**Assignment No :- 4**

**Title :-** Design a User Interface in Python.

**Problem Statement :-** To Design a User Interface in Python.

**Hardware/Software Requirements:-**

Desktop, 1GB RAM, Hard Disk of 200 MB , VSCode , 64 bit Operating System, Windows 11, Programming Language-Python, tkinter Library .

**Theory:-**

1. The Tools Available for Python Website:-

1. PyCharm IDE

1. PyCharm, as an IDE, offers intuitive auto-completion, tips, PEP8 checks, and other code quality enhancement features. You can also rely on it for intelligent automated code refactoring, testing assistance, and code inspections, amongst other features.
2. PyCharm supports some of the most well-known frameworks like Django, Flask, web2py, etc. The IDE has a few built-in database and developer tools for debugging and profiling, each of which seamlessly integrates with Visual Studio Code for further functionality.
3. Its NumPy and Matplotlib library integrations extend its scientific computation features. PyCharm’s remote development, cross-technology interoperability, and integrated testing facilities round off this IDE's overarching feature-set.

2. Jupyter Notebook

1. Jupyter Notebook has everything you'll ever need for interactive development, documentation, and code execution. The Notebook comes equipped with a console-based approach for document authoring.
2. Firstly, it provides you with a browser-based web application component that combines mathematics, rich media, computations, and text explanations. Secondly, it can list inputs and outputs used in calculations, along with rich-media enhanced objects and other related content.
3. This feature helps you edit code with in-browser inspections and commit automatic syntax highlighting. You can also manage indentation and tab completion with Jupyter. This tool uses the Markdown markup language, which is not limited to plain-text for its code commentary feature.
4. With Jupyter’s LaTeX integration, you can insert mathematical notations within markdown cells and render them natively with MathJax.
5. Keras
6. Keras is a high-level neural network API library spearheading Python ML and Deep Learning projects. Keras helps you extend your Python AI neural network programming ecosystem with back-ends like TensorFlow, Microsoft Cognitive Toolkit, PlaidML, Theano, and many others.
7. Keras generates detail-rich and actionable error messages to allow you to troubleshoot your code extensively. You can adapt to the deep learning framework for any number of workflow use cases, given its capability to mesh with various infrastructural configurations—be it a GPU cluster or a whole TPU pod.
8. Integrating Keras can help deep learning specialists reduce their ML cognitive load. The open-source library is widely adopted for its user-friendliness, extensibility, and modular programming approach.

4. Pip Package

* 1. Python's Pip Package is a default package manager for Python and comes bundled with every installation. Pip helps you leverage packages in other local or remote repositories, provided they abide by Python Enhancement Proposal 503.
  2. Pip manages complete package lists with corresponding package version numbers. It logs your package management in a requirements file to help you catalog the same packages for a different desktop or virtual environment.
  3. Pip remains a faithful, readily available package manager, suitable for beginner and advanced package library management. Nonetheless, many developers and teams rely on third-party package management solutions like PyPL as Pip is available with Python 3 and its subsequent versions.

5. Python Anywhere

1. Python Anywhere earned its reputation for its reliability in taking care of end-to-end SDLC needs. This PaaS (Platform as a Service) allows you to develop, run, and host Python programs on the web.
2. With its built-in Python NumPy, SciPy, Mechanize, BeautifulSoup, and PyCrypto libraries, you can commit changes and deploy automated updates directly from your browser.
3. Python Anywhere helps you deploy your codebase rapidly on its AWS EC2-based servers. Decentralized hosting enables Python-based remote research, learning, and development.

6. Scikit-Learn

1. Scikit-Learn’s open-source library is ready and waiting to assist you with your Python-scripted ML goals. Scikit’s predictive analysis tools expedite object classification, assist with continuous-valued attribute prediction and regression, clustering of support-vector machines, gradient boosting, random forests, and automatic grouping of similar objects.
2. Scikit has advanced ML tools like dimensionality reduction and model selection for advanced ML analysis. It’s built on the Matplotlib, NumPy, and SciPy libraries, each of which is ideal for any bespoke ML context.

7. Sphinx

1. Sphinx is another Python documentation that can be seen as an apt alternative to Jupyter Notebook. It generates highly articulate Python documentation in HTML, LaTeX, ePub, and other formats.
2. Sphinx can further help you set up automatic, language-specific indices and testing for code snippets. Sphinx comes equipped with several third-party extensions besides its built-in docstrings libraries.

