The Restaurant Billing Software

A PROJECT REPORT SUBMITTED IN PARTIAL FULLFILLMENT OF THE REQUIREMENT FOR THE AWARD OF THREE-YEAR DIPLOMA

IN

COMAPUTER ENGINEERING

GOVT. OF NCT DELHI

BOARD OF TECHNICAL EDUCATION DELHI

AMBEDKAR INSTITUTE OF TECHNOLOGY

SHAKARPUR, DELHI-110092

SESSION-2017 -2020

UNDER SUPERVISION OF
MR. H.S BHATIA SIR
IN-CHARGE
(COMPUTER ENGINEERING)

SUBMITTED BY
ROHIT KUMAR MAHATO
ROLL NO-1702051050
COMPUTER ENGINEERING

CERTIFICATE

This is to certify that thesis entitled “ ***The Restaurant Billing Software****” is submitted by* ***Rohit Kumar Mahato****, Roll no.* ***1702051050****, in the partial fulfilment of the requirement for the award of three year diploma in Computer Engineering,* ***AMBEDKAR INSTITUTE OF TECHNOLOGY*** *is a record of work done by him under my supervision and guidance.*

Under the guidance of

Mr. HS BHATIA

(Project In-charge)

DECLARATION

*This is to declare that this report has been written **by Rohit Kumar Mahato**. No part is plagiarized from other sources. All information include from other sources have been duly acknowledge. I aver if any part of the report is founder to be plagiarized, I'll be fully responsible.*

Rohit Kumar Mahato

*Roll no.:- **1702051050***

CE – 6th Sem

COPYRIGHT

Mr. H.S. Bhatia

(H.O.D.)

Department of Computer engineering,

Ambedkar institute of technology (AIT)

Shakarpur, NEW DELHI.

ACKNOWLEDGEMENT

We take this opportunity to express my deepest and sincere gratitude to our supervisor ***Mr. HS BHATIA*** *, HOD, Department of Computer engineering in Ambedkar institute of technology for his insightful advice, motivating suggestions, invaluable guidance, help and support in successful completion of this project and also for his constant encouragement and advice throughout our diploma.*

We express our deep gratitude to ***Shubha mam****, teacher, Department of Computer engineering in Ambedkar Institute of technology for her regular support, co-operation, and co-ordination.*

The in-time facilities provided by the department throughout the diploma are also equally acknowledgeable.

We would like to convey our thanks to the teaching and non teaching staff of the Department of Computer Engineering, AIT for their invaluable help and support throughout the period of Diploma. We are also grateful to all our classmates for their help, encouragement and invaluable suggestions.

Finally, yet more importantly, I would like to express my deep appreciation to my parents, sister and brother for their perpetual support and encouragement throughout the diploma period.

Rohit Kumar Mahato
1702051050

ABSTRACT

*This project is titled as **Restaurant Billing Software** is Restaurant Billing Software which helps in operating and managing a restaurant efficiently. It is a software used by the restaurant to streamline the process of catering to the customers and conveniently improve the employee work rate and overall productivity of the restaurant. This software is developed in Python, which mainly focuses on the management of billing in a restaurant.*

Restaurant Billing Software is a windows-based application written for 64-bit Windows Operating System. It is designed to help restaurant employee manage their bills. This software is easy for both beginners and advance. It features a familiar and well thought-out, an attractive user interface, combined with strong searching insertion and reporting capabilities.

It is a software used by the restaurant to streamline the process of catering to the customers and conveniently improve the employee work rate and overall productivity of the restaurant. Aim of this software is to serve customers what they prefer, reduce the waiting time and maximize potential of the restaurant through its vast features.

This software has mainly four modules: -

- ➢ *Insertion to database Module - User Friendly input screen.*
- ➢ *Extraction from databased module – Attractive output screen.*
- ➢ *Generating Bill: - Bill is being generated*
- ➢ *Searching Bill - Search for a specific bill using unique bill ID*

Billing is an easy to use and comprehensive GST Invoicing & Billing App for Retail and Restaurant. It runs both on mobile and computer. This GST

compliant point of sale (POS) makes it easier for you to keep track of your business and pay more importance to your business growth.
Surveys show that the billing systems are fast replacing the cash drawers. Unlike cash drawers, the billing systems has the option of maintaining your stock, keeping a tab on your employees, offer customer loyalty and do more in addition to billing. Furthermore, it also helps in reducing the cost of maintaining your business.
Just Billing GST Software lets you conveniently generate invoices, run customer loyalty program, manage and monitor sales, procurement activities, optimize inventory, manage accounts, taxes and reports at the store level or consolidated analysis at the cloud back office anytime. Look modern and surprise your customer by sending invoice using SMS.
Multi store retail management become much easier and real time with cloud based back office management. Just Billing works with or without Internet. And all this comes at an affordable price, without the added baggage of maintaining an in-house IT infrastructure!

Windows POS Software

Just Billing is a unique POS (Point of Sale) solution for micro, small and medium size business. The features perfectly match the business requirements of any Food joints, Service providers or Retail businesses.

Just Billing Windows is also helpful for multi-chain stores since it runs on cloud, which makes it easier to map the on-goings of multiple stores at the same time.

Increase profit by maintaining customer loyalty, updated account book, inventory, taxes and many more. Check your business reports from anywhere, anytime with real time updates from your back office portal. It is powerful enough to take over the entire store, billing and back office management, thereby helping you save time, money and energy.
This growth hacking solution is in your fingertips now.

Benefits of GST Billing Software

As a business everyone's focus is on growth. But a good technology solution enables and expedite your growth faster.

Increase profit

Increase revenue and optimize cost by strategic reports anywhere anytime.

Save time

Manage queue with faster billing, auto-inventory & account update, no manual effort required.

Enhanced business image

Groom your business with latest technology, send SMS/Email invoice.

More organized

Tap to bill, auto-update the inventory and accounts, and stay organized.

More customers

Offer customer loyalty and discounts to increase the customer base.

Prevent misuse

Auto-updating feature of inventory & accounting prevent misuse of materials and theft.

Accuracy

Say goodbye to human errors with automated billing, discounts & complex GST calculations.

Reconciliation

Automated reconciliation of sales, purchase, inventory, payments and GST.

INDEX

Technology Used

Python

***Python** is a general purpose, dynamic, high-level, and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.*

Python is easy to learn yet powerful and versatile scripting language, which makes it attractive for Application Development.

Python's syntax and dynamic typing with its interpreted nature make it an ideal language for scripting and rapid application development.

Python supports multiple programming pattern, including object-oriented, imperative, and functional or procedural programming styles.

Python is not intended to work in a particular area, such as web programming. That is why it is known as multipurpose programming language because it can be used with web, enterprise, 3D CAD, etc.

We don't need to use data types to declare variable because it is dynamically typed so we can write a=10 to assign an integer value in an integer variable.

Python makes the development and debugging fast because there is no compilation step included in Python development, and edit-test-debug cycle is very fast.

- *Python -2 VS Python -3*
  - *Python 3 syntax is simpler and easily understandable whereas Python 2 syntax is comparatively difficult to understand.*
  - *Python 3 default storing of strings is Unicode whereas Python 2 stores need to define Unicode string value with "u."*
  - *Python 3 value of variables never changes whereas in Python 2 value of the global variable will be changed while using it inside for-loop.*
  - *Python 3 exceptions should be enclosed in parenthesis while Python 2 exceptions should be enclosed in notations.*

- *Python 3 rules of ordering comparisons are simplified whereas Python 2 rules of ordering comparison are complex.*
- *Python 3 offers Range() function to perform iterations whereas, In Python 2, the xrange() is used for iterations.*

Python features

Python provides many useful features which make it popular and valuable from the other programming languages. It supports object-oriented programming, procedural programming approaches and provides dynamic memory allocation. We have listed below a few essential features.

I. *Easy to learn and use*
   Python is easy to learn as compared to other programming languages. Its syntax is straightforward and much the same as the English language. There is no use of the semicolon or curly-bracket, the indentation defines the code block. It is the recommended programming language for beginners.
II. *Expressive Language*

Python can perform complex tasks using a few lines of code. A simple example, the hello world program you simply type ***print("Hello World")****. It will take only one line to execute, while Java or C takes multiple lines.*

III. *Interpreted Language*

Python is an interpreted language; it means the Python program is executed one line at a time. The advantage of being interpreted language, it makes debugging easy and portable.

IV. *Cross-platform Language*

Python can run equally on different platforms such as Windows, Linux, UNIX, and Macintosh, etc. So, we can say that Python is a portable language. It enables programmers to develop the software for several competing platforms by writing a program only once.

V. *Free and Open Source*

Python is freely available for everyone. It is freely available on its official website www.python.org. It has a large community across the world that is *dedicatedly working towards make new python modules and functions. Anyone can contribute to the Python community. The open-source means, "Anyone can download its source code without paying any penny."*

VI. Object-Oriented Language

Python supports object-oriented language and concepts of classes and objects come into existence. It supports inheritance, polymorphism, and encapsulation, etc. The object-oriented procedure helps to programmer to write reusable code and develop applications in less code.

VII. Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our Python code. It converts the program into byte code, and any platform can use that byte code.

VIII. Large Standard Library

It provides a vast range of libraries for the various fields such as machine learning, web developer, and also for the scripting. There are various machine learning libraries, such as Tensor flow, Pandas, Numpy, Keras, and Pytorch, etc. Django, flask, pyramids are the popular framework for Python web development.

IX. GUI Programming Support

Graphical User Interface is used for the developing Desktop application. PyQT5, Tkinter, Kivy are the libraries which are used for developing the web application.

X. Integrated

It can be easily integrated with languages like C, C++, and JAVA, etc. Python runs code line by line like C,C++ Java. It makes easy to debug the code.

XI. Embeddable

The code of the other programming language can use in the Python source code. We can use Python source code in another programming language as *well. It can embed other language into our code.*

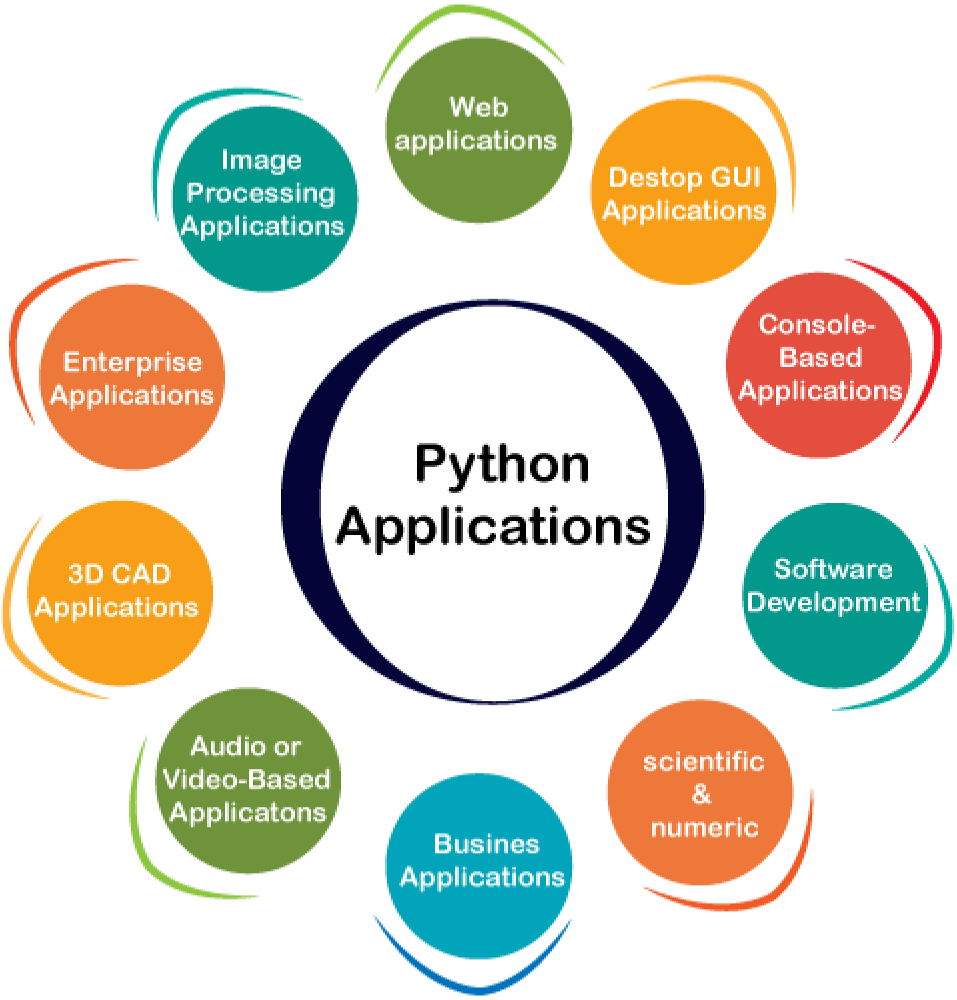
XII. Dynamic Memory Allocation

In Python, we don't need to specify the data-type of the variable. When we assign some value to the variable, it automatically allocates the memory to the variable at run time. Suppose we are assigned integer value 15 to x, then we don't need to write int x = 15. Just write x = 15.

Python Applications

Python is known for its general-purpose nature that makes it applicable in almost every domain of software development. Python makes its presence in every emerging field. It is the fastest-growing programming language and can develop any application.

Here, we are specifying application areas where Python can be applied.



***1**) Web Applications*

We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, beautifulSoup, Feedparser, etc. One of Python web-framework named Django is used on Instagram. Python provides many useful frameworks, and these are given below:

- *Django and Pyramid framework(Use for heavy applications)*
- *Flask and Bottle (Micro-framework)*
- *Plone and Django CMS (Advance Content management)*

2) Desktop GUI Applications

The GUI stands for the Graphical User Interface, which provides a smooth interaction to any application. Python provides a Tk GUI library to develop a user interface. Some popular GUI libraries are given below.

- *Tkinter or Tk*
- *wxWidgetM*
- *Kivy (used for writing multitouch applications)*
- *PyQt or Pyside*

3) Console-based Application

Console-based applications run from the command-line or shell. These applications are computer program which are used commands to execute. This kind of application was more popular in the old generation of computers. Python can develop this kind of application very effectively. It is famous for having REPL, which means the Read-Eval-Print Loop that makes it the most suitable language for the command-line applications.

Python provides many free library or module which helps to build the command-line apps. The necessary IO libraries are used to read and write. It helps to parse argument and create console help text out-of-the-box. There are also advance libraries that can develop independent console apps.

4) Software Development

Python is useful for the software development process. It works as a support language and can be used to build control and management, testing, etc.

- *SCons is used to build control.*
- *Buildbot and Apache Gumps are used for automated continuous compilation and testing.*
- *Round or Trac for bug tracking and project management.*

5) Scientific and Numeric

This is the era of Artificial intelligence where the machine can perform the task the same as the human. Python language is the most suitable language for Artificial intelligence or machine learning. It consists of many scientific and mathematical libraries, which makes easy to solve complex calculations.

Implementing machine learning algorithms require complex mathematical calculation. Python has many libraries for scientific and numeric such as Numpy, Pandas, Scipy, Scikit-learn, etc. If you have some basic knowledge of Python, you need to import libraries on the top of the code. Few popular frameworks of machine libraries are given below.

