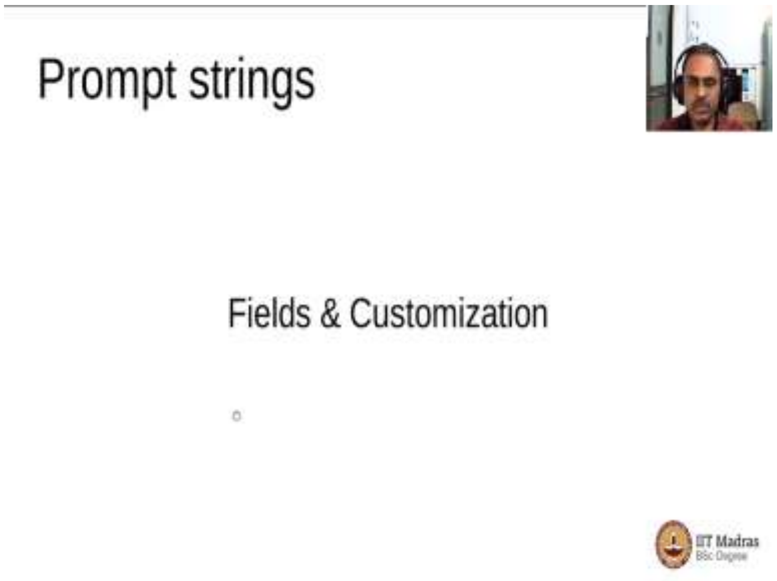
System Commands
Professor. Gandham Phanikumar
Metallurgical and Materials Engineering
Indian Institute of Technology, Madras

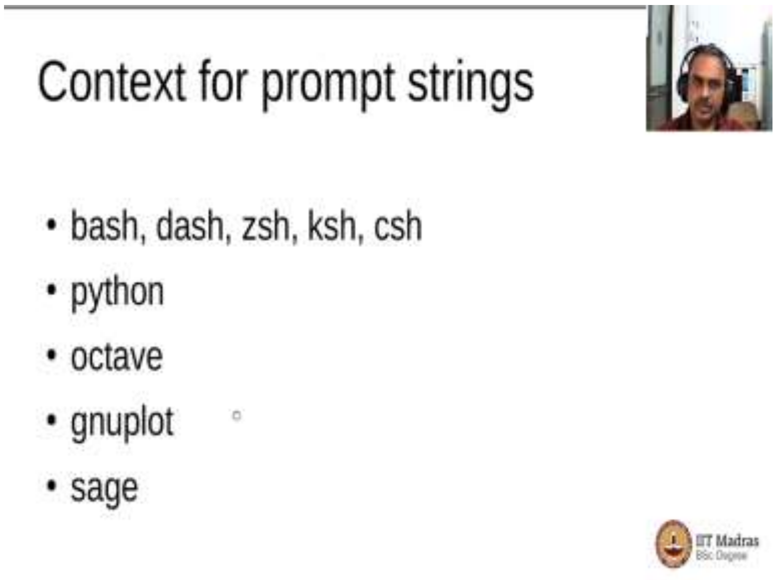
Prompt Strings

(Refer Slide Time: 0:15)



Prompt strings

Fields & Customization



Context for prompt strings

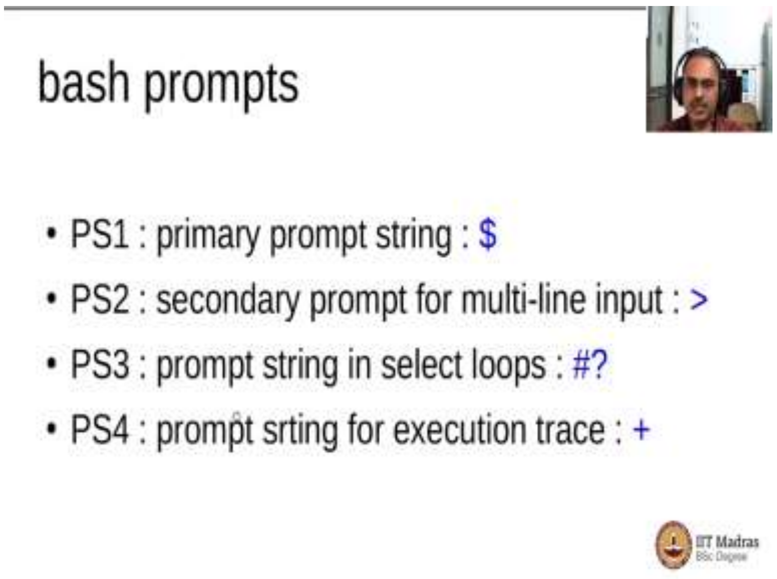
- bash, dash, zsh, ksh, csh
- python
- octave
- gnuplot
- sage

So, here we come to the discussion about the Prompt Strings. And what are the fields that you can insert as a part of the prompt string and customizing for your needs. So, the context for prompt string has come already when we were using the bash shell So, there are lot of shells in the Linux operating system like dash shell, z shell, k shell, and c shell apart from the bash. E Each of these will have a default configuration for the prompt.

