**JAYPEE INSTITUE OF INFORMATION TECHNOLOGY**



**IDE FOR BLIND**

**Minor Project Report**

**BATCH B4**

**GROUP 3**

**Submitted To:** **Submitted By:**

Mrs. Sakshi Agarwal Akash [16103318]

Akarsh Sharma [16103315]

Rajat Kumar [16103167]

Rishabh Dhawan[16103213]

**ACKNOWLEDGEMENT**

On the very outset of this report, we would like to extend our deepest gratitude to our teacher Mrs. SAKSHI AGARWAL, who provided us with the opportunity of completing our project and also guided us with her valuable suggestions to enhance the quality of this project. Without her active guidance, support, cooperation and encouragement this project would not have taken its present shape.

We extend our gratitude to Jaypee Institute of Information Technology for giving us this opportunity. We also acknowledge with deep sense of reverence, our gratitude towards our parents, members of family and friends who have always supported us morally.

Thanking you,

Akash

Akarsh Sharma

Rajat Kumar

Rishabh Dhawan

**CONTENTS**

1. INTRODUCTION

2. PROBLEM STATEMENT

3. METHODOLOGY

4. RESULTS

5. CONCLUSION AND FUTURE WORK

6. REFERENCES

**INTRODUCTION**

Sustainability is the development that satisfies the needs of the present without compromising the capacity of future generations, guaranteeing the balance between economic growth, care for the environment and social well-being.

Since 2000, there has been enormous progress in achieving the target of universal primary education. Achieving inclusive and quality education for all reaffirms the belief that education is one of the most powerful and proven vehicles for sustainable development. This goal ensures that all girls and boys complete free primary and secondary schooling by 2030. It also aims to provide equal access to affordable vocational training, to eliminate gender and wealth disparities, and achieve universal access to a quality higher education.

This project **IDE FOR BLIND** will endeavor to provide a coding environment **(IDE)** tailored to the needs of the blind to help them earn a basic earning through the use of a useful skill. This Integrated development is to help those coders who have lost their sight to use their coding skills even now in an environment friendly and adaptive to there need to code and develop in python.This project deals with all the basic codes of Python. It helps the user/coder by interacting with the user in speech format.

This project consist of a text editor, dedicated input window, descriptive output window, list box with autocomplete suggestions, dedicated hotkeys binded with different keyboard shortcuts for various functions like complie, autocomplete, code in speech format, save, open file etc.

**PROBLEM STATEMENT**

In World there are disparities in education and still equal access to all levels of education isn’t assured .Education if provided to all it might provide with vocational training for person with **disabilities**.

Inequality within and among countries that discriminates on the basis of the social, economic and political, age, sex, **disability**, race, ethnicity, origin, religion or economic or other status.

**Approach towards the problem**

In order to deal with various problems discussed above this project helps in providing the needs of the blind to help them earn a basic earning through the use of a useful skill. Project helps the person with sight disability in coding with Python language by speaking the code line by line , guiding the user on where the cursor is placed, output of current complied code, errors and autocompleting various syntax in code.

**Sustainable Development Goal**

**Goal :** Quality Education

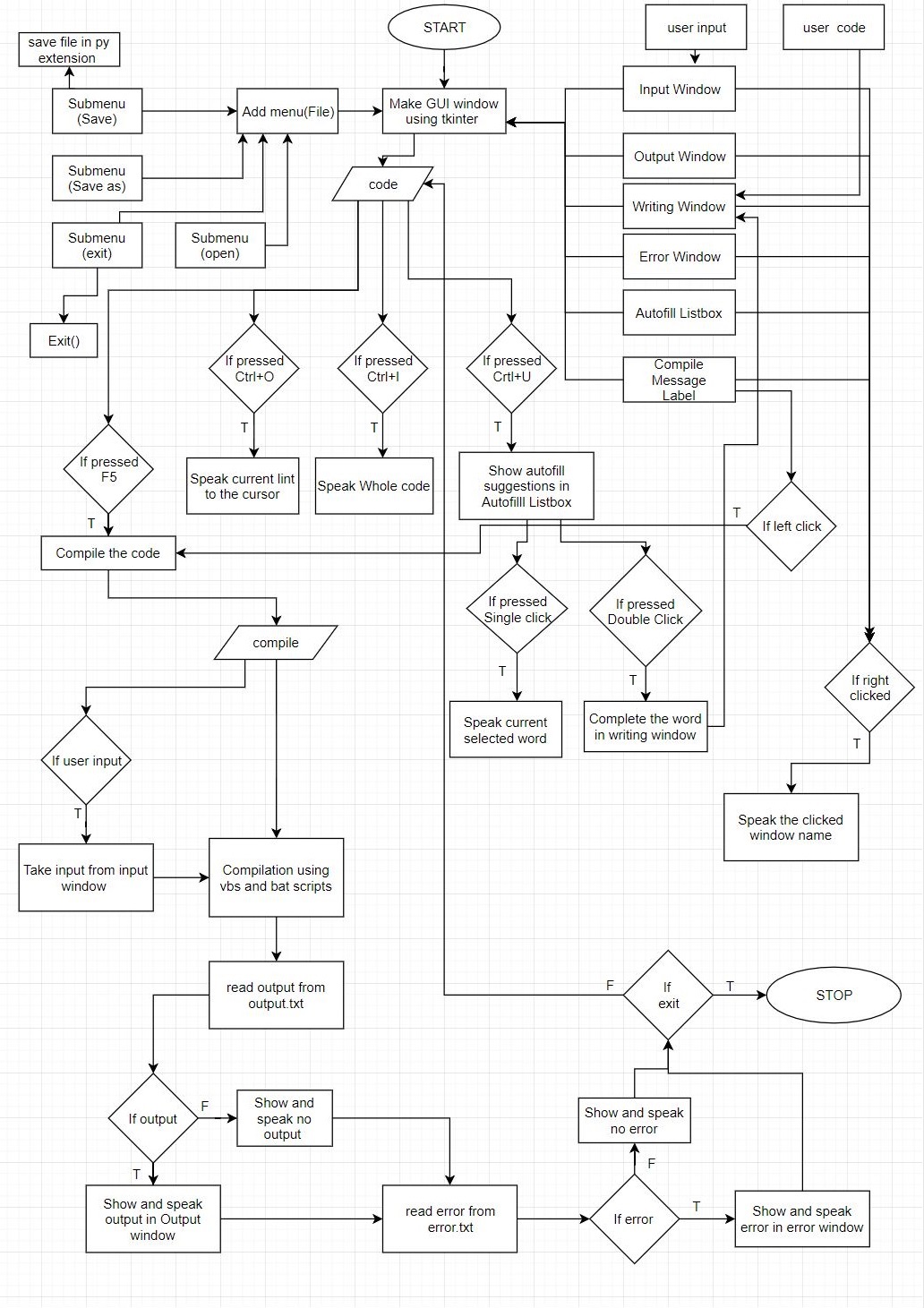
**Target 4.5** : By 2030, eliminate gender disparities in education and ensure equal access to all levels of education and vocational training for the vulnerable, including persons with **disabilities**, indigenous peoples and children in vulnerable situations

**Goal** : Reduce inequality within and among countries

**Target 10.2** : By 2030, empower and promote the social, economic and political inclusion of all, irrespective of age, sex, **disability**, race, ethnicity, origin, religion or economic or other status

**METHODOLOGY**

**MODULE DIAGRAM:**



**MODULE DESCRIPTION:**

**1. GUI(Tkinter):** To develop GUI(graphical user interface) of project,tkinter a standard Python interface is used.Tkinter is a [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) binding to the [Tk](https://en.wikipedia.org/wiki/Tk_(software)) [GUI](https://en.wikipedia.org/wiki/Graphical_user_interface) toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's [de facto standard](https://en.wikipedia.org/wiki/De_facto_standard) GUI. As with most other modern Tk bindings, Tkinter is implemented as a Python wrapper around a complete [Tcl](https://en.wikipedia.org/wiki/Tcl) interpreter embedded in the Python interpreter. Tkinter calls are translated into Tcl commands which are fed to this embedded interpreter, thus making it possible to mix Python and Tcl in a single application.

It comes with various widgets that have been used in the project:

* Text Box
* List Box
* Message Label
* Menu
* Key binding
* MessageBox

**2. PYTTSX3:** Pyttsx is a good text to speech conversion library in python but it was written only in python2 untill now .Even some fair amount of googling didn’t help much to get a tts library compatible with Python3.

There is however , one library gTTS which works perfectly in python3 but it needs internet connection to work since it relies on google to get the audio data.But Pyttsx is completely offline and works seemlesly and has multiple tts-engine support.The codes in this repos are slightly modified version of the pyttsx module of python 2.x and is a clone from westonpace’s repo. The purpose of creating this repo is to help those who want to have an offline tts lib for Python3 and don’t want to port it from python2 to python3 themselves.

**3. JEDI(an awesome autocompletion/static analysis library for Python):** Jedi is a static analysis tool for Python that can be used in IDEs/editors. Its historic focus is autocompletion, but does static analysis for now as well. Jedi is fast and is very well tested.Jedi has support for two different goto functions. It’s possible to search for related names and to list all names in a Python file and infer them.

Jedi is pretty simple and allows you to concentrate on writing a good text editor, while still having very good IDE features for Python.

**class jedi.Script(source=None, line=None, column=None, path=None, encoding='utf-8', sys\_path=None, environment=None)**

A Script is the base for completions, goto or whatever you want to do with Jedi.You can either use the source parameter or path to read a file. Usually you’re going to want to use both of them (in an editor).

