to filter such words and convert to the same type of sequenced data using stemming. Also, this is an important technique to get row data from a set of sentence and removal of redundant data also known as normalization.

Python Designing GUI

Python GUI Examples (Tkinter Tutorial)

In this tutorial, we will learn how to develop graphical user interfaces by writing some Python code and GUI examples using the Tkinter package. Tkinter package is shipped (sent) with Python as a standard package, so we don't need to install anything to use it.

You want to build a GUI? Great, here are different fantastic open source libraries to get you started.

1. Tkinter

- ✓ If there were a single package which might be called the "standard" GUI toolkit for Python, it would be Tkinter.
- ✓ a popular graphical interface and language pairing first popularized in the early 90s.

Tkinter is a very powerful package. If you already have installed Python, you may use IDLE which is the integrated IDE that is shipped with Python, this IDE is written using Tkinter. Sounds Cool!!

2. **WxPython**

WxPython brings the wxWidgets cross-platform GUI library from its native C++ to Python.

WxPython is a slightly more modern approach to, which looks a little more native than Tkinter across different operating systems as it does not attempt to create its own set of widgets (although these can be themed to look much like native components)

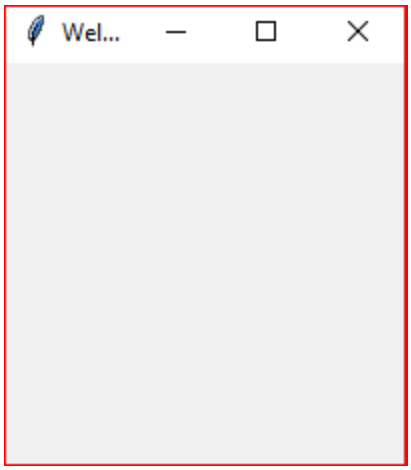
Create Your First GUI Application

First, we will import THE Tkinter package and create a window and set its title:

```
from tkinter import *

window = Tk()

window.title("Welcome to python GUI app")
```

window.mainloop()

output of the above code:



Creating Button and label

from tkinter import *

window = Tk()

window.title("Welcome to my App")

window.geometry('400x400')

lbl = Label(window, text="Username")

lbl.grid(column=0, row=0)

lbl = Label(window, text="Password")

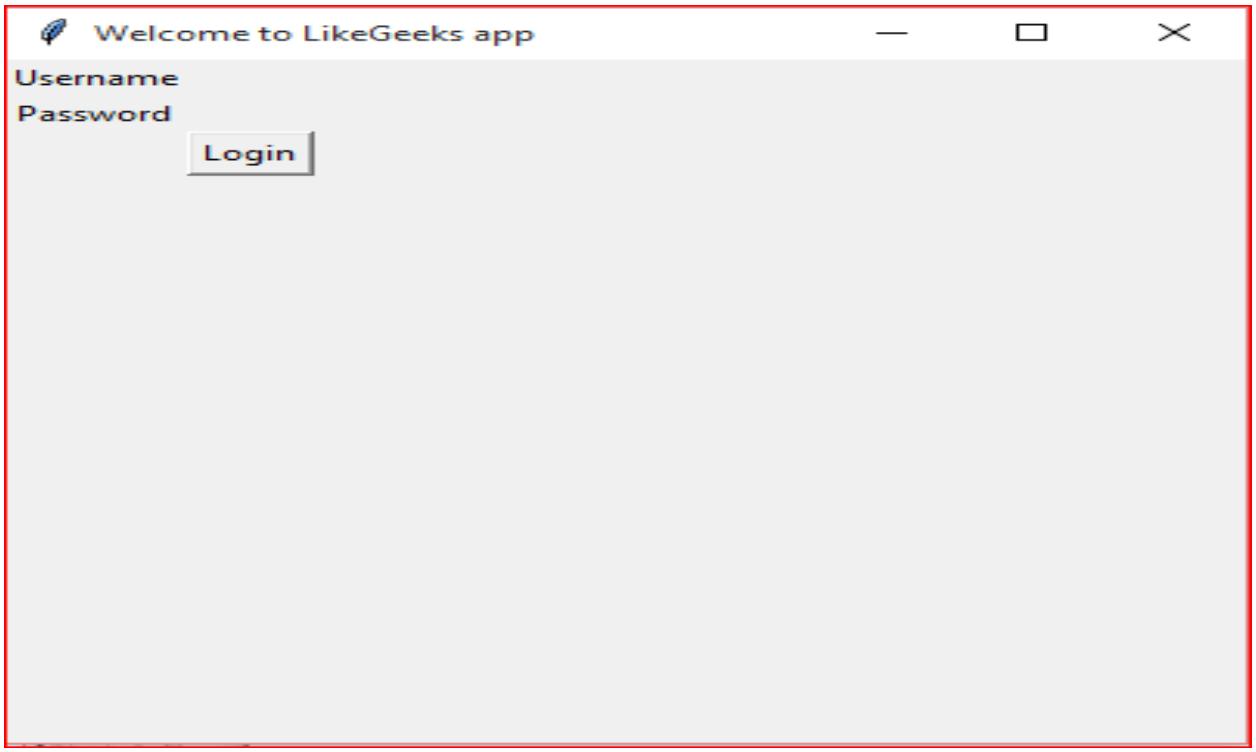
lbl.grid(column=0, row=1)