8. Selenium

1. Selenium is the go-to tool for agile testing in Python. The tool helps you conduct manual, automated, and cross-browser testing for Python-based web applications.
2. With Selenium, you can even convert automated tests into manual function tests, depending on your needs. You can write purpose-driven, custom test scripts or cases for any software.
3. Its wide range of customization makes it easy to implement Selenium testing functions across tailored software builds.
4. The entire comprehensive Selenium testing suite is open-source. If your Python program runs on multiple platforms, you can use Selenium for cross-platform testing. Selenium meets Maven, Jenkins, and Docker testing requirements.

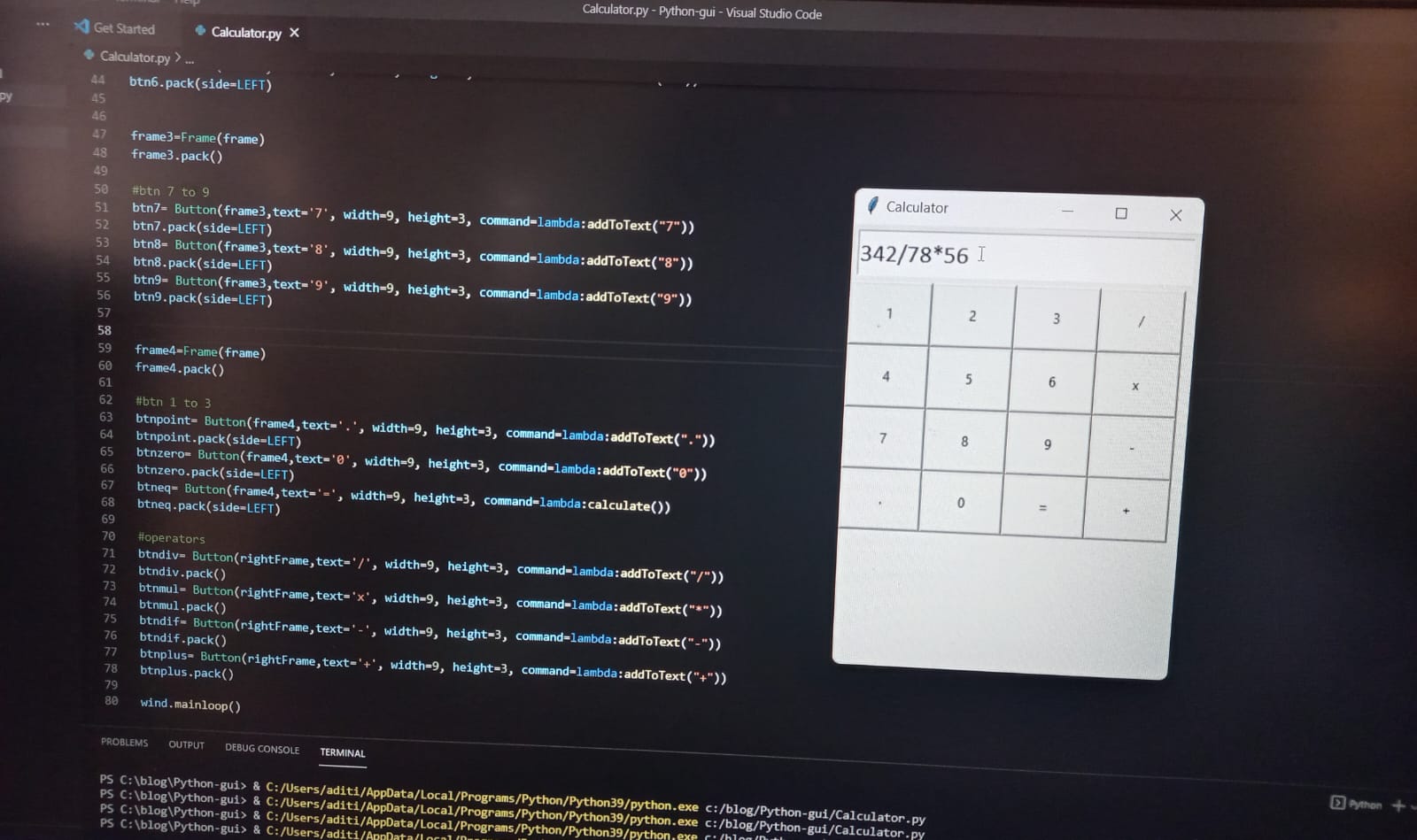
9. Sublime Text

1. Since its inception, Sublime Text has been a feature-rich, lightweight IDE that has taken the development community by storm. Its highly context-aware auto-completion feature and syntax definition engine help you rapidly hone your Python skills.
2. Sublime Text renders many beginner woes insignificant with its extensive Python API documentation. The adaptive UI offers a highly aesthetic programming interface that intuitively switches to any platform's native color scheme.
3. The latest version of Sublime Text allows you to leverage the updated Python API, but you can still enjoy backward compatibility with packages for older Python versions.
4. If you use Raspberry Pi, Apple Silicon, or Linux Arm64, you can use Sublime Text for cross-platform Python support. Managing projects within Sublime Text is effortless, with its multi-tab selection and navigational properties.