The script might be analyzed in a different sys.path than Jedi:

* if sys\_path parameter is not None, it will be used as sys.path for the script.
* if sys\_path parameter is None and VIRTUAL\_ENV environment variable is defined, sys.path for the specified environment will be guessed (see jedi.evaluate.sys\_path.get\_venv\_path()) and used for the script.
* otherwise sys.path will match that of Jedi.

|  |  |
| --- | --- |
| **Parameters:** | * **source** ([str](https://docs.python.org/3/library/stdtypes.html#str)) – The source code of the current file, separated by newlines. * **line** ([int](https://docs.python.org/3/library/functions.html#int)) – The line to perform actions on (starting with 1). * **column** ([int](https://docs.python.org/3/library/functions.html#int)) – The column of the cursor (starting with 0). * **path** ([str](https://docs.python.org/3/library/stdtypes.html#str) or [None](https://docs.python.org/3/library/constants.html#None)) – The path of the file in the file system, or ''if it hasn’t been saved yet. * **encoding** ([str](https://docs.python.org/3/library/stdtypes.html#str)) – The encoding of source, if it is not a Unicode object (default 'utf-8'). * **sys\_path** ([Environment](https://jedi.readthedocs.io/en/latest/docs/api.html#jedi.api.environment.Environment)) – sys.path to use during analysis of the script * **environment** – TODO |

**completions**()

Return **classes.Completion** objects. Those objects contain information about the completions, more than just names.

|  |  |
| --- | --- |
| **Returns:** | Completion objects, sorted by name and \_\_ comes last. |
| **Return type:** | list of **classes.Completion** |

**4.PYNPUT:** This library allows you to control and monitor input devices. It contains subpackages for each type of input device supported:

**pynput.mouse:** Contains classes for controlling and monitoring a mouse or trackpad.

**pynput.keyboard:** Contains classes for controlling and monitoring the keyboard.

All modules mentioned above are automatically imported into the pynput package. To use any of them, import them from the main package.

**5. VB AND BAT SCRIPTS:** VBScript is an Active Scripting language developed by Microsoft that is modelled on Visual Basic. It allows Microsoft Windows system administrators to generate powerful tools for managing computers with error handling, subroutines, and other advanced programming constructs.

A batch file is a kind of script file in DOS, OS/2 and Microsoft Windows. It consists of a series of commands to be executed by the command-line interpreter, stored in a plain text file.

We have executed the user’s code through bat script and we used VBScript to run it in the background.

**RESULTS**

**HARDWARE REQUIREMENTS:**

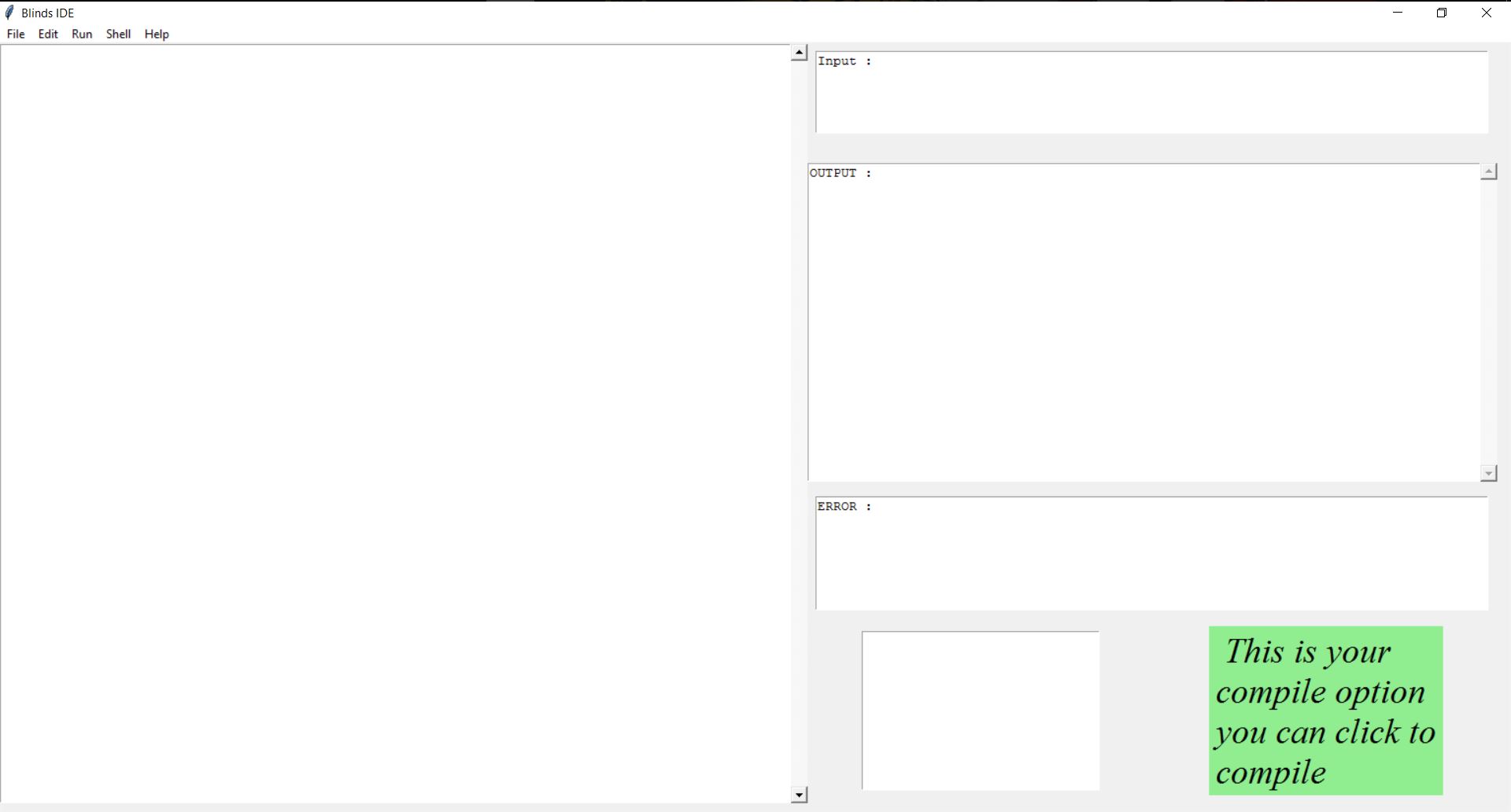
* Braille Keyboard:The Braille Keyboard for the Blind and Visually Impaired is a braille keyboard designed for use by individuals who are blind or have low vision. This 104-key keyboard is equipped with clear braille labels that allow the original key legends to show through enabling both blind and sighted users access to the keyboard.
* A computer with minimum of 1GB of RAM and Dual Core Processor with windows Operating System.
* A speaker.