And by recognizing the nature of the prompt, you can actually go on to see if there is any configuration change that you want to do. There are also other command line utilities such as Python, which can be run on the command line to try out certain pieces of code before you put them together in a Jupyter notebook or in a Python script.

So, the prompt for the Python or octave, which is a MATLAB compatible, numerical package and language on Linux, and then gnuplot, which is a very powerful plotting tool available on Linux, and sage math, which is a symbolic computing package, which is comparable and perhaps better than tools like Mathematica or Maple. So, all these languages will have a command line environment where the prompt can be different. And one can also customize those.

(Refer Slide Time: 1:50)



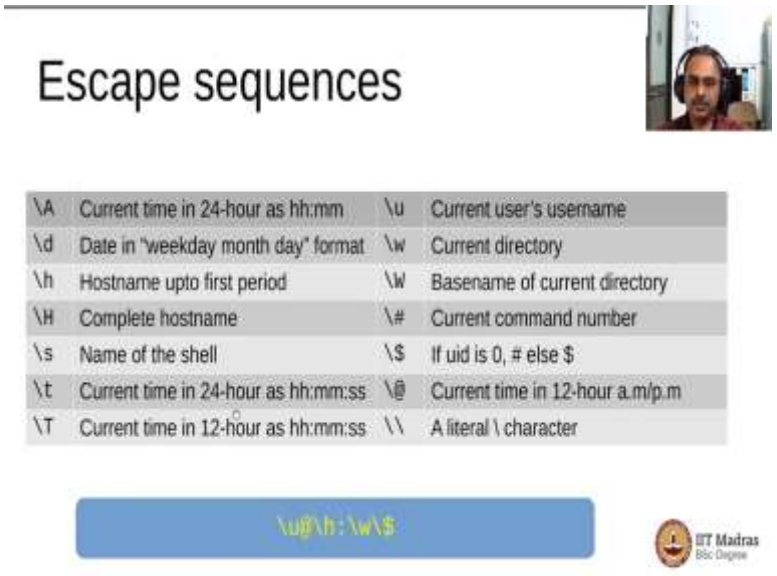
bash prompts

- PS1 : primary prompt string : $
- PS2 : secondary prompt for multi-line input : >
- PS3 : prompt string in select loops : #?
- PS4 : prompt srting for execution trace : +

There are four bash prompts that are configured. And what you see on the screen when we open a bash shell is the primary prompt, which usually ends with dollar. And then there is a secondary prompt, which is shown when we have a multi-line input, we did not close the codes, and then we need to close it by supplying some more input. So, in such situations, the secondary prompt will be shown, and that is usually the greater than symbol.

The third prompt string is used when we run a bash command in a select loop, which is part of the shell scripts that we have explored in the past. And the fourth prompt string is shown when every command that is executed by the bash is to be displayed on the screen, because we have used the option set minus x which is to basically make the execution trace so, that it helps us in debugging the script.

(Refer Slide Time: 2:51)



Escape sequences

\A	Current time in 24-hour as hh:mm	\u	Current user's username
\d	Date in "weekday month day" format	\w	Current directory
\h	Hostname upto first period	\W	Basename of current directory
\H	Complete hostname	\#	Current command number
\s	Name of the shell	\$	If uid is 0, # else $
\t	Current time in 24-hour as hh:mm:ss	\@	Current time in 12-hour a.m/p.m
\T	Current time in 12-hour as hh:mm:ss	\\	A literal \ character

```
\u@\h:\w\$
```

