**Project Report**

on

**Notepad**

in Python

**INDEX**

|  |  |
| --- | --- |
| **TOPIC** | **PAGE NO.** |
| ACKNOWLEDGEMENT |  |
| PREFACE |  |
| ABOUT THE PROJECT |  |
| SYSTEM ANALYSIS & DESIGNING |  |
| ABOUT JAVA |  |
| OBJECTIVES |  |
| CODING |  |
| OUTPUT |  |
| BIBLIOGRAPHY |  |

**ACKNOWLEDGEMENT**

I want to express a deep sense of Gratitude all those people who have supported me to accomplish my job training project .First foremost I would like to thank the management of **UIET Kurukshetra University** who has given me the opportunity to accomplish my training for partial fulfillment of the requirement for the degree of B.Tech to Kurukshetra university .

I am thankful to **Anand Mahajan** for providing me opportunity to complete my project job and their guidance. I am thankful to Mr. Anand Mahajan, Senior Faculty, Mahajan Computers who guided me during my project and spares their valuable times for us and for taking keen interest.

**PREFACE**

I feel great pleasure in presenting this work in partial fulfillment of the condition laid down by the **UIET, KURUKSHETRA** for the degree of **B.Tech (Bachelor of Technology)**. In this work great care has been taken to present the detail simple in lucid manner and fairly precision

I sincerely assure that the method of presentation of this project will enable to grasp the details with satisfaction if it does so, I shall simply feel rewarded for the pain taken only the readers can judge the existent to which I have succeeded in doing justice. This is my sincere wish and hope that this project will be of great undertaking the same project in future.

While every effort has been taken to avoid printing and other errors, I oblige for any error that might have inadvertently crept in.

**ABOUT THE PROJECT**

**Introduction**

Notepad is a common text-only ([plain text](https://en.wikipedia.org/wiki/Plain_text)) editor. The resulting files—typically saved with the [.txt](https://en.wikipedia.org/wiki/Text_file) extension—have no format tags or styles, making the program suitable for editing system files to use in a [DOS](https://en.wikipedia.org/wiki/DOS) environment and, occasionally, source code for later [compilation](https://en.wikipedia.org/wiki/Compiler) or [execution](https://en.wikipedia.org/wiki/Scripting_language), usually through a [command prompt](https://en.wikipedia.org/wiki/Command_line_interpreter). It is also useful for its negligible use of system resources; making for quick load time and processing time, especially on under-powered hardware. Notepad supports both left-to-right and right-to-left based languages. Historically, Notepad did not treat [newlines](https://en.wikipedia.org/wiki/Newline) in [Unix](https://en.wikipedia.org/wiki/Unix)- or [classic Mac OS](https://en.wikipedia.org/wiki/Classic_Mac_OS)-style text files correctly. However, on 8th May 2018, Microsoft announced that they had fixed this issue.[[11]](https://en.wikipedia.org/wiki/Microsoft_Notepad#cite_note-11) Notepad offers only the most basic text manipulation functions, such as finding text. Only newer versions of Windows include an updated version of Notepad with a search and replace function. However, it has much less functionality in comparison to full-scale editors.

In all versions of Windows, Notepad uses a built-in [window class](https://en.wikipedia.org/wiki/Window_class) named *EDIT* and the maximum file size that Notepad can open is dependent on operating system limitations on the size of the *EDIT* window class, with the limit being different for each version of Windows. Due to the operating system limit of the *EDIT* window class, the Notepad version shipped with [Windows 3.0](https://en.wikipedia.org/wiki/Windows_3.0), [Windows 3.1](https://en.wikipedia.org/wiki/Windows_3.1) and [Windows 3.11](https://en.wikipedia.org/wiki/Windows_3.11) could not open files larger than 54 KB ([kilobytes](https://en.wikipedia.org/wiki/Kilobyte)) and Microsoft recommended not to open files larger than 45 KB, with the official workaround advice provided by Microsoft being "Use another text editor",[[12]](https://en.wikipedia.org/wiki/Microsoft_Notepad#cite_note-12) but this limit was extended to 64 KB in [Windows 95](https://en.wikipedia.org/wiki/Windows_95) (and remained the same in [Windows 98](https://en.wikipedia.org/wiki/Windows_98) and [Windows Me](https://en.wikipedia.org/wiki/Windows_Me)), with users now directed to [WordPad](https://en.wikipedia.org/wiki/WordPad) to open larger files. On the Notepad version shipped with Windows XP the limit was 32 MB ([megabytes](https://en.wikipedia.org/wiki/Megabyte)) with the application displaying the message "The file is too large for Notepad. Use another editor to edit the file" if the user attempted to open a file larger than 32 MB.[[13]](https://en.wikipedia.org/wiki/Microsoft_Notepad#cite_note-13) Newer versions of Notepad can open files at least up to 58 MB ([megabytes](https://en.wikipedia.org/wiki/Megabyte)) in size,[[14]](https://en.wikipedia.org/wiki/Microsoft_Notepad#cite_note-14) and on [Windows 8.1](https://en.wikipedia.org/wiki/Windows_8.1) the Notepad application is able to open files at least as large as 512 MB (megabytes) but fails to open 1 GB (gigabyte) files displaying the same message that Windows XP users would see ("The file is too large for Notepad. Use another editor to edit the file").[[15]](https://en.wikipedia.org/wiki/Microsoft_Notepad#cite_note-15)

Up to [Windows 95](https://en.wikipedia.org/wiki/Windows_95), [Fixedsys](https://en.wikipedia.org/wiki/Fixedsys) was the only available display [font](https://en.wikipedia.org/wiki/Font) for Notepad. [Windows NT 4.0](https://en.wikipedia.org/wiki/Windows_NT_4.0) and [98](https://en.wikipedia.org/wiki/Windows_98) introduced the ability to change this font. As of [Windows 2000](https://en.wikipedia.org/wiki/Windows_2000), the default font was changed to [Lucida Console](https://en.wikipedia.org/wiki/Lucida_Console). The font setting, however, only affects how the text is shown to the user and how it is printed, not how the file is saved to disk. The default font was changed to [Consolas](https://en.wikipedia.org/wiki/Consolas) on Windows 8.

Up to [Windows Me](https://en.wikipedia.org/wiki/Windows_Me), there were almost no keyboard shortcuts and no line-counting feature. Starting with [Windows 2000](https://en.wikipedia.org/wiki/Windows_2000), shortcuts for common tasks like new, open and save were added, as well as a status-bar with a line counter (available only when word-wrap is disabled).

In the [Windows NT](https://en.wikipedia.org/wiki/Windows_NT)-based versions of Windows, Notepad can edit traditional 8-bit text files as well as [Unicode](https://en.wikipedia.org/wiki/Unicode) text files (both [UTF-8](https://en.wikipedia.org/wiki/UTF-8) and [UTF-16](https://en.wikipedia.org/wiki/UTF-16), and in case of UTF-16, [both little-endian and big-endian](https://en.wikipedia.org/wiki/Endianness)).

