1.write a calculator program in python ..?

```
# pip install tkinter

import tkinter as tk

import tkinter.messagebox

from tkinter.constants import SUNKEN

window = tk.Tk()

window.title('Calculator-GeeksForGeeks')

frame = tk.Frame(master=window, bg="skyblue", padx=10)

frame.pack()

entry = tk.Entry(master=frame, relief=SUNKEN, borderwidth=3, width=30)

entry.grid(row=0, column=0, columnspan=3, ipady=2, pady=2)

def myclick(number):

        entry.insert(tk.END, number)

def equal():

        try:

                y = str(eval(entry.get()))

                entry.delete(0, tk.END)

                entry.insert(0, y)

        except:

                tkinter.messagebox.showinfo("Error", "Syntax Error")

def clear():

        entry.delete(0, tk.END)


button_1 = tk.Button(master=frame, text='1', padx=15,

                                        pady=5, width=3, command=lambda: myclick(1))

button_1.grid(row=1, column=0, pady=2)

button_2 = tk.Button(master=frame, text='2', padx=15,
                                               pady=5, width=3, command=lambda: myclick(2))
button_2.grid(row=1, column=1, pady=2)
button_3 = tk.Button(master=frame, text='3', padx=15,
                                               pady=5, width=3, command=lambda: myclick(3))
button_3.grid(row=1, column=2, pady=2)
button_4 = tk.Button(master=frame, text='4', padx=15,
                                               pady=5, width=3, command=lambda: myclick(4))
button_4.grid(row=2, column=0, pady=2)
button_5 = tk.Button(master=frame, text='5', padx=15,
                                               pady=5, width=3, command=lambda: myclick(5))
button_5.grid(row=2, column=1, pady=2)
button_6 = tk.Button(master=frame, text='6', padx=15,
                                               pady=5, width=3, command=lambda: myclick(6))
button_6.grid(row=2, column=2, pady=2)
button_7 = tk.Button(master=frame, text='7', padx=15,
                                               pady=5, width=3, command=lambda: myclick(7))
button_7.grid(row=3, column=0, pady=2)
button_8 = tk.Button(master=frame, text='8', padx=15,
                                               pady=5, width=3, command=lambda: myclick(8))
button_8.grid(row=3, column=1, pady=2)
button_9 = tk.Button(master=frame, text='9', padx=15,
                                               pady=5, width=3, command=lambda: myclick(9))
button_9.grid(row=3, column=2, pady=2)
button_0 = tk.Button(master=frame, text='0', padx=15,
                                               pady=5, width=3, command=lambda: myclick(0))
button_0.grid(row=4, column=1, pady=2)

button_add = tk.Button(master=frame, text="+", padx=15,
                                        pady=5, width=3, command=lambda: myclick('+'))
button_add.grid(row=5, column=0, pady=2)


button_subtract = tk.Button(
        master=frame, text="-", padx=15, pady=5, width=3, command=lambda: myclick('-'))
button_subtract.grid(row=5, column=1, pady=2)
button_multiply = tk.Button(
        master=frame, text="*", padx=15, pady=5, width=3, command=lambda: myclick('*'))
button_multiply.grid(row=5, column=2, pady=2)


button_div = tk.Button(master=frame, text="/", padx=15,
                                        pady=5, width=3, command=lambda: myclick('/'))
button_div.grid(row=6, column=0, pady=2)


button_clear = tk.Button(master=frame, text="clear",
        padx=15, pady=5, width=12, command=clear)
button_clear.grid(row=6, column=1, columnspan=2, pady=2)


button_equal = tk.Button(master=frame, text="=", padx=15,
                                                pady=5, width=9, command=equal)
button_equal.grid(row=7, column=0, columnspan=3, pady=2)


window.mainloop()
```

2. write a program to concatenate.reverse and slice a string ?