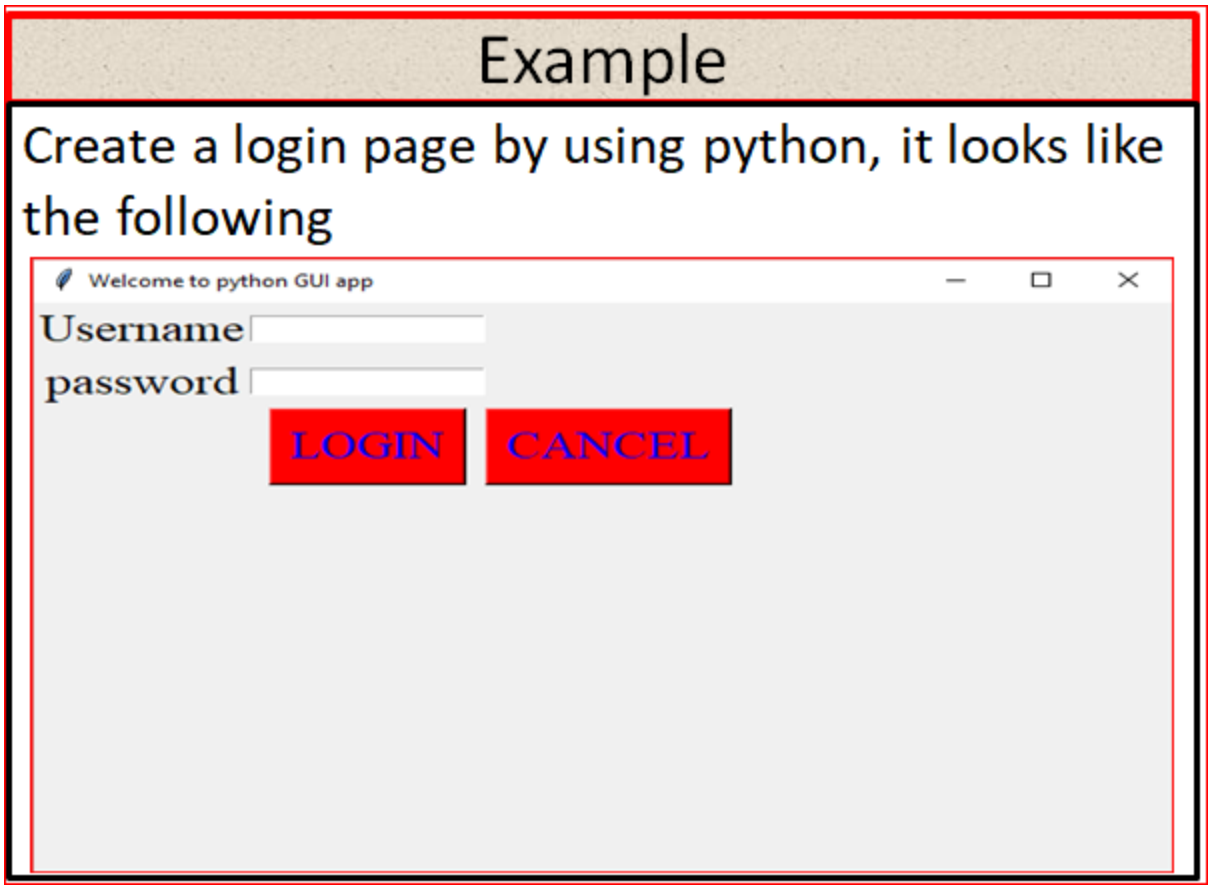
btn = Button(window, text="Login")

btn.grid(column=1, row=2)

window.mainloop()

The output of the above is:





Answer

```
from tkinter import *

window = Tk()

window.title('Welcome to python GUI app')

window.geometry('600x600')

lbl=Label(window,text="Username",font=("Times New Roman",20))#if there is a problem
like'label'is not defined it means that your 'l' s small  letter

lbl.grid(column=0,row=0)

txt = Entry(window,width=20)

txt.grid(column=1, row=0)

lbl=Label(window,text="password",font=("Times New Roman",20))

lbl.grid(column=0,row=1)
```

```
txt = Entry(window,width=20)#if there is a problem like'entry'is not defined it means that your 'e' is small  letter

txt.grid(column=1, row=1)

btn=Button(window,text='LOGIN',bg='red',fg='blue',font=("Times New Roman",20))

btn.grid(column=1,row=2)

btn=Button(window,text='CANCEL',bg='red',fg='blue',font=("Times New Roman",20))

btn.grid(column=2,row=2)

rad1=Radiobutton(window,text='Male', value=1)

rad1.grid(column=3,row=3)

rad2=Radiobutton(window,text='Female', value=2)

rad2.grid(column=4,row=3)

window.mainloop()
```

Exercise

- Create a python Gui for the following with the button submit

Username ➔textbox

Password ➔textbox

Usertype ➔use combobox

Sex ➔use radiobutton

ANSWER

Answer

```
from tkinter import *

from tkinter.ttk import *#using Combobox class from ttk library

window=Tk()

window.title("My GUI")

window.geometry("500x500")


lbl=Label(window,text="USERNAME", font=('times new roman',20))

lbl.grid(column=0,row=0)
```

```
lbl=Label(window,text="PASSWORD", font=('times new roman',20))
lbl.grid(column=0,row=1)
lbl=Label(window,text="AGE", font=('times new roman',20))
lbl.grid(column=0,row=2)
combo=Combobox(window)
combo['values']=(20,21,22,23,24,25,26)
combo.grid(column=1,row=2)
txt=Entry(window,width=20)
txt.grid(column=1,row=0)
txt=Entry(window,width=20)
txt.grid(column=1,row=1)
btn=Button(window,text="LOGIN")
btn.grid(column=1,row=4)
btn=Button(window,text="CANCEL")
btn.grid(column=2,row=4)
window.mainloop()
```