Notepad also has a simple built-in logging function. Each time a file that starts with .LOG is opened, the program inserts a text timestamp on the last line of the file.[[16]](https://en.wikipedia.org/wiki/Microsoft_Notepad#cite_note-16)[[17]](https://en.wikipedia.org/wiki/Microsoft_Notepad#cite_note-17)

Notepad accepts text from the Windows [clipboard](https://en.wikipedia.org/wiki/Clipboard_(software)). When clipboard data with multiple formats is pasted into Notepad, the program only accepts text in the CF\_TEXT format.[[18]](https://en.wikipedia.org/wiki/Microsoft_Notepad#cite_note-18) This is useful for stripping embedded font type and style codes from formatted text, such as when copying text from a web page and pasting into an email message or other [WYSIWYG](https://en.wikipedia.org/wiki/WYSIWYG) text editor. Formatted text can be temporarily pasted into Notepad, and then immediately copied again in stripped format to paste into the other program.

Notepad can print files, but doesn't print correctly if *Word Wrap* is turned on. Headers, footers, and margins can be set and adjusted when preparing to print a file under *Page Setup*. The date, file name, and other information can be placed in the headers and footers with various codes consisting of an ampersand ('&') followed by a letter.

**INTRODUCTION TO PYTHON**

**About Python**

Python is one of those rare languages which can claim to be both *simple* and *powerful*. You will find yourself pleasantly surprised to see how easy it is to concentrate on the solution to the problem rather than the syntax and structure of the language you are programming in.

The official introduction to Python is:

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

I will discuss most of these features in more detail in the next section.

**Story behind the name**

Guido van Rossum, the creator of the Python language, named the language after the BBC show "Monty Python's Flying Circus". He doesn't particularly like snakes that kill animals for food by winding their long bodies around them and crushing them.

**Features of Python**

**Simple**

Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English! This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself.

**Easy to Learn**

As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned.

**Free and Open Source**

Python is an example of a *FLOSS* (Free/Libré and Open Source Software). In simple terms, you can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python.

**High-level Language**

When you write programs in Python, you never need to bother about the low-level details such as managing the memory used by your program, etc.

**Portable**

Due to its open-source nature, Python has been ported to (i.e. changed to make it work on) many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features.

You can use Python on GNU/Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE and PocketPC!

You can even use a platform like [Kivy](http://kivy.org/" \t "_blank) to create games for your computer *and* for iPhone, iPad, and Android.

**Interpreted**

This requires a bit of explanation.

A program written in a compiled language like C or C++ is converted from the source language i.e. C or C++ into a language that is spoken by your computer (binary code i.e. 0s and 1s) using a compiler with various flags and options. When you run the program, the linker/loader software copies the program from hard disk to memory and starts running it.

Python, on the other hand, does not need compilation to binary. You just *run* the program directly from the source code. Internally, Python converts the source code into an intermediate form called bytecodes and then translates this into the native language of your computer and then runs it. All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc. This also makes your Python programs much more portable, since you can just copy your Python program onto another computer and it just works!

**Object Oriented**

Python supports procedure-oriented programming as well as object-oriented programming. In *procedure-oriented* languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In *object-oriented* languages, the program is built around objects which combine data and functionality. Python has a very powerful but simplistic way of doing OOP, especially when compared to big languages like C++ or Java.

**Extensible**

If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can code that part of your program in C or C++ and then use it from your Python program.

**Embeddable**

You can embed Python within your C/C++ programs to give *scripting* capabilities for your program's users.

**Extensive Libraries**

The Python Standard Library is huge indeed. It can help you do various things involving regular expressions,documentation generation, unit testing, threading, databases, web browsers, CGI, FTP, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI (graphical user interfaces), and other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the *Batteries Included* philosophy of Python.

Besides the standard library, there are various other high-quality libraries which you can find at the [Python Package Index](http://pypi.python.org/pypi).

**Summary**

Python is indeed an exciting and powerful language. It has the right combination of performance and features that make writing programs in Python both fun and easy.

**Python 3 versus 2**

You can ignore this section if you're not interested in the difference between "Python version 2" and "Python version 3". But please do be aware of which version you are using. This book is written for Python version 3.

Remember that once you have properly understood and learn to use one version, you can easily learn the differences and use the other one. The hard part is learning programming and understanding the basics of Python language itself. That is our goal in this book, and once you have achieved that goal, you can easily use Python 2 or Python 3 depending on your situation.

## What Programmers Say

You may find it interesting to read what great hackers like ESR have to say about Python:

* Eric S. Raymond is the author of "The Cathedral and the Bazaar" and is also the person who coined the term Open Source. He says that [Python has become his favorite programming language](http://www.python.org/about/success/esr/). This article was the real inspiration for my first brush with Python.
* Bruce Eckel is the author of the famous 'Thinking in Java' and 'Thinking in C++' books. He says that no language has made him more productive than Python. He says that Python is perhaps the only language that focuses on making things easier for the programmer. Read the [complete interview](http://www.artima.com/intv/aboutme.html) for more details.
* Peter Norvig is a well-known Lisp author and Director of Search Quality at Google (thanks to Guido van Rossum for pointing that out). He says that [writing Python is like writing in pseudocode](https://news.ycombinator.com/item?id=1803815). He says that Python has always been an integral part of Google. You can actually verify this statement by looking at the [Google Jobs](http://www.google.com/jobs/index.html) page which lists Python knowledge as a requirement for software engineers.

# Migration strategies

Making a new release of software that is backwards incompatible is a high risk strategy. When people need to rewrite their software, or maintain separate versions of their source code to support both versions of a language or a framework, the risk is that they never make the transition and stay on the old version forever, or worse, that they switch to another framework.

For that reason Python versions 2.6 and 2.7 include both several forward compatibility features to enable you to write code for both Python 2 and Python 3, as well as support for migrating in the form of 2to3, a program and package that can convert code from Python 2 to Python 3. There are other techniques and strategies you can use and there are also different ways to use 2to3. Which strategy to use depends very much on what type of software you are converting.

## Only supporting Python 3

The easiest case is when you only need to support one version of Python at a time. In these cases you can just convert your code to Python 3 and forget about Python 2. With this strategy you will first use the 2to3 tool to make an automatic conversion of most of the changes and then fix every problem that remains manually in the Python 3 code. You will probably also want to go through all of the converted code and clean it up, as the 2to3 conversions may not always be the most elegant solution for your case.

## Separate branches for Python 2 and Python 3

If you need to continue to support Python 2, the simplest case is having a branch in your source tree with the code for Python 2 and another branch for Python 3. You will then have to make every change on the two different branches, which is a bit more work, but feasible if the code doesn’t change that often.

One problem with this strategy is that your distribution becomes complex, because you now have two distributions and you need to make sure that Python 3 users get the Python 3 version and Python 2 users get the Python 2 version of your package. Solutions for this are documented in [Distributing packages](http://python3porting.com/2to3.html#distribution-section).

## Converting to Python 3 with 2to3

In complex cases you can support both Python 2 and Python 3 by maintaining the source code in a Python 2 version and converting it with 2to3 for Python 3 support. That means you will have to run 2to3 each time you have made a code change so you can test it under Python 3, but on the other hand 2to3 deals with many of the differences.

To do this you need a script that performs the 2to3 conversion, because doing all the steps manually quickly becomes really boring. Since you need to do the conversion every time you have changed something so you can run the tests under Python 3, you want to run the conversion only on the files that have been modified as the conversion can be rather slow. That means that the conversion script should compare time stamps on the files to see which ones have been modified and convert only them, which means the script can not be a trivial shell script.