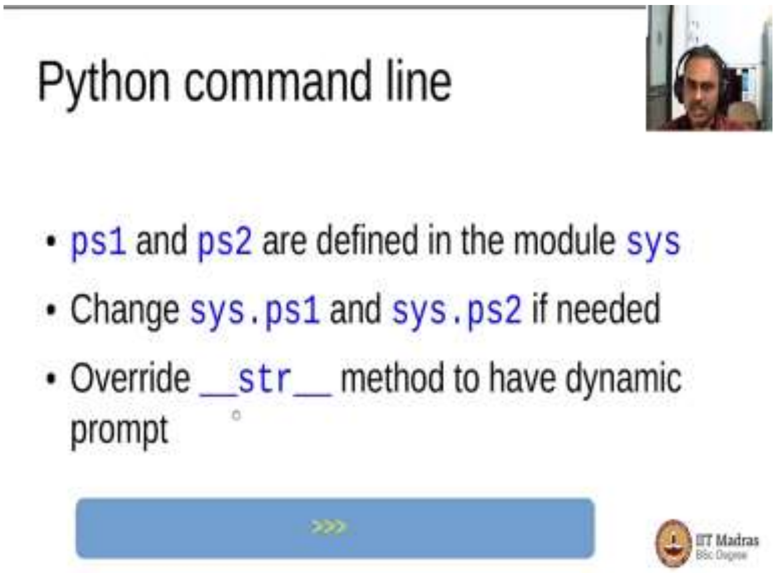
IIT Madras
BSc Degree

Now there are certain escape sequences that you could use while configuring the values for these for prompt strings. So, that what is displayed on the screen can be changed. So, you could use the time in different formats the current time or current time in 24 hour format with the seconds also displayed or current time in 12 hour format with the second display. You could have the name of the shell displayed the hostname either up to the first period, or the complete hostname.

You could also have the date displayed if you wish the current user's user name, which is also default in the bash shell and Ubuntu that is displayed. You can have the current directory that is displayed which is also part of the default for the open to bash. And then you can also have the command line number so, that you can keep track on how many commands you are running. And you can go back to the history also.

And you can also use the literal characters to escape the sequence by using a backslash. So, the default value of the ps1 is what is given at the bottom slash u at slash h colon slash w slash dollar. So, the meaning would be the user name at the machine name colon, the current directory followed by dollar symbol if the user is not super user.

(Refer Slide Time: 4:22)



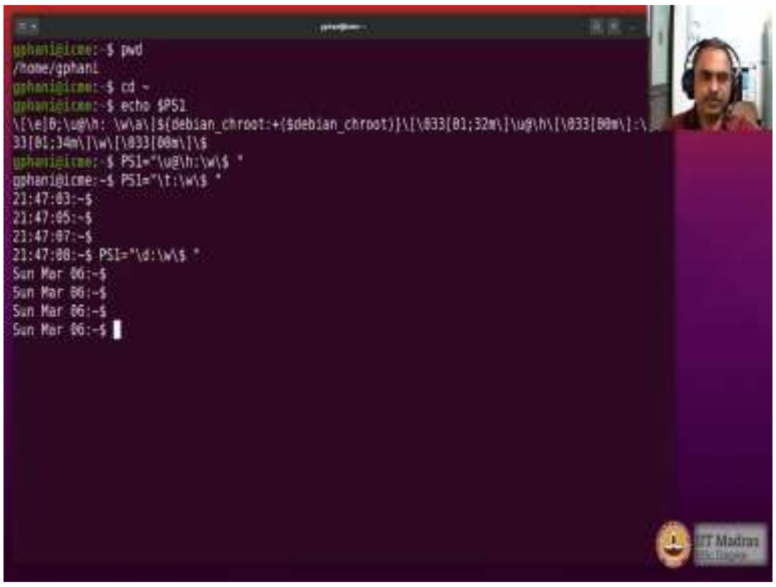
Python command line

- ps1 and ps2 are defined in the module sys
- Change sys.ps1 and sys.ps2 if needed
- Override __str__ method to have dynamic prompt

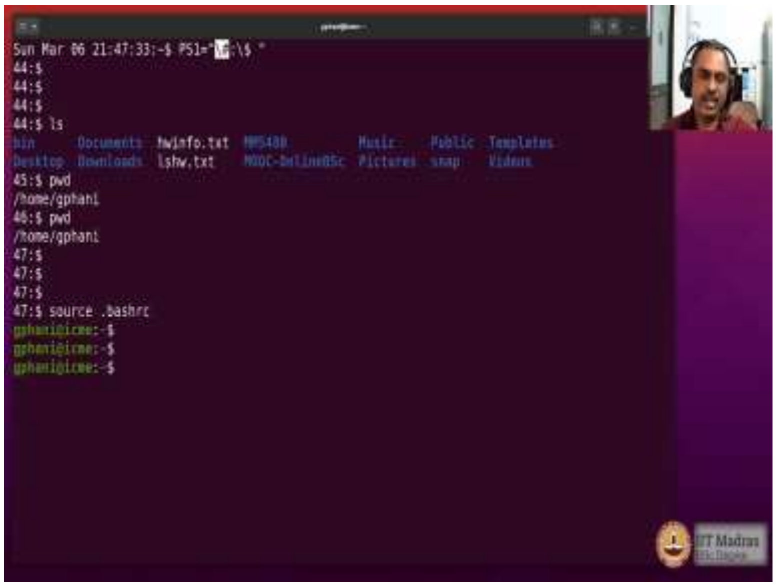
```
>>>
```

In the case of Python command line, the ps1 and ps2 are defined in the module sys and they can be modified as a part of your code. And by overriding the string function, you could actually also make it dynamic. And by default, the Python command line has the command prompt which is three greater than symbols. Now, let us look at a demo of this customization in the shell.