**SOFTWARE REQUIREMENTS:**

* Python 3.5.2
* Python Libraries:
  + Pyttsx3
  + Jedi
  + Tkinter
  + pynput

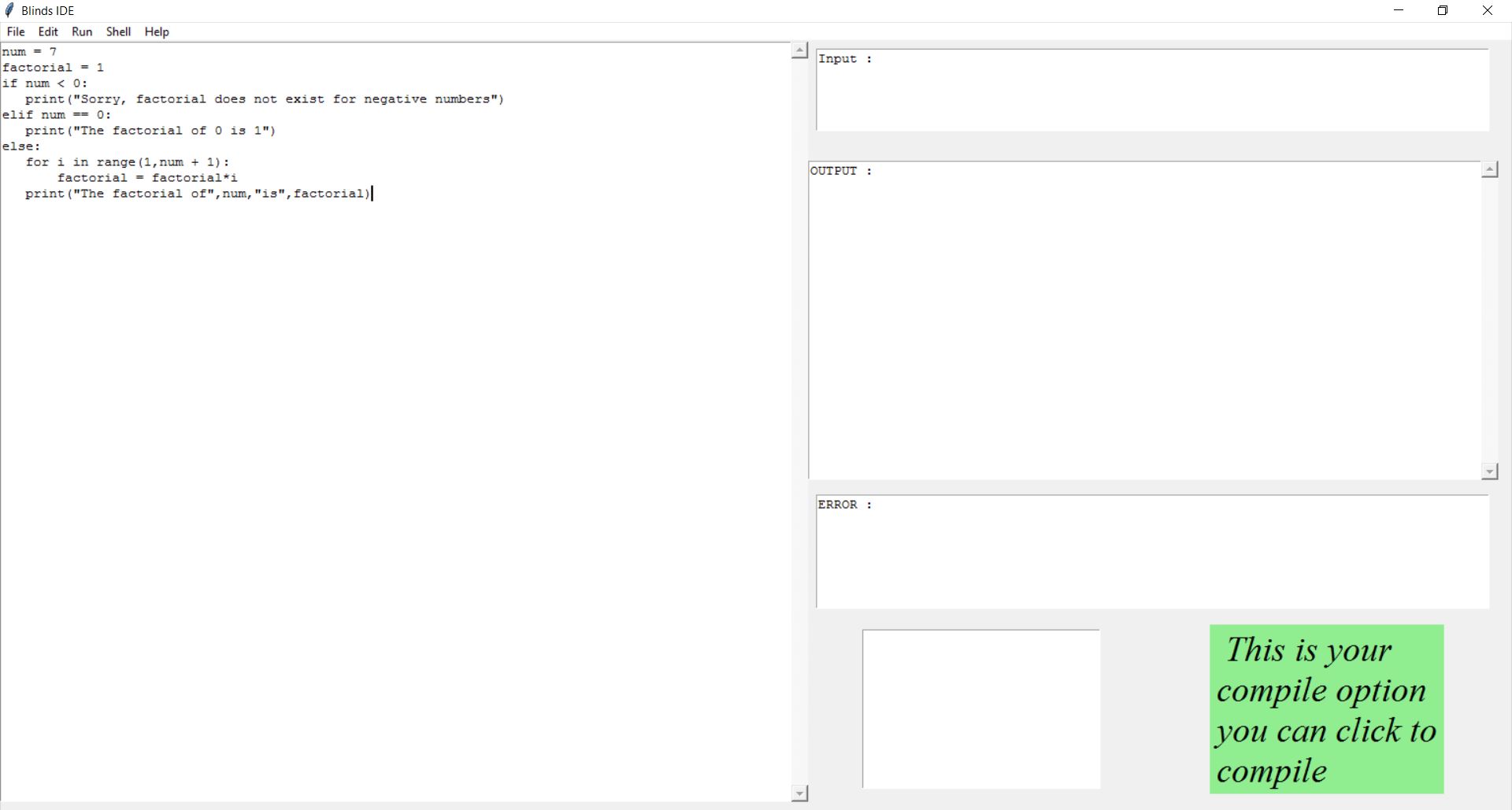
**SCREEN SHOT**

1. **Application Window:**

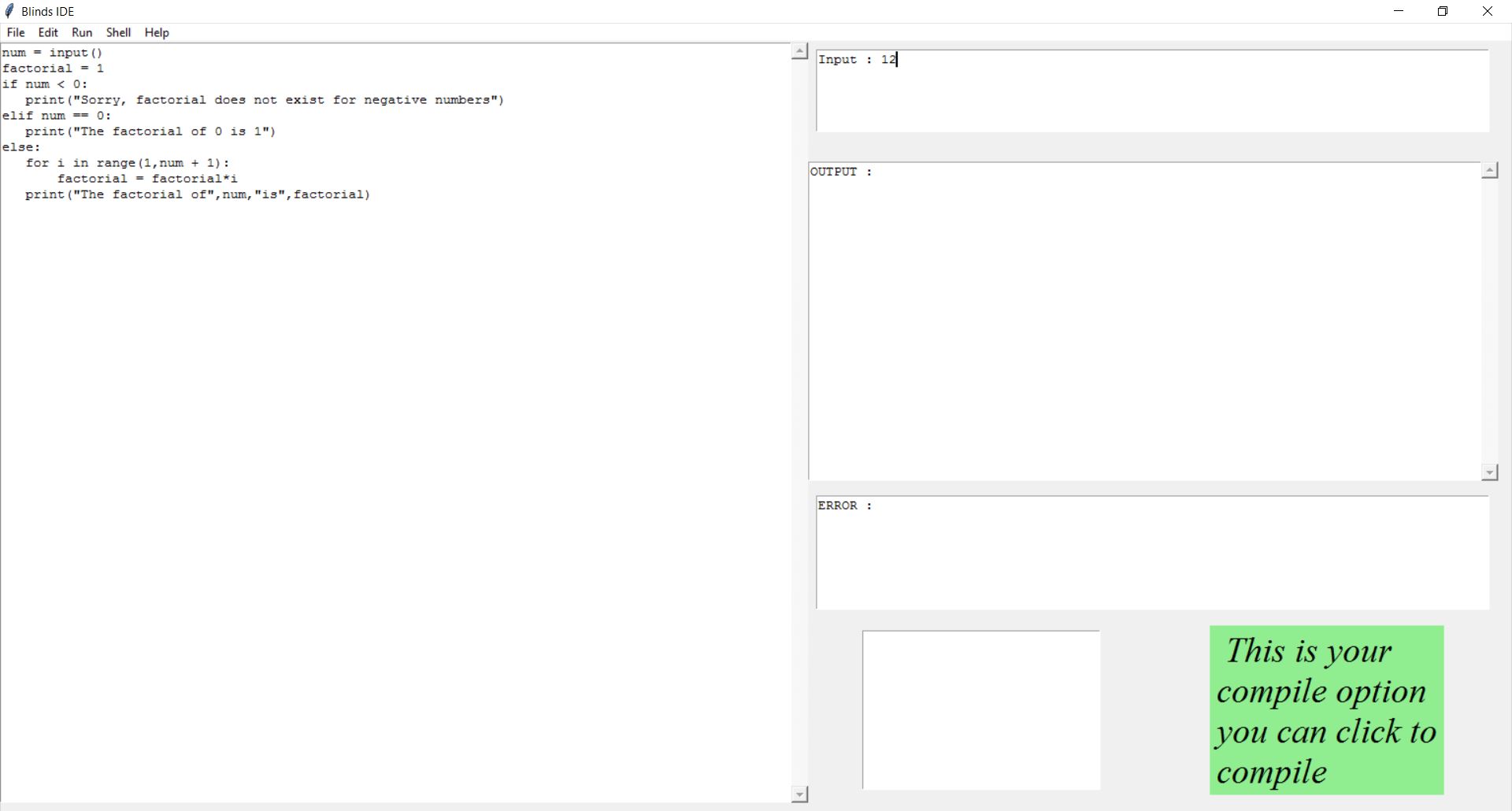


It is the main window of the application which opens when the user executes the program.User can right click on any of the window and the program will speak in which window the user is.

1. **Writing Window:**In writing window(left) user can write his/her code.The user can press ‘**Ctrl+I**’ to listen the whole code,’**Ctrl+o**’ to listen the currentline to the cursor,’**Ctrl+u**’ to get the suggested words in the autofill window, and **‘F5**’ to execute the code.