How to Insert Image

#A pattern of regularly spaced horizontal and vertical lines

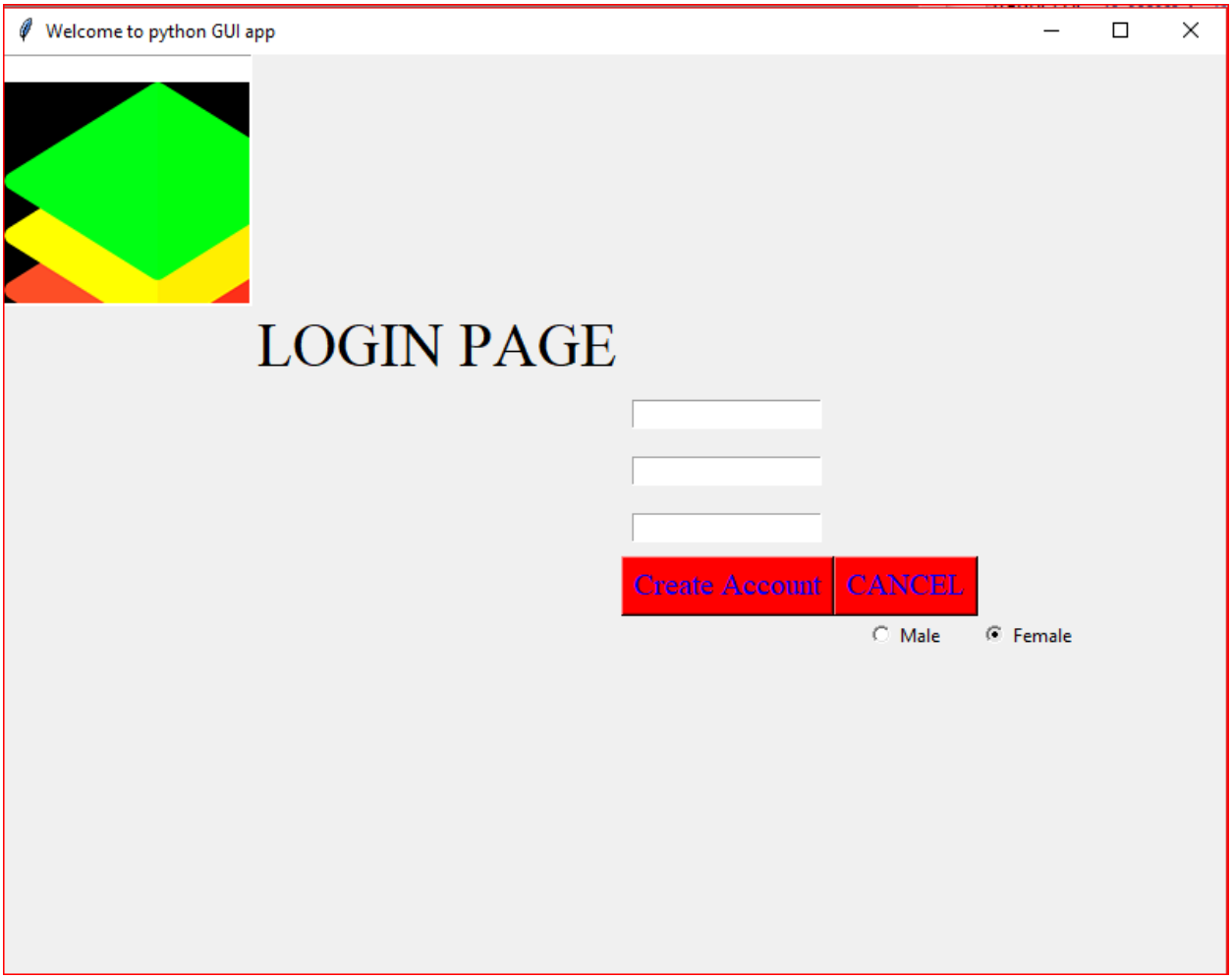
```
import sqlite3
con = sqlite3.connect('hu.db')
from tkinter import *
window = Tk()
window.geometry('800x600')
```

```
username=StringVar()

password=StringVar()

email=StringVar()


window.title('Welcome to python GUI app')

def database():

    con=sqlite3.connect('hu.db')

    with con:

        cursor=con.cursor()

        cursor.execute('CREATE TABLE IF NOT EXISTS login2(username TEXT,password
TEXT,email TEXT)')

        cursor.execute("INSERT INTO login2(username,password,email)
VALUES(?,?,?)",(username.get(),password.get(),email.get()))

        con.commit()




text1 = Text(window, height=10, width=20)

photo=PhotoImage(file='C:/Users/DELL/Desktop/new photo/img/image.PNG')

text1.insert(END,'\n')

text1.image_create(END, image=photo)

text1.pack(side=TOP)

text1.grid(column=0,row=0)
```

```
lbl=Label(window,text="LOGIN PAGE",font=("Times New Roman",30))#if there is a problem like'label'is not defined it means that your 'l' s small  letter

#lbl.pack(side=TOP)

lbl.grid(column=4,row=1)


lbl=Label(window,text="Username",textvariable=username,font=("Times New Roman",20))#if there is a problem like'label'is not defined it means that your 'l' s small letter

lbl.grid(column=4,row=4)

txt = Entry(window,width=20)

txt.grid(column=5, row=4)

lbl=Label(window,text="password",textvariable=password,font=("Times New Roman",20))

lbl.grid(column=4,row=5)


lbl=Label(window,text="Email",textvariable=email,font=("Times New Roman",20))

lbl.grid(column=4,row=6)


txt = Entry(window,width=20)#if there is a problem like'entry'is not defined it means that your 'e' is small  letter

txt.grid(column=5, row=5)


txt = Entry(window,width=20)#if there is a problem like'entry'is not defined it means that your 'e' is small  letter

txt.grid(column=5, row=6)

btn=Button(window,text='Create Account',command=database,bg='red',fg='blue',font=("Times New Roman",15))

btn.grid(column=5,row=7)
```

```
btn=Button(window,text='CANCEL',bg='red',fg='blue',font=("Times New Roman",15))
btn.grid(column=7,row=7)
rad1=Radiobutton(window,text='Male', value=1)
rad1.grid(column=7,row=8)
rad2=Radiobutton(window,text='Female', value=2)
rad2.grid(column=8,row=8)
window.mainloop()
```

The output looks like the following



How to connect python with database (SQLite3)

In this tutorial we will create a **Simple Login Application in Python**. Python has a design philosophy which emphasizes code readability. That's why python is very easy to use especially for beginners who just started programming. It is very easy to learn the syntax emphasizes readability and it can reduce time consuming in developing. So let's now do the coding.

Getting started

First you will have to download & install the Python IDLE's, here's the link for the Integrated Development and Learning Environment for Python https://www.python.org/downloads/.

Installing SQLite Browser (SQLite Studio)

After you installed Python, we will now then install the SQLite, here's the link for the DB Browser for SQLite http://sqlitebrowser.org/.

Part1: Designing part

Importing Modules

After setting up the installation and the database run the IDLE and click file and then new file. After that a new window will appear containing a black file this will be the text editor for the python.