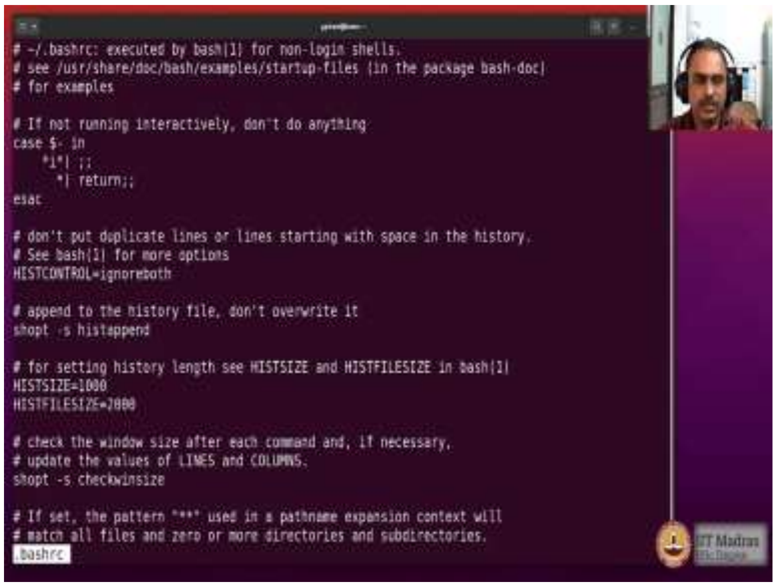
(Refer Slide Time: 4:49)



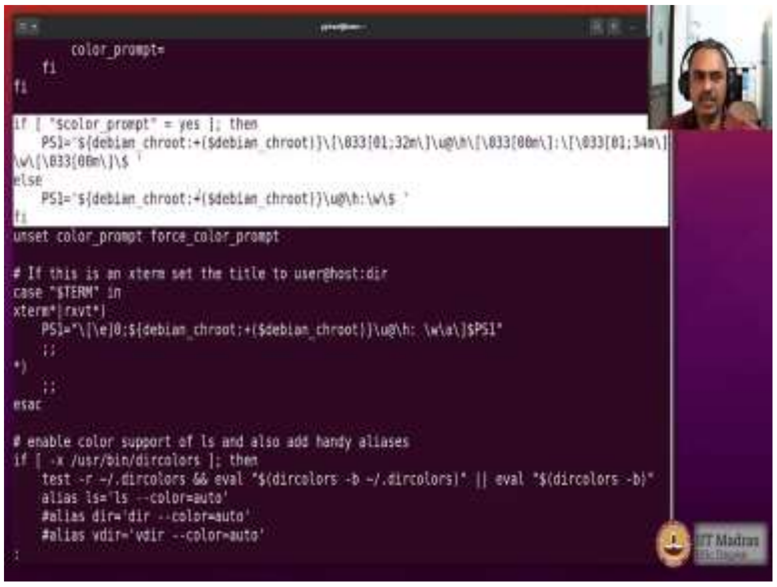
```
gphani@icme:~$ pwd
/home/gphani
gphani@icme:~$ cd ~
gphani@icme:~$ echo $PS1
\[\e]0;\u@\h: \w\a\]${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\
33[01;34m\]\w\[\033[00m\]\$
gphani@icme:~$ PS1="\u@\h:\w\$ "
gphani@icme:~$ PS1="\t:\w\$ "
21:47:03:~$
21:47:05:~$
21:47:07:~$
21:47:08:~$ PS1="\d:\w\$ "
Sun Mar 06:~$
Sun Mar 06:~$
Sun Mar 06:~$
Sun Mar 06:~$
```



```
Sun Mar 06 21:47:33:~$ PS1="\#:\$ "
44:$
44:$
44:$
44:$ ls
bin      Documents  hwinfo.txt  MM5480         Music     Public  Templates
Desktop  Downloads  lshw.txt    MOOC-OnlineBSc Pictures  snap    Videos
45:$ pwd
/home/gphani
46:$ pwd
/home/gphani
47:$
47:$
47:$
47:$ source .bashrc
gphani@icme:~$
gphani@icme:~$
gphani@icme:~$
```



```
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

# If not running interactively, don't do anything
case $- in
    *i*) ;;
      *) return;;
esac

# don't put duplicate lines or lines starting with space in the history.
# See bash(1) for more options
HISTCONTROL=ignoreboth

# append to the history file, don't overwrite it
shopt -s histappend

# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
HISTSIZE=1000
HISTFILESIZE=2000

# check the window size after each command and, if necessary,
# update the values of LINES and COLUMNS.
shopt -s checkwinsize

# If set, the pattern "**" used in a pathname expansion context will
# match all files and zero or more directories and subdirectories.
.bashrc
```



```
        color_prompt=
    fi
fi

if [ "$color_prompt" = yes ]; then
    PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]
\w\[\033[00m\]\$ '
else
    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
fi
unset color_prompt force_color_prompt

# If this is an xterm set the title to user@host:dir
case "$TERM" in
xterm*|rxvt*)
    PS1="\[\e]0;${debian_chroot:+($debian_chroot)}\u@\h: \w\a\]$PS1"
    ;;
*)
    ;;
esac

# enable color support of ls and also add handy aliases
if [ -x /usr/bin/dircolors ]; then
    test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" || eval "$(dircolors -b)"
    alias ls='ls --color=auto'
    #alias dir='dir --color=auto'
    #alias vdir='vdir --color=auto'
:
```

