Acknowledgement

We would like to extend our sincere and heartfelt gratitude to our Computer Science teacher, Mrs.R.Elakkiya who helped us in this endeavor and has been very cooperative throughout. Without her help, cooperation, guidance and encouragement, our project wouldn't be what it has evolved to be.

We also thank Mr.Sandeep Kumar for his tireless efforts and support for making this project possible.

We extend our heartfelt thanks to our faculty for their guidance and constant supervision, as well as, for providing us with the necessary information regarding the project.

We are also thankful to our parents for their cooperation and encouragement.

Last but not least, gratitude to all our friends who helped us complete this project within a tight limited time frame.

Prajan R
Thowthikan PR
Suhaa Rajesh

Overview of Python

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactic constructions than other languages.

Guido Van Rossum conceived python in the late 1980s. It was released in 1991 at centrum wiskunde & information (CWI) in the Netherlands as a successor to the ABC language.

- Interpreted Language: python is processed at runtime by python interpreter.
- Object-Oriented Language: It supports object-oriented features and techniques of programming.
- Interactive Programming Language: Users can interact with the python interpreter directly for writing programs.
- Easy Language: Python is easy to learn, especially for beginners.
- Straightforward Syntax: The formation of python syntax is simple and straightforward, which also makes it popular.
- Easy to Read: Python source-code is clearly defined and visible to the eyes.
- Portable: Python codes can be run on a wide variety of hardware platforms having the same interface.
- Extendable: users can add low level-modules to the python interpreter.
- Scalable: Python provides an improved structure for supporting large programs then shell-scripts

Python and its Applications

Python is used to create web and desktop applications, and some of the most popular web applications like Instagram, YouTube, Spotify all have been developed in python. You can also develop the next big thing by using python.

Table of Contents

Synopsis

This project is a Complex Number Calculator. It enables the user to solve problems involving complex numbers.

The user simply has to enter the real and imaginary values in the respective entry boxes, select the required operator and press the evaluate button for the program to process and display the result.

The software provides a handful of basic arithmetic operators that the user can make use of during evaluation. Once the user hits the evaluate button, the result is produced almost immediately and accuracy is ensured.

The software also provides another interesting feature where the user can enter the equation as a whole instead of having to enter it one by one. By clicking the “Solve by expression” button, the user is taken to another window where this feature can be utilized.

Although the project mainly focuses on complex numbers, calculations involving real numbers can be done as well.

This program provides a clear and comprehensible set of functions that is easy to use and vastly effective for any library.

MySQL Tables

```
mysql> use calc;
Database changed
mysql> show tables;
+----------------+
| Tables_in_calc |
+----------------+
| history        |
+----------------+
1 row in set (0.06 sec)

mysql> select * from history;
+-------------+
| VALUE       |
+-------------+
| 83692j      |
| (43+5436j)  |
| (56+645j)   |
| (6457+567j) |
| 546.0       |
| 4567.0      |
| 657.0       |
| 4356.0      |
+-------------+
8 rows in set (0.00 sec)

mysql>
```

Note: There is one main database [history] where all the results of the performed calculations are stored. As the user keeps performing calculations the results get added to the history and the same can be deleted upon request.