- *SciPy*
- *Scikit-learn*
- *NumPy*
- *Pandas*
- *Matplotlib*

6) Business Applications

Business Applications differ from standard applications. E-commerce and ERP are an example of a business application. This kind of application requires extensively, scalability and readability, and Python provides all these features.

Oddo is an example of the all-in-one Python-based application which offers a range of business applications. Python provides a Tryton platform which is used to develop the business application.

7) Audio or Video-based Applications

Python is flexible to perform multiple tasks and can be used to create multimedia applications. Some multimedia applications which are made by using Python are TimPlayer, cplay, etc. The few multimedia libraries are given below.

- *Gstreamer*
- *Pyglet*
- *QT Phonon*

8) 3D CAD Applications

The CAD (Computer-aided design) is used to design engineering related architecture. It is used to develop the 3D representation of a part of a system. Python can create a 3D CAD application by using the following functionalities.

- *Fandango (Popular)*
- *CAMVOX*
- *HeeksCNC*
- *AnyCAD*
- *RCAM*

9) Enterprise Applications

Python can be used to create applications that can be used within an Enterprise or an Organization. Some real-time applications are OpenERP, Tryton, Picalo, etc.

10) Image Processing Application

Python contains many libraries that are used to work with the image. The image can be manipulated according to our requirements. Some libraries of image processing are given below.

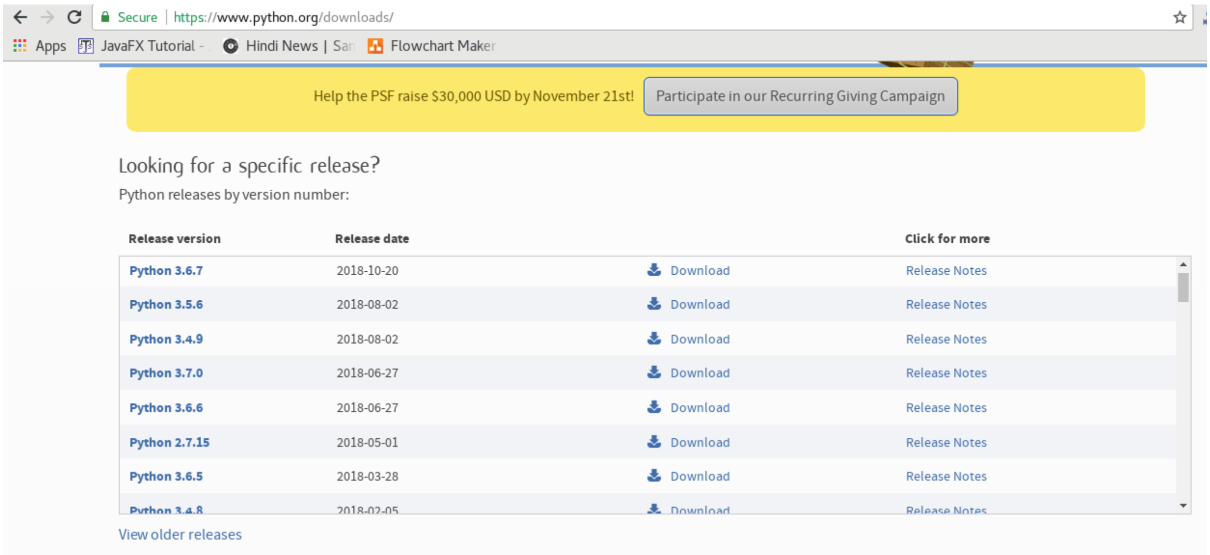
- *OpenCV*
- *Pillow*
- *SimpleITK*

How to Install Python (Environment Set-up)

In this section of the tutorial, we will discuss the installation of Python on various operating systems.

Installation on Windows

Visit the link https://www.python.org/downloads/ to download the latest release of Python. In this process, we will install Python 3.6.7 on our Windows operating system.



Help the PSF raise $30,000 USD by November 21st! Participate in our Recurring Giving Campaign

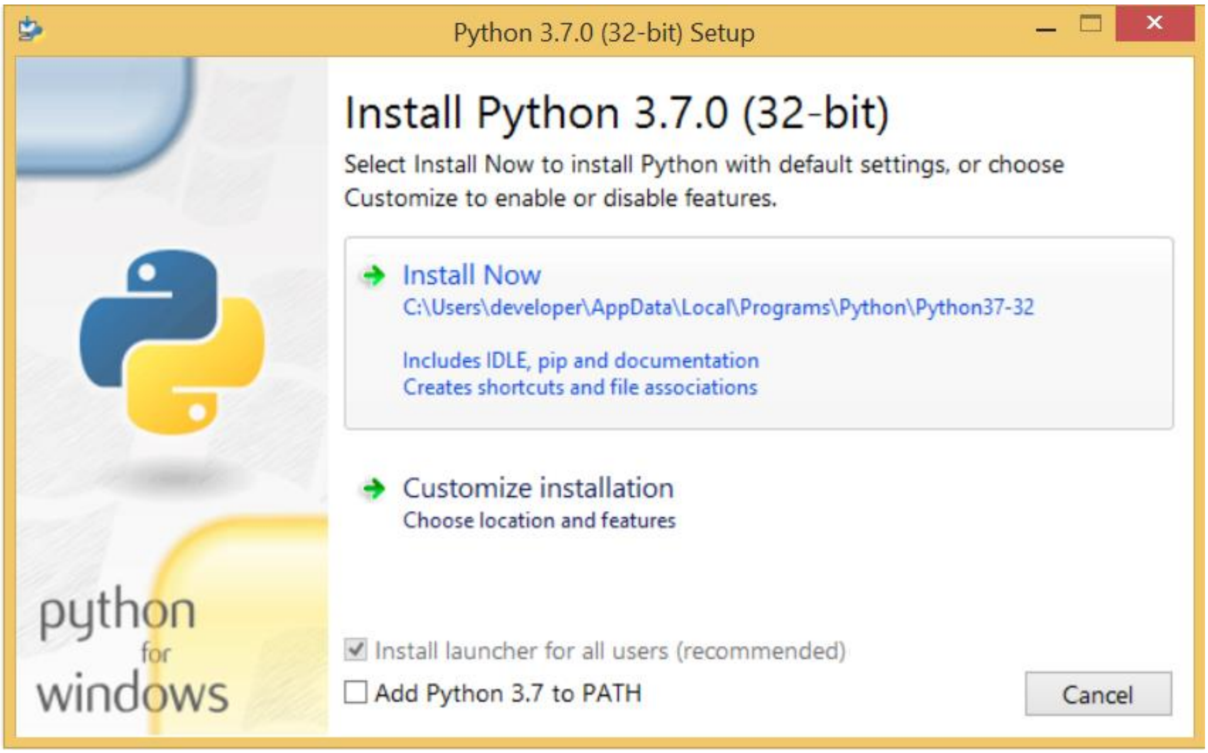
Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.6.7	2018-10-20	Download	Release Notes
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

View older releases

Double-click the executable file, which is downloaded; the following window will open. Select Customize installation and proceed.



Install Python 3.7.0 (32-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\developer\AppData\Local\Programs\Python\Python37-32

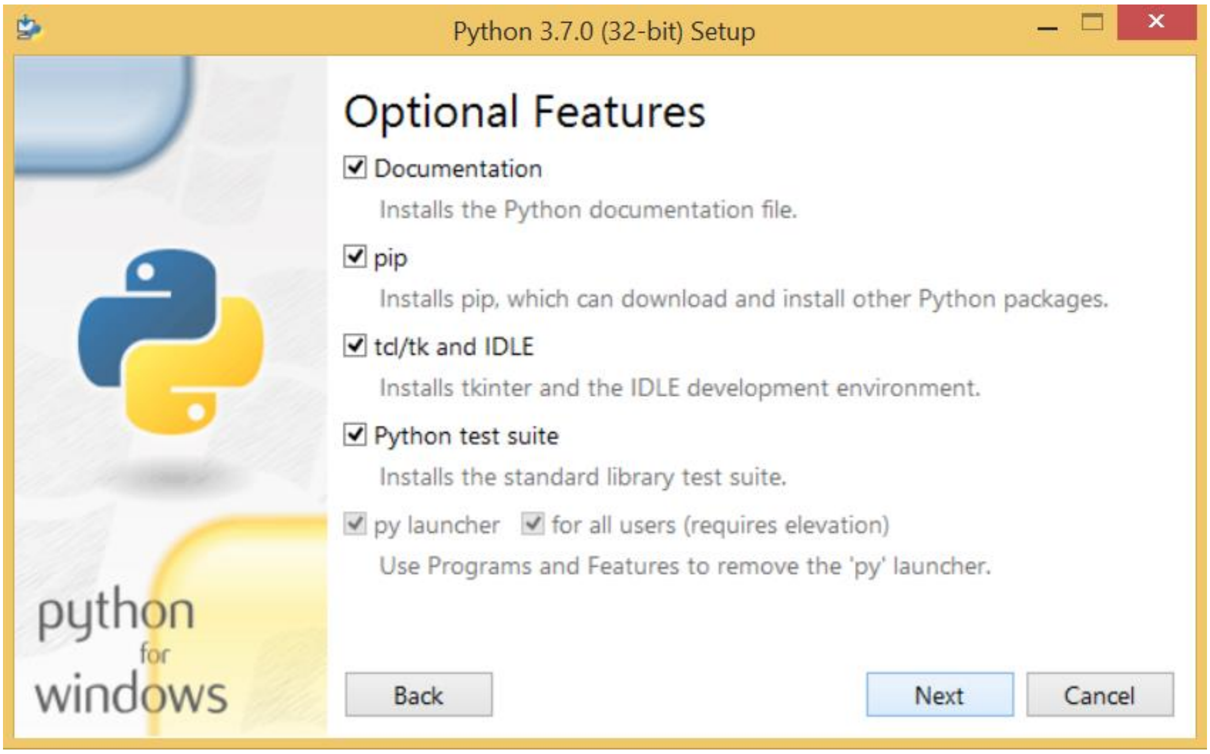
Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Install launcher for all users (recommended)

□ Add Python 3.7 to PATH

The following window shows all the optional features. All the features need to be installed and are checked by default; we need to click next to continue.



Python 3.7.0 (32-bit) Setup

Optional Features

☑ Documentation
Installs the Python documentation file.

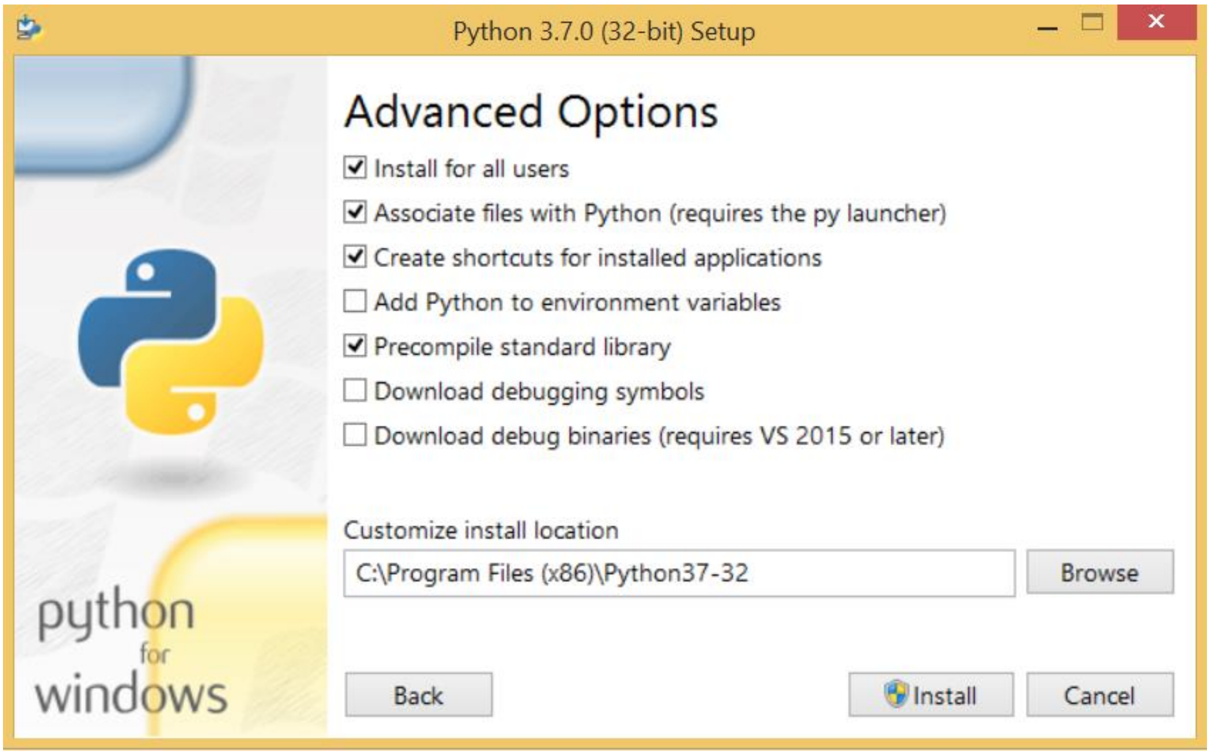
☑ pip
Installs pip, which can download and install other Python packages.

☑ tcl/tk and IDLE
Installs tkinter and the IDLE development environment.

☑ Python test suite
Installs the standard library test suite.

☑ py launcher ☑ for all users (requires elevation)
Use Programs and Features to remove the 'py' launcher.

The following window shows a list of advanced options. Check all the options which you want to install and click next. Here, we must notice that the first check-box (install for all users) must be checked.



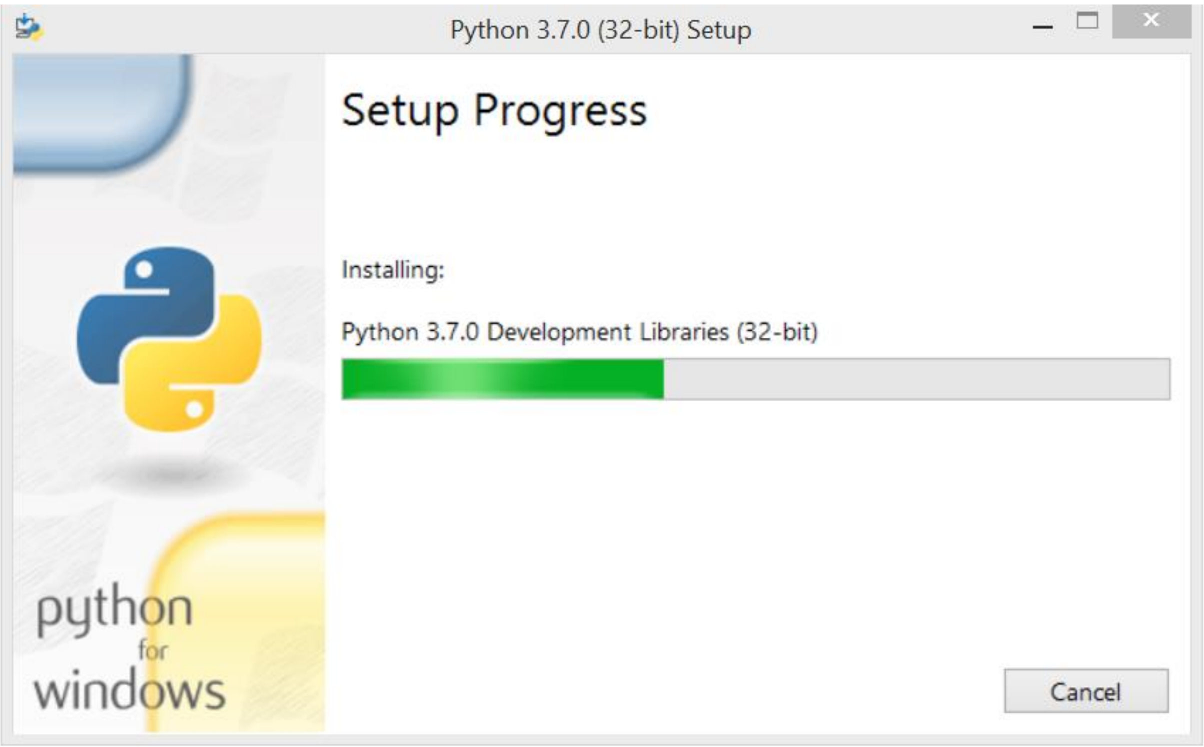
Python 3.7.0 (32-bit) Setup

Advanced Options

☑ Install for all users
☑ Associate files with Python (requires the py launcher)
☑ Create shortcuts for installed applications
□ Add Python to environment variables
☑ Precompile standard library
□ Download debugging symbols
□ Download debug binaries (requires VS 2015 or later)

Customize install location
C:\Program Files (x86)\Python37-32

Now, we are ready to install python-3.6.7. Let's install it.