You can of course write these conversion scripts yourself, but you might not need to. If you are using Distutils it has support for running 2to3 as a part of the build process. This also solves the distribution problems, as this way you can distribute only Python 2 code and 2to3 will be run on that code during install when installed on Python 3. That way you don’t have to have separate packages or even two copies of the code in your package. [Distributing packages](http://python3porting.com/2to3.html#distribution-section) also has information on this.

However, the lazy coders approach here would be to use Distribute, as it includes some extensions to the 2to3-story.

### Using Distribute to support the 2to3 conversion

Distribute[[1]](http://python3porting.com/strategies.html#distribute) is a fork of Phillip J. Eby’s popular Setuptools package and provides Python 3 compatibility, as well as extensions simplifying the support of Python 2 and Python 3 from the same source. Basically what Distribute has done is to extend the principles of the Distutils build\_py\_2to3 command and integrated 2to3 into all parts of the packaging story.

These changes will be merged back into Setuptools during 2013, but at the time of writing Setuptools doesn’t support Python 3.

With Distribute you can add a few extra parameters in the setup.py file to have 2to3 run the conversion at build time. This means you only need to have one version of the source in your version control system and you therefore only need to fix bugs once. You also need only one source release, so you only have to release the software once and there is only one package to download and install for both Python 2 and Python 3.

You still need to run your tests under all versions of Python that you want to support, but Distribute includes a test command that will convert your code with2to3 before running the tests. You can easily set up your package to use that. Then testing becomes just running python setup.py test once for every python version you want to support.

The main drawback with this solution is that you can’t use the earliest versions of 2to3, because they are too buggy. In practice it means you need to have Python 3.1 or later installed on the target machine. This is generally not a problem, as most platforms that support Python 3 already use Python 3.1 for that support.

You can find examples of how to set up your module or package to use Distribute for your Python 3 support under [Supporting multiple versions of Python with Distribute](http://python3porting.com/2to3.html#usingdistribute) as well as in the standard Distribute documentation[[2]](http://python3porting.com/strategies.html#distributedoc).

## Python 2 and Python 3 without conversion

In many cases it’s often perfectly feasible to modify the code so that it runs under both Python 2 and Python 3 without needing any conversion, although you have to apply several tricks to avoid the incompatibilities between Python 2 and Python 3.

Python 2.6 and 2.7 have a lot of forward compatibility, making supporting Python 2.6 and Python 3 much easier than supporting Python 2.5 and Python 3. Supporting 2.5 or even older versions means you have to employ more tricks. Python 3.3 also re-introduces the u'' literal for strings, which helps with one of the major difficulties in supporting Python 3.

Benjamin Peterson’s excellent six module[[3]](http://python3porting.com/strategies.html#six) also helps by wrapping much of the incompatibilities, and since the need to support older Python versions is shrinking, supporting both Python 2 and Python 3 without conversion is becoming the preferred method.

There are also cases where you can’t use Distribute, or don’t want to. You may need to distribute your code in a format that is not installable with Distutils and therefore not Distribute. In those cases you can’t use Distribute’s 2to3 support and then using 2to3 is more work and not using 2to3 becomes a more attractive prospect.

Even if you do use 2to3 for your project as a whole, you still may end up with having to write some code so it runs on both Python 2 and Python 3 without conversion. This is useful for bootstrapping scripts and setup scripts or if your code generates code from strings, for example to create command line scripts. You can of course have two separate strings depending on the Python version, or even run 2to3 on the string using lib2to3. However, in these cases it’s generally easier to make the generated code snippets run on all Python versions without 2to3.

My recommendation for the development workflow if you want to support Python 3 without using 2to3 is to run 2to3 on the code once and then fix it up until it works on Python 3. Only then introduce Python 2 support into the Python 3 code, using six where needed. Add support for Python 2.7 first, and then Python 2.6. Doing it this way can sometimes result in a very quick and painless process.

There is also a tool called python-modernize which will do a 2to3-type conversion of your code, but it will keep Python 2 compatibility together with the six library. This can be a good start.

More information on the techniques necessary to do this is in the chapter[Supporting Python 2 and 3 without 2to3 conversion](http://python3porting.com/noconv.html#noconv-chapter).

## Using 3to2

The 2to3 tool is flexible enough for you to define what changes should be done by writing “fixers”. Almost any kind of Python code conversion is imaginable here and 3to2[[4]](http://python3porting.com/strategies.html#to2) is a set of fixers written by Joe Amenta that does the conversion from Python 3 to Python 2. This enables you to write your code for Python 3 and then convert it to Python 2 before release.

However, there is no Distribute support for 3to2 and also Python 2.5 or earlier do not include the required lib2to3 package. Therefore 3to2 currently remains only an interesting experiment, although this may change in the future.

## Which strategy is for you?

### Applications

Unless your code is a reusable package or framework you probably do not need to support older versions of Python, unless some of your customers are stuck on Python 2 while others demand that you support Python 3. In most cases you can just drop Python 2 support completely.

### Python modules and packages

If you are developing some sort of module or package that other Python developers use you would probably like to support both Python 2 and Python 3 at the same time. The majority of your users will run Python 2 for some time to come, so you want to give them access to new functionality, but if you don’t support Python 3, the users of Python 3 must find another package to fulfill their need.

Today you typically only need to support Python 2.7, Python 3.4 and Python 3.5. These versions have enough backwards and forwards compatibility to make it easy to make code that runs on both Python 2 and Python 3. So this is the recommended strategy for reusable packages.

## Summary

In general, if you write end-user software, you can just switch to Python 3, starting with a one-time run of 2to3 on your code. If you write a Python package you want to support both Python 2 and Python 3 at the same time, and you can drop Python 2.5 support, try first to support Python 2 and 3 without 2to3conversion.

ABOUT DATABASE

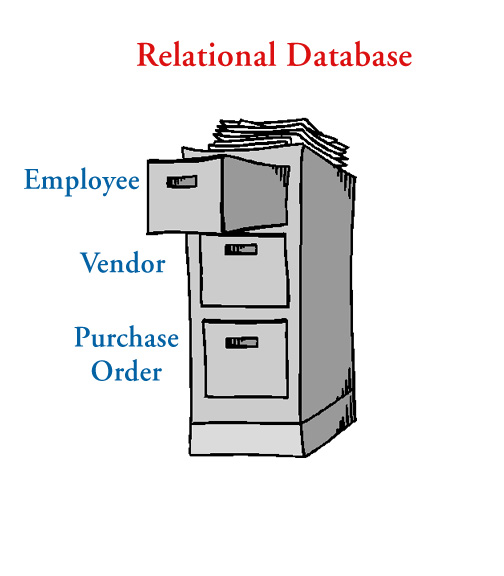
#### What is a database?

A *database* is a collection of data which is organized in a manner that can be easily retrieved. They consist of fields, records, and files, much like a telephone book. A *field* is a single piece of information (your last name is one field); a *record* is one complete set of fields (your entry in the telephone book is a record); and a *file* is a collection of records (the entire telephone book). When you look at the phone book for someone's telephone number, you are the "search engine" for the telephone book database. The only difference is that the search engine included with database programs is probably a little faster than you are at retrieving information.

#### What is a Relational Database?

A *relational database* is a single database spread across multiple tables. Think of a database as a file cabinet and each drawer of the file cabinet is a table.