Coding

```
from tkinter import *
import cmath
import mysql.connector
from tkinter import messagebox

mys = mysql.connector.connect(host="localhost", user="root",
password="12345", database="calc")
cursor = mys.cursor()

if mys.is_connected():
   print("Database integrated!")

Main_window = Tk()
Main_window.configure(bg="#202020")
Main_window.iconbitmap("C:/Users/admin/Desktop/proimg/calcicon_
3.ico")

lis_values, lis_operations = [], []
final_value = 0
Error_case = 0
Func_case =  0
Stat_window = None
prev_condition = 0

tuple_font1=("Lucida Bright",10,"bold")
tuple_font=("Lucida Bright",12,"bold")

def Operation_func(operation = None):
   global  Error_case
   imag = imag_part.get()
   real = real_part.get()
   if (not real) and (not imag):
       pass
   else:
      try:
         if not real:
            real = 0
         else:
            real = float(real)
```

```
            if not imag:
              imag = 0
            else:
               imag = float(imag)

         except ValueError:
            label_result.config(text="Invalid input")
            real_part.delete(0, END)
            imag_part.delete(0, END)
            Error_case = 1

         else:
            z = complex(real, imag)
            real_part.delete(0, END)
            imag_part.delete(0, END)
            lis_values.append(z)

            if operation:
               lis_operations.append(operation)

def Evaluate():
   global final_value
   global  Error_case
   final_value = 0
   Operation_func()
   if Error_case == 0:
      while True:
         if len(lis_operations) == 0:
            break

         elif lis_operations[0] == "+":
            lis_values[0] = lis_values[0] + lis_values[1]
            del lis_values[1]

         elif lis_operations[0] == "-":
            lis_values[0] = lis_values[0] - lis_values[1]
            del lis_values[1]

         elif lis_operations[0] == "/":
            try:
               lis_values[0] = lis_values[0] / lis_values[1]
            except ZeroDivisionError:
```

```
                label_result.config(text="Division by zero is not possible")
                Error_case = 1
                break
            else:
                del lis_values[1]

        elif lis_operations[0] == "*":
            lis_values[0] = lis_values[0] * lis_values[1]
            del lis_values[1]

        lis_operations.pop(0)
    if Error_case == 0:
        real_part = round(final_value.real, 3)
        imag_part = round(final_value.imag, 3)
        final_value = complex(real_part, imag_part)
        final_value = lis_values[0]
        if (final_value.imag == 0):
            final_value = final_value.real
        label_result.config(text=final_value)
        st = "insert into history values('%s')" %(str(final_value),)
        cursor.execute(st)
        mys.commit()

def Clear():
    global final_value
    global Func_case
    global  Error_case
    if (Func_case == 1 ):
        label_result.config(text=final_value)
        Func_case = 0

    else:
        global lis_values, lis_operations
        lis_values, lis_operations = [], []
        final_value = 0
        label_result.config(text=final_value)
        Error_case = 0

def power():
    global final_value
    pow_val = float(label_power.get())
    final_value = pow(final_value, pow_val)
```

```
    real_part = round(final_value.real, 3)
    imag_part = round(final_value.imag, 3)
    final_value = complex(real_part, imag_part)
    label_power.delete(0, END)
    if (final_value.imag == 0):
        final_value = final_value.real
    lis_values = [final_value]
    label_result.config(text=lis_values)
    st = "insert into history values('%s')" % (str(final_value),)
    cursor.execute(st)
    mys.commit()

def Argument():
    global Func_case
    phase = round(cmath.phase(final_value), 3)
    if final_value:
        label_result.config(text=("arg(" + str(final_value) + ") = " + str(phase)+
"rad"))
        Func_case = 1

    else:
        return

def Modulus():
    global Func_case
    mod = round(abs(final_value), 3)
    if final_value:
        label_result.config(text=("|" + str(final_value) + "| = " + str(mod)))
        Func_case = 1
    else:
        return

def Conj():
    global final_value
    global lis_values
    if final_value:
        final_value = final_value.conjugate()
        lis_values = [final_value]
        label_result.config(text=final_value)
    else:
        pass
```

```
def His_Prev():
    global final_value
    global lis_values
    global prev_condition
    global ind
    global his_len, his
    if prev_condition == 0:
        ind = -1
        cursor.execute("select * from history")
        his = cursor.fetchall()
        his_len = len(his)
        prev_condition = 1
        if his_len == 0:
            pass
        else:
            val = eval(his[ind][0])
            final_value = val
            lis_values = [final_value]
            label_result.config(text=final_value)

    elif prev_condition == 1:
        if his_len == 0:
            pass
        elif ind == (-his_len):
            pass
        else:
            ind += -1
            val = eval(his[ind][0])
            final_value = val
            lis_values = [final_value]
            label_result.config(text=final_value)

def His_Next():
    global ind, final_value, lis_values
    if prev_condition == 0:
        pass
    elif prev_condition == 1:
        if ind != -1:
            ind += 1
            val = eval(his[ind][0])
            final_value = val
            lis_values = [final_value]
```

```
            label_result.config(text=final_value)


def clear_history():
    answer = messagebox.askyesno("Clear History", "Do you really want to
clear the history?")
    if answer  == True:
        cursor.execute("DELETE FROM history;")
        mys.commit()
    else:
        pass
#solve by stat window 2

def Solve_by_stat():
    global Stat_window

    if not Stat_window:
        global stat_val
        stat_val = 0
        Stat_window = Toplevel()
        Stat_window.title("Solve by Expression")
        Stat_window.configure(bg="#202020")

Stat_window.iconbitmap("C:/Users/admin/Desktop/proimg/calcicon_3.
ico")
    else:
        Stat_window.deiconify()

    def Solve():
        exp = Stat_Entry.get()
        global stat_val
        if exp:
            stat_val = eval(exp)
            label_result_stat.config(text=stat_val)
            Stat_Entry.delete(0, END)
        else:
            pass

    def Pass_to_main():
        global final_value
        global lis_values
        if stat_val:
```

```
            final_value = stat_val
        label_result.config(text=final_value)
        lis_values = [final_value]
        Stat_window.iconify()
        st = "insert into history values('%s')" %(str(final_value),)
        cursor.execute(st)
        mys.commit()

    def Clear_stat():
        global stat_val
        stat_val = 0
        label_result_stat.config(text=stat_val)
        Stat_Entry.delete(0, END)

    def close():
        global Stat_window
        Stat_window.destroy()
        Stat_window = None

    def table_show():
        if table_state.get() == 1:
            table = Table(Stat_window)

    Stat_Entry = Entry(Stat_window, width=45, bd =3 )
    Stat_Entry.grid(row=0, column=0, columnspan=4,padx=5,pady=10)

    button_solve = Button(Stat_window,image=img_win2_solve,
text="Solve",command=Solve, bg="#202020", width=50, height=50,
borderwidth=0)
    button_solve.grid(row=1, column=0)

    button_clear_stat = Button(Stat_window,
text="Clear",image=img_win2_clear, command=Clear_stat, bg="#202020",
width=50, height=50, borderwidth=0)
    button_clear_stat.grid(row=1, column=1, padx=5)

    pass_button = Button(Stat_window, text="Pass value to main window",
command=Pass_to_main, image=img_win2_pass,  bg="#202020",
width=150, height=50, borderwidth=0)
    pass_button.grid(row=1,column=2,columnspan=3)

    label_result_stat = Label(Stat_window,image=img_win2_result,
```

```
width=250, height=50,
text=stat_val,fg="white",bg="#202020",compound="center",font=tuple_fon
t2)
   label_result_stat.grid(row=3, column=0, columnspan=4,
pady=5,padx=5)

   Stat_window.protocol("WM_DELETE_WINDOW", close)

img_win2_solve =
PhotoImage(file="C:/Users/Admin/Desktop/proimg/solve_3.png")
img_win2_clear =
PhotoImage(file="C:/Users/Admin/Desktop/proimg/clear_6.png")
img_win2_pass =
PhotoImage(file="C:/Users/Admin/Desktop/proimg/pass_3.png")
img_win2_result =
PhotoImage(file="C:/Users/Admin/Desktop/proimg/bord_4.png")


Main_window.title('Complex Number Calculator')

label2 = Label(Main_window, text="Real part", width=15, height=2,
font=tuple_font, bg="#202020", foreground="#ECF8F8")
label2.grid(row=0, columnspan=2, column=0)

real_part = Entry(Main_window, width=20, bd=5)
real_part.grid(row=1, columnspan=2, column=0)

label1 = Label(Main_window, text="Imaginary part", width=15, height=2,
font=tuple_font, bg="#202020", foreground="#ECF8F8")
label1.grid(row=0, columnspan=2, column=2)

imag_part = Entry(Main_window, width=20, bd=5)
imag_part.grid(row=1, columnspan=2, column=2)

img_add =
PhotoImage(file="C:/Users/Admin/Desktop/proimg/plus_8.png")

button_add = Button(Main_window, image=img_add, command=lambda
: Operation_func("+"), fg="red", bg="#202020", width=50, height=50,
borderwidth=0)
button_add.grid(row=0, column=4)
```