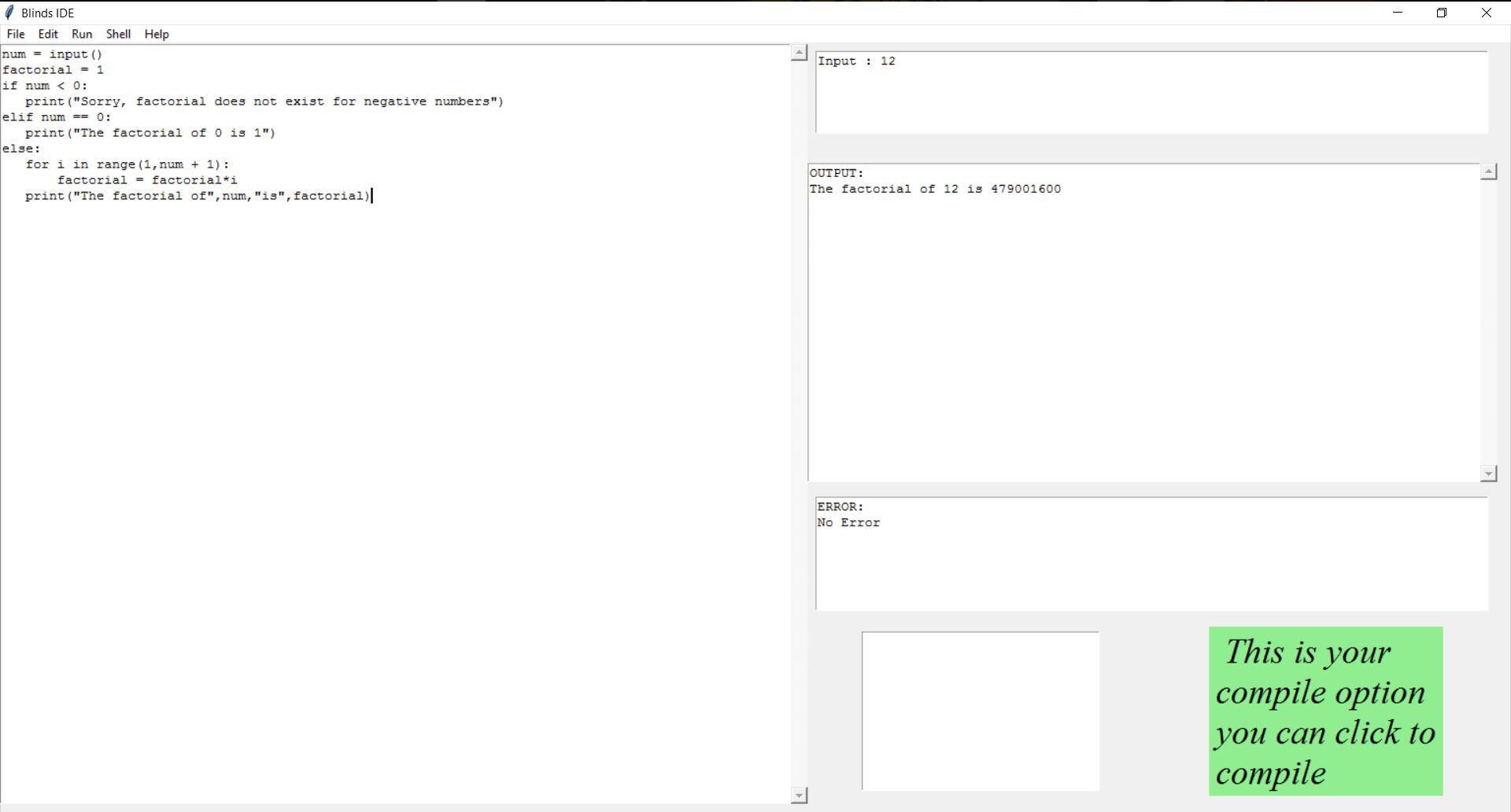


1. **Input Window:**



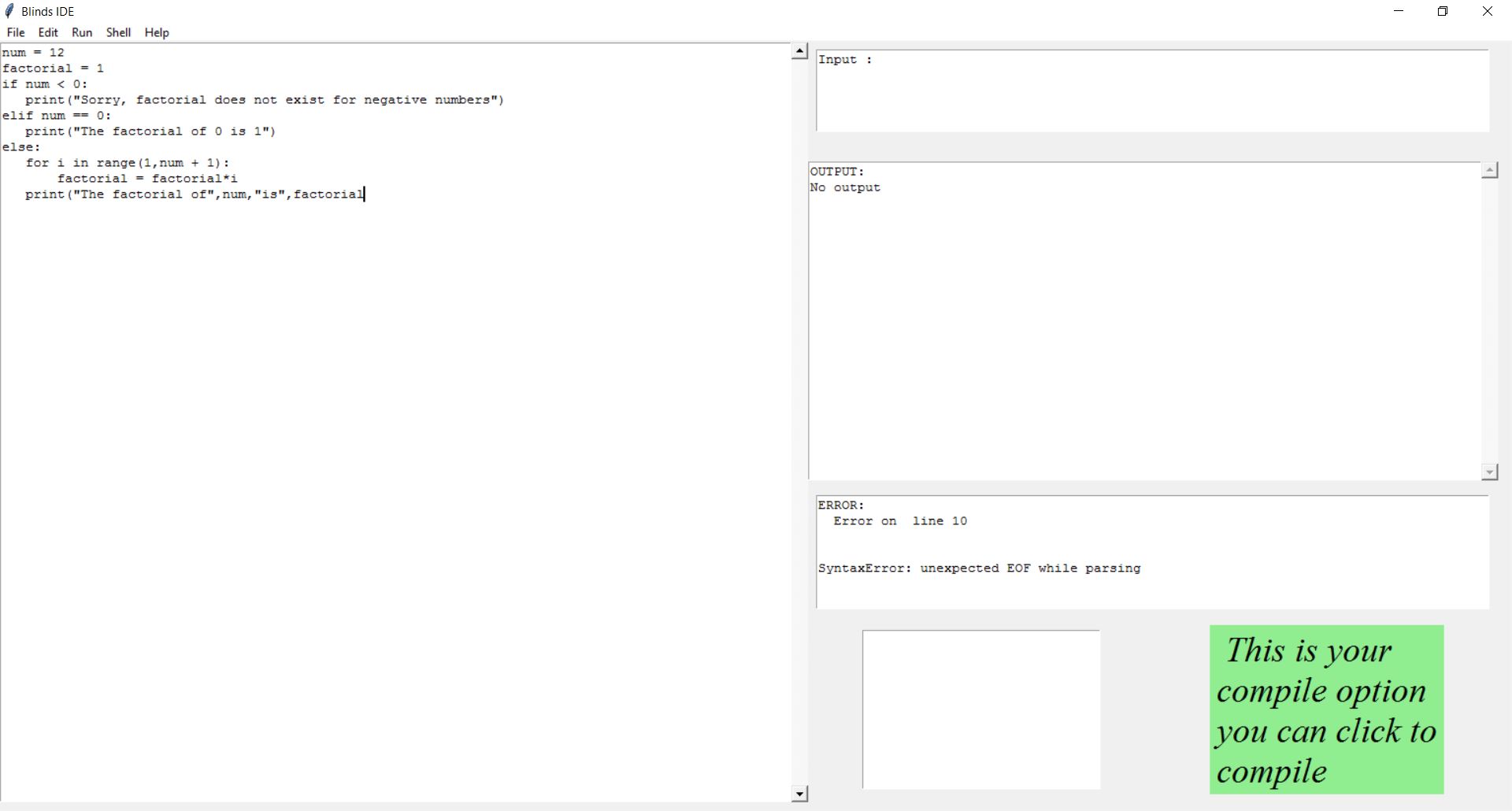
In input window (top right) user can give space separated inputs as per requirement of his/her’s code.

1. **Output Window:**



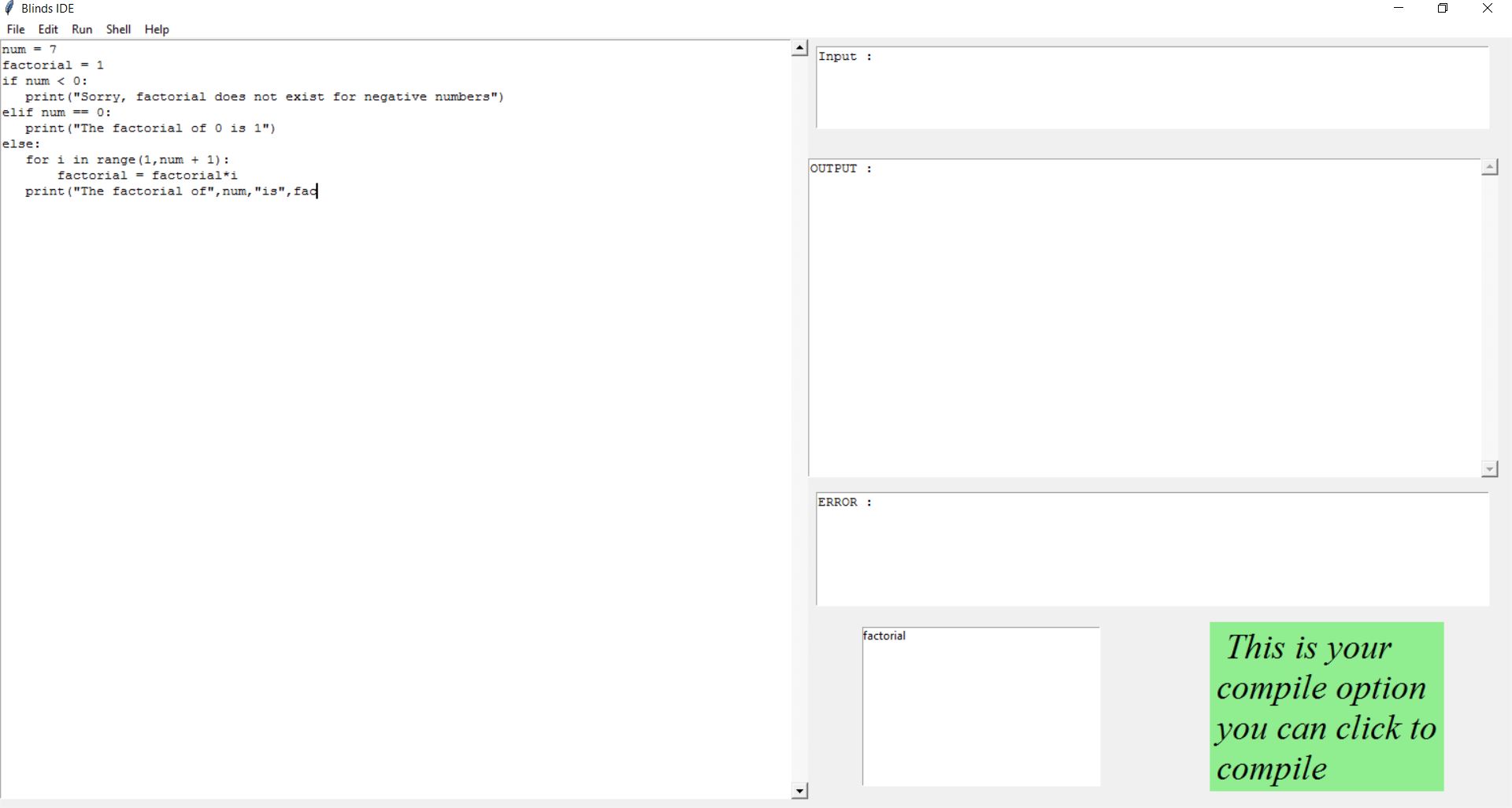
In output window(middle right) output is being shown and spoken.If there is no output then ‘No Output’ is shown and spoken.

1. **Error Window:**



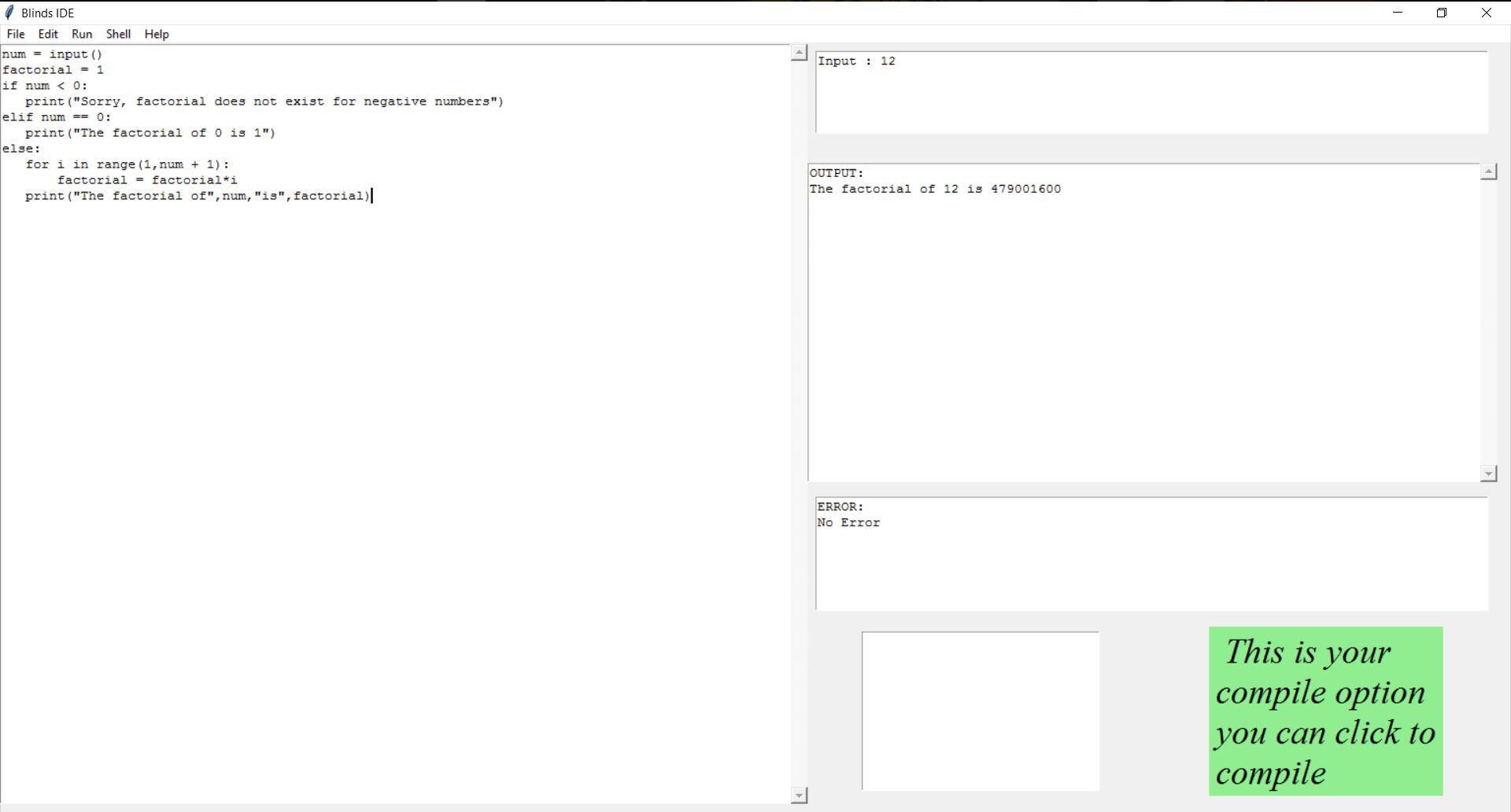
In error window(middle right) error is shown and spoken and the text cursor is moved to the error line. If there is no error then ‘No Error’ is shown and spoken.