Then copy code that I provided below and paste it inside the IDLE text editor

```
1. from tkinter import *
2. import sqlite3
```

Setting up the Main Frame

After importing the modules, we will now then create the main frame for the application. To do that just copy the code below and paste it inside the IDLE text editor.

```
1. root = Tk()
2. root.title("Python: Simple Login Application")
3. width = 400
```

4. height = 280
5. screen_width = root.winfo_screenwidth()
6. screen_height = root.winfo_screenheight()
7. root.geometry(‘500x500’)
8. root.resizable(0, 0)

Designing the Layout

After creating the Main Frame we will now add some layout to the application. Just kindly copy the code below and paste it inside the IDLE text editor.

```
#=====================VARIABLES===================================

USERNAME = StringVar()
PASSWORD = StringVar()

#==================FRAMES===================================
 Top = Frame(root, bd=2,  relief=RIDGE)
 Top.pack(side=TOP, fill=X)
Form = Frame(root, height=200)
 Form.pack(side=TOP, pady=20)
  #============================LABELS============================
  lbl_title = Label(Top, text = "Python: Simple Login Application", font=('arial', 15))
  lbl_title.pack(fill=X)
  lbl_username = Label(Form, text = "Username:", font=('arial', 14), bd=15)
 lbl_username.grid(row=0, sticky="e")
  lbl_password = Label(Form, text = "Password:", font=('arial', 14), bd=15)
  lbl_password.grid(row=1, sticky="e")
  lbl_text = Label(Form)
  lbl_text.grid(row=2, columnspan=2)
#=====================ENTRY WIDGETS=======================
 username = Entry(Form, textvariable=USERNAME, font=(14))
  username.grid(row=0, column=1)
  password = Entry(Form, textvariable=PASSWORD, show="*", font=(14))
  password.grid(row=1, column=1)
```

```
#=====================BUTTON WIDGETS==========================
btn_login = Button(Form, text="Login", width=45, command=Login)
btn_login.grid(pady=25, row=3, columnspan=2)
btn_login.bind('<Return>', Login)
```

Part 2: Data base connection

Creating the Database Connection

Then after setting up the design we will now create the database function. To do that just simply copy the code below and paste it inside the IDLE text editor

```
#=================METHODS==================================
def Database():
    global conn, cursor
    conn = sqlite3.connect("pythontut.db")
    cursor = conn.cursor()
    cursor.execute("CREATE TABLE IF NOT EXISTS `member` (mem_id INTEGER NOT NULL PRIMARY KEY  AUTOINCREMENT, username TEXT, password TEXT)")
    cursor.execute("SELECT * FROM `member` WHERE `username` = 'admin' AND `password` = 'admin'")
    if cursor.fetchone() is None:
        cursor.execute("INSERT INTO `member` (username, password) VALUES('admin', 'admin')")
        conn.commit()
```

Creating the Main Function

This is the main function where the Entry will be check if there is a user exist in the database, after login correctly a new window will pop up. To do that just simply copy the code below then paste it inside the IDLE text editor.

Note: It is better if you use the following code next to import modules

```
def Login(event=None):
    Database()
   if USERNAME.get() == "" or PASSWORD.get() == "":
        lbl_text.config(text="Please complete the required field!", fg="red")
    else:
        cursor.execute("SELECT * FROM `member` WHERE `username` = ? AND `password` =
?", (USERNAME.get(), PASSWORD.get()))
        if cursor.fetchone() is not None:
            HomeWindow()
           USERNAME.set("")
            PASSWORD.set("")
            lbl_text.config(text="")
        else:
            lbl_text.config(text="Invalid username or password", fg="red")
            USERNAME.set("")
            PASSWORD.set("")
    cursor.close()
    conn.close()

def HomeWindow():

    global Home

    root.withdraw()

    Home = Toplevel()

    Home.title("Python: Simple Login Application")

    width = 600

    height = 500

    screen_width = root.winfo_screenwidth()

    screen_height = root.winfo_screenheight()

    root.resizable(0, 0)

    Home.geometry(‘700x700’)

    lbl_home = Label(Home, text="Successfully Login!", font=('times new roman', 20)).pack()

    btn_back = Button(Home, text='Back', command=Back).pack(pady=20, fill=X)


def Back():

    Home.destroy()

    root.deiconify()
```

Initializing the Application

After finishing the function save the application as 'index.py'. This function will run the code and check if the main is initialize properly. To do that copy the code below and paste it inside the IDLE text editor.

```
1. #=========================INITIALIATION==============================
2. if __name__ == '__main__':
3.     root.mainloop()
```

The step is finished. If you are unable to understand the above step, copy and paste the following code inside the IDLE text editor