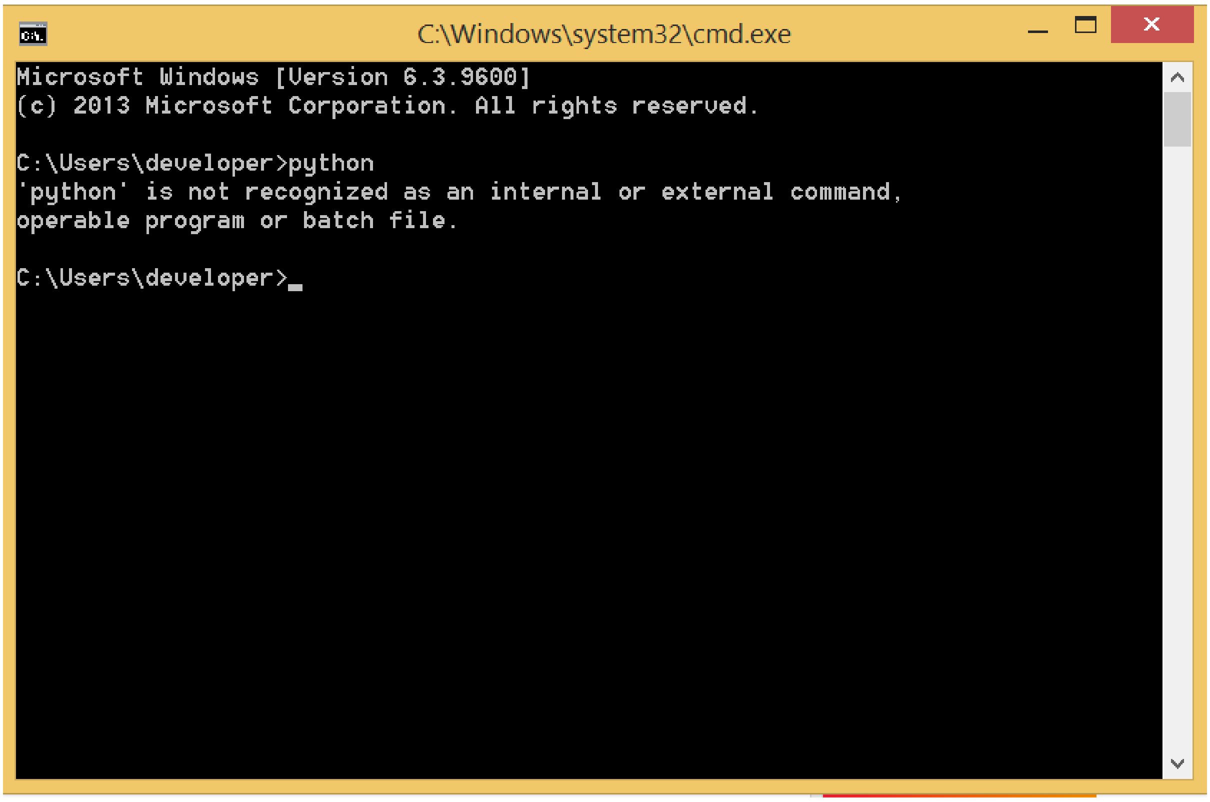
Python 3.7.0 (32-bit) Setup

Setup Progress

Installing:

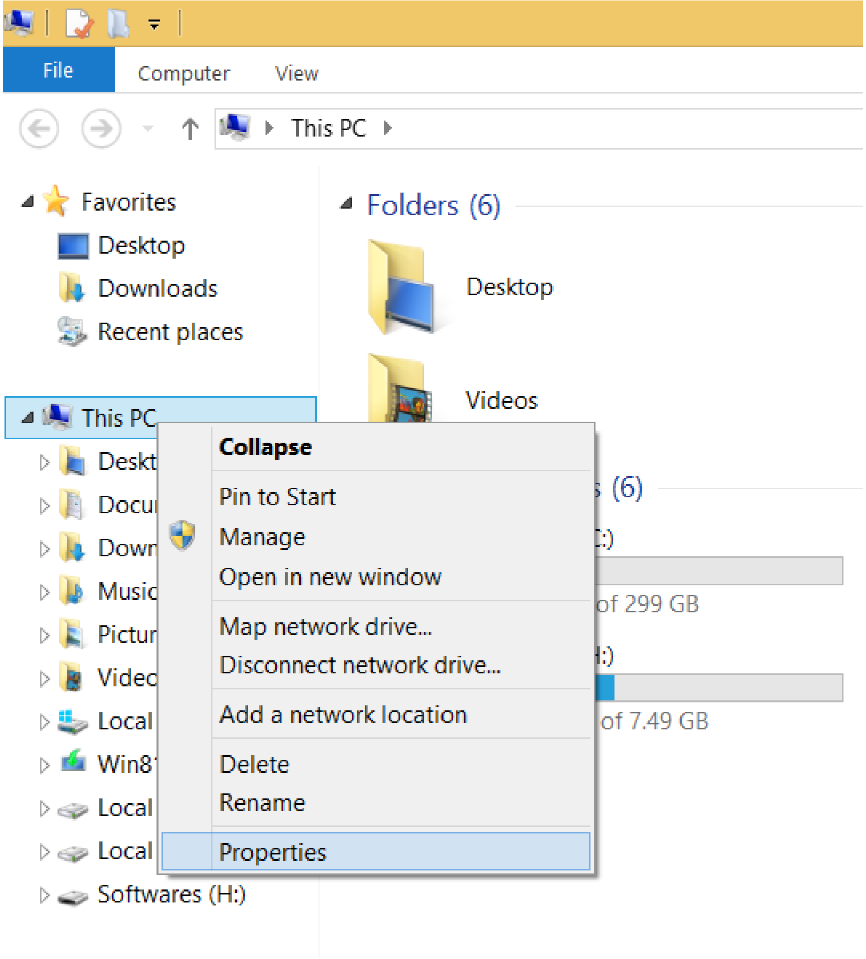
Python 3.7.0 Development Libraries (32-bit)

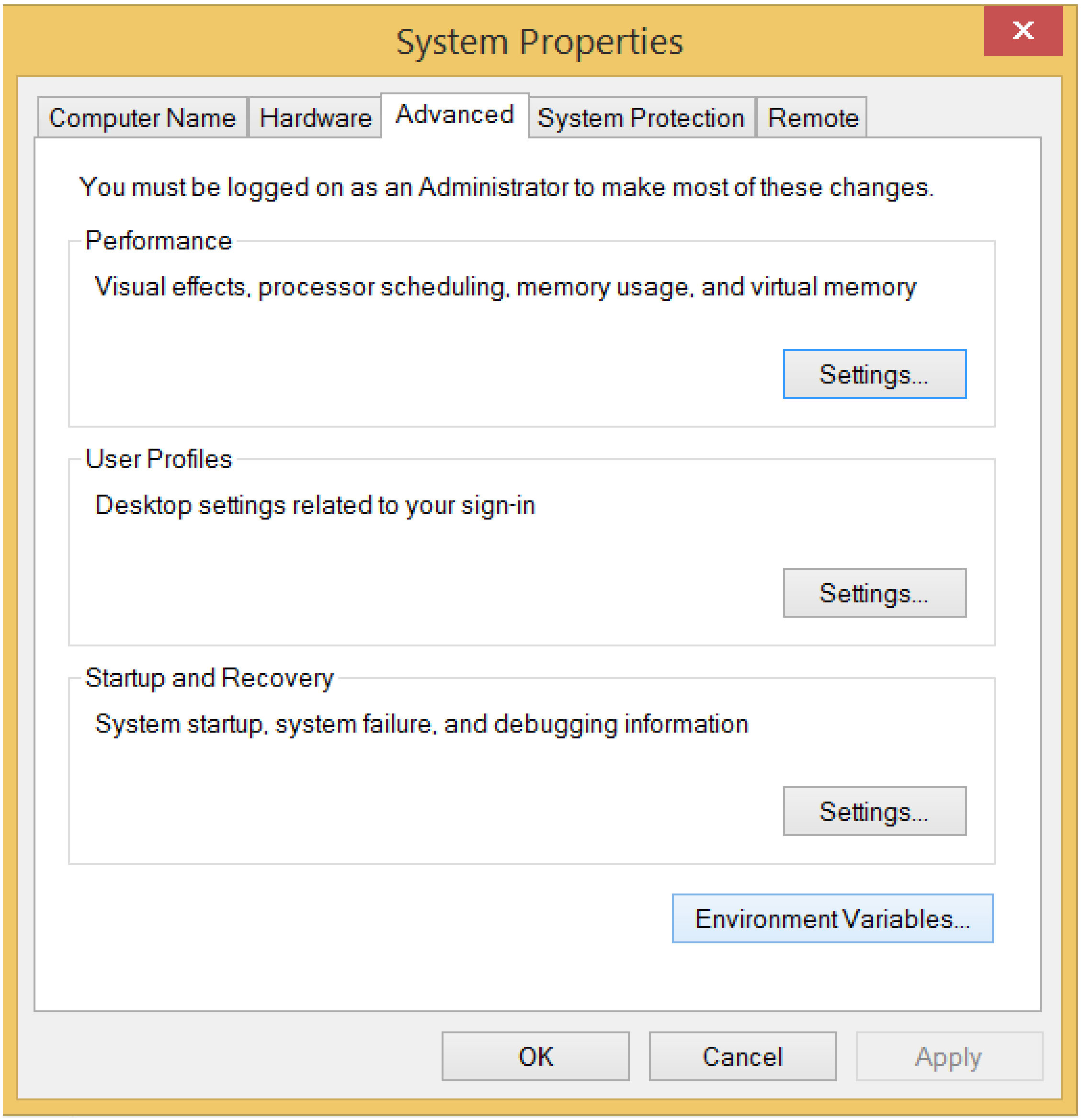
*Now, try to run python on the command prompt. Type the command **python** in case of python2 or python3 in case of **python3**. It will show an error as given in the below image. It is because we haven't set the path.*