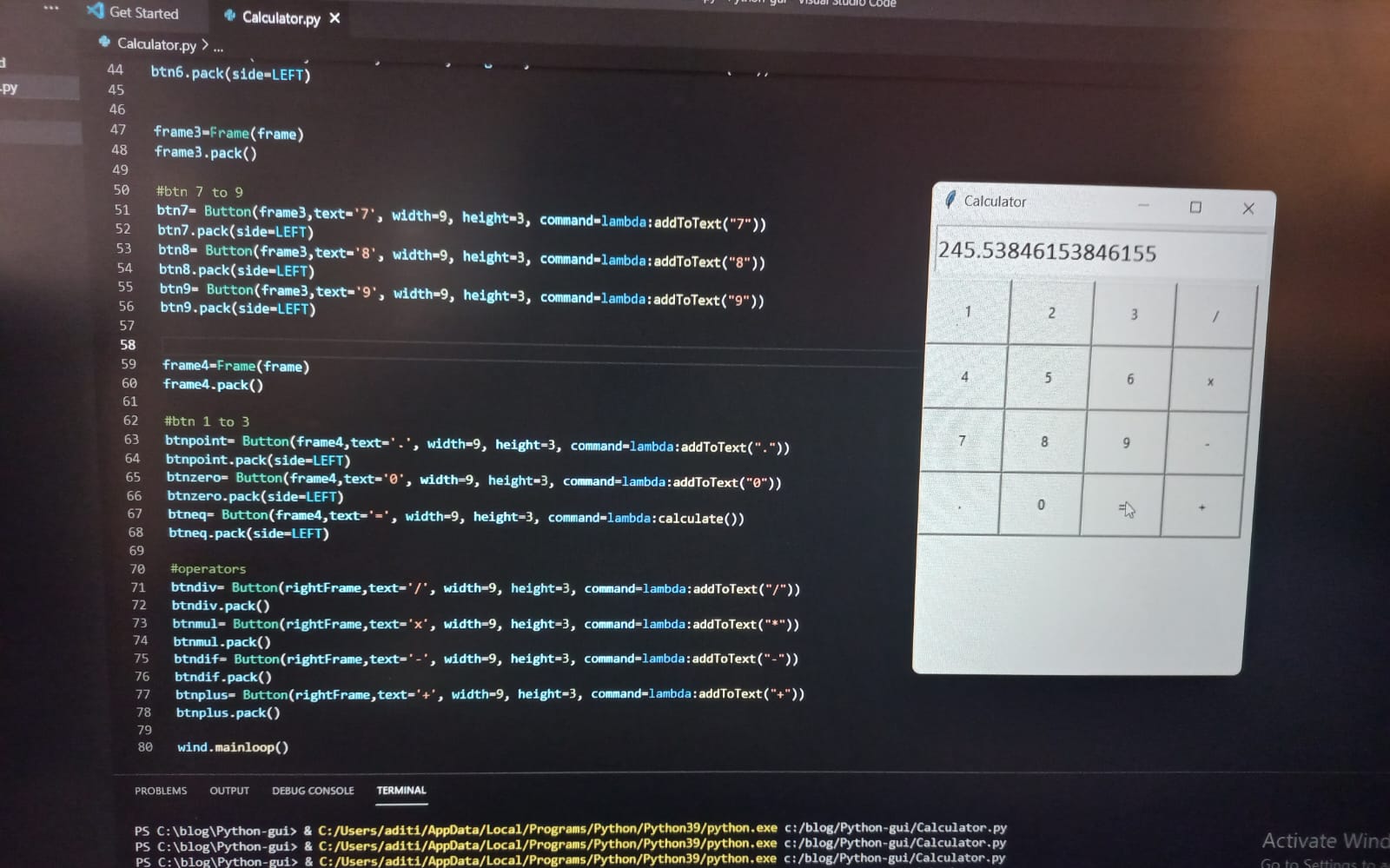
10. BeautifulSoup

1. BeautifulSoup is a web scraping framework that comes in handy when accessing, managing, or manipulating Python web app data. The tool works in synergy with any parser, supporting a wide range of Python idioms for operating on the parse tree.
2. Integrate your APIs with BeautifulSoup to effortlessly scrape app and website data. Additionally, use BeautifulSoup to obtain data in CSV spreadsheets, HTML, XML, and JS data from Python sites.
3. Calculator’s UI Using Python

While development of Calculators UI on VSCode using Python , tkinter library is needed to import.



1. Result of 342/78\*56 on Calculators UI



1. Code of UI:-

from tkinter import \*

wind= Tk() wind.title("Calculator") wind.geometry("300x400")

text= Entry(wind, font=("calibri", 16)) text.pack(fill=X, padx=5, pady=5, ipady=5)

def addToText(n): text.insert(END,n) def calculate():

result= eval(text.get()) text.delete(0,END) text.insert(0,result)

frame=Frame(wind) frame.pack(side=TOP,anchor=NW)

rightFrame=Frame(frame) rightFrame.pack(side=RIGHT)

frame1=Frame(frame) frame1.pack()

#btn 1 to 3 btn1= Button(frame1,text='1', width=9, height=3, command=lambda:addToText("1")) btn1.pack(side=LEFT) btn2= Button(frame1,text='2', width=9, height=3, command=lambda:addToText("2")) btn2.pack(side=LEFT) btn3= Button(frame1,text='3', width=9, height=3, command=lambda:addToText("3")) btn3.pack(side=LEFT)

frame2=Frame(frame) frame2.pack()