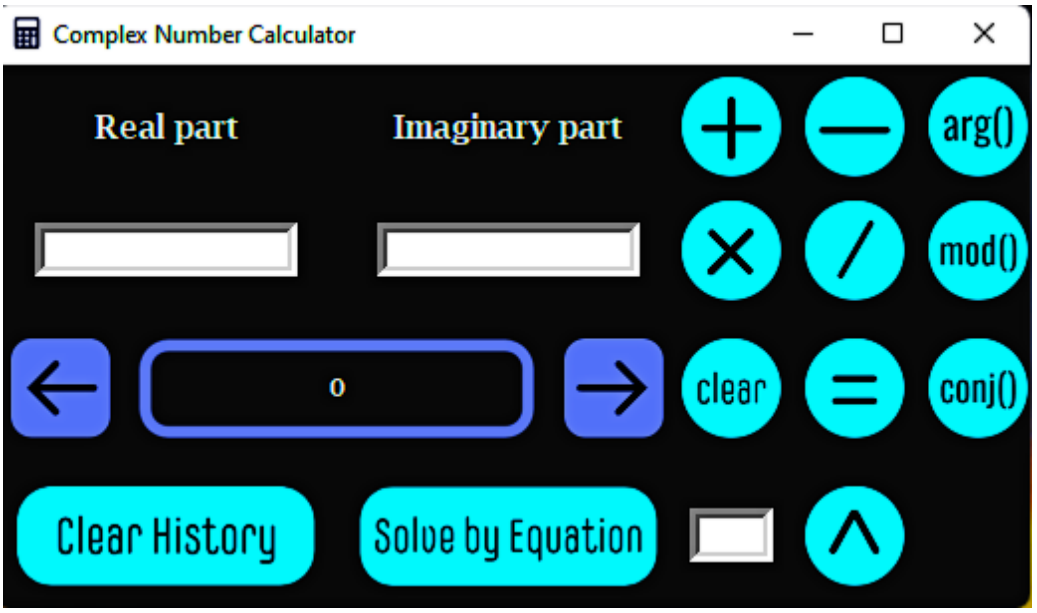
```
img_subtract =
PhotoImage(file="C:/Users/Admin/Desktop/proimg/minus_3.png")

button_subtract = Button(Main_window, image=img_subtract,
command=lambda : Operation_func("-"), fg="red", bg="#202020",
width=50, height=50, borderwidth=0)
button_subtract.grid(row=0, column=5, pady=5,padx=10)

img_multiply =
PhotoImage(file="C:/Users/Admin/Desktop/proimg/multiply_3.png")

button_multiply = Button(Main_window, image=img_multiply,
command=lambda : Operation_func("*"), fg="red", bg="#202020",
width=50, height=50, borderwidth=0)
button_multiply.grid(row=1, column=4, pady=5)

img_divide =
PhotoImage(file="C:/Users/Admin/Desktop/proimg/divide_3.png")

button_divide = Button(Main_window, image=img_divide,
command=lambda : Operation_func("/"), fg="red", bg="#202020",
width=50,height=50, borderwidth=0)
button_divide.grid(row=1, column=5, pady=5)

img_clear=
PhotoImage(file="C:/Users/Admin/Desktop/proimg/clear_3.png")

button_clear = Button(Main_window, text="Clear", command=Clear,
image=img_clear, fg="red", bg="#202020", width=50,height=50,
borderwidth=0)
button_clear.grid(row=2, column=4, pady=5)

img_evaluate =
PhotoImage(file="C:/Users/Admin/Desktop/proimg/eval_1.png")

button_evaluate = Button(Main_window, text="=", command=Evaluate,
image=img_evaluate, fg="red", bg="#202020", width=50,height=50,
borderwidth=0)
button_evaluate.grid(row=2, column=5, pady=5)

img_arg=
PhotoImage(file="C:/Users/Admin/Desktop/proimg/arg_1.png")
```

```
button_arg = Button(Main_window, text="arg()",
command=Argument,image=img_arg, fg="red", bg="#202020",
width=50,height=50, borderwidth=0)
button_arg.grid(row=0, column=6, pady=5)

img_mod=
PhotoImage(file="C:/Users/Admin/Desktop/proimg/mod_1.png")

button_mod = Button(Main_window, text="mod()", command=Modulus,
image=img_mod, fg="red", bg="#202020", width=50,height=50,
borderwidth=0)
button_mod.grid(row=1, column=6, pady=5)

img_conj =
PhotoImage(file="C:/Users/Admin/Desktop/proimg/conj_2.png")

button_conj = Button(Main_window, text="conj()", command=Conj,
image=img_conj, fg="red", bg="#202020", width=50,height=50,
borderwidth=0)
button_conj.grid(row=2, column=6, pady=5,)

img_previous =
PhotoImage(file="C:/Users/Admin/Desktop/proimg/left_arrow.png")

button_previous = Button(Main_window, text="Solve by expression",
image=img_previous, fg="red", bg="#202020", width=50,height=50,
borderwidth=0, command=His_Prev)
button_previous.grid(row = 2, column=0, pady=5,padx=5)

img_next =
PhotoImage(file="C:/Users/Admin/Desktop/proimg/right_arrow.png")

button_next = Button(Main_window, text="Solve by expression",
image=img_next, fg="red", bg="#202020", width=50,height=50,
borderwidth=0, command=His_Next)
button_next.grid(row = 2, column=3, pady=5,padx=5)

img_result =
PhotoImage(file="C:/Users/Admin/Desktop/proimg/bord_2.png")
tuple_font2=("Lucida Bright",10,"bold")
```