Reversing Strings Through Slicing

Slicing is a useful technique that allows you to extract items from a given sequence using different combinations of **integer indices** known as offsets. When it comes to slicing strings, these offsets define the index of the first character in the slicing, the index of the character that stops the slicing, and a value that defines how many characters you want to jump through in each iteration.

To slice a string, you can use the following syntax:

```
a_string[start:stop:step]
```

Your offsets are start, stop, and step. This expression extracts all the characters from start to stop − 1 by step. You're going to look more deeply at what all this means in just a moment.

All the offsets are optional, and they have the following default values:

Offset	Default Value
start	0
stop	len(a_string)
step	1

Here, start represents the index of the first character in the slice, while stop holds the index that stops the slicing operation. The third offset, step, allows you to decide how many characters the slicing will jump through on each iteration.

The step offset allows you to fine-tune how you extract desired characters from a string while skipping others:

```
>>> letters = "AaBbCcDd"

>>> # Get all characters relying on default offsets
>>> letters[::]
'AaBbCcDd'
>>> letters[:]
'AaBbCcDd'

>>> # Get every other character from 0 to the end
>>> letters[::2]
'ABCD'
```

```
>>> # Get every other character from 1 to the end
>>> letters[1::2]
'abcd'
```

Here, you first slice letters without providing explicit offset values to get a full copy of the original string. To this end, you can also use a slicing that omits the second colon (:). With step equal to 2, the slicing gets every other character from the target string. You can play around with different offsets to get a better sense of how slicing works.

Why are slicing and this third offset relevant to reversing strings in Python? The answer lies in how step works with negative values. If you provide a negative value to step, then the slicing runs backward, meaning from right to left.

For example, if you set step equal to -1, then you can build a slice that retrieves all the characters in reverse order:

```
>>> letters = "ABCDEF"

>>> letters[::-1]
'FEDCBA'

>>> letters
'ABCDEF'
```

This slicing returns all the characters from the right end of the string, where the index is equal to len(letters) - 1, back to the left end of the string, where the index is 0. When you use this trick, you get a copy of the original string in reverse order without affecting the original content of letters.

This function takes three arguments, with the same meaning of the offsets in the slicing operator, and returns a slice object representing the set of indices that result from calling `range(start, stop, step)`.

You can use `slice()` to emulate the slicing `[::-1]` and reverse your strings quickly. Go ahead and run the following call to `slice()` inside square brackets:

```
>>> letters = "ABCDEF"

>>> letters[slice(None, None, -1)]
```

3. why is python a popular prpgramming language ..?

1) Easy to Learn and Use

Python language is incredibly easy to use and learn for new beginners and newcomers. The python language is one of the most accessible programming languages available because it has simplified syntax and not complicated, which gives more emphasis on natural language. Due to its ease of learning and usage, python codes can be easily written and executed much faster than other programming languages.

When **Guido van Rossum** was creating python in the 1980s, he made sure to design it to be a general-purpose language. One of the main reasons for the popularity of python would be its simplicity in syntax so that it could be easily read and understood even by amateur developers also.

Must read: **Free excel courses!**

One can also quickly experiment by changing the code base of python because it is an interpreted language which makes it even more popular among all kinds of developers.

2) Mature and Supportive Python Community

Python was created more than 30 years ago, which is a lot of time for any community of programming language to grow and mature adequately to support developers ranging from beginner to expert levels. There are plenty of documentation, guides and Video Tutorials for Python language are available that learner and developer of any skill level or ages can use and receive the support required to enhance their knowledge in python programming language.

Many students get introduced to computer science only through Python language, which is the same language used for in-depth research projects. The community always guides learners who learn data science.

If any programming language lacks developer support or documentation, then they don't grow much. But python has no such kind of problems because it has been here for a very long time. The python developer community is one of the most incredibly active programming language communities.

This means that if somebody has an issue with python language, they can get instant support from developers of all levels ranging from beginner to expert in the community. Getting help on time plays a vital role in the development of the project, which otherwise might cause delays.

Our learners also read – Learn python free courses!