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\developer>python
'python' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\developer>_
```

To set the path of python, we need to the right click on "my computer" and go to Properties → Advanced → Environment Variables.


File
Computer
View
This PC
Favorites
Desktop
Downloads
Recent places
This PC
Deskt
Docu
Down
Music
Pictur
Videc
Local
Win8
Local
Local
Softwares (H:)
Folders (6)
Desktop
Videos
s (6)
C:)
of 299 GB
H:)
of 7.49 GB
Collapse
Pin to Start
Manage
Open in new window
Map network drive...
Disconnect network drive...
Add a network location
Delete
Rename
Properties



System Properties

You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage, and virtual memory

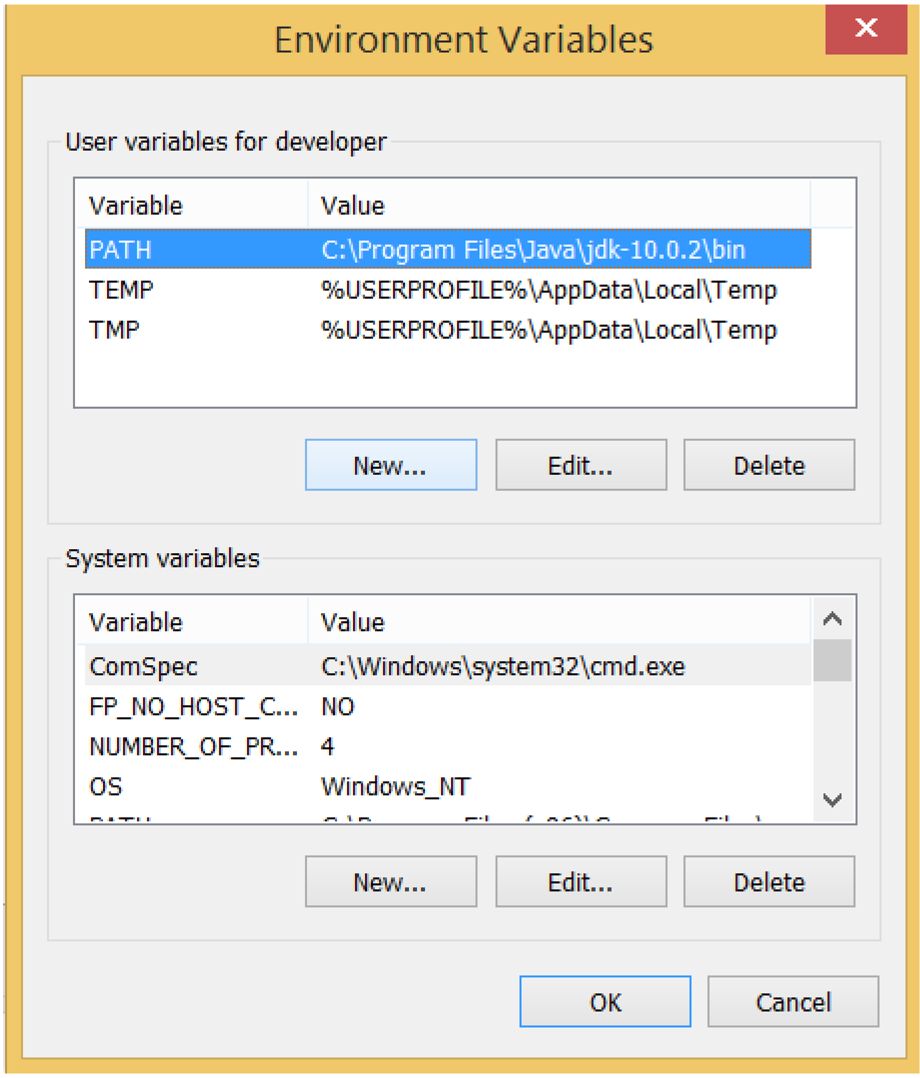
User Profiles

Desktop settings related to your sign-in

Startup and Recovery

System startup, system failure, and debugging information

Add the new path variable in the user variable section.



Environment Variables

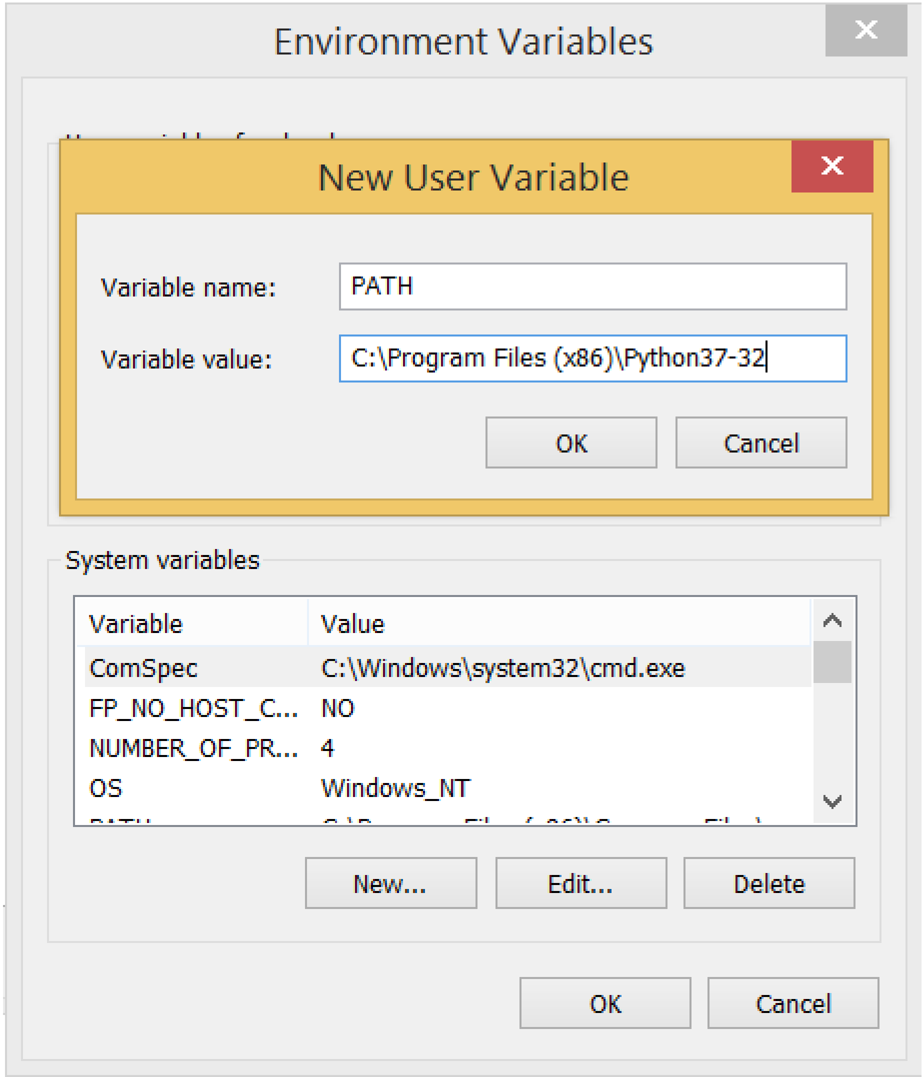
User variables for developer

Variable	Value
PATH	C:\Program Files\Java\jdk-10.0.2\bin
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

System variables

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
FP_NO_HOST_C...	NO
NUMBER_OF_PR...	4
OS	Windows_NT

*Type **PATH** as the variable name and set the path to the installation directory of the python shown in the below image.*



Variable name: PATH

Variable value: C:\Program Files (x86)\Python37-32

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
FP_NO_HOST_C...	NO
NUMBER_OF_PR...	4
OS	Windows_NT

Now, the path is set; we are ready to run python on our local system. Restart CMD, and type ***python again****. It will open the python interpreter shell where we can execute the python statements.*

Installation on Mac

To install python3 on MacOS, visit the link https://www.javatpoint.com/how-to-install-python-on-macand follow the instructions given in the tutorial.

Installation on Mac

To install python3 on MacOS, visit the link https://www.javatpoint.com/how-to-install-python-on-macand follow the instructions given in the tutorial.

Installation on CentOS

To install Python3 on CentOS, visit the link https://www.javatpoint.com/how-to-install-python-on-centosand follow the instructions given in the tutorial.

Installation on Ubuntu

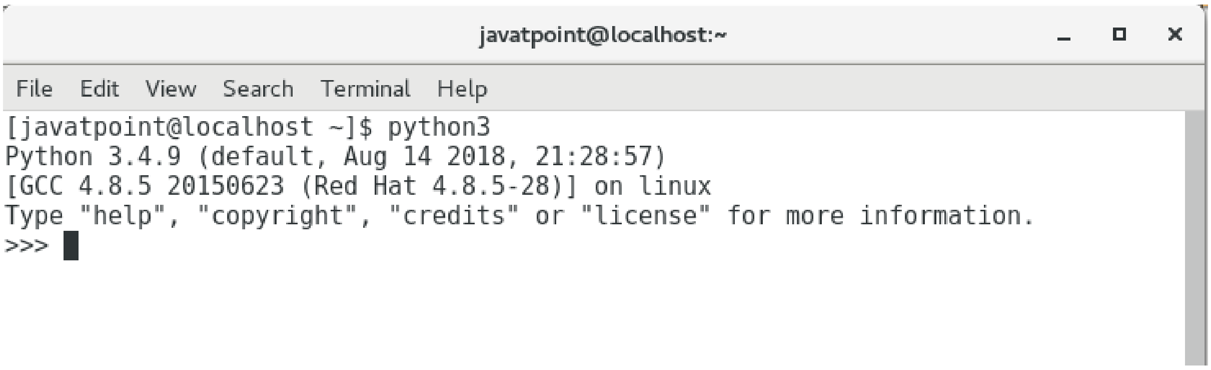
To install Python3 on Ubuntu, visit the link https://www.javatpoint.com/how-to-install-python-in-ubuntu and follow the instructions given in the tutorial.

First Python Program

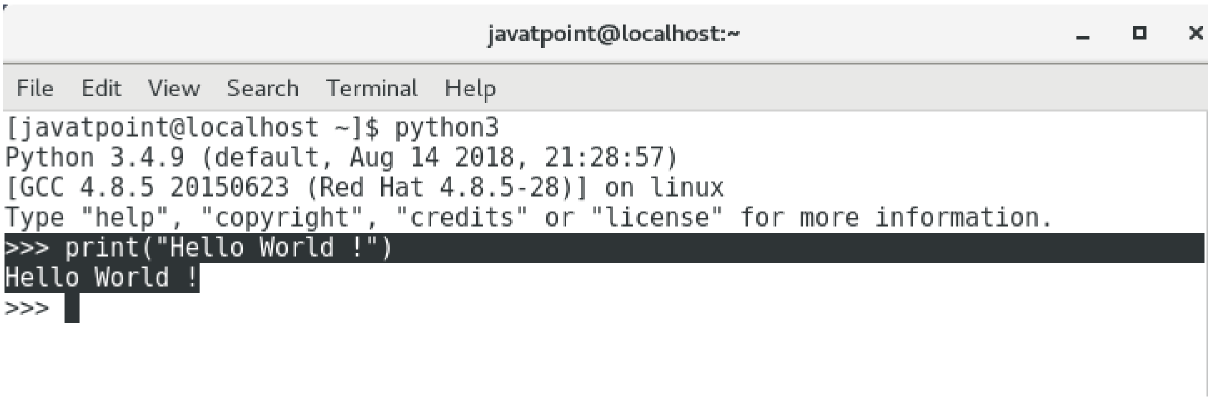
In this Section, we will discuss the basic syntax of Python, we will run a simple program to print ***Hello World*** *on the console.*

Python provides us the two ways to run a program:

- *Using Interactive interpreter prompt*
- *To open the interactive mode, open the terminal (or command prompt) and type python (python3 in case if you have Python2 and Python3 both installed on your system).*
- *It will open the following prompt where we can execute the Python statement and check their impact on the console.*



```
[javatpoint@localhost ~]$ python3
Python 3.4.9 (default, Aug 14 2018, 21:28:57)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```



```
[javatpoint@localhost ~]$ python3
Python 3.4.9 (default, Aug 14 2018, 21:28:57)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World !")
Hello World !
>>> 
```

- *Here, we get the message **"Hello World !"** printed on the console.*
- *Using a script file*

Open IDLE, Visual Studio Code or PyCharm, PYPY, Jupiter, Anaconda, Spider etc to run your program.

And Run the following command,

Print("Hello World")

By running this program we'll get output as   Hello World

Python IDLE

*Every Python installation comes with an **Integrated Development and Learning***

***Environment**, which you'll see shortened to IDLE or even IDE. These are a class of applications that help you write code more efficiently. While there are many IDEs for you to choose from, Python IDLE is very bare-bones, which makes it the perfect tool for a beginning programmer.*

Python IDLE comes included in Python installations on Windows and Mac. If you're a Linux user, then you should be able to find and download Python IDLE using your package manager. Once you've installed it, you can then use Python IDLE as an interactive interpreter or as a file editor.

Tkinter

Python provides several different options for writing GUI based programs. These are listed below:

- ***Tkinter**: It is easiest to start with. Tkinter is Python's standard GUI (graphical user interface) package. It is the most commonly used toolkit for GUI programming in Python.*
- ***JPython**: It is the Python platform for Java that is providing Python scripts seamless access o Java class Libraries for the local machine.*
- ***wxPython**: It is an open-source, cross-platform GUI toolkit written in C++. It is one of the alternatives to Tkinter, which is bundled with Python.*

There are many other interfaces available for GUI. But these are the most commonly used ones. In this, we will learn about the basic GUI programming using Tkinter

It is the standard GUI toolkit for Python. Fredrik Lundh wrote it. For modern Tk binding, Tkinter is implemented as a Python wrapper for the Tcl Interpreter embedded within the interpreter of Python. Tk provides the following widgets:

- *button*
- *canvas*
- *combo-box*
- *frame*
- *level*
- *check-button*
- *entry*
- *level-frame*

- menu
- list - box
- menu button
- message
- tk_optoinMenu
- progress-bar
- radio button
- scroll bar
- separator
- tree-view, and many more.

Creating a GUI program using this Tkinter is simple. For this, programmers need to follow the steps mentioned below:

1. Import the module Tkinter
2. Build a GUI application (as a window)
3. Add those widgets that are discussed above
4. Enter the primary, i.e., the main event's loop for taking action when the user triggered the event.

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

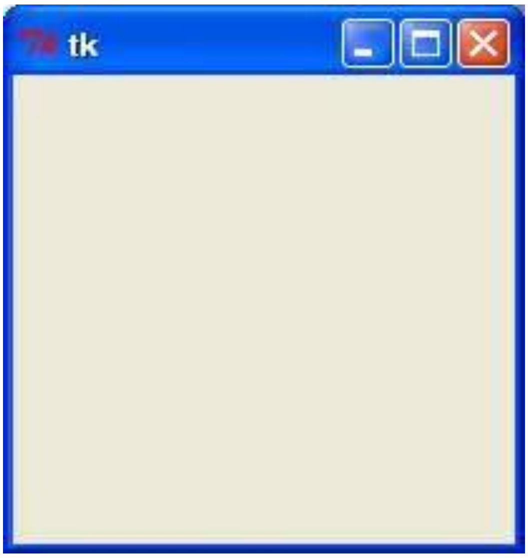
- Import the Tkinter module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Example

```
#!/usr/bin/python

import Tkinter
top = Tkinter.Tk()
```

```
# Code to add widgets will go here...
top.mainloop()
```

This would create a following window −



In this program, it is shown how Tkinter is used via Python to built windows along with some buttons and the events that are programmed using these buttons.

```
import tkinter as tk
from tkinter import *
from tkinter import ttk

class karl( Frame ):
   def __init__( self ):
      tk.Frame.__init__(self)
      self.pack()
      self.master.title("Karlos")
      self.button1 = Button( self, text = "CLICK HERE", width = 25,
                        command = self.new_window )
      self.button1.grid( row = 0, column = 1, columnspan = 2, sticky = W+E+N+S )
   def new_window(self):
      self.newWindow = karl2()
class karl2(Frame):
   def __init__(self):
      new =tk.Frame.__init__(self)
      new = Toplevel(self)
      new.title("karlos More Window")
      new.button = tk.Button(  text = "PRESS TO CLOSE", width = 25,
                         command = self.close_window )
      new.button.pack()
   def close_window(self):
      self.destroy()
def main():
   karl().mainloop()
```

```
if __name__ == '__main__':
   main()
```

- Dimensions
- Fonts
- Colors
- Cursors
- Anchors
- Bitmaps

- ***The pack()***: This method manages the geometry of widgets in blocks
- ***The grid()***: This method organizes widgets in a tabular structure
- ***The place()***: This method organizes the widgets to place them in a specific position

Visual Studio Code

Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging.

First and foremost, it is an editor that gets out of your way. The delightfully frictionless edit-build-debug cycle means less time fiddling with your environment, and more time executing on your ideas

Visual Studio Code supports macOS, Linux, and Windows - so you can hit the ground running, no matter the platform.

At its heart, Visual Studio Code features a lightning fast source code editor, perfect for day-to-day use. With support for hundreds of languages, VS Code helps you be instantly productive with syntax highlighting, bracket-matching, auto-indentation, box-selection, snippets, and more. Intuitive keyboard shortcuts, easy customization and community-contributed keyboard shortcut mappings let you navigate your code with ease.

Introduction to restaurant billing Software

A restaurant management software helps in operating and managing a restaurant efficiently. It is a software used by the restaurant to streamline the process of catering to the customers and conveniently improve the employee work rate and overall productivity of the restaurant. Aim of this software is to serve customers what they prefer, reduce the waiting time and maximize potential of the restaurant through its vast features.

Features/ benefits of a restaurant management software

A restaurant management software is only as good as the features it provides. The features should be diverse and must cater to every type and size of restaurants. These features help monitor all employees and reduce any wrongdoings. Some of the features are listed below:

- *A management software or a point-of-sale is the base to any restaurant's growth and enhances productivity.*
- *Inventory control*
- *Table management*
- *Enhanced staff timesheets*
- *Gift cards and loyalty programs for customers*
- *Easy and customizable menu setup*
- *Advanced reporting and data*
- *Round-the-clock technical assistance*
- *Integration availability for growth*
- *User-friendly order management*
- *Generate accurate financial statements*
- *Cost savings*
- *Reservation system*

- *Control to the software from any device and from anywhere*
- *Track sales of each item or product*
- *Shorter queues*
- *Workforce management*
- *Improved employee synergy*
- *Cloud storage*
- *Data security*

Restaurants of all types want to improve their profit and "make it big." By implementing a free and open source restaurant management software, the probability of growth will increase by letting the staff lay all their attention on catering to the customers. This software is crucial in terms of updating and selecting menus, placing orders, or receiving payments. It is a platform designed to reduce human efforts and maximize business potential.

Objective Restaurant Billing Software

The prime Objective of Restaurant Billing Software is to create a full fledge Android and Windows based application. Which could locate & type of the cuisine entered by the user. The user not only finds all the restaurants in the city, but also one can make choice of the best restaurant based on the rating & cuisine one chooses to have.

Modules

- *Admin :*

  a) *This module can take care all the operations such as manage the menu of items in the restaurant. View the orders and manages the employees and also manage the financial transaction that can be involved in the restaurant maintenance. Maintain the quantity details of the items.*

  b) *Some of the operations can be included as future scope of the project that can be maintained in the transaction those can be entitled in the transactions.*

- *User :*

a) *User can responsible to check the menu item. In the menu item they can choose items can be ordered. So, that ordering of an item can be finished so that they can maintain the complete order information at their own end.*

Restaurant Billing Software will fetch the user friendliness to make their transactions. It consumes their resources also i.e. effective utilization of time by both restaurant maintainer and also user who will avail this facility.