Example. All of the employee information is kept in the first drawer; all of the vendor information is kept in the second drawer; and all of the purchase orders are kept in the third drawer. To connect each of these drawers (which is what makes it "relational"), a set piece of data from one drawer has to be present in one of the other drawers.



For instance, a purchase order will have a vendor name, address, & phone number on it, along with the items purchased, purchase price, and any discounts. If the vendor name is the "set" piece of data, it is entered in the purchase order table, but the address & phone number are retrieved from the vendor table instead of having to re-type that information each time it is needed in the purchase order table.

#### When Should I Use Access to Store Data?

Databases, like Access, are used to store large quantities of information. The information can be viewed, sorted, manipulated, retrieved, and printed in various ways. The database gives you the flexibility to obtain this data in multiple formats. If the information you need to store is vast, interrelated, and you need to retrieve it fast & accurately, Access is probably the way to go.

#### Terms and Definitions

Field

The smallest piece of data in a table.

Cell

Where one row and one column intersect on a table.

Record

A collection of all fields relating to one person, place, or thing (i.e., Cindy Ireland's mailing address) occupying a single row of a table.

Table

A collection of records (i.e., all WSU employee's mailing address information) with one record occupying a single row with one or many fields per row. A database can consist of multiple tables.

Query

Asks specific questions about the data in the database.

Forms

A method of entering, displaying, and searching data in other than the spreadsheet-like environment of tables and queries (a graphical user interface).

Reports

Summarizes and formats data in the "look" that you want for either table or query data.

Object

Components that make up a database (i.e., tables, forms, queries, reports, macros, & modules).

What is Database?

1. A database is a collection of data that is associated with a topic.
2. A database is an organized collection of related information.

Types of Databases:

The two main types of databases are Flat and relational:

1. The flat database systems are jammed together into one single table.
2. Relational database system is more complex and hence is good at cutting down on storage space or duplicated material – the database is split into multiple tables and each table holds a portion of total data.
3. To read a full article on TechNet about Relational Database design: Click here.

Table Creation:

If you want to create a table, you would need to open a new database or a preexisting one. Once you open the database, you can create a table three ways. You can create a table using:

1. A Wizard.
2. Design View.
3. By entering data.

Wizard View:

The wizard view is of course the easiest way to go. Let’s look below and see how it is sed: Simply follow the directions and prompts. You can see that, by choosing sample fields, you can then enter them into your new table. Remember, a table is where you are going to begin to enter your data to create content for your database.

Design View:

Design view is a little more difficult, but has the same basic functionality. Make sure that you know both. On the bottom of the dialog box below is a little pane that will help you along. Pay attention to what you need to do, as well as use help and the office assistant you are completely lost.

#### 

##### Objective of the Project

A text editor is program that allows you to open, view, and edit plain text files. Unlike word processors, text editors do not add formatting to text, instead focusing on editing functions for plain text.

Text editors are used by a wide variety of people, for a wide variety of purposes. Software programmers and web developers use text editors to write and edit in programming and markup languages. This is one of the primary purposes of text editors, and many of the features of text editing software are built to help these users read and write code. However, text editors are ideal tools for anyone who needs to write quickly and simply, read source code, or create text files.

If you have Windows, you already have the default Notepad application installed on your system. Notepad is a very basic text editor with minimal features and low capabilities. If you only need a text editor for small tasks, Notepad will work just fine. However, Notepad's limitations become apparent if you try to do some more heavy duty text editing.

With more advanced text editors come more advanced and useful features. Syntax highlighting, one of the most useful tools in a text editor's repertoire, color codes text based on the programming/markup language it written in. Other staple text editing features not included in Notepad are large file support, advanced find and replace, vertical selection editing, and document comparison, and more.

If you're looking for a Notepad replacement, many text editors are available online for download and purchase. The quality of such text editors varies greatly, and it's a good idea to research the different kinds available to find one that matches your needs before making a decision. Most text editors that you can buy allow you to download a trial version before committing to a purchase, and this is a great way to figure out what works for you.

**Project Category Tools and Environment**

* **Operating system: - Windows**
* **Front End Tools: - Python**

**Hardware and Software Requirements:**

**Hardware required:**

* IBM compatible Micro computer with hard disk i.e. PC/XT or PC/ AT
* Personal Computer
* HP DeskJet 3500 series Printer for the printouts.

**Software required:**

* Windows
* Python 3.7

**Systems Analysis and Design**

Systems analysis and design refers to the process of examining a business situation with the intent of improving it through better procedures and methods. Systems development can generally be thought of as having two major components:

1. Systems Analysis

2. Systems Design.

Systems Analysis is the process of gathering and interpreting facts, diagnosing problems and using the information to recommend improvement to the system. In brief, we can say that analysis specifies what the system should do. Design states hew to accomplish the objective.

Systems Design is the process of planning a new system or replace or complement an existing system. But before this planning can be done, we must thoroughly understand the existing system and determine how computers can best be used to make its operation more effective.

Systems study may be defined as "a study of the operations of a set of connected elements and of the interconnections between these elements". It shows clearly that one cannot ignore any part or element of a system without first finding out the effect that element has on the operation of the system as a whole. We can understand this with the help of systems analysis.

There is a difference between "systems approach" and "systems analysis" also.

The systems approach shows a set of procedure for solving a particular problem. It applies scientific methods to observe, clarify, identify and solve a problem with special care being taken to understand the inter- relatedness between elements and their system characteristics.

However, systems analysis is a management technique which helps us in designing a new system or improving an existing system.

System development, a process consisting of the two major steps of systems analysis and design, starts when management or sometimes system development personnel feel that a new system or an improvement in the existing system is required. The systems development life cycle is classically, thought of as the set of activities- that analysts, designers and users carry out to develop and implement an information system.

The systems development life cycle consists of the following activities:

· Preliminary investigation

· Determination of system requirements

· Design of system

· Development of software

· Systems testing

· Implementation, evaluation and maintenance

Preliminary investigation

A request to take assistance from information systems can be made for many reasons, but in each case someone in the organization initiates the request When the request is made. the first systems activity the preliminary investigation begins. This activity has three parts:

i. request, clarification

ii. feasibility study

iii. request approval

Determination of system requirements

At the heart of systems analysis is a detailed understanding of all important facets of the business area under investigation. The key questions are

· What is being done?

· How is it being done?

· How frequently does it occur?

· How great is the volume of transactions or decisions?

· How well is the task being performed?

· Does a problem exist?

· If a problem exists, how serious is it? What is the underlying cause?

To answer the above questions, systems analysts discuss with different category of persons to collect the facts about the business process and their opinions of why things happen as they do and their views for changing the existing process. During analysis. Data are collected on the available files, decision points and transactions handled by the present system. Some tools are used in analysis like data flow diagrams, interviews, on-site observations and questionnaires. Detail investigations also require the study of manuals and reports. Once the structured analysis is completed, analyst has a firm understanding of what is to be done?

· Design of system

The design of an information system produces the details that clearly describe how a system will meet the requirements identified during systems analysis. Systems specialists often refer to this stage as logical design, in contrast to the process of developing program software, which is referred to as physical design.