#btn 4 to 6 btn4= Button(frame2,text='4', width=9, height=3, command=lambda:addToText("4")) btn4.pack(side=LEFT) btn5= Button(frame2,text='5', width=9, height=3, command=lambda:addToText("5")) btn5.pack(side=LEFT) btn6= Button(frame2,text='6', width=9, height=3, command=lambda:addToText("6")) btn6.pack(side=LEFT)

frame3=Frame(frame) frame3.pack()

#btn 7 to 9 btn7= Button(frame3,text='7', width=9, height=3, command=lambda:addToText("7")) btn7.pack(side=LEFT) btn8= Button(frame3,text='8', width=9, height=3, command=lambda:addToText("8")) btn8.pack(side=LEFT) btn9= Button(frame3,text='9', width=9, height=3, command=lambda:addToText("9")) btn9.pack(side=LEFT)

frame4=Frame(frame) frame4.pack()

#btn 1 to 3

btnpoint= Button(frame4,text='.', width=9, height=3, command=lambda:addToText(".")) btnpoint.pack(side=LEFT) btnzero= Button(frame4,text='0', width=9, height=3, command=lambda:addToText("0")) btnzero.pack(side=LEFT) btneq= Button(frame4,text='=', width=9, height=3, command=lambda:calculate()) btneq.pack(side=LEFT)

#operators btndiv= Button(rightFrame,text='/', width=9, height=3, command=lambda:addToText("/")) btndiv.pack() btnmul= Button(rightFrame,text='x', width=9, height=3, command=lambda:addToText("\*")) btnmul.pack() btndif= Button(rightFrame,text='-', width=9, height=3, command=lambda:addToText("-")) btndif.pack() btnplus= Button(rightFrame,text='+', width=9, height=3, command=lambda:addToText("+")) btnplus.pack()

wind.mainloop()

**Conclusion:-** Hence, We developed of UI using Python.