The code is:

```
#============importing modules=================

from tkinter import *

import sqlite3

#========creating the main function====================

def Login(event=None):

    Database()

    if USERNAME.get() == "" or PASSWORD.get() == "":

        lbl_text.config(text="Please complete the required field!", fg="red")

    else:

        cursor.execute("SELECT * FROM `member` WHERE `username` = ? AND `password` = ?", (USERNAME.get(), PASSWORD.get()))

        if cursor.fetchone() is not None:

            HomeWindow()
```

```
        USERNAME.set("")
        PASSWORD.set("")
        lbl_text.config(text="")
    else:
        lbl_text.config(text="Invalid username or password", fg="red")
        USERNAME.set("")
        PASSWORD.set("")
cursor.close()
conn.close()


root = Tk()
root.title("Python: Simple Login Application")
width = 400
height = 280
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()
root.geometry('500x500')
root.resizable(0, 0)
#===========VARIABLES==========================
USERNAME = StringVar()
PASSWORD = StringVar()
```

```
#========FRAMES==================================

Top = Frame(root, bd=2,  relief=RIDGE)

Top.pack(side=TOP, fill=X)

Form = Frame(root, height=200)

Form.pack(side=TOP, pady=20)

#==============LABELS============================

lbl_title = Label(Top, text = "Python: Simple Login Application", font=('arial', 15))

lbl_title.pack(fill=X)

lbl_username = Label(Form, text = "Username:", font=('arial', 14), bd=15)

lbl_username.grid(row=0, sticky="e")

lbl_password = Label(Form, text = "Password:", font=('arial', 14), bd=15)

lbl_password.grid(row=1, sticky="e")

lbl_text = Label(Form)

lbl_text.grid(row=2, columnspan=2)

#==============ENTRY WIDGETS====================

username = Entry(Form, textvariable=USERNAME, font=(14))

username.grid(row=0, column=1)

password = Entry(Form, textvariable=PASSWORD, show="*", font=(14))

password.grid(row=1, column=1)

#===========BUTTON WIDGETS===================
```

```
btn_login = Button(Form, text="Login", width=45, command=Login)

btn_login.grid(pady=25, row=3, columnspan=2)

btn_login.bind('<Return>', Login)

#==================METHODS====================

def Database():

    global conn, cursor

    conn = sqlite3.connect("pythontut.db")

    cursor = conn.cursor()

    cursor.execute("CREATE TABLE IF NOT EXISTS `member` (mem_id INTEGER
NOT NULL PRIMARY KEY  AUTOINCREMENT, username TEXT, password TEXT)")

    cursor.execute("SELECT * FROM `member` WHERE `username` = 'admin' AND
`password` = 'admin'")

    if cursor.fetchone() is None:

        cursor.execute("INSERT INTO `member` (username, password) VALUES('admin',
'admin')")

        conn.commit()

def HomeWindow():

    global Home

    root.withdraw()

    Home = Toplevel()

    Home.title("Python: Simple Login Application")

    width = 600
```

```
    height = 500

    screen_width = root.winfo_screenwidth()

    screen_height = root.winfo_screenheight()

    root.resizable(0, 0)

    Home.geometry('700x700')

    lbl_home = Label(Home, text="Successfully Login!", font=('times new roman',
20)).pack()

    btn_back = Button(Home, text='Back', command=Back).pack(pady=20, fill=X)

def Back():

    Home.destroy()

    root.deiconify()

#===============INITIALIATION=====================

if __name__ == '__main__':

    root.mainloop()
```

Example 2: python program that can create account using username and password with BD Browser for sqlite

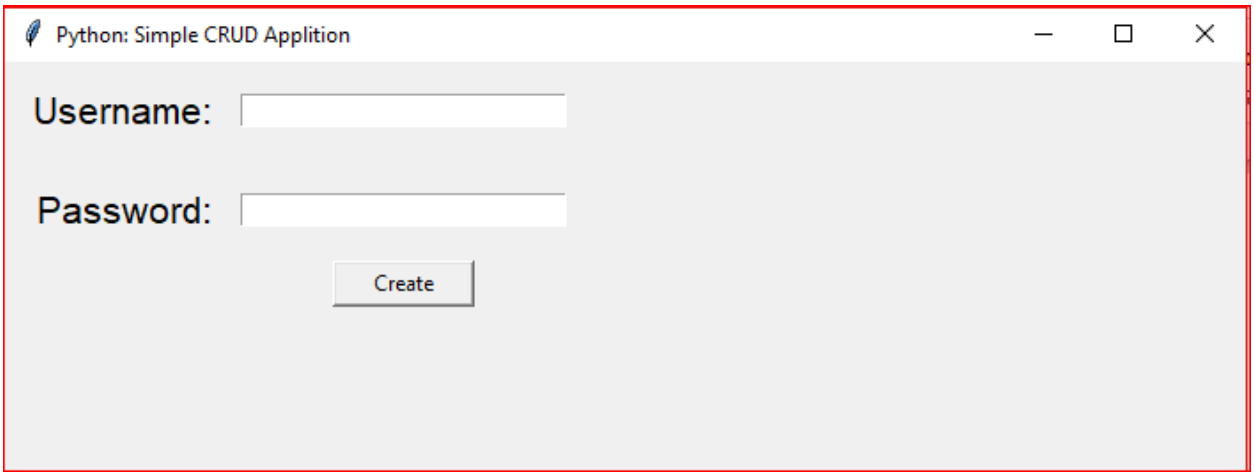
```
from tkinter import*

import sqlite3

import tkinter.ttk as ttk

import tkinter.messagebox as tkMessageBox

root = Tk()

root.title("Python: Simple CRUD Applition")

root.geometry('700x800')

#================VARIABLES==========================

USERNAME = StringVar()

PASSWORD = StringVar()

txt_result = Label(root)

#=============insert into database===================

def Create():

    if   USERNAME.get() == "" or PASSWORD.get() == "":

        txt_result.config(text="Please complete the required field!", fg="red")

    else:

        Database()
```

```
        cursor.execute("INSERT INTO `account` (username, password) VALUES(?, ?)",
(str(USERNAME.get()), str(PASSWORD.get())))

        conn.commit()

        USERNAME.set("")

        PASSWORD.set("")

        cursor.close()

        conn.close()

        txt_result.config(text="Created a data!", fg="green")

def Read():

    read()

def Update():

    Update()

def Delete():

    Delete()

def Exit():

    Exit()

#============For Database creation==========(Table create)

def Database():

    global conn, cursor

    conn = sqlite3.connect('hu1.db')

    cursor = conn.cursor()
```