The main objectives of the software are:-

- *Availability : The all transaction of restaurant stored permanently in the database admin can see the data in the availability of any information.*
- *Maintain Cost : Reduce the cost of maintenance. It is stand-able application so no required of cost for maintain it.*

- *Flexibility : The software modifiable dependable on changing needs of the users. It is portable to different computer systems.*

Workflow

Work Flow in the restaurant will be done in the following way : -

- *The data entry operator will login to the software.*
- *Then waiter will come to the customer, takes his order.*
- *Data entry operator will enter the order of the customer.*
- *Then, the software automatically calculates the bill and generate the bill of the entered order details.*
- *The waiter will takes the order to the customer.*
- *Waiter will give him, his bill.*

Customer will pay for his orders.

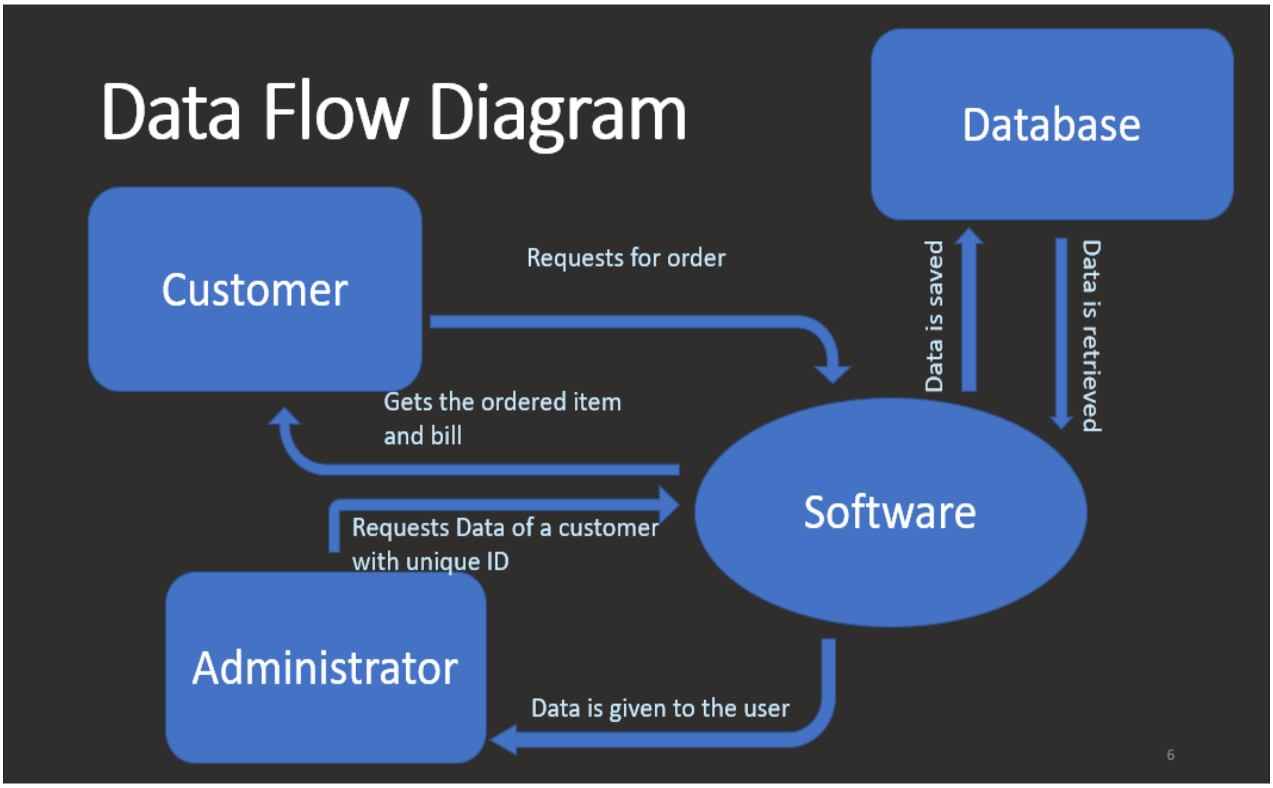
Data Flow Diagram

Need for software based application

A restaurant billing software is a solution to make your billing process faster and easier. It delivers seamless connectivity and flexibility to keep track of the diverse functionalities of your restaurant while boosting your productivity. Whether it is kitchen display system or petty cash accounting, home delivery or dine-in management, it gives you an insight into day-to-day financial tasks – to give the desired results in a customer-oriented and forward-thinking manner.

Some of the main need for a software based application are:-

- *Simple Menu Setup.*



Data Flow Diagram

Database

Customer

Requests for order

Data is saved

Data is retrieved

Gets the ordered item and bill

Software

Requests Data of a customer with unique ID

Administrator

Data is given to the user

6

- *User friendly order managements.*
- *Customer Management.*

- *Better customer service.*
- *Support auto GST calculation feature.*

Code Of the software

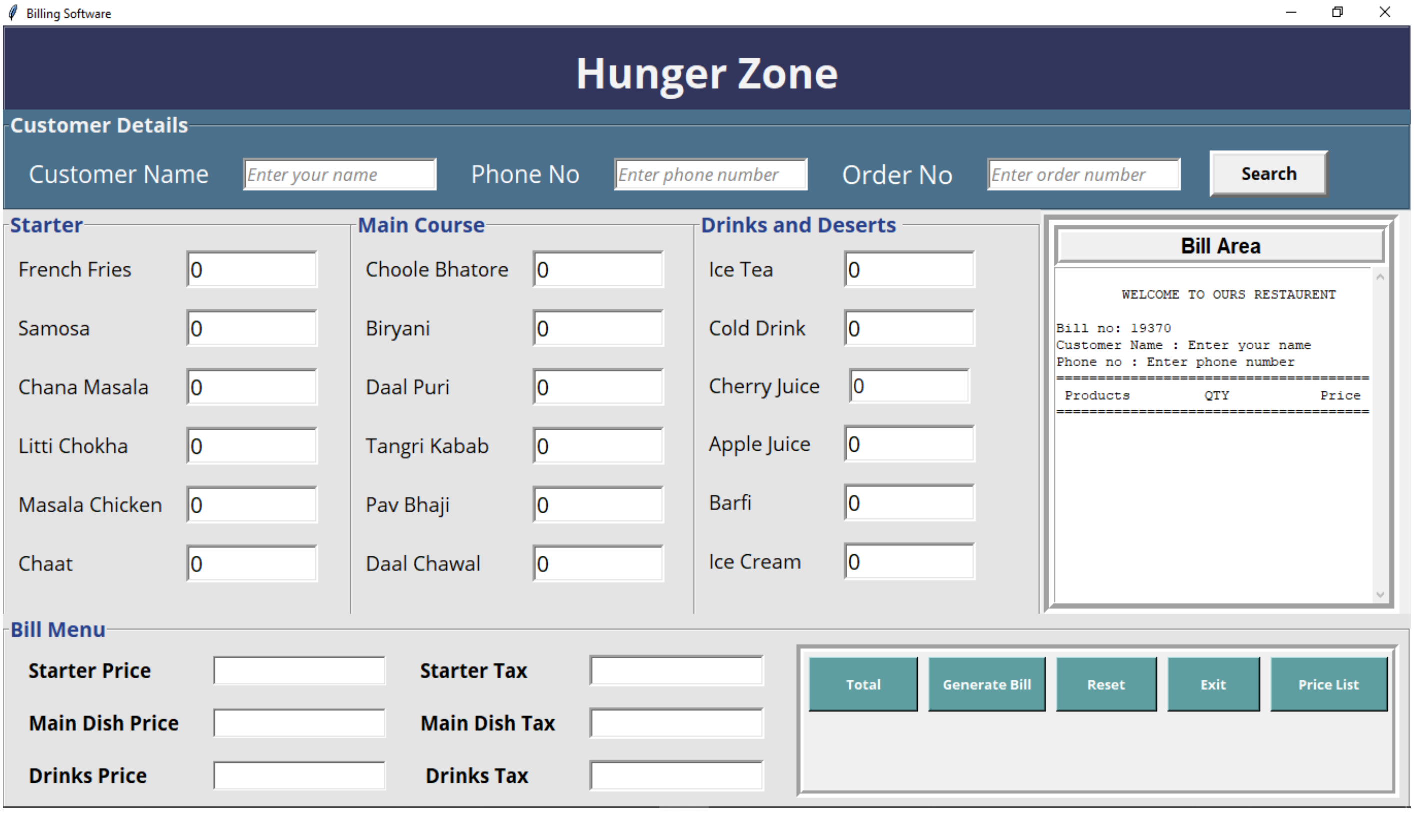
- *Main Window / Invoice :*

- *In this area, we place customer’s order requests.*
- *Firstly, customers details are filled and then order is placed.*
- *After placing order total prices and taxes on each item is calculated.*
- *Finally, we generate bill or invoice of orders placed.*

- *In main, window we have provided the ‘Price List’ to check price of each item.*

Code :-

```
from tkinter import *

import random

import math


import os

from tkinter import messagebox

class Bill:

    def __init__(self, root):
```



Customer Name	Enter your name	Phone No	Enter phone number	Order No	Enter order number

French Fries	0	Choole Bhatore	0	Ice Tea	0
Samosa	0	Biryani	0	Cold Drink	0
Chana Masala	0	Daal Puri	0	Cherry Juice	0
Litti Chokha	0	Tangri Kabab	0	Apple Juice	0
Masala Chicken	0	Pav Bhaji	0	Barfi	0
Chaat	0	Daal Chawal	0	Ice Cream	0

```
        WELCOME TO OURS RESTAURENT

Bill no: 19370
Customer Name : Enter your name
Phone no : Enter phone number
======================================
 Products          QTY            Price
======================================
```