1. **Autofill Listbox:**



In autofill Listbox(bottom right-left) the suggested words(python syntax words ,function names and variables).On single left clicking and of the suggested words the program will speak the word and double left clicking will insert the word in the writing window.

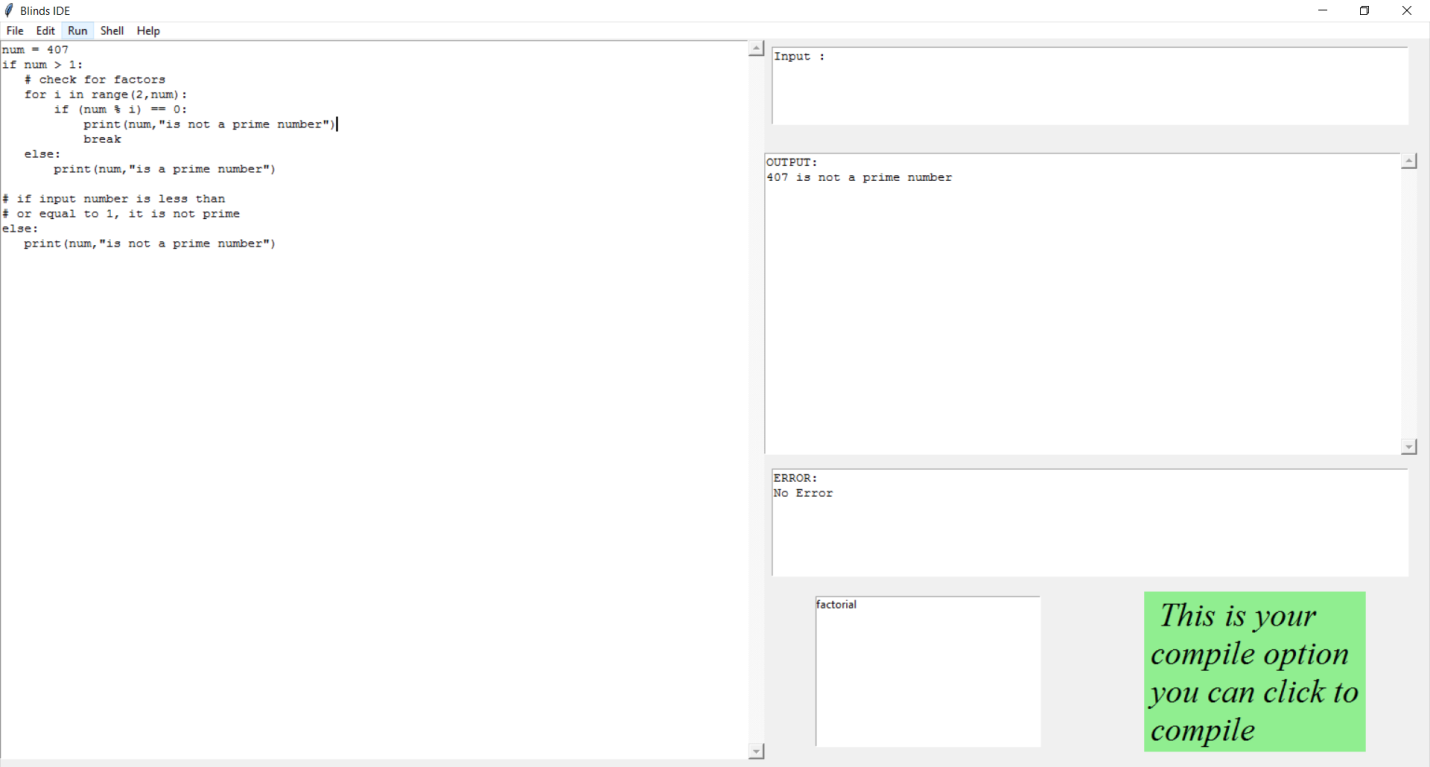
1. **Compile Message Label:**



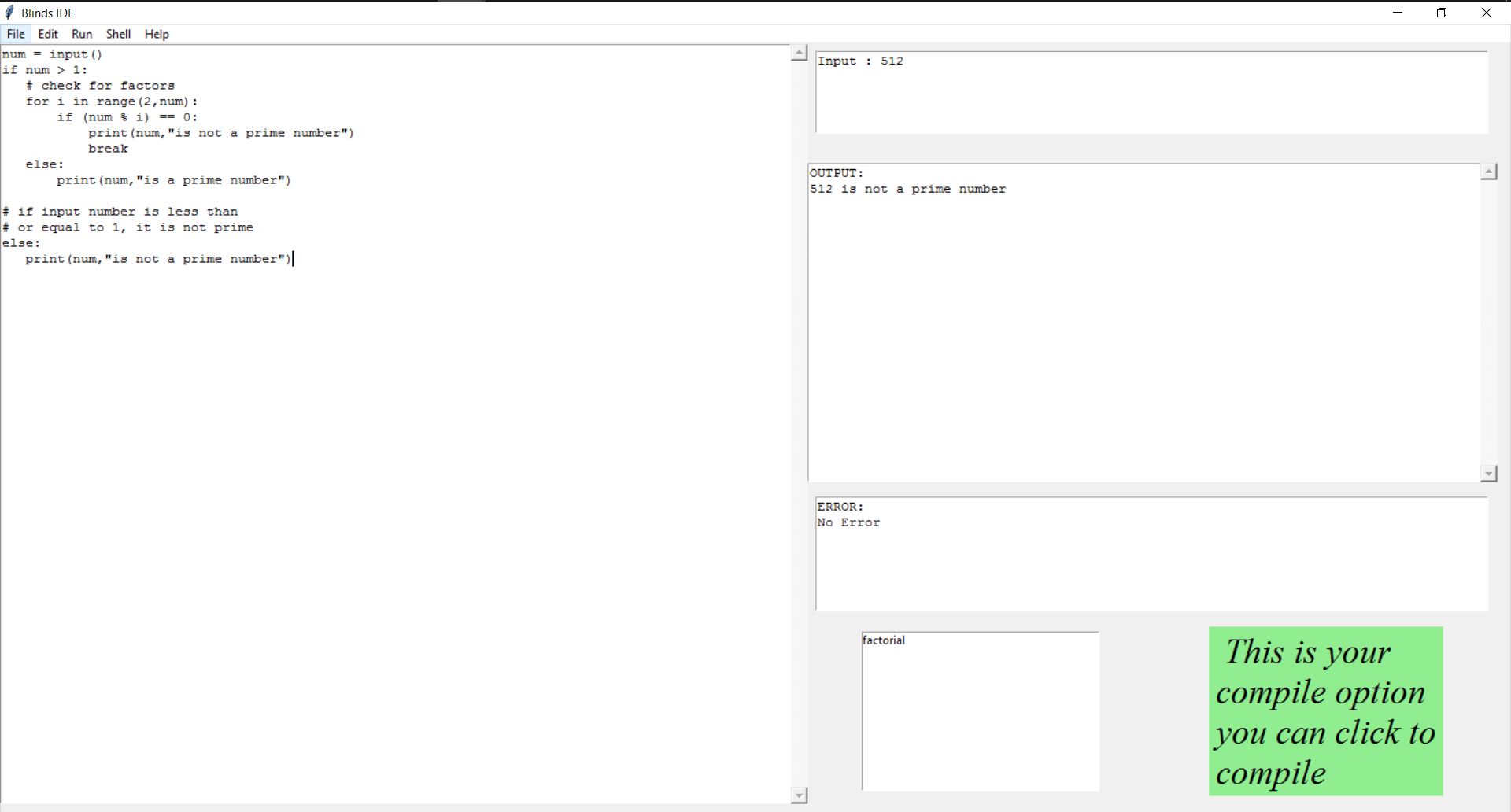
User can click Compile Message Label(bottom right-right) to execute his/hers code.