Systems analysts begin the design process by identifying reports and other outputs system will produce. Then the specific data on each are pinpointed. The systems design also describes the data to be input, calculated or stored. Individual data items and calculation procedures are written in detail. Designers select file structures and storage devices, such as magnetic disk, magnetic tape or even paper files. Procedures they write tell how to process the data and produce the output. The documents containing the design specifications portray the design in many different ways-charts, tables, and special symbols. The detailed design information is passed on to the programming staff for the purpose of software development. Designers are responsible for providing programmers with complete and clearly out lined software specifications.

Development of software

Software developers may install purchased software or they may develop new, custom- designed programs. The choice depends on the cost of each option, the time available to develop software and the availability of programmers. Generally it has been observed that programmers are part of permanent professional staff in a big organization. In smaller organization, without programmers, outside programming services may be hired or retained on a contractual basis. Programmers are also responsible for documenting the program, providing an explanation of how and why certain procedures are coded in specific ways. Documentation is essential to test the program and carry on maintenance once the application has been installed.

· Systems Testing

During systems testing, the system is used experimentally to ensure that the software does not fail. In other words, we can say that it will run according to its specifications and in the way users expect. Special test data are input for processing, and the results examined. A limited number of users may be allowed to use the system so that analyst can see whether they try to use it in unforeseen ways. It is desirable to discover any surprises before the organization implements the system and depends on it.

Implementation, evaluation and maintenance

Implementation is the process of having systems personnel check out and put new equipment into use, train users, install the new application and construct any files of data needed to use it. This phase is less creative than system design. Depending on the size of the organization that will be involved in using the application and the risk involved in its use, systems developers may choose to test the operation in only one area of the firm with only one or two persons. Sometimes, they will run both old and new systems in parallel way to com- pare the results. In still other situations, system developers stop using the old system one day and start using the new one the next.

Evaluation of the system is performed to identify its strengths and weaknesses. The actual evaluation can occur along any of the following dimensions:

1. Operational Evaluation: Assessment of the manner in which the system functions, including case of use, response time, overall reliability and level of utilization

ii. Organisational Impact: Identification and measurement of benefits to the organisation in such areas as financial concerns, operational efficiency and competitive impact.

iii. User Manager Assessment: Evaluation of the attitudes of senior and user manager within the organisation, as well as end-users.

iv . Development Performance: Evaluation of the development process in accordance with such yardsticks as overall development time and effort, conformance to budgets and standards and other project management criteria.

Maintenance is necessary to eliminate errors in the working system during its working life and to tune the system to any variations in its working environment often small system deficiencies are found as a system is brought into operations and changes are made to remove them. System planners must always plan for resource availability to carry out these maintenance functions. The importance of maintenance is to continue to bring the new system to standards.

Coding

import tkinter

import os

from tkinter import \*

from tkinter.messagebox import \*

from tkinter.filedialog import \*

class Notepad:

\_\_root = Tk()

# default window width and height

\_\_thisWidth = 300

\_\_thisHeight = 300

\_\_thisTextArea = Text(\_\_root)

\_\_thisMenuBar = Menu(\_\_root)

\_\_thisFileMenu = Menu(\_\_thisMenuBar, tearoff=0)

\_\_thisEditMenu = Menu(\_\_thisMenuBar, tearoff=0)

\_\_thisHelpMenu = Menu(\_\_thisMenuBar, tearoff=0)

# To add scrollbar

\_\_thisScrollBar = Scrollbar(\_\_thisTextArea)

\_\_file = None

def \_\_init\_\_(self, \*\*kwargs):

# Set icon

try:

self.\_\_root.wm\_iconbitmap("Notepad.ico")

except:

pass

# Set window size (the default is 300x300)

try:

self.\_\_thisWidth = kwargs['width']

except KeyError:

pass

try:

self.\_\_thisHeight = kwargs['height']

except KeyError:

pass

# Set the window text

self.\_\_root.title("Untitled - Notepad")

# Center the window

screenWidth = self.\_\_root.winfo\_screenwidth()

screenHeight = self.\_\_root.winfo\_screenheight()

# For left-alling

left = (screenWidth / 2) - (self.\_\_thisWidth / 2)

# For right-allign

top = (screenHeight / 2) - (self.\_\_thisHeight / 2)

# For top and bottom

self.\_\_root.geometry('%dx%d+%d+%d' % (self.\_\_thisWidth,

self.\_\_thisHeight,

left, top))

# To make the textarea auto resizable

self.\_\_root.grid\_rowconfigure(0, weight=1)

self.\_\_root.grid\_columnconfigure(0, weight=1)

# Add controls (widget)

self.\_\_thisTextArea.grid(sticky=N + E + S + W)

# To open new file

self.\_\_thisFileMenu.add\_command(label="New",

command=self.\_\_newFile)

# To open a already existing file

self.\_\_thisFileMenu.add\_command(label="Open",

command=self.\_\_openFile)

# To save current file

self.\_\_thisFileMenu.add\_command(label="Save",

command=self.\_\_saveFile)

# To create a line in the dialog

self.\_\_thisFileMenu.add\_separator()

self.\_\_thisFileMenu.add\_command(label="Exit",

command=self.\_\_quitApplication)

self.\_\_thisMenuBar.add\_cascade(label="File",

menu=self.\_\_thisFileMenu)

# To give a feature of cut

self.\_\_thisEditMenu.add\_command(label="Cut",

command=self.\_\_cut)

# to give a feature of copy

self.\_\_thisEditMenu.add\_command(label="Copy",

command=self.\_\_copy)

# To give a feature of paste

self.\_\_thisEditMenu.add\_command(label="Paste",

command=self.\_\_paste)

# To give a feature of editing

self.\_\_thisMenuBar.add\_cascade(label="Edit",

menu=self.\_\_thisEditMenu)

# To create a feature of description of the notepad

self.\_\_thisHelpMenu.add\_command(label="About Notepad",

command=self.\_\_showAbout)

self.\_\_thisMenuBar.add\_cascade(label="Help",

menu=self.\_\_thisHelpMenu)

self.\_\_root.config(menu=self.\_\_thisMenuBar)

self.\_\_thisScrollBar.pack(side=RIGHT, fill=Y)

# Scrollbar will adjust automatically according to the content

self.\_\_thisScrollBar.config(command=self.\_\_thisTextArea.yview)

self.\_\_thisTextArea.config(yscrollcommand=self.\_\_thisScrollBar.set)

def \_\_quitApplication(self):

self.\_\_root.destroy()

# exit()

def \_\_showAbout(self):

showinfo("Notepad", "Rahul UIET, Kurukshetras")

def \_\_openFile(self):

self.\_\_file = askopenfilename(defaultextension=".txt",

filetypes=[("All Files", "\*.\*"),

("Text Documents", "\*.txt")])

if self.\_\_file == "":

# no file to open

self.\_\_file = None

else:

# Try to open the file

# set the window title

self.\_\_root.title(os.path.basename(self.\_\_file) + " - Notepad")

self.\_\_thisTextArea.delete(1.0, END)

file = open(self.\_\_file, "r")

self.\_\_thisTextArea.insert(1.0, file.read())

file.close()

def \_\_newFile(self):

self.\_\_root.title("Untitled - Notepad")

self.\_\_file = None

self.\_\_thisTextArea.delete(1.0, END)

def \_\_saveFile(self):

if self.\_\_file == None:

# Save as new file

self.\_\_file = asksaveasfilename(initialfile='Untitled.txt',

defaultextension=".txt",

filetypes=[("All Files", "\*.\*"),

("Text Documents", "\*.txt")])

if self.\_\_file == "":

self.\_\_file = None

else:

# Try to save the file

file = open(self.\_\_file, "w")

file.write(self.\_\_thisTextArea.get(1.0, END))

file.close()

# Change the window title

self.\_\_root.title(os.path.basename(self.\_\_file) + " - Notepad")

else:

file = open(self.\_\_file, "w")

file.write(self.\_\_thisTextArea.get(1.0, END))

file.close()

def \_\_cut(self):

self.\_\_thisTextArea.event\_generate("<<Cut>>")

def \_\_copy(self):

self.\_\_thisTextArea.event\_generate("<<Copy>>")

def \_\_paste(self):

self.\_\_thisTextArea.event\_generate("<<Paste>>")

def run(self):

# Run main application

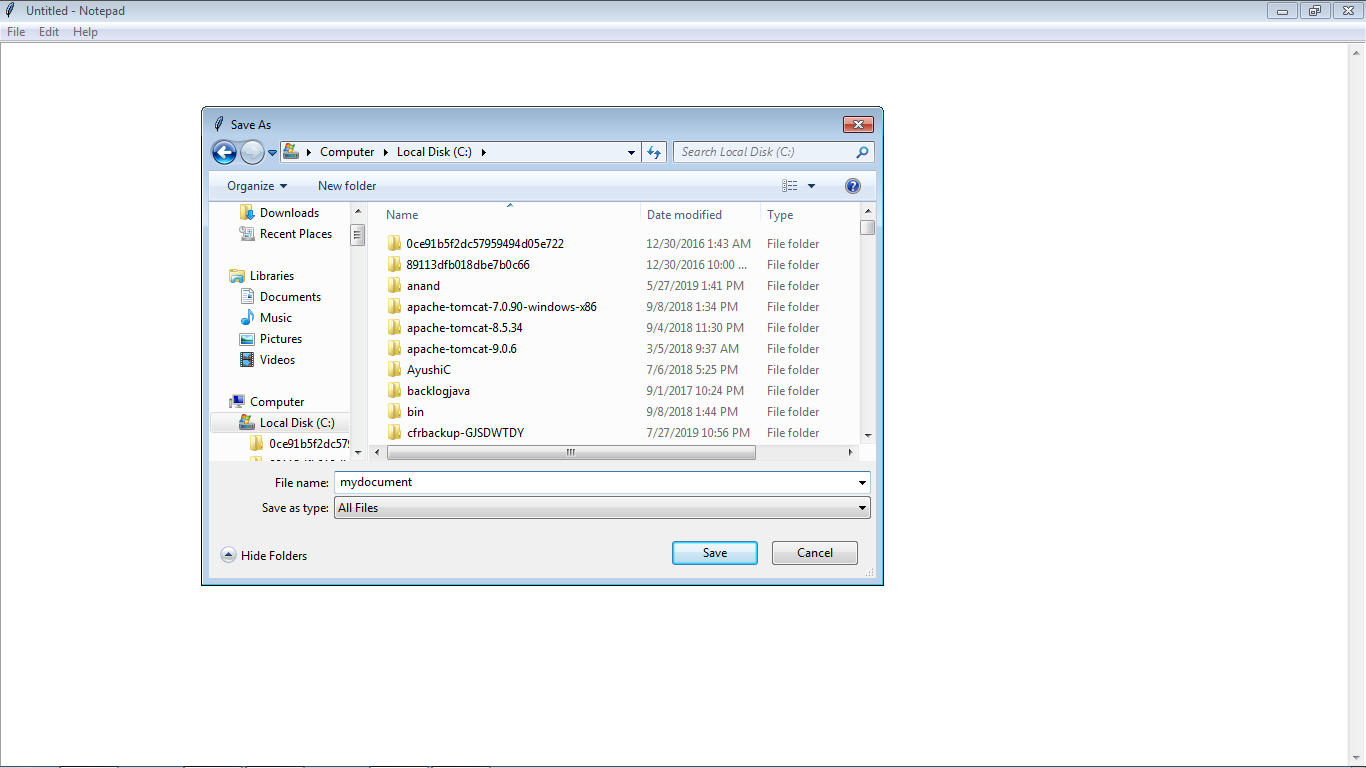
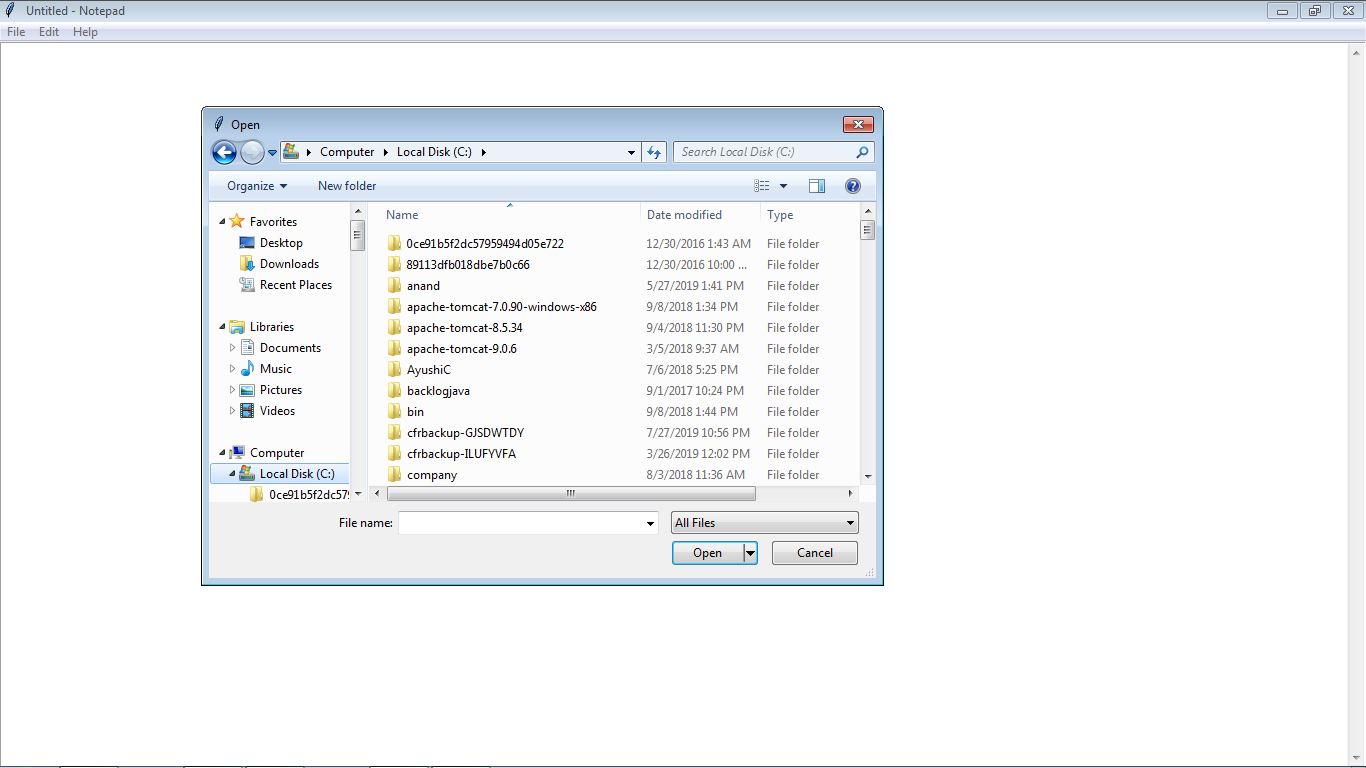
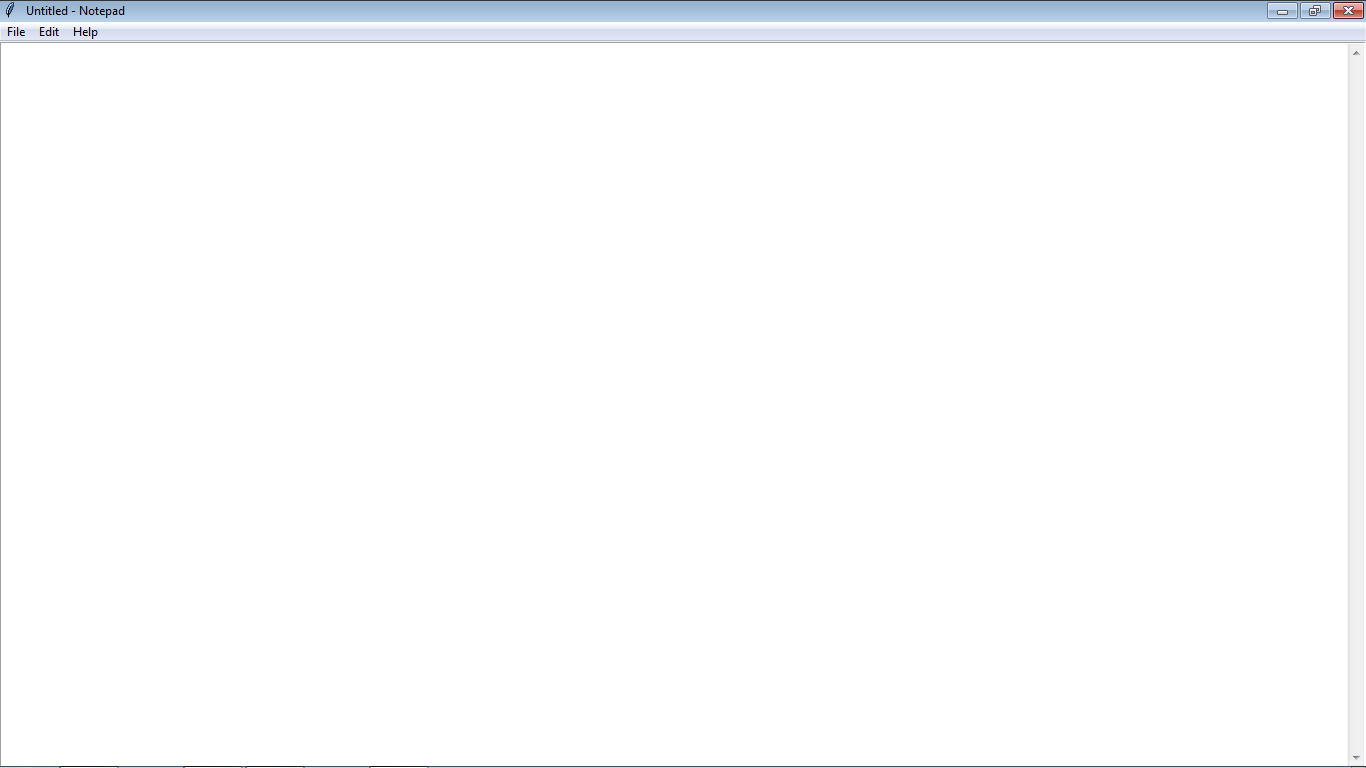