Starter Price		**Starter Tax**	
Main Dish Price		**Main Dish Tax**	
Drinks Price		**Drinks Tax**	

```
        #pass

        self.root=root

        self.root.geometry("1200x1024+0+0")

        self.root.title("Billing Software")
```

```
title= Label(self.root, text= "Billing Software", bd=12, relief=GROOVE, bg="cadet blue", fg= "white", font=("ariel", 30, "bold"),pady=2).pack(fill=X)

##### creating customer frame    #####


F1 = LabelFrame(self.root,bd=10, relief= GROOVE, text = "Customer Details", font=("times new roman", 15, "bold"), fg="coral", bg="royalblue2")

F1.place(x=0,y=80,relwidth=1)

#####    creating customers items / products

F2 = LabelFrame(self.root,bd=10, relief= GROOVE, text = "Cosmetics Frame", font=("times new roman", 15, "bold"), fg="coral", bg="royalblue2")

F2.place(x=5,y=180,width=325,height=380)


#####   creating frame for grocerry product    #######


F3 = LabelFrame(self.root,bd=10, relief= GROOVE, text = "Cosmetics Frame", font=("times new roman", 15, "bold"), fg="coral", bg="royalblue2")

F3.place(x=340,y=180,width=325,height=380)



#####   creating frame for Cold Drinks   #######


F4 = LabelFrame(self.root,bd=10, relief= GROOVE, text = "Cold Drinks", font=("times new roman", 15, "bold"), fg="coral", bg="royalblue2")

F4.place(x=670,y=180,width=325,height=380)
```

```
#####   Bill Area   #######


F5 = Frame(self.root,bd=10, relief= GROOVE)

F5.place(x=1010,y=180,width=345,height=380)



#########~~~~~~~~~~~~~~    Varialbles  ~~~~~~~~~~~~~~#########

###########          Cosmetics              #####

self.soap=IntVar()

self.face_cream=IntVar()

self.face_wash=IntVar()

self.spray=IntVar()

self.gell=IntVar()

self.loshan=IntVar()


##########    Grocery vars ############



self.rice = IntVar()

self.food_oil = IntVar()

self.daal = IntVar()

self.wheat = IntVar()

self.sugar = IntVar()

self.tea = IntVar()


##########     Cold Drinks    #############
```

```
self.mazaa = IntVar()
self.cock = IntVar()
self.frooti = IntVar()
self.thumbs_up = IntVar()
self.limca = IntVar()
self.sprite = IntVar()

#########      Total Product Price & Tax Variable         ################

self.cosmetics_price  = StringVar()
self.grocery_price = StringVar()
self.cold_drink_price = StringVar()



self.cosmetics_pric_tax = StringVar()
self.grocery_tax = StringVar()
self.cold_drink_tax = StringVar()

###########    Customer    #########

self.c_name = StringVar()
self.c_phon = StringVar()
self.bill_no = StringVar()
x=random.randint(1000,99999)
self.bill_no.set(str(x))
```

```
        self.search_bill = StringVar()


        # customer label

        c_name_lbl= Label(F1, text="Customer Name",bg="royalblue2", fg="white", font=("times new roman", 19,"bold")).grid(row=0, column=0,padx=20,pady=5)
        c_name_txt = Entry(F1, width=20,textvariable = self.c_name, font="arial 15",bd=7, relief=SUNKEN).grid(row=0,column=1, padx=10, pady=5)

        # for phone no

        c_phone_lbl= Label(F1, text="Phone No",bg="royalblue2", fg="white", font=("times new roman", 19,"bold")).grid(row=0, column=2,padx=20,pady=5)
        c_phone_txt = Entry(F1, width=20,textvariable = self.c_phon, font="arial 15",bd=7, relief=SUNKEN).grid(row=0,column=3, padx=10, pady=5)

        # for order no

        c_order_no_lbl= Label(F1, text="Order No",bg="royalblue2", fg="white", font=("times new roman", 19,"bold")).grid(row=0, column=4,padx=20,pady=5)
        c_order_no_txt = Entry(F1, width=20, textvariable = self.search_bill, font="arial 15",bd=7, relief=SUNKEN).grid(row=0,column=5, padx=10, pady=10)

        bill_btn= Button(F1, text="GO",command=self.find_bill, width=4, bd= 7, font= "arial 12 bold").grid(row=0,column=6, padx= 10, pady=10)
```

```
   #######  Cosmetics Frame   ########



  #  F2 = LabelFrame(self.root,bd=10, relief= GROOVE, text = "Cosmetics
Frame", font=("times new roman", 15, "bold"), fg="coral", bg="royalblue2")

  #   F2.place(x=0,y=80,width=1, height=)


        ##########   Cosmetics Items    ####################


        bath_lbl= Label(F2, text="Bath Soap", font=("times new
roman",16,"bold"),bg= "royal blue2", fg="lightgreen").grid(row=0, column=0,
padx=10, pady=10, sticky="w")

        bath_txt= Entry(F2, width=10, textvariable = self.soap, font=("times new
roman",16,"bold"),bd=5, relief=SUNKEN).grid(row=0, column=1 , padx=10,
pady=10)



  face_cream_lbl = Label(F2, text="Face Cream", font=("times new
roman",16,"bold"),bg= "royal blue2", fg="lightgreen").grid(row=1, column=0,
padx=10, pady=10, sticky="w")

        face_cream_txt= Entry(F2, width=10, textvariable = self.face_cream,
font=("times new roman",16,"bold"),bd=5, relief=SUNKEN).grid(row=1,
column=1 , padx=10, pady=10)
```

```
Face_wah_lbl= Label(F2, text="Face Wash", font=("times new roman",16,"bold"),bg= "royal blue2", fg="lightgreen").grid(row=2, column=0, padx=10, pady=10, sticky="w")

Face_wah_txt= Entry(F2, width=10, textvariable = self.face_wash ,font=("times new roman",16,"bold"),bd=5, relief=SUNKEN).grid(row=2, column=1 , padx=10, pady=10)
```

```
hair_spray_lbl= Label(F2, text="Hair Spary", font=("times new roman",16,"bold"),bg= "royal blue2", fg="lightgreen").grid(row=3, column=0, padx=10, pady=10, sticky="w")

hair_spray_txt= Entry(F2, width=10, textvariable = self.spray, font=("times new roman",16,"bold"),bd=5, relief=SUNKEN).grid(row=3, column=1 , padx=10, pady=10)
```

```
Hair_Gel_lbl= Label(F2, text="Hair Gel", font=("times new roman",16,"bold"),bg= "royal blue2", fg="lightgreen").grid(row=4, column=0, padx=10, pady=10, sticky="w")

Hair_gel_txt= Entry(F2, width=10,textvariable = self.gell, font=("times new roman",16,"bold"),bd=5, relief=SUNKEN).grid(row=4, column=1 , padx=10, pady=10)
```

```
body_lotion_lbl= Label(F2, text="Body Lotion", font=("times new roman",16,"bold"),bg= "royal blue2", fg="lightgreen").grid(row=5, column=0, padx=10, pady=10, sticky="w")

body_lotion_txt= Entry(F2, width=10, textvariable = self.loshan, font=("times new roman",16,"bold"),bd=5, relief=SUNKEN).grid(row=5, column=1 , padx=10, pady=10)
```

```
##########   Grocery  items    ####################

        rice_lbl= Label(F3, text="Rice", font=("times new roman",16,"bold"),bg= "royal blue2", fg="lightgreen").grid(row=0, column=0, padx=10, pady=10, sticky="w")

        rice_txt= Entry(F3, width=10, textvariable = self.rice, font=("times new roman",16,"bold"),bd=5, relief=SUNKEN).grid(row=0, column=1 , padx=10, pady=10)


        food_oil_lbl = Label(F3, text="Food Oil", font=("times new roman",16,"bold"),bg= "royal blue2", fg="lightgreen").grid(row=1, column=0, padx=10, pady=10, sticky="w")

        food_oil_txt= Entry(F3, width=10, textvariable = self.food_oil, font=("times new roman",16,"bold"),bd=5, relief=SUNKEN).grid(row=1, column=1 , padx=10, pady=10)


        Daal_lbl= Label(F3, text="Daal", font=("times new roman",16,"bold"),bg= "royal blue2", fg="lightgreen").grid(row=2, column=0, padx=10, pady=10, sticky="w")


 Daal_txt= Entry(F3, width=10,  textvariable = self.daal, font=("times new roman",16,"bold"),bd=5, relief=SUNKEN).grid(row=2, column=1 , padx=10, pady=10)


        Wheat_lbl= Label(F3, text="Wheat floor", font=("times new roman",16,"bold"),bg= "royal blue2", fg="lightgreen").grid(row=3, column=0, padx=10, pady=10, sticky="w")
        Wheat_spray_txt= Entry(F3, width=10,  textvariable = self.wheat, font=("times new roman",16,"bold"),bd=5, relief=SUNKEN).grid(row=3, column=1 , padx=10, pady=10)



        sugar_lbl= Label(F3, text="Sugar", font=("times new roman",16,"bold"),bg= "royal blue2", fg="lightgreen").grid(row=4, column=0, padx=10, pady=10, sticky="w")

        sugar_txt= Entry(F3, width=10, textvariable = self.sugar, font=("times new roman",16,"bold"),bd=5, relief=SUNKEN).grid(row=4, column=1 , padx=10, pady=10)



        Tea_lbl= Label(F3, text="Tea", font=("times new roman",16,"bold"),bg= "royal blue2", fg="lightgreen").grid(row=5, column=0, padx=10, pady=10, sticky="w")

        Tea_txt= Entry(F3, width=10,  textvariable = self.tea, font=("times new roman",16,"bold"),bd=5, relief=SUNKEN).grid(row=5, column=1 , padx=10, pady=10)




##########  Cold Drinks    ####################

              maza_lbl= Label(F4, text="Mazaa", font=("times new roman",16,"bold"),bg= "royal blue2", fg="lightgreen").grid(row=0, column=0, padx=10, pady=10, sticky="w")

        maza_txt= Entry(F4, width=10,  textvariable = self.mazaa, font=("times new roman",16,"bold"),bd=5, relief=SUNKEN).grid(row=0, column=1 , padx=10, pady=10)
        cock_lbl = Label(F4, text="Cock", font=("times new roman",16,"bold"),bg= "royal blue2", fg="lightgreen").grid(row=1, column=0, padx=10, pady=10, sticky="w")

        cock_txt= Entry(F4, width=10,  textvariable = self.cock, font=("times new roman",16,"bold"),bd=5, relief=SUNKEN).grid(row=1, column=1 , padx=10, pady=10)



        frooti_lbl= Label(F4, text="Frooti", font=("times new roman",16,"bold"),bg= "royal blue2", fg="lightgreen").grid(row=2, column=0, padx=10, pady=10, sticky="w")

        frooti_txt= Entry(F4, width=10, textvariable = self.frooti, font=("times new roman",16,"bold"),bd=5, relief=SUNKEN).grid(row=2, column=1 , padx=10, pady=10)



        thumbs_up_lbl= Label(F4, text="Thumbs Up", font=("times new roman",16,"bold"),bg= "royal blue2", fg="lightgreen").grid(row=3, column=0, padx=10, pady=10, sticky="w")

        thumbs_up_txt= Entry(F4, width=10,  textvariable = self.thumbs_up, font=("times new roman",16,"bold"),bd=5, relief=SUNKEN).grid(row=3, column=1 , padx=10, pady=10)



        limca_lbl= Label(F4, text="Limca", font=("times new roman",16,"bold"),bg= "royal blue2", fg="lightgreen").grid(row=4, column=0, padx=10, pady=10, sticky="w")

        limca_txt= Entry(F4, width=10,  textvariable = self.limca, font=("times new roman",16,"bold"),bd=5, relief=SUNKEN).grid(row=4, column=1 , padx=10, pady=10)
        sprite_lbl= Label(F4, text="Sprite", font=("times new roman",16,"bold"),bg= "royal blue2", fg="lightgreen").grid(row=5, column=0, padx=10, pady=10, sticky="w")

        sprite_txt= Entry(F4, width=10, textvariable = self.sprite, font=("times new roman",16,"bold"),bd=5, relief=SUNKEN).grid(row=5, column=1 , padx=10, pady=10)



       #############     Bill Area      #############


        bill_title = Label(F5, text="Bill Area", font= "arial 15 bold", bd="7", relief= GROOVE).pack(fill=X)

        scrlbr = Scrollbar(F5, orient=VERTICAL)

        self.txtarea = Text(F5, yscrollcommand=scrlbr.set)

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

        scrlbr.config(command=self.txtarea.yview)

        self.txtarea.pack(fill=BOTH, expand=1)



###################   Button Area   ###########################


        F6 = LabelFrame(self.root,bd=10, relief= GROOVE, text = "Bill Menu", font=("times new roman", 15, "bold"), fg="coral", bg="royalblue2")

        F6.place(x=0,y=560,relwidth=1, height=185)
        m1_lbl = Label(F6, text="Total Cosmetic Price", bg="RoyalBlue",
fg="white", font=("times new roman",14,"bold")).grid(row=0, column=0,
padx=20, pady=1, sticky="w")

        m1_txt = Entry(F6, width=18, textvariable = self.cosmetics_price,
font="arial 10 bold", bd=7, relief=SUNKEN).grid(row=0, column=1, padx=10,
pady=10)


        m2_lbl = Label(F6, text="Total Grocery Price", bg="RoyalBlue", fg="white",
font=("times new roman",14,"bold")).grid(row=1, column=0, padx=20, pady=1,
sticky="w")

        m2_txt = Entry(F6, width=18, textvariable = self.grocery_price, font="arial
10 bold", bd=7, relief=SUNKEN).grid(row=1, column=1, padx=10, pady=10)


        m3_lbl = Label(F6, text="Total Cold Drinks Price", bg="RoyalBlue",
fg="white", font=("times new roman",14,"bold")).grid(row=2, column=0,
padx=20, pady=1, sticky="w")

        m3_txt = Entry(F6, width=18, textvariable = self.cold_drink_price,
font="arial 10 bold", bd=7, relief=SUNKEN).grid(row=2, column=1, padx=10,
pady=10)




 c1_lbl = Label(F6, text="Cosmetic Tax", bg="RoyalBlue", fg="white",
font=("times new roman",14,"bold")).grid(row=0, column=2, padx=20, pady=1,
sticky="w")

        c1_txt = Entry(F6, width=18, textvariable = self.cosmetics_pric_tax,
font="arial 10 bold", bd=7, relief=SUNKEN).grid(row=0, column=3, padx=10,
pady=10)
```

```
c2_lbl = Label(F6, text="Grocery Tax", bg="RoyalBlue", fg="white", font=("times new roman",14,"bold")).grid(row=1, column=2, padx=20, pady=1, sticky="w")

c2_txt = Entry(F6, width=18, textvariable = self.grocery_tax, font="arial 10 bold", bd=7, relief=SUNKEN).grid(row=1, column=3, padx=10, pady=10)


c3_lbl = Label(F6, text="Cold Drinks Tax", bg="RoyalBlue", fg="white", font=("times new roman",14,"bold")).grid(row=2, column=2, padx=20, pady=1, sticky="w")

c3_txt = Entry(F6, width=18, textvariable = self.cold_drink_tax, font="arial 10 bold", bd=7, relief=SUNKEN).grid(row=2, column=3, padx=10, pady=10)



#################    Button #########################


btn = Frame(F6, bd=7, relief=GROOVE)

btn.place(x=750, width=580, height=145)



total_btn = Button(btn, text="Total", command=self.total, bg="cadetblue", fg="white", pady=11, width=10,bd=2, font="arial 15 bold").grid(row=0, column=0, padx=5, pady=5)


B_bill_btn = Button(btn,command=self.bill_area, text="Generate Bill", bg="cadetblue", fg="white", pady=11, width=10,bd=2, font="arial 15 bold").grid(row=0, column=1, padx=5, pady=5)

Clear_btn = Button(btn,command=self.clear_data, text="Reset", bg="cadetblue", fg="white", pady=11, width=10,bd=2, font="arial 15 bold").grid(row=0, column=2, padx=5, pady=5)
```

```
        Exit_btn = Button(btn, command=self.Exit, text="Exit", bg="cadetblue",
fg="white", pady=11, width=10,bd=2, font="arial 15 bold").grid(row=0,
column=3, padx=5, pady=5)

        self.welcome_bill()

    def total(self):

        self.c_s_p=self.soap.get()*40

        self.c_c_p=self.face_cream.get()*120

        self.c_f_p=self.face_wash.get()*60

        self.c_sp_p=self.spray.get()*180

        self.c_gl_p=self.gell.get()*140

        self.c_l_p=self.loshan.get()*180


        self.total_cosmetic_price = float(

                            self.c_s_p+

                            self.c_c_p+

                            self.c_f_p+

                            self.c_sp_p+

                            self.c_gl_p+

                            self.c_l_p

                            )



       self.cosmetics_price.set("Rs. "+str(self.total_cosmetic_price ))

        self.c_tax=round(self.total_cosmetic_price*0.05,3)

        self.cosmetics_pric_tax.set("Rs. "+str(self.c_tax))


        self.g_r_p=self.rice.get()*40
self.g_foil_p=self.food_oil.get()*120
self.g_dl_p=self.daal.get()*60
self.g_wht_p=self.wheat.get()*180
self.g_sg_p=self.sugar.get()*140
self.g_t_p=self.tea.get()*180

self.total_grocery_price = float(
            self.g_r_p+
            self.g_foil_p+
            self.g_dl_p+
            self.g_wht_p+
            self.g_sg_p+
            self.g_t_p
            )
self.grocery_price.set("Rs. "+str(self.total_grocery_price ))
self.g_tax=round(self.total_grocery_price*0.05,3)
self.grocery_tax.set("Rs. "+str(self.g_tax))

self.cd_mz_p=self.mazaa.get()*60
self.cd_ck_p=self.cock.get()*60
self.cd_frt_p=self.frooti.get()*50
self.cd_th_p=self.thumbs_up.get()*45
self.cd_lim_p=self.limca.get()*40
self.cd_sp_p=self.sprite.get()*60

self.total_drinks_price = float(
                        self.cd_mz_p+
                        self.cd_ck_p+
                        self.cd_frt_p+
                        self.cd_th_p+
                        self.cd_lim_p+
                        self.cd_sp_p
                        )
        self.cold_drink_price.set("Rs. "+str(self.total_drinks_price ))
        self.cd_tax=round(self.total_drinks_price*0.05,3)
        self.cold_drink_tax.set("Rs. "+str(self.cd_tax))

        self.total_bill = float(self.total_cosmetic_price + self.total_drinks_price +
self.total_grocery_price + self.c_tax + self.g_tax + self.cd_tax)



    def Exit(self):
        op=messagebox.askyesno("Exit","Do you really want to exit?")
        if op>0:                         #====================== exit fn
=============================#
            self.root.destroy()

    def welcome_bill(self):
        self.txtarea.delete("1.0",END)
        self.txtarea.insert(END,"\n \tWELCOME TO OURS RESTAURENT\t \n")
        self.txtarea.insert(END,f"\nBill no: {self.bill_no.get()}")
        self.txtarea.insert(END,f"\nCustomer Name : {self.c_name.get()}")
        self.txtarea.insert(END,f"\nPhone no : {self.c_phon.get()}")

self.txtarea.insert(END,f"\n=========================================")
        self.txtarea.insert(END,f"\n Products \t\tQTY \t\tPrice")

self.txtarea.insert(END,f"\n=========================================")



    def bill_area(self):
        self.welcome_bill()
        if self.c_name.get() == "" or self.c_phon.get() == "":
            messagebox.showerror("Error", "Customer details are must")
        elif self.cosmetics_price == "Rs. 0.0" or self.grocery_price == "Rs. 0.0" or
self.cold_drink_price == "Rs. 0.0":
            messagebox.showerror("Error", "No product purchased")
        else:
            if self.soap.get()!=0:
                self.txtarea.insert(END,f"\n Bath
Soap\t\t{self.soap.get()}\t\t{self.c_s_p}")
            if self.face_cream.get()!=0:
                self.txtarea.insert(END,f"\n Face
Cream\t\t{self.face_cream.get()}\t\t{self.c_c_p}")

if self.face_wash.get()!=0:
                self.txtarea.insert(END,f"\n Face
Wash\t\t{self.face_wash.get()}\t\t{self.c_f_p}")
            if self.spray.get()!=0:
```