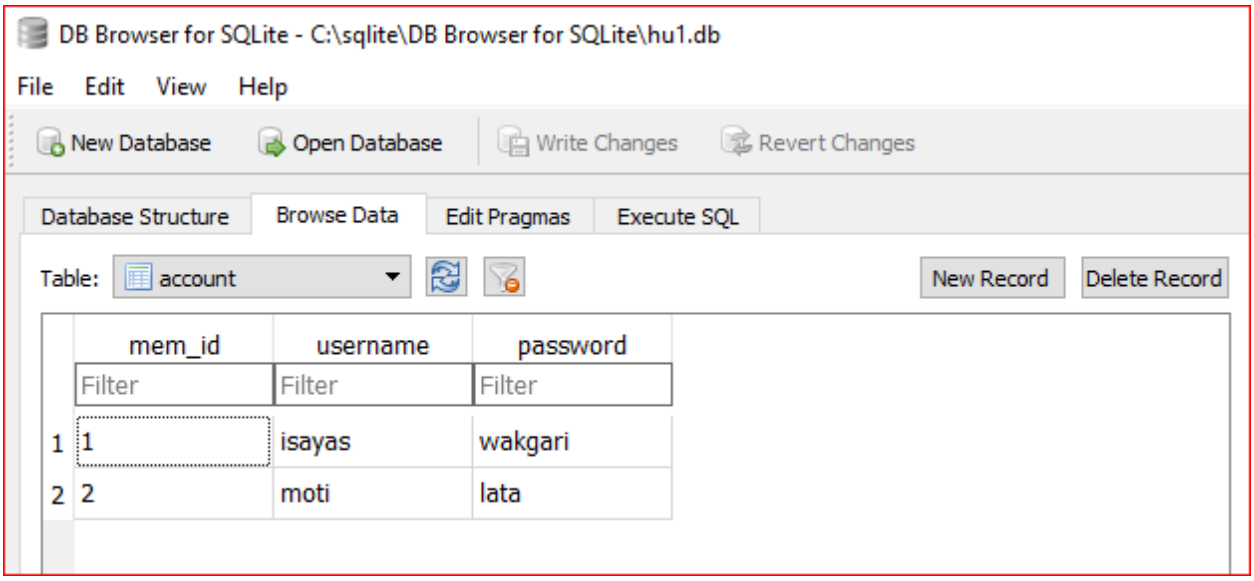
```
    cursor.execute("CREATE TABLE IF NOT EXISTS `account` (mem_id INTEGER
PRIMARY KEY AUTOINCREMENT NOT NULL, username TEXT, password TEXT)")

#=================LABEL WIDGET=================

txt_username = Label(root, text="Username:", font=('arial', 16), bd=15)

txt_username.grid(row=4, stick="e")

txt_password = Label(root, text="Password:", font=('arial', 16), bd=15)

txt_password.grid(row=5, stick="e")

txt_result = Label(root)

#===========ENTRY WIDGET===================

username = Entry(root, textvariable=USERNAME, width=30)

username.grid(row=4, column=1)

password = Entry(root, textvariable=PASSWORD, show="*", width=30)

password.grid(row=5, column=1)


#============BUTTONS WIDGET====================

btn_create = Button(root, width=10, text="Create", command=Create)

btn_create.grid(column=1,row=6)

#btn_read = Button(root, width=10, text="Read", command=Read )

#btn_read.grid(column=2,row=6)

#btn_update = Button(root, width=10, text="Update", state=DISABLED)

#btn_update.grid(column=3,row=6)
```

```
#btn_delete = Button(root, width=10, text="Delete", state=DISABLED)

#btn_delete.grid(column=4,row=6)

#btn_exit = Button(root, width=10, text="Exit", command=Exit)

#btn_exit.grid(column=5,row=6)

#============INITIALIZATION======================

if __name__ == '__main__':

    root.mainloop()
```

When you write the above code on the python editor you will get the following GUI



When you enter username and password and click on create button , the data is inserted into db.



Python: Simple CRUD Application

```
from tkinter import*

import sqlite3

import tkinter.ttk as ttk

import tkinter.messagebox as tkMessageBox



def Create():

    if  FIRSTNAME.get() == "" or LASTNAME.get() == "" or GENDER.get() ==
"" or ADDRESS.get() == "" or USERNAME.get() == "" or PASSWORD.get() ==
"":

        txt_result.config(text="Please complete the required field!", fg="red")

    else:

        Database()

        cursor.execute("INSERT INTO `member` (firstname, lastname, gender,
address, username, password) VALUES(?, ?, ?, ?, ?, ?)", (str(FIRSTNAME.get()),
str(LASTNAME.get()), str(GENDER.get()), str(ADDRESS.get()),
str(USERNAME.get()), str(PASSWORD.get())))

        conn.commit()
```

```
        FIRSTNAME.set("")
        LASTNAME.set("")
        GENDER.set("")
        ADDRESS.set("")
        USERNAME.set("")
        PASSWORD.set("")
        cursor.close()
        conn.close()
        txt_result.config(text="Created a data!", fg="green")



def Read():
    tree.delete(*tree.get_children())
    Database()
    cursor.execute("SELECT * FROM `member` ORDER BY `lastname` ASC")
    fetch = cursor.fetchall()
    for data in fetch:
        tree.insert('', 'end', values=(data[1], data[2], data[3], data[4], data[5], data[6]))
        cursor.close()
```

```
        #conn.close()
        txt_result.config(text="Successfully read the data from database", fg="black")




def Exit():
    result = tkMessageBox.askquestion('Python: Simple CRUD Applition', 'Are you sure you want to exit?', icon="warning")
    if result == 'yes':
        root.destroy()
        exit()
root = Tk()
root.title("Python: Simple CRUD Applition")
#screen_width = root.winfo_screenwidth()
#screen_height = root.winfo_screenheight()
#width = 900
#height = 500
#x = (screen_width/2) - (width/2)
#y = (screen_height/2) - (height/2)
```

```
root.geometry('1200x500')

root.resizable(0, 0)


FIRSTNAME = StringVar()

LASTNAME = StringVar()

GENDER = StringVar()

ADDRESS = StringVar()

USERNAME = StringVar()

PASSWORD = StringVar()


Top = Frame(root, width=900, height=50, bd=8, relief="raise")

Top.pack(side=TOP)

Left = Frame(root, width=300, height=500, bd=8, relief="raise")

Left.pack(side=LEFT)

Right = Frame(root, width=600, height=500, bd=8, relief="raise")

Right.pack(side=RIGHT)

Forms = Frame(Left, width=300, height=450)

Forms.pack(side=TOP)

Buttons = Frame(Left, width=300, height=100, bd=8, relief="raise")
```

```
Buttons.pack(side=BOTTOM)

RadioGroup = Frame(Forms)

Male = Radiobutton(RadioGroup, text="Male", variable=GENDER,
value="Male", font=('arial', 16)).pack(side=LEFT)

Female = Radiobutton(RadioGroup, text="Female", variable=GENDER,
value="Female", font=('arial', 16)).pack(side=LEFT)



txt_title = Label(Top, width=900, font=('arial', 24), text = "Python: Simple CRUD
Application")

txt_title.pack()

txt_firstname = Label(Forms, text="Firstname:", font=('arial', 16), bd=15)

txt_firstname.grid(row=0, stick="e")

txt_lastname = Label(Forms, text="Lastname:", font=('arial', 16), bd=15)

txt_lastname.grid(row=1, stick="e")

txt_gender = Label(Forms, text="Gender:", font=('arial', 16), bd=15)

txt_gender.grid(row=2, stick="e")

txt_address = Label(Forms, text="Address:", font=('arial', 16), bd=15)

txt_address.grid(row=3, stick="e")

txt_username = Label(Forms, text="Username:", font=('arial', 16), bd=15)
```