**TEST CASES:**

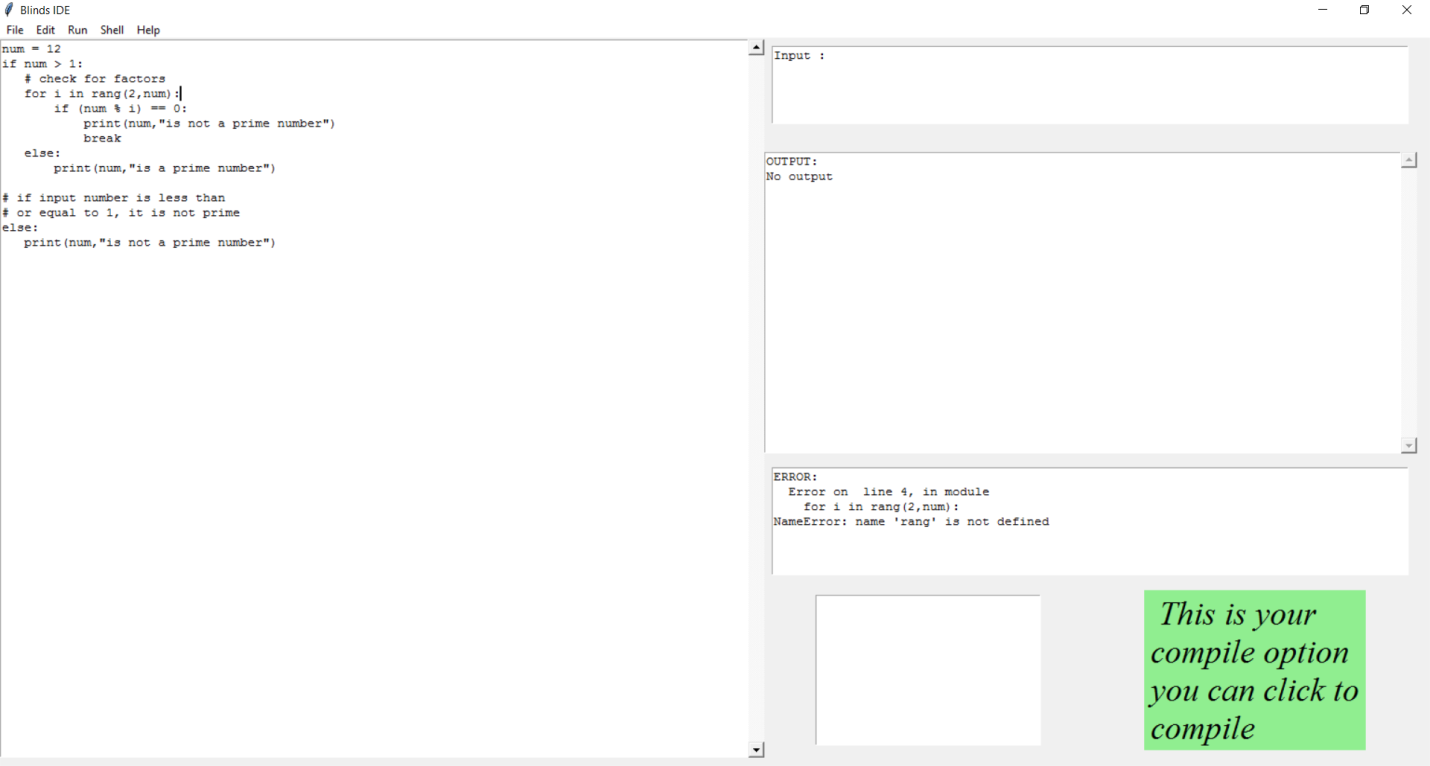
1. Prime Number Program:

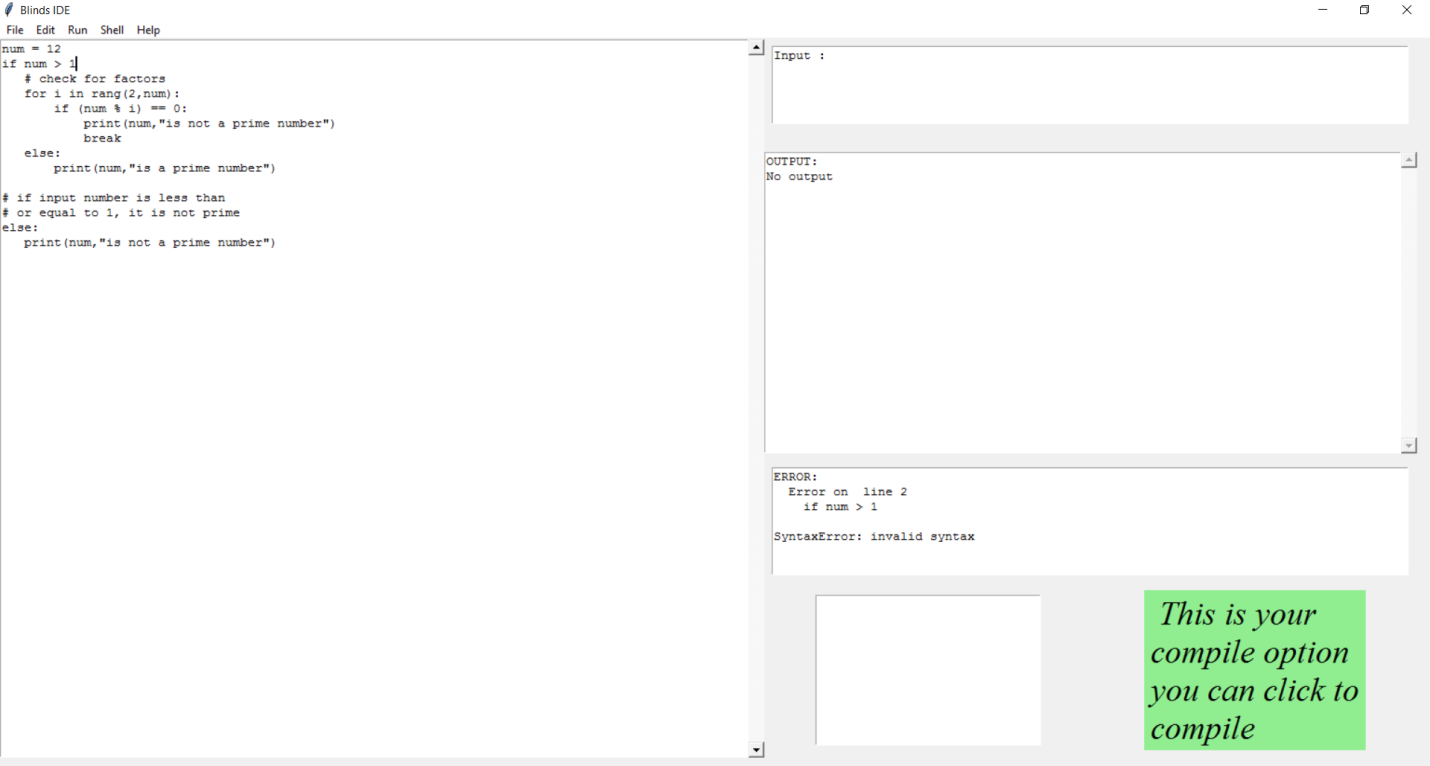
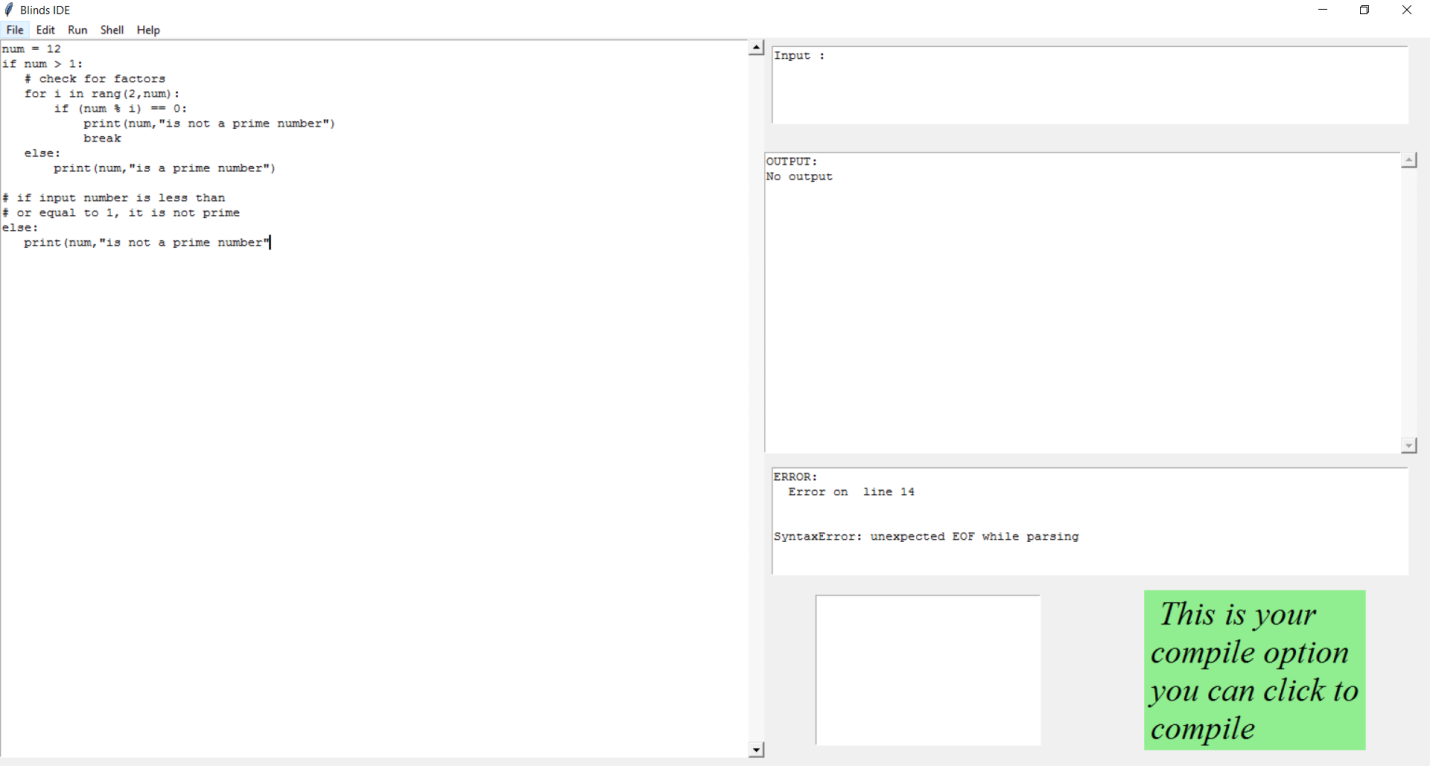
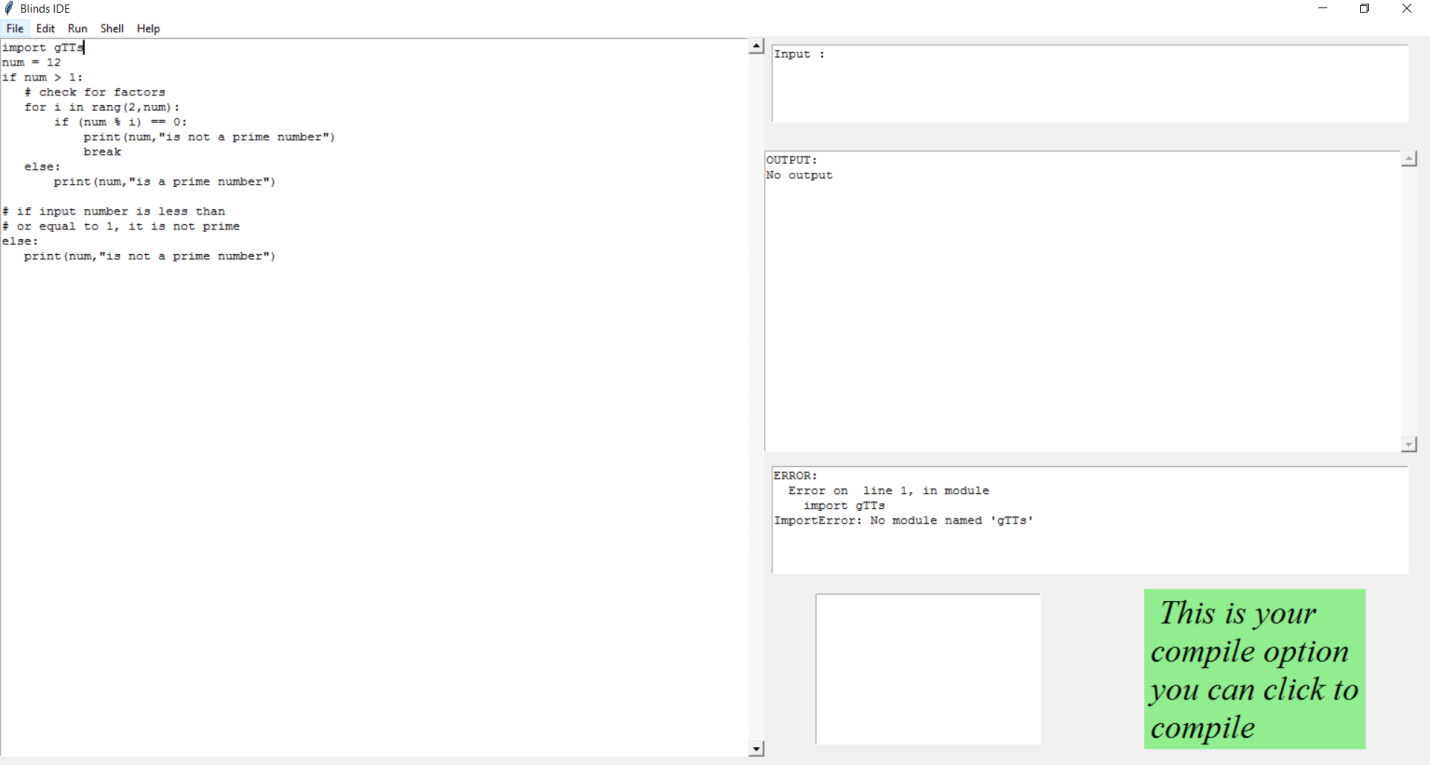
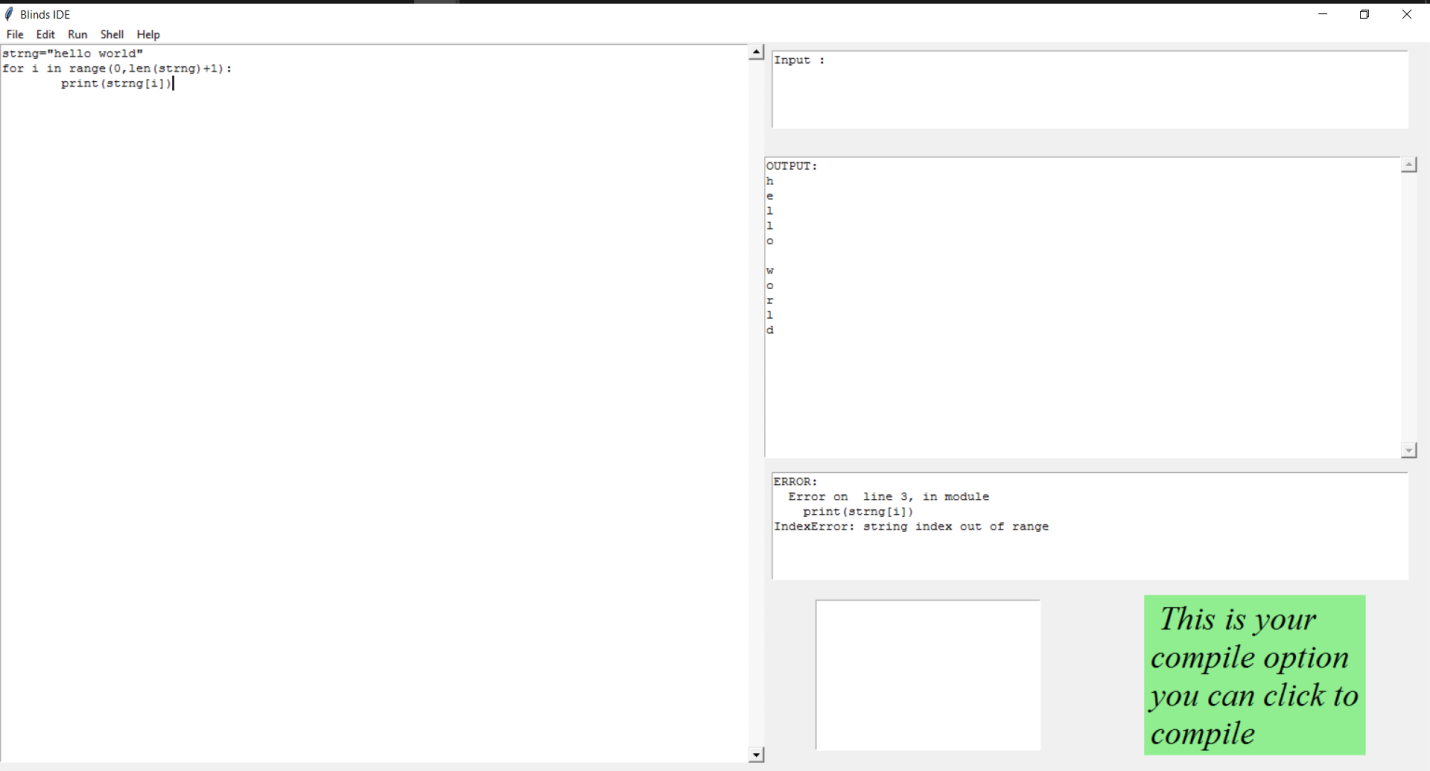
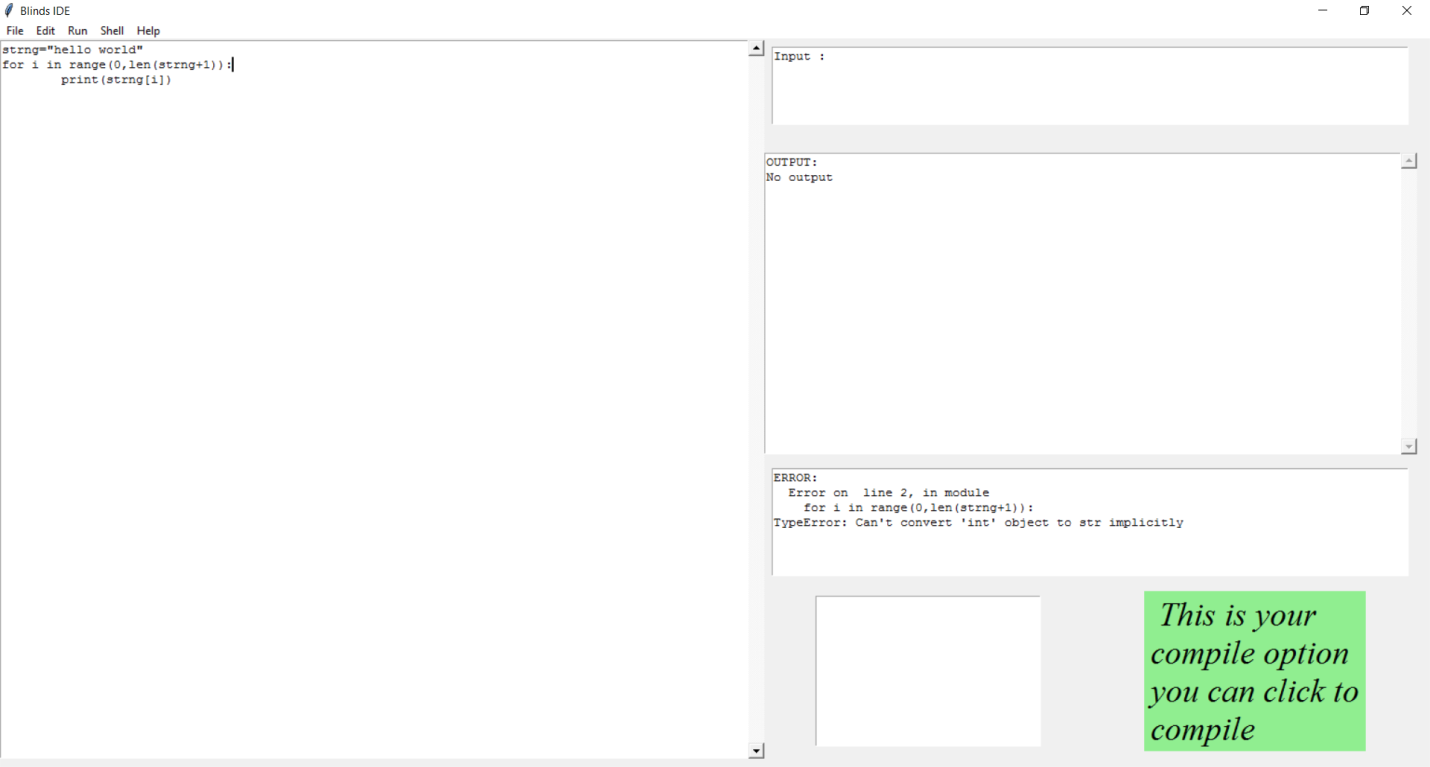
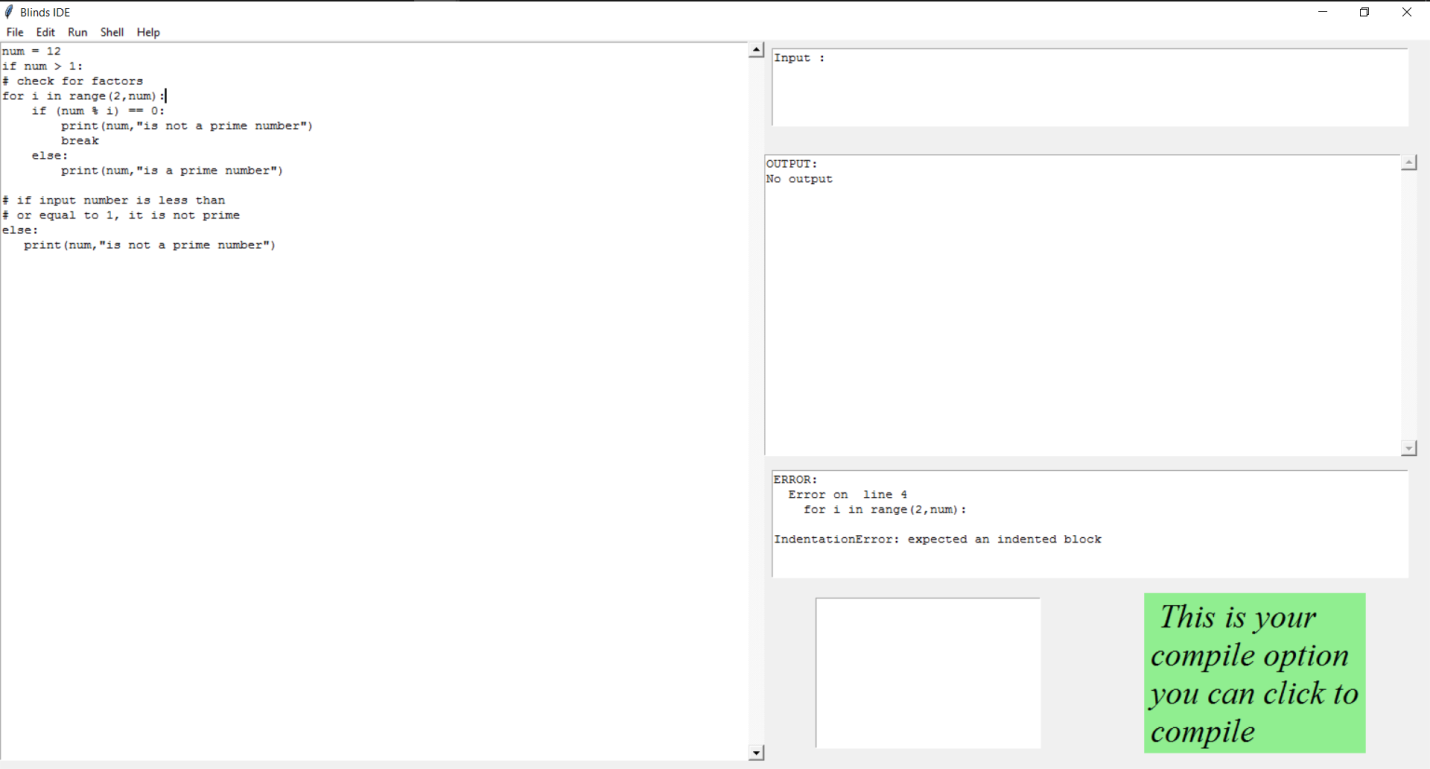
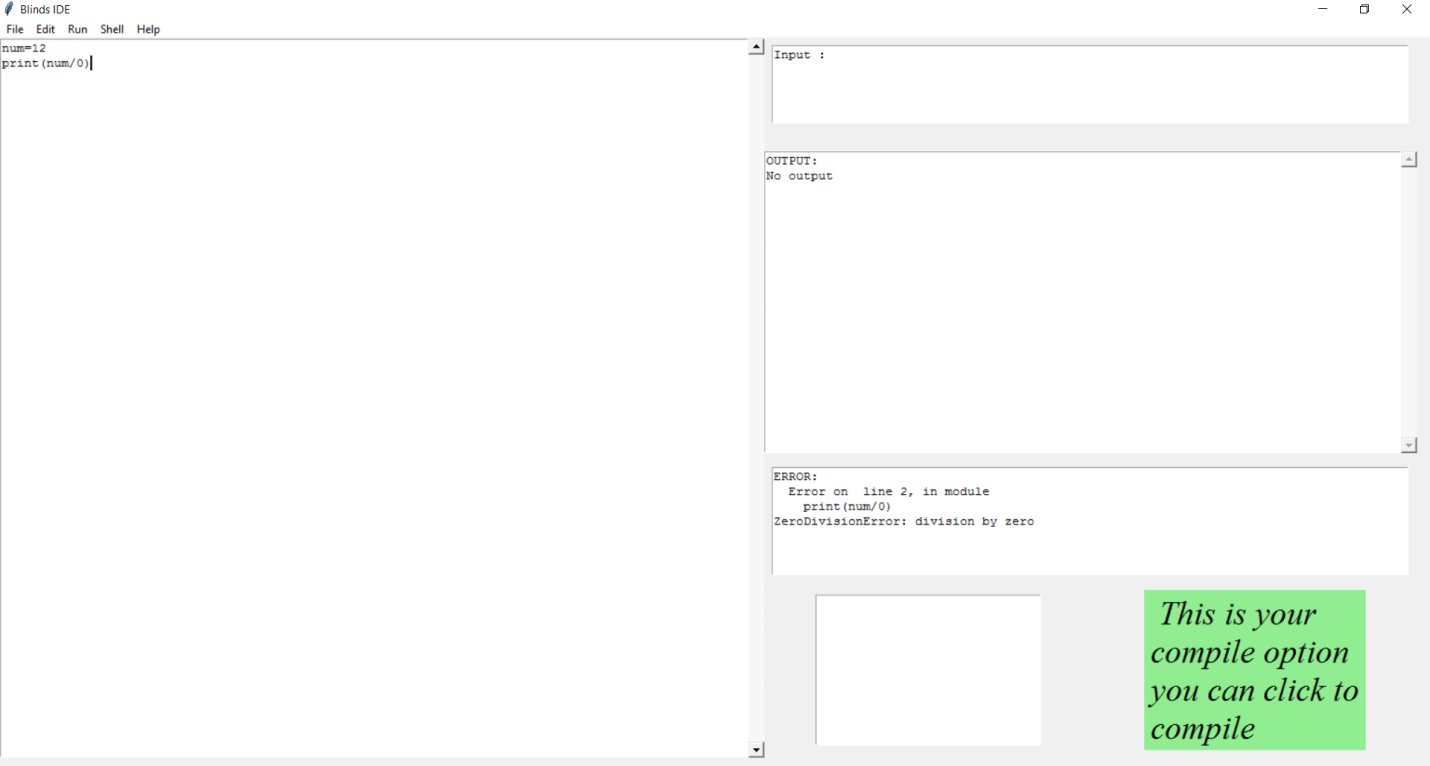