3) Support from Renowned Corporate Sponsors

Programming languages grows faster when a corporate sponsor backs it. For example, PHP is backed by Facebook, Java by Oracle and Sun, Visual Basic & C# by Microsoft. Python Programming language is heavily backed by Facebook, Amazon Web Services, and especially Google.
Google adopted python language way back in 2006 and have used it for many applications and platforms since then. Lots of Institutional effort and money have been devoted to the training and success of the python language by Google. They have even created a dedicated portal only for python. The list of support tools and documentation keeps on growing for python language in the developers' world.

4) Hundreds of Python Libraries and Frameworks

Due to its corporate sponsorship and big supportive community of python, python has excellent libraries that you can use to select and save your time and effort on the initial cycle of development. There are also lots of cloud media services that offer cross-platform support through library-like tools, which can be extremely beneficial.
Libraries with specific focus are also available like nltk for natural language processing or scikit-learn for machine learning applications.

There are many frameworks and libraries are available for python language, such as:

- matplotib for plotting charts and graphs
- SciPy for engineering applications, science, and mathematics
- BeautifulSoup for HTML parsing and XML
- NumPy for scientific computing
- Django for server-side web development

5) Versatility, Efficiency, Reliability, and Speed

Ask any python developer, and they will wholeheartedly agree that the python language is efficient, reliable, and much faster than most modern languages. Python can be used in **nearly any kind of environment**, and one will not face any kind of performance loss issue irrespective of the platform one is working.
One more best thing about versatility of python language is that it can be used in many varieties of environments such as mobile applications, desktop applications, web development, hardware programming, and many more. The versatility of python makes it more attractive to use due to its high number of applications.

6) Big data, Machine Learning and Cloud Computing

Cloud Computing, Machine Learning, and Big Data are some of the hottest trends in the computer science world right now, which helps lots of organizations to transform and improve their processes and workflows.
Python language is the second most popular used tool after R language for data science and analytics. Lots of many data processing workloads in the organization are powered by python language only. Most of the research and development takes place in python language due to its many applications, including ease of analyzing and organizing the usable data.

4.what are the other framework that can be used with python..?

Python is the go-to programming language for Data Science. Besides its inherent simplicity, what makes Python most appealing is that it is backed by a wide range of Python frameworks.
Python frameworks offer a well-defined structure for app development. Since they can automate the implementation of some standard solutions, they not only reduce the development time significantly but also allow Developers to focus on the core application logic instead of routine elements. Long story short – they make the job of Developers much easier and make Python one of the best programming language

1. Full-Stack Framework

A full-stack framework, also known as enterprise framework, is the one-stop solution for all development needs. These have built-in libraries configured to work seamlessly together. They support the development of databases, frontend interfaces, and backend services

2. Microframework

Microframeworks are lightweight, minimalistic web application frameworks that have limited functionalities and features. Usually, microframeworks offer only those components that are required for building an application. They lack many additional functionalities and features like database abstraction layer, form validation, web template engine, authentication functionality, authorization, input validation, and input sanitation.

3. Asynchronous Framework

The asynchronous framework is the latest to join the Python framework bandwagon. It is a unique microframework that lets Developers handle and manage large sets of concurrent connections. These frameworks feed on Python's Asyncio library.

Things to consider while choosing a Python framework

Python has consequently ranked as the number one programming language loved by Data Scientists and Developers alike. There are multiple reasons which make Python the best programming language for Data Science. So, rest assured, there's no shortage of Python frameworks. However, the abundance of Python frameworks might also become overwhelming while choosing the right framework for your application.

Here are two things you should consider while deciding on a Python framework:

- First, evaluate the size and complexity of your project. If you have to develop a large system packed with features and requirements, a full-stack framework will be the right choice. On the contrary, if the project at hand is small and straightforward, you can work with a microframework

6. Full Form Of WSGI..?

WSGI Servers

A Web Server Gateway Interface (WSGI) server implements the web server side of the WSGI interface for running Python web applications.

.