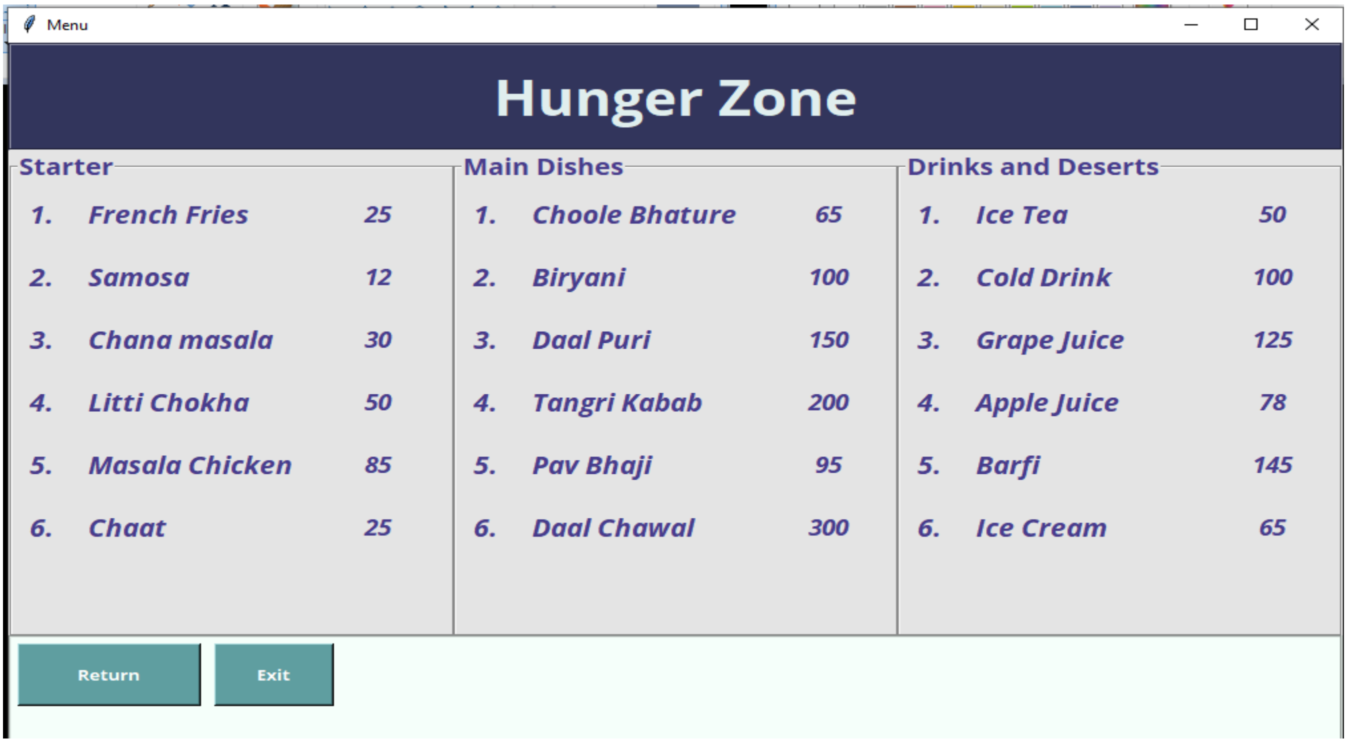
```
            self.txtarea.insert(END,f"\n
Spray\t\t{self.spray.get()}\t\t{self.c_sp_p}")

         if self.gell.get()!=0:

            self.txtarea.insert(END,f"\n Hair Gell
\t\t{self.gell.get()}\t\t{self.c_gl_p}")

         if self.loshan.get()!=0:

            self.txtarea.insert(END,f"\n Body
Loshan\t\t{self.loshan.get()}\t\t{self.c_l_p}")


         if self.rice.get()!=0:

            self.txtarea.insert(END,f"\n Rice\t\t{self.rice.get()}\t\t{self.g_r_p}")

         if self.food_oil.get()!=0:

            self.txtarea.insert(END,f"\n Food
Oil\t\t{self.food_oil.get()}\t\t{self.g_foil_p}")

         if self.daal.get()!=0:

            self.txtarea.insert(END,f"\n Daal\t\t{self.daal.get()}\t\t{self.g_dl_p}")

         if self.wheat.get()!=0:

            self.txtarea.insert(END,f"\n
Wheat\t\t{self.wheat.get()}\t\t{self.g_wht_p}")

         if self.sugar.get()!=0:

            self.txtarea.insert(END,f"\n Sugar
\t\t{self.sugar.get()}\t\t{self.g_sg_p}")




         if self.tea.get()!=0:

            self.txtarea.insert(END,f"\n Tea\t\t{self.tea.get()}\t\t{self.g_t_p}")


         if self.mazaa.get()!=0:
            self.txtarea.insert(END,f"\n
Mazaa\t\t{self.mazaa.get()}\t\t{self.cd_mz_p}")
        if self.cock.get()!=0:
            self.txtarea.insert(END,f"\n
Cock\t\t{self.cock.get()}\t\t{self.cd_ck_p}")
        if self.frooti.get()!=0:
            self.txtarea.insert(END,f"\n
Frooti\t\t{self.frooti.get()}\t\t{self.cd_frt_p}")
        if self.thumbs_up.get()!=0:
            self.txtarea.insert(END,f"\n Thumbs
Up\t\t{self.thumbs_up.get()}\t\t{self.cd_th_p}")
        if self.limca.get()!=0:
            self.txtarea.insert(END,f"\n Limca
\t\t{self.limca.get()}\t\t{self.cd_lim_p}")
        if self.sprite.get()!=0:
            self.txtarea.insert(END,f"\n
Sprite\t\t{self.sprite.get()}\t\t{self.cd_sp_p}")



        self.txtarea.insert(END,f"\n--------------------------------------")
        if self.cosmetics_pric_tax.get() != "Rs. 0.0":
            self.txtarea.insert(END,f"\n Cosmetic
Tax\t\t\t{self.cosmetics_pric_tax.get()}")

        if self.grocery_tax.get() != "Rs. 0.0":
            self.txtarea.insert(END,f"\n Grocery Tax\t\t\t{self.grocery_tax.get()}")

        if self.cold_drink_tax.get() != "Rs. 0.0":
            self.txtarea.insert(END,f"\n Cold Drinks
Tax\t\t\t{self.cold_drink_tax.get()}")


        self.txtarea.insert(END,f"\n Total Bill : \t\t\tRs. {self.total_bill}")

        self.txtarea.insert(END,f"\n--------------------------------------")


        self.save_bill()


  def save_bill(self):

     op = messagebox.askyesno("Save Bill","Do you want to save the Bill?")

     if op>0:

        self.bill_data = self.txtarea.get("1.0",END)

        f1 = open("D:\\A1 Ideas\\bills/"+str(self.bill_no.get())+".txt","w")

        f1.write(self.bill_data)

        f1.close()

        messagebox.showinfo("Saved",f"Bill no. : {self.bill_no.get()} save
successfully")

     else:

        return




  def find_bill(self):


present="no"

     for I in os.listdir("D:\\A1 Ideas\\bills/"):

        if( I.split(".")[0])==self.search_bill.get():

            f1=open(f"D:\\A1 Ideas\\bills/{I}","r")
```

```
          self.txtarea.delete("1.0",END)
          #   self.txtarea.insert(END,f1)
          for d in f1:
             self.txtarea.insert(END,d)
          f1.close()
          present="yes"
    if present=="no":
       messagebox.showerror("Error","Invalid Bill No.")


  def clear_data(self):
    op=messagebox.askyesno("Reset","Do you really want to Reset data?")
    if op>0:                              #===================== exit fn
============================#


       self.soap.set(0)
       self.face_cream.set(0)
       self.face_wash.set(0)
       self.spray.set(0)
       self.gell.set(0)
       self.loshan.set(0)




  ##########    Grocery vars ###########
       self.rice.set(0)
       self.food_oil.set(0)
       self.daal.set(0)
```

```
        self.wheat.set(0)

        self.sugar.set(0)

        self.tea.set(0)


        ##########     Cold Drinks    #############


        self.mazaa.set(0)

        self.cock.set(0)

        self.frooti.set(0)

        self.thumbs_up.set(0)

        self.limca.set(0)

        self.sprite.set(0)


        #########       Total Product Price & Tax Variable
################


        self.cosmetics_price.set("")

        self.grocery_price.set("")

        self.cold_drink_price.set("")

        self.cosmetics_pric_tax.set("")




        self.grocery_tax.set("")

        self.cold_drink_tax.set("")


        ###########    Customer    #########
        self.c_name.set("")
        self.c_phon.set("")
        self.bill_no.set("")
        x=random.randint(1000,99999)
        self.bill_no.set(str(x))

        self.search_bill.set("")
        self.welcome_bill()



root= Tk()
obj = Bill(root)
root.mainloop()
## main window closed
```

Code to change price of items

```
from tkinter import*
```



Hunger Zone

Starter			Main Dishes			Drinks and Deserts		
1.	French Fries	25	1.	Choole Bhature	65	1.	Ice Tea	50
2.	Samosa	12	2.	Biryani	100	2.	Cold Drink	100
3.	Chana masala	30	3.	Daal Puri	150	3.	Grape Juice	125
4.	Litti Chokha	50	4.	Tangri Kabab	200	4.	Apple Juice	78
5.	Masala Chicken	85	5.	Pav Bhaji	95	5.	Barfi	145
6.	Chaat	25	6.	Daal Chawal	300	6.	Ice Cream	65

```
import tkinter.messagebox



left1 = Tk()

left1.geometry("1600x800+0+0")

left1.title("Change Price")


label4 = Label(left1, font = ('arial',50,'bold'), text ="Change Price", fg = "steel blue", bd = 10, anchor = 'w')

label4.grid(row = 0)



fries_inp_p = StringVar()
```

```
Sandwich_inp_p = StringVar()

burger_inp_p = StringVar()

drinks_inp_p = StringVar()

Pasta_inp_p = StringVar()

Tacos_inp_p = StringVar()


def update():

        f = open('value.txt','r')

        line = f.readlines()

        fries_p = float(line[0])

        Sandwich_p = float(line[1])

        burger_p = float(line[2])

        drinks_p = float(line[3])

        Pasta_p = float(line[4])

        Tacos_p = float(line[5])

        f.close()


        f2 = open('value.txt','w')

        try:

                CoF1 = float(fries_inp_p.get())

        except Exception as e:

                if fries_inp_p.get() != "":

                        tkinter.messagebox.showinfo('Error','Incorrect Input')




                fries_inp_p.set("")
                f2.write(str(fries_p)+"\n")
        else:
                f2.write(str(CoF1)+"\n")


        try:
                CoS1 = float(Sandwich_inp_p.get())
        except Exception as e:
                if Sandwich_inp_p.get() != "":
                        tkinter.messagebox.showinfo('Error','Incorrect Input')
                Sandwich_inp_p.set("")
                f2.write(str(Sandwich_p)+"\n")
        else:
                f2.write(str(CoS1)+"\n")


        try:
                CoB1 = float(burger_inp_p.get())
        except Exception as e:
                if burger_inp_p.get() != "":
                        tkinter.messagebox.showinfo('Error','Incorrect Input')
                burger_inp_p.set("")
                f2.write(str(burger_p)+"\n")
        else:
                f2.write(str(CoB1)+"\n")



try:
    CoD1 = float(drinks_inp_p.get())
except Exception as e:
    if drinks_inp_p.get() != "":
        tkinter.messagebox.showinfo('Error','Incorrect Input')
    drinks_inp_p.set("")
    f2.write(str(drinks_p)+"\n")
else:
    f2.write(str(CoD1)+"\n")


try:
    CoP1 = float(Pasta_inp_p.get())
except Exception as e:
    if Pasta_inp_p.get() != "":
        tkinter.messagebox.showinfo('Error','Incorrect Input')
    Pasta_inp_p.set("")
    f2.write(str(Pasta_p)+"\n")
else:
    f2.write(str(CoP1)+"\n")


try:
    CoC1 = float(Tacos_inp_p.get())
except Exception as e:
    if Tacos_inp_p.get() != "":



            tkinter.messagebox.showinfo('Error','Incorrect Input')
        Tacos_inp_p.set("")
        f2.write(str(Tacos_p))
    else:
        f2.write(str(CoC1))

    #f.write(str(Sandwich_p) +"\n"+ str(Pasta_p) +"\n"+ str(Tacos_p) +"\n"+ str(fries_p) +"\n"+ str(burger_p) +"\n"+ str(drinks_p))

    tkinter.messagebox.showinfo('Update Box','Successfully Updated')
    f2.close()

def reset1():
    fries_inp_p.set("")
    Sandwich_inp_p.set("")
    burger_inp_p.set("")
    drinks_inp_p.set("")
    Pasta_inp_p.set("")
    Tacos_inp_p.set("")

def qexit1():
    left1.destroy()

def printfn():



f3 = open('value.txt','r')
```

```
        liness = f3.readlines()
        fries_p = float(liness[0])
        Sandwich_p = float(liness[1])
        burger_p = float(liness[2])
        drinks_p = float(liness[3])
        Pasta_p = float(liness[4])
        Tacos_p = float(liness[5])
        print("fry "+str(fries_p))
        print("Sandwich "+str(Sandwich_p))
        print("chi tikka "+str(Tacos_p))
        print("Pasta "+str(Pasta_p))
        print("burger "+str(burger_p))
        print("drinks "+str(drinks_p))
        f3.close()

def backfn():
        left1.destroy()
        import question



fries = Label(left1, font=('arial',16,'bold'),text = "Fries", bd = 16, anchor = 'w' )
fries.grid(row=1,column=0,sticky = E)
txt_fries = Entry(left1,font=('arial',16,'bold'), textvariable=fries_inp_p, bd=10,
insertwidth =4,bg = "powder blue", justify ='right')
```

```
txt_fries.grid(row=1,column=1)

Sandwich = Label(left1, font=('arial',16,'bold'),text = "Sandwich", bd = 16,
anchor = 'w' )

Sandwich.grid(row=2,column=0,sticky = E)

txt_Sandwich = Entry(left1,font=('arial',16,'bold'),
textvariable=Sandwich_inp_p, bd=10, insertwidth =4,bg = "powder blue",
justify ='right')

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



burger = Label(left1, font=('arial',16,'bold'),text = "Burger", bd = 16, anchor =
'w' )

burger.grid(row=3,column=0,sticky = E)

txt_burger = Entry(left1,font=('arial',16,'bold'), textvariable=burger_inp_p,
bd=10, insertwidth =4,bg = "powder blue", justify ='right')

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


drinks = Label(left1, font=('arial',16,'bold'),text = "Drinks", bd = 16, anchor = 'w'
)

drinks.grid(row=4,column=0,sticky = E)

txt_drinks = Entry(left1,font=('arial',16,'bold'), textvariable=drinks_inp_p,
bd=10, insertwidth =4,bg = "powder blue", justify ='right')

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


Pasta = Label(left1, font=('arial',16,'bold'),text = "Pasta", bd = 16, anchor = 'w' )
```

```
Pasta.grid(row=5,column=0,sticky = E)

txt_Pasta = Entry(left1,font=('arial',16,'bold'), textvariable=Pasta_inp_p, bd=10, insertwidth =4,bg = "powder blue", justify ='right')

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


Tacos = Label(left1, font=('arial',16,'bold'),text = "Tacos", bd = 16, anchor = 'w' )

Tacos.grid(row=6,column=0,sticky = E)

txt_Tacos = Entry(left1,font=('arial',16,'bold'), textvariable=Tacos_inp_p, bd=10, insertwidth =4,bg = "powder blue", justify ='right')

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



btn_update = Button(left1, padx= 16, pady= 8, bd= 16, fg= "black", font=('arial',16,'bold'), width=10, text= "Update", bg= "powder blue",command = update)

btn_update.grid(row=7, column= 1)


btn_reset1 = Button(left1, padx= 16, pady= 8, bd= 16, fg= "black", font=('arial',16,'bold'), width=10, text= "Reset", bg= "powder blue",command = reset1)

btn_reset1.grid(row=7, column= 2)


btn_exit1 = Button(left1, padx= 16, pady= 8, bd= 16, fg= "black", font=('arial',16,'bold'), width=10, text= "Exit", bg= "powder blue",command = qexit1)

btn_exit1.grid(row=7, column= 3)
```

```
btn_print = Button(left1, padx= 16, pady= 8, bd= 16, fg= "black", font=('arial',16,'bold'), width=10, text= "print", bg= "powder blue",command = printfn)

btn_print.grid(row=7, column= 4)


btn_back = Button(left1, padx= 16, pady= 8, bd= 16, fg= "black", font=('arial',16,'bold'), width=10, text= "Back", bg= "powder blue",command = backfn)

btn_back.grid(row=8, column= 3)




left1.mainloop()
```