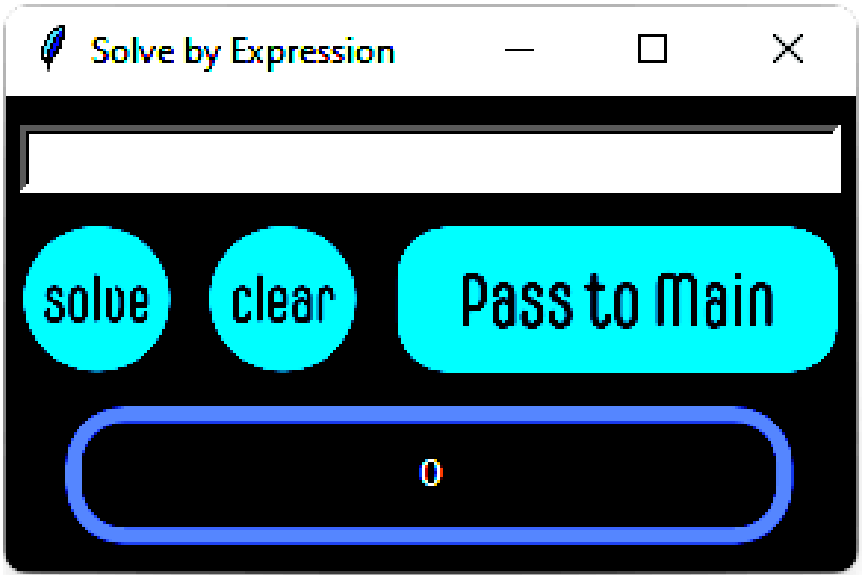
```
label_result = Label(Main_window,image=img_result, text=final_value,
height=50, bg="#202020",fg="white", width=200,
font=tuple_font2,compound='center')
label_result.grid(row=2,column=1, columnspan=2,pady=10, padx=5)

img_solveby =
PhotoImage(file="C:/Users/Admin/Desktop/proimg/solveby_3.png")

button_solvebystat = Button(Main_window, text="Solve by expression",
command=Solve_by_stat,image=img_solveby,  bg="#202020", width=150,
height=50, borderwidth=0)
button_solvebystat.grid(row = 3, column=2, columnspan=2,padx=10,
pady=10)

img_clearhistory =
PhotoImage(file="C:/Users/Admin/Desktop/proimg/clearhistory_1.png
")

button_clear_history = Button(Main_window, text="Clear history",
image=img_clearhistory,  bg="#202020", width=150, height=50,
borderwidth=0)
button_clear_history.grid(row=3, column=0, columnspan=2,padx=10,
pady=10)

img_power =
PhotoImage(file="C:/Users/Admin/Desktop/proimg/exp_4.png")

button_power = Button(Main_window, text="^", command=power,
image=img_power,  bg="#202020", width=50,height=50, borderwidth=0)
button_power.grid(row=3, column=5)

label_power = Entry(Main_window, width=5, bd=5)
label_power.grid(row=3, column=4)

button_clear_history = Button(Main_window, text="Clear history",
command=clear_history)
button_clear_history.grid(row=3, column=0)

Main_window.mainloop()
```