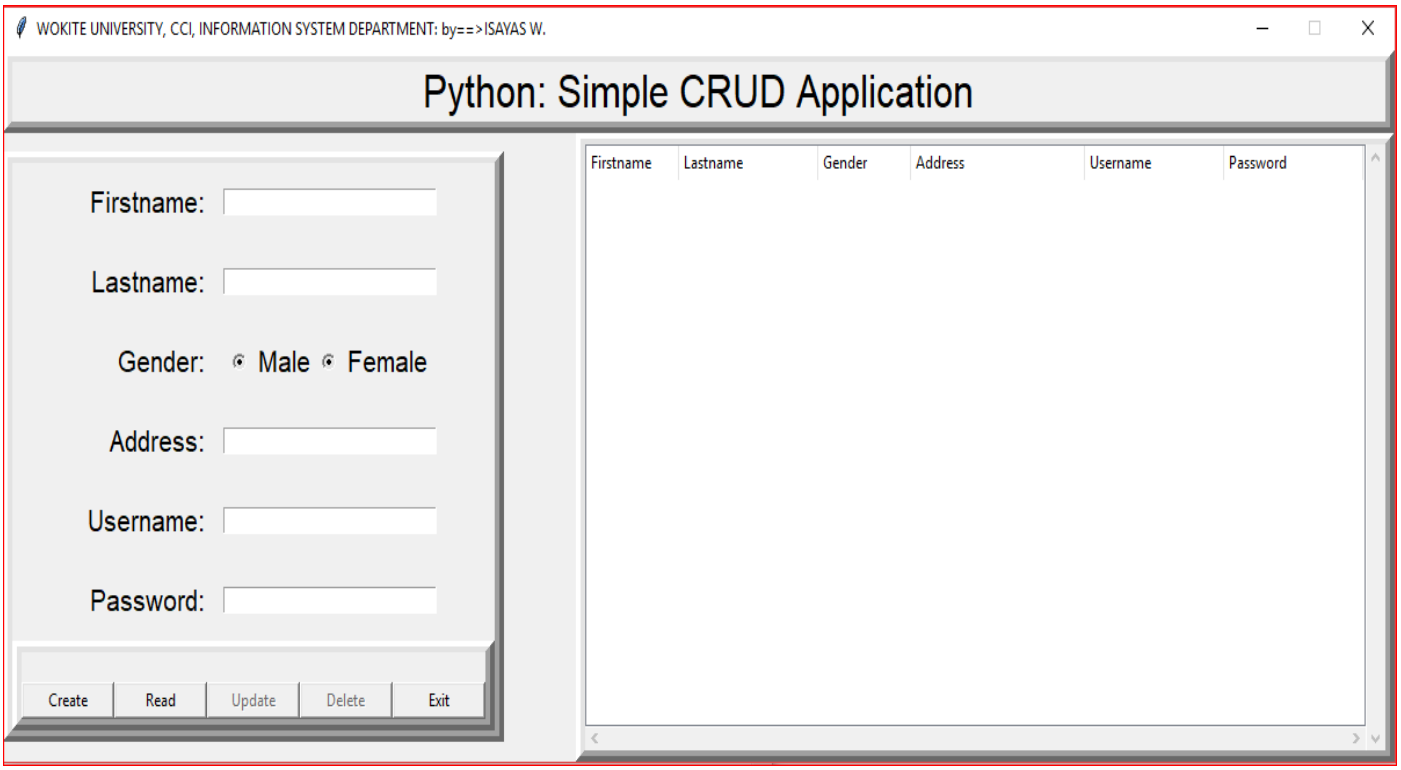
```
txt_username.grid(row=4, stick="e")

txt_password = Label(Forms, text="Password:", font=('arial', 16), bd=15)

txt_password.grid(row=5, stick="e")

txt_result = Label(Buttons)

txt_result.pack(side=TOP)


firstname = Entry(Forms, textvariable=FIRSTNAME, width=30)

firstname.grid(row=0, column=1)

lastname = Entry(Forms, textvariable=LASTNAME, width=30)

lastname.grid(row=1, column=1)

RadioGroup.grid(row=2, column=1)

address = Entry(Forms, textvariable=ADDRESS, width=30)

address.grid(row=3, column=1)

username = Entry(Forms, textvariable=USERNAME, width=30)

username.grid(row=4, column=1)

password = Entry(Forms, textvariable=PASSWORD, show="*", width=30)

password.grid(row=5, column=1)


btn_create = Button(Buttons, width=10, text="Create", command=Create)
```

```
btn_create.pack(side=LEFT)

btn_read = Button(Buttons, width=10, text="Read", command=Read )

btn_read.pack(side=LEFT)

btn_update = Button(Buttons, width=10, text="Update", state=DISABLED)

btn_update.pack(side=LEFT)

btn_delete = Button(Buttons, width=10, text="Delete", state=DISABLED)

btn_delete.pack(side=LEFT)

btn_exit = Button(Buttons, width=10, text="Exit", command=Exit)

btn_exit.pack(side=LEFT)


scrollbary = Scrollbar(Right, orient=VERTICAL)

scrollbarx = Scrollbar(Right, orient=HORIZONTAL)

tree = ttk.Treeview(Right, columns=("Firstname", "Lastname", "Gender",
"Address", "Username", "Password"), selectmode="extended", height=500,
yscrollcommand=scrollbary.set, xscrollcommand=scrollbarx.set)

scrollbary.config(command=tree.yview)

scrollbary.pack(side=RIGHT, fill=Y)

scrollbarx.config(command=tree.xview)

scrollbarx.pack(side=BOTTOM, fill=X)

tree.heading('Firstname', text="Firstname", anchor=W)
```

```
tree.heading('Lastname', text="Lastname", anchor=W)
tree.heading('Gender', text="Gender", anchor=W)
tree.heading('Address', text="Address", anchor=W)
tree.heading('Username', text="Username", anchor=W)
tree.heading('Password', text="Password", anchor=W)
tree.column('#0', stretch=NO, minwidth=0, width=0)
tree.column('#1', stretch=NO, minwidth=0, width=80)
tree.column('#2', stretch=NO, minwidth=0, width=120)
tree.column('#3', stretch=NO, minwidth=0, width=80)
tree.column('#4', stretch=NO, minwidth=0, width=150)
tree.column('#5', stretch=NO, minwidth=0, width=120)
tree.column('#6', stretch=NO, minwidth=0, width=120)
tree.pack()


def Database():
    global conn, cursor
    conn = sqlite3.connect('pythontut.db')
    cursor = conn.cursor()
```

```
    cursor.execute("CREATE TABLE IF NOT EXISTS `member` (mem_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, firstname TEXT, lastname TEXT, gender TEXT, address TEXT, username TEXT, password TEXT)")



if __name__ == '__main__':
    root.mainloop()
```