Code for very first UI, which is login system

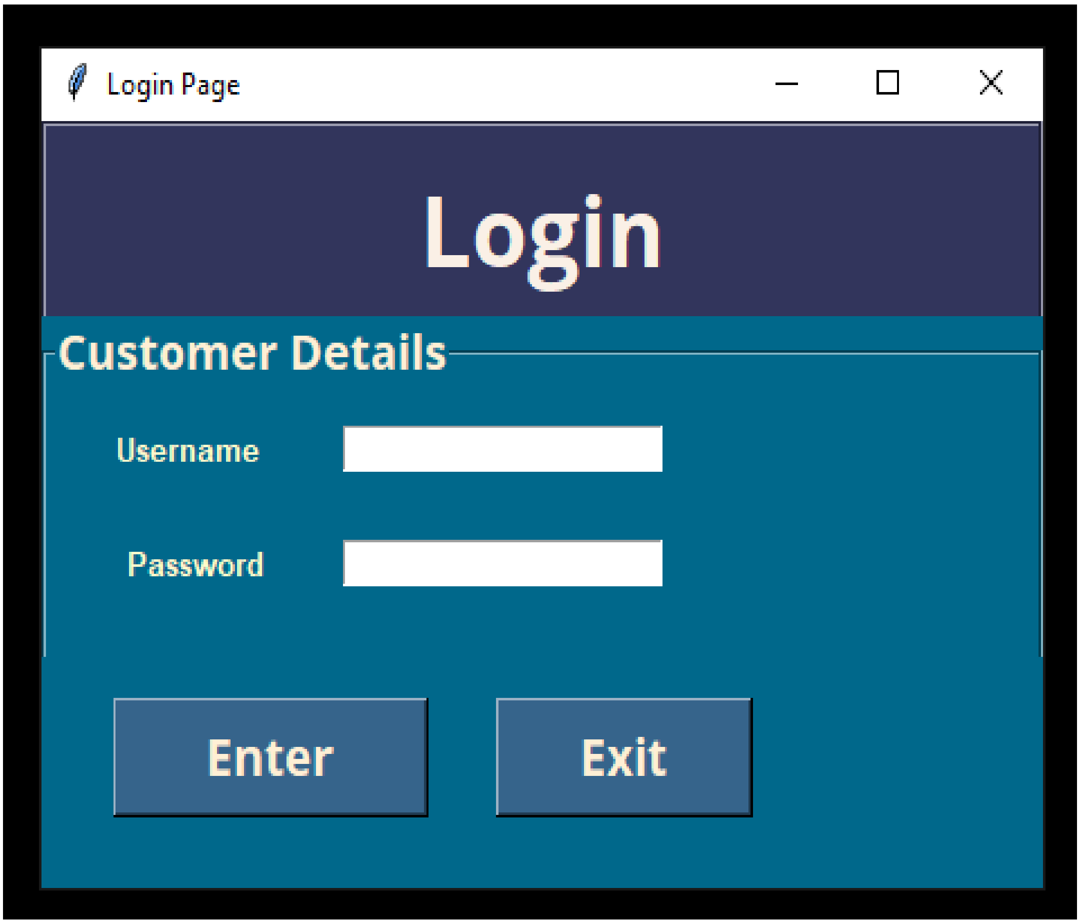
Login :

- *Login features allows only access to admin*
- *When 'Enter' is pressed it takes us/admin to main billing page*

Login :

- *This also decreases the probability of un-authorised accessing*

- *This feature makes data safe from unauthorised personals*
- *This features increases security*



Code :-

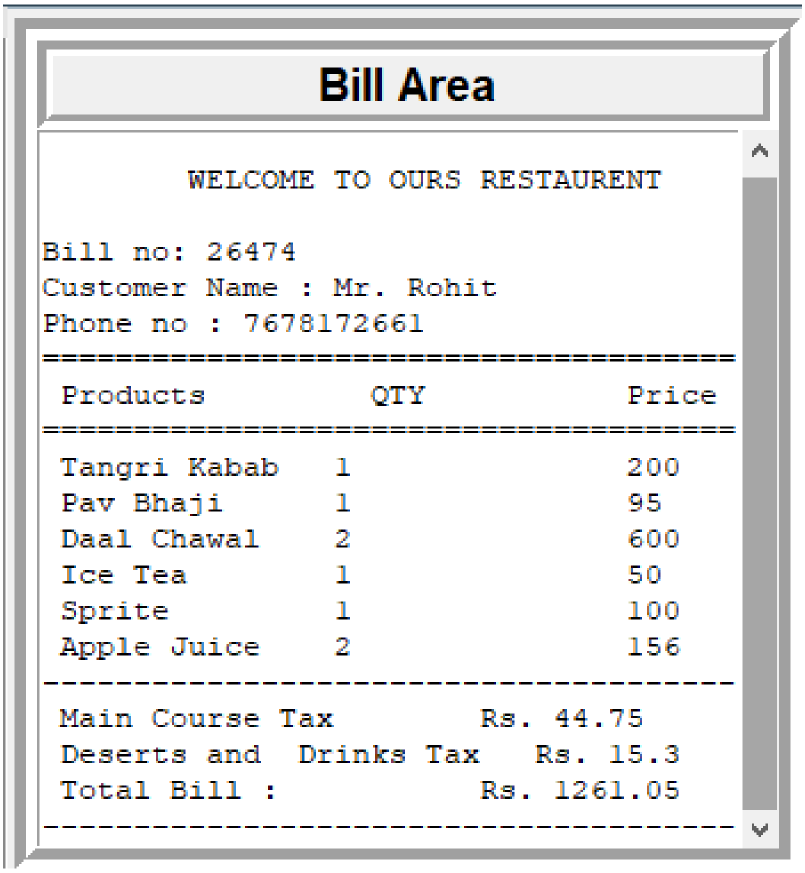
```
from tkinter import*
import tkinter.messagebox
root1 = Tk()
root1.geometry("420x220+0+0")
root1.title("Login Page")

name_inp = StringVar()
password_inp = StringVar()
```

```
def enter():
        if name_inp.get() == "harry" and password_inp.get() == "1234":
                root1.destroy()
                import question
        else:
                tkinter.messagebox.showinfo('Error','Authentication Failed')
                name_inp.set("")
                password_inp.set("")

def destroy():
        root1.destroy()


label = Label(root1, font = ('arial',50,'bold'), text ="Login", fg = "steel blue", bd = 10, anchor = 'w')
label.grid(row = 0)



label1 = Label(root1, text="Username")
label2 = Label(root1, font =('slant',10,'bold'),text="Pin")

entry1 = Entry(root1, textvariable = name_inp)
entry2 = Entry(root1, textvariable = password_inp)

label1.grid(row=1, sticky=E)
```

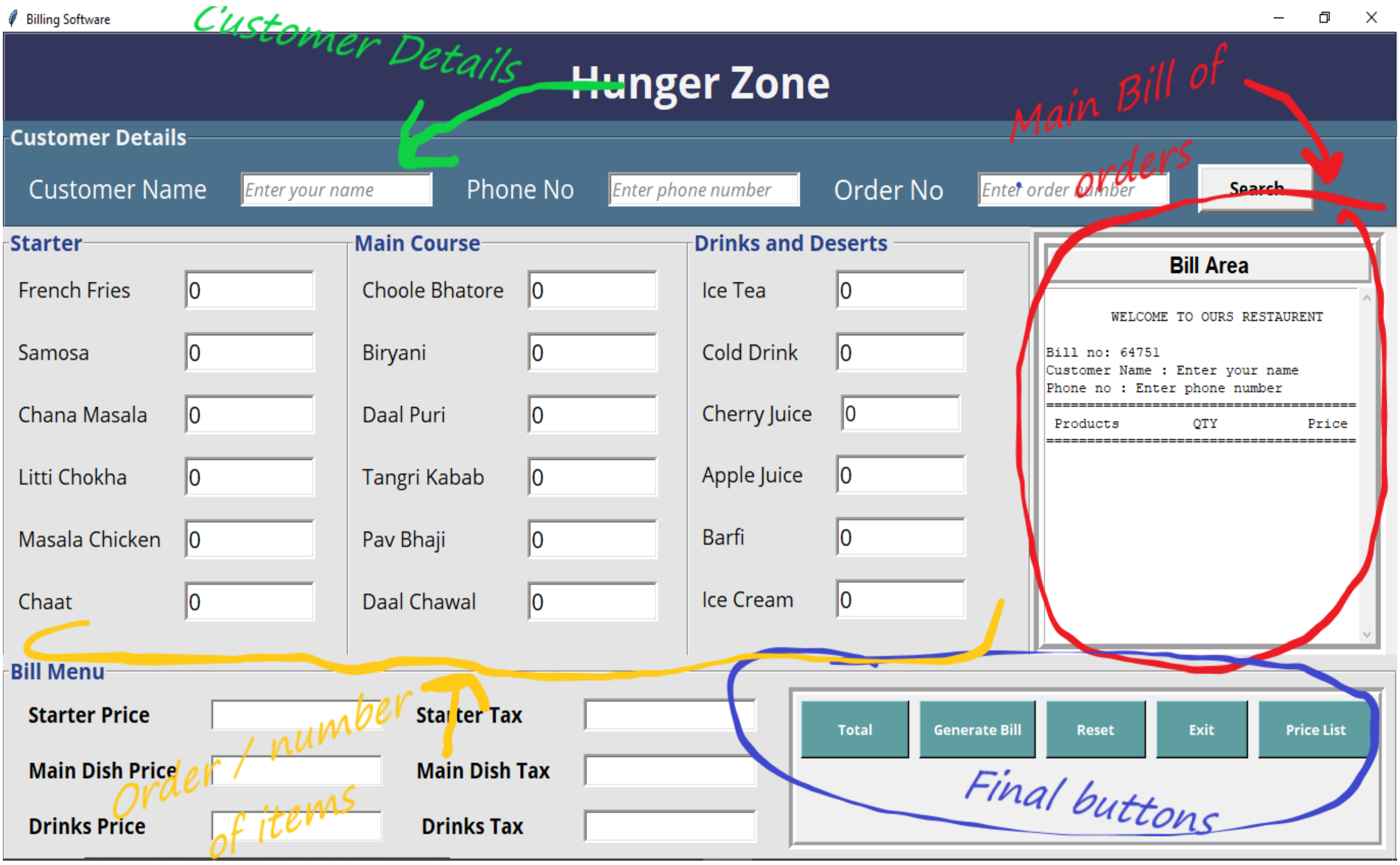
```
label2.grid(row=2,sticky=E)
entry1.grid(row=1,column=1)
entry2.grid(row=2,column=1)
enter_btn = Button(root1, text="Enter", command= enter)
enter_btn.grid(row=3, column=1)

exit_btn = Button(root1, padx= 1, text="Exit", command= destroy)
exit_btn.grid(row=3,column=2)

root1.mainloop()
```

Invoice

- *Billing Area :*
  - ○ *Billing is to have a printed overall order list and their price*
  - ○ *Billing includes name, phone number are must*
  - ○ *“Bill no” is auto generated unique number*



```
        WELCOME TO OURS RESTAURENT

Bill no: 26474
Customer Name : Mr. Rohit
Phone no : 7678172661
======================================
 Products         QTY            Price
======================================
 Tangri Kabab    1              200
 Pav Bhaji       1              95
 Daal Chawal     2              600
 Ice Tea         1              50
 Sprite          1              100
 Apple Juice     2              156
--------------------------------------
 Main Course Tax        Rs. 44.75
 Deserts and  Drinks Tax   Rs. 15.3
 Total Bill :           Rs. 1261.05
--------------------------------------
```



Customer Name: Enter your name | Phone No: Enter phone number | Order No: Enter order number

French Fries	0	Choole Bhatore	0	Ice Tea	0
Samosa	0	Biryani	0	Cold Drink	0
Chana Masala	0	Daal Puri	0	Cherry Juice	0
Litti Chokha	0	Tangri Kabab	0	Apple Juice	0
Masala Chicken	0	Pav Bhaji	0	Barfi	0
Chaat	0	Daal Chawal	0	Ice Cream	0

Starter Price		Starter Tax	
Main Dish Price		Main Dish Tax	
Drinks Price		Drinks Tax	

```
           WELCOME TO OURS RESTAURENT

Bill no: 64751
Customer Name : Enter your name
Phone no : Enter phone number
======================================
 Products          QTY           Price
======================================
```

Advantages of the software

- *Saving in labor due to self-service system.*
- *Faster work is done*
- *Low cost of operation*
- *Easy to re-correct and re-enter newly fresh data*
- *No complexity in customer is data as all gets an unique data*

Disadvantages of the software

Limitations in the software are :-

-

- *There is lack of bill deletion once created.*
- *Dependent on Operating system like Windows OS , it requires different coding for other Operating Systems.*
- *Any problem in software make system not working purposefully.*

Conclusion

- *We have a system which takes the necessary choice of the customer according to the filter like category of the food. This is achieved through an easy to use graphical interface menu options. The users can add any number of items to the cart from any of the available food categories.*
- *Customer according to the choice payment is done. And the customer gets a bill generated on a hard copy.*
- *Using this software users saves their time by faster ordering, and this software saves the extra effort of the man. And hence all this saves the capital of the owner of the restaurant.*
- *Project is mainly implemented in restaurants and dhabas. I take this opportunity to all those who helped me in completing this project and implementation.*
- *I’m immensely grateful to my esteemed project guide Mr. H.S. Bhatia (Head of Department of Computer Engg.) guidance, without which this work would not have been possible.*
- *This project and implementation has contributed a lot to my knowledge that has proved to be a value addition for me.*

Future enhancement

This application can be easily implemented under various situations. We can add new features as and when we require. Reusability is possible as and when require in this application. There is flexibility in all the modules.

SOFTWARE SCOPE *–*

- *Extensibility:- This software is extendable in ways that its original developers may not expect. The following principles enhances extensibility like hide data structure, void traversing multiple links or methods, avoid case statements on object type and distinguish public and private operations.*
- *Reusability:- It is possible as and when require in this application. We can update it next version. Reusable software reduces design, coding and testing cost by amortizing effort over several designs. Reducing the amount of code also simplifies understanding, which increases the likelihood that the code is correct. We follow up both types of reusability: sharing of newly written code within a project and reuse of previously written code on new projects.*