Output Screen

Main Window:



Window - 2:



Conclusion

The Simple Complex Number Calculator is an easy, efficient and quick way to operate and solve complex numbers and related problems of any level of complexity with absolutely no effort.

The software takes care of all the requirements that one might need during the calculation of complex numbers and ensures that the result generated is accurate. The software provides options for various arithmetic operators that the user can make use of just by clicking on the respected buttons during evaluation.

It also provides a feature where a complex equation can be solved directly. All these features and functions are user-friendly and easy to use making it the perfect software for solving complex number equations.

Limitations

1. This system requires knowledgeable people to use the system.
2. Advanced calculations are not possible.
3. The specific “j” variable must be used while using the “Solve by Equation” feature.
4. Doesn’t provide online services for any users.

Software and Hardware Requirements

1. Operating System: Windows 11 pro

2. Processor: AMD Ryzen 3 4300G with Radeon Graphics 3.80 GHz

3. Motherboard: Intel HP G6-1000 DA0R13MB6E0

4. RAM: 4.00 GB

5. System Type: 64-bit Operating System, x64-Based Processor

6. Laptop 14.1 Inch, Keyboard & Mouse

Bibliography & References

1. Computer Science - NCERT (Class XI and XII)
2. Computer Science with Python – Sumita Arora
3. GeeksforGeeks
4. W3Schools
5. Codemy - YouTube
6. Github