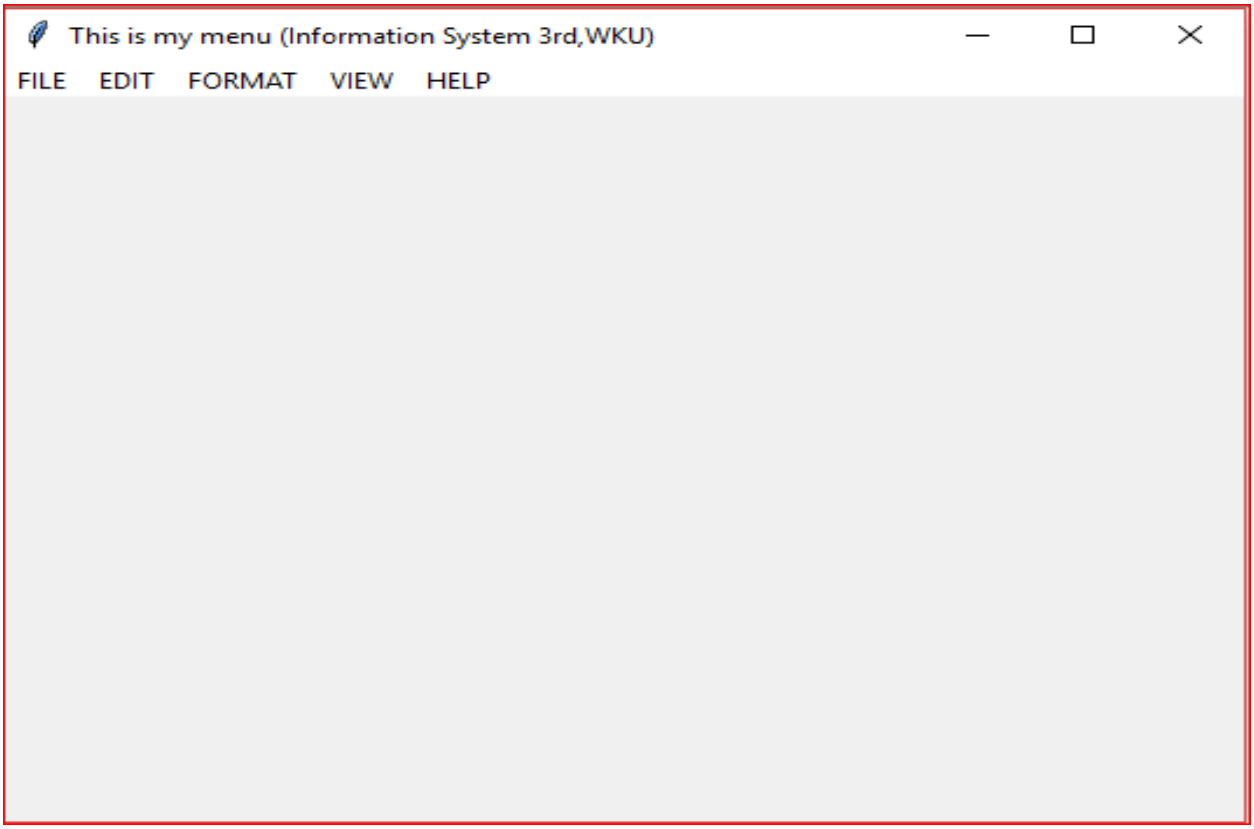
The output of the above code looks like the following:



Creating Editors

```
from tkinter import *

from tkinter import Menu

#def donothing():

    #print(donothing)

window=Tk()

menubar = Menu(window)

filemenu= Menu(menubar)

menubar.add_cascade(label='File',menu=filemenu)

#filemenu.add_command(label='new',command=donothing)

window.config(men=menubar)

window.mainloop()
```

Write a python code that produce the following editor



Answer

```
from tkinter import *

from tkinter import Menu

def donothing():

    print(a)

window=Tk()
```

```
window.geometry('500x500')

menubar = Menu(window)

filemenu=Menu(menubar,tearoff=0)

editmenu=Menu(menubar,tearoff=0)

formatmenu=Menu(menubar,tearoff=0)

helpmenu=Menu(menubar,tearoff=0)

viewmenu=Menu(menubar,tearoff=0)

menubar.add_cascade(label='File',menu=filemenu)

filemenu.add_command(label="New", command=donothing)

filemenu.add_command(label="Open", command=donothing)

filemenu.add_command(label="Save", command=donothing)

filemenu.add_command(label="Save as", command=donothing)

filemenu.add_command(label="Exit", command=donothing)

menubar.add_cascade(label='Edit',menu=editmenu)

menubar.add_cascade(label='Format',menu=formatmenu)

menubar.add_cascade(label='View',menu=viewmenu)

menubar.add_cascade(label='Help',menu=helpmenu)

window.config(menu=menubar)

window.mainloop()
```