1. Prime Number Program with user input



1. Prime Number program with Name Error:



1. Prime Number program with Syntax Error:
2. Prime Number program with EOF Error:
3. Prime Number program with Import Error:
4. Program with Index Error:
5. Program with TypeError:
6. Program with Indentation Error:
7. Program with ZeroDivisionError:

**CONCLUSION AND FUTURE WORK:**

The project successfully runs for Python programs that consist of loops, conditional statements, data types, user input. With the help of text to speech, display of errors with following line number and moves the cursor to that line in the code(text editor), auto complete, options to open an existing file, save file and save as file.

The project also handles complex codes and compiles it giving correct output and descriptive errors.

Future work can be done with this project in the following fields:

* Syntax correction with ML
* Error correction(more specific)
* Debugging
* Automatic import the required libaries
* Voice assistance with options like :
* Cut
* Copy
* Paste
* Autowrite basic/ mostly used codes(like prime no., factorial,etc.)
* Tutorial for advanced topic

**REFERENCES**

1. Pyttsx3:<https://pypi.org/project/pyttsx3/>
2. Jedi:<https://pypi.org/project/jedi/>
3. Pypnut: <https://pypi.org/project/pynput/>
4. Tkinter: <http://effbot.org/tkinterbook/text.htm>
5. Tkinter Binding Key: <http://effbot.org/tkinterbook/tkinter-events-and-bindings.htm>
6. Repeated Timer: <https://stackoverflow.com/questions/3393612/run-certain-code-every-n-seconds>
7. Error solutions(mostly): <https://stackoverflow.com/>