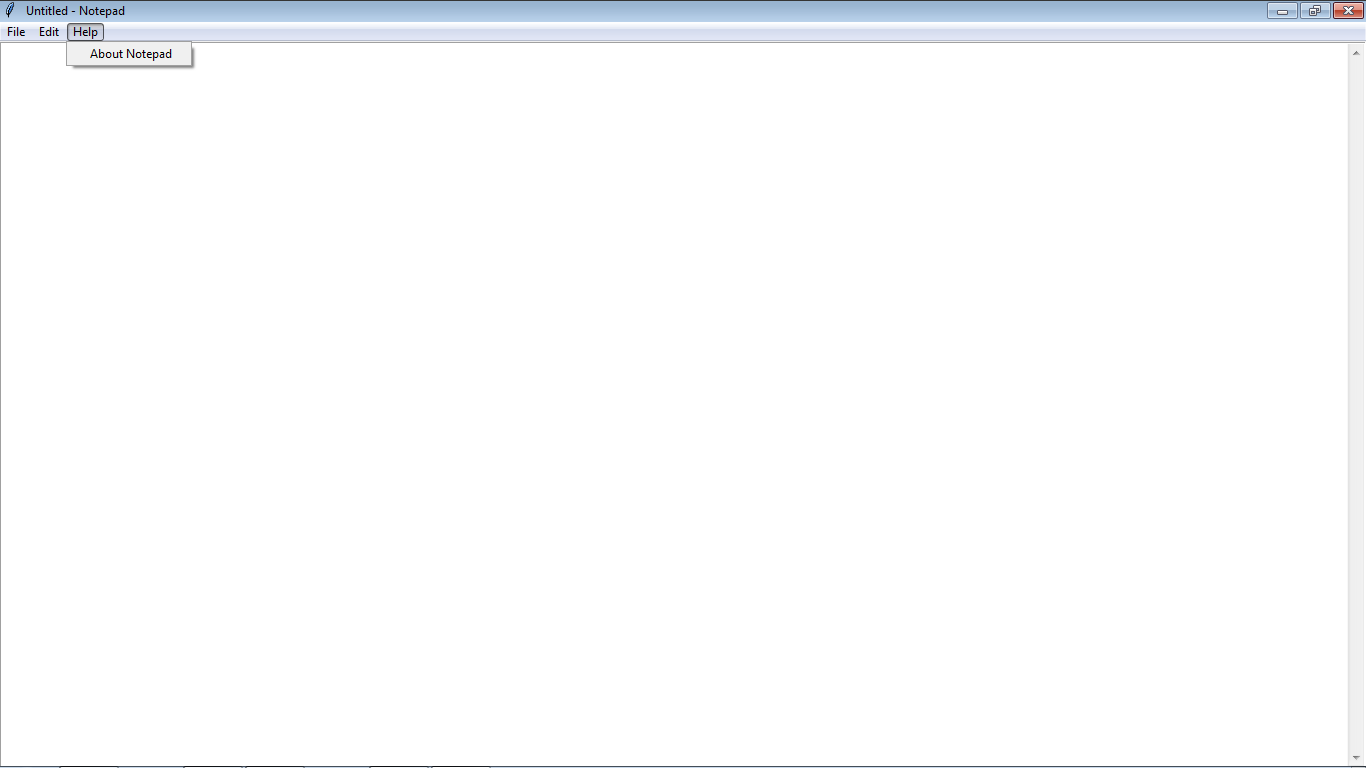
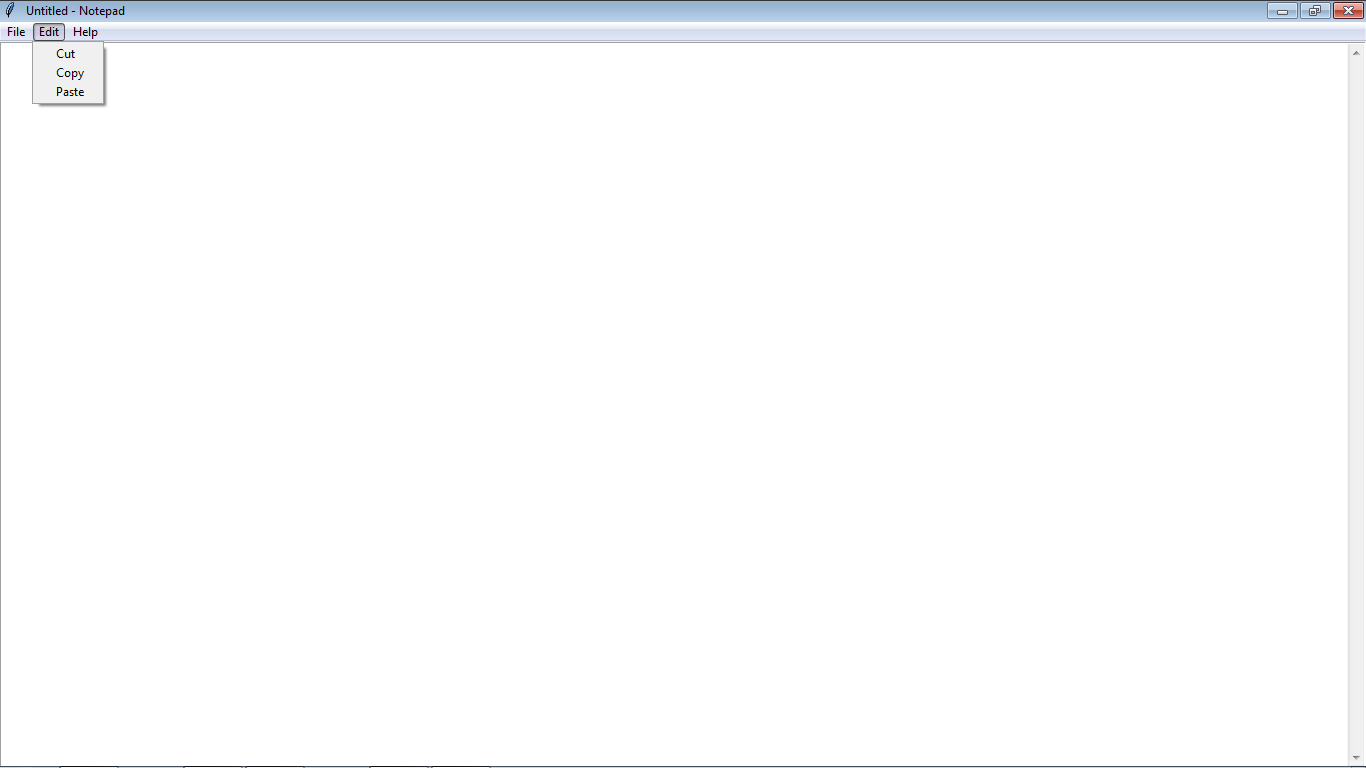
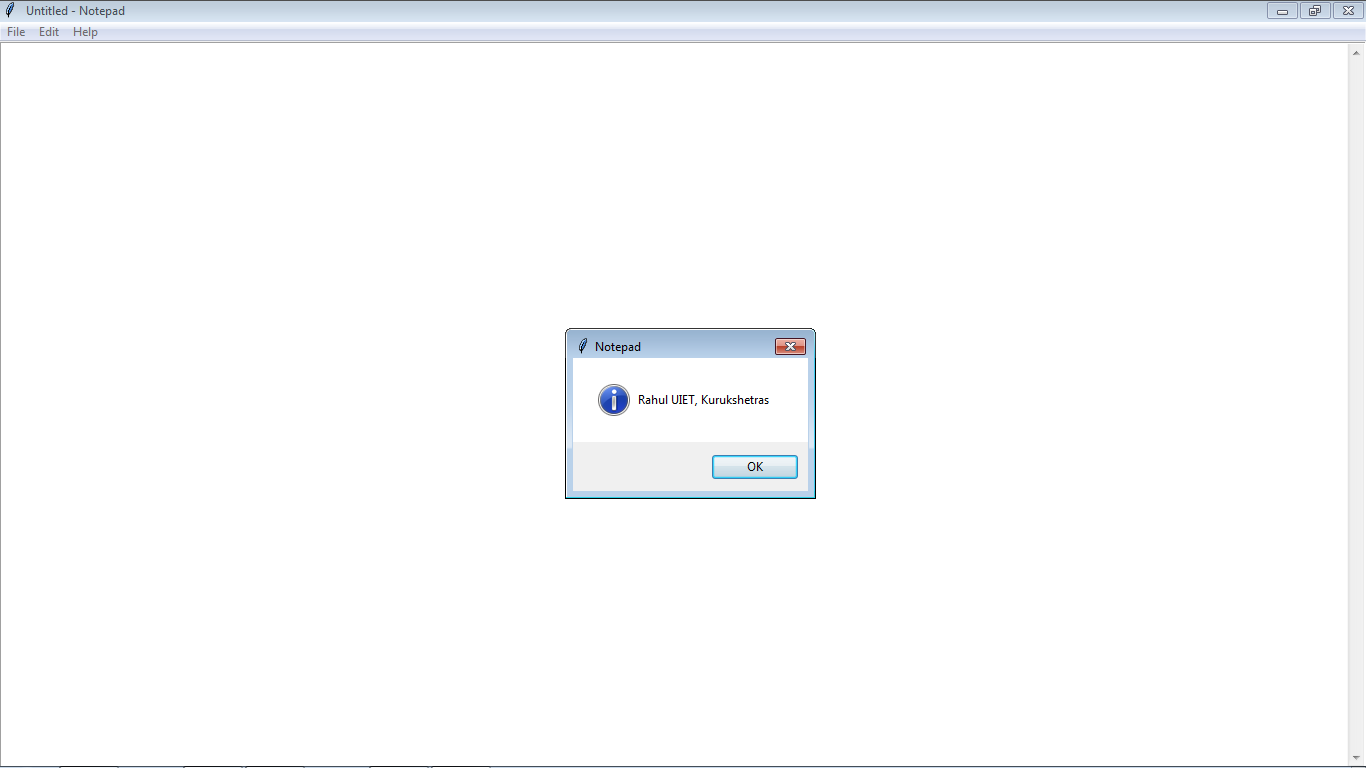
self.\_\_root.mainloop()

# Run main application

notepad = Notepad(width=600, height=400)

notepad.run()

Screenshots



Bibliography

* <https://www.codecademy.com/tracks/python>
* <http://www.tutorialspoint.com/python/>
* <https://www.codementor.io/community/topic/python>
* <http://www.pythonchallenge.com/>
* <https://developers.google.com/edu/python/?hl=en>