So, the first prompt sting what we have here already you can make out that it is the user name at the host name which is the machine name ICME colon, and then the tilde is a short form for the home directory followed by dollar. So, where I am, So, I am at the home directory and CD tilde will bring me to the home directory. So, tilde is a short form or home directory. So, the prompt string is already have in the home directory here.

So, let us see what is the value that is given per ps1. And now you see this fairly long string is there. Some of these which are coming here are actually for the color display, So, you know that it is displays in green color. So, color displays shown, do not worry about those, come here and notice what is actually given. So, the user name at the hostname is there, followed by the current directory and then dollar.

So, I will show you what would be the difference if we reconfigure the ps1 to the four parts that we have seen just now. ps1 is equal to slash u at slash h colon slash w slash dollar. Now, you see that the color of the prompt has changed. So, some of these other quotes which I mentioned here and here the ending so, these kind of things are all basically for color. Other than that, if you see the display, it is the same text, which means that this is all that actually matters for us.

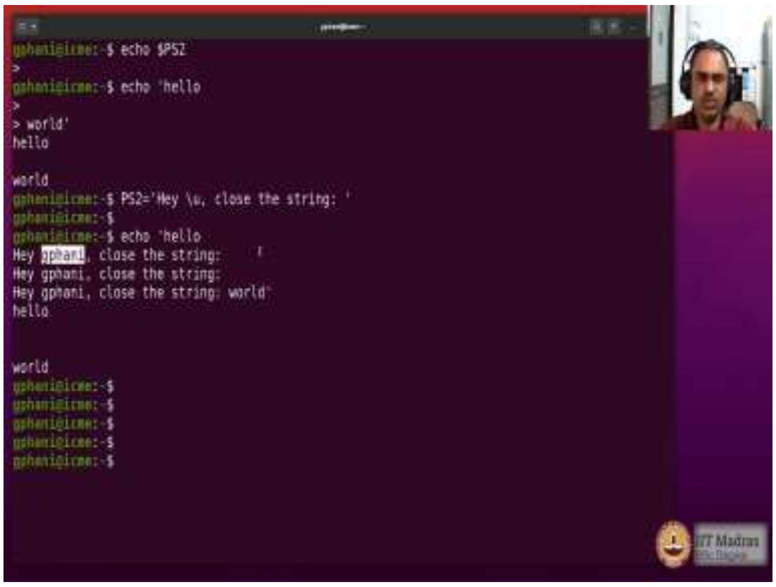
So, we can now configure it with some more fields. And let us try that out. Instead of username and hostname, what I would like to do is slash t. And now you see that the time is being displayed. So, including seconds it can be displayed as a part of the prompt string. And instead of t, I would actually use d which gives me the date. So, the date is matching, only the date part is actually displayed in the prompt.

So, for those who are liking that format can be can choose that way. So, you could also have the date and time together as part of the prompt string. And you could also have hash to list what is the number of the command. So, 44th command is currently what is there. So, I put LS and then the number changes to 45. So, is the next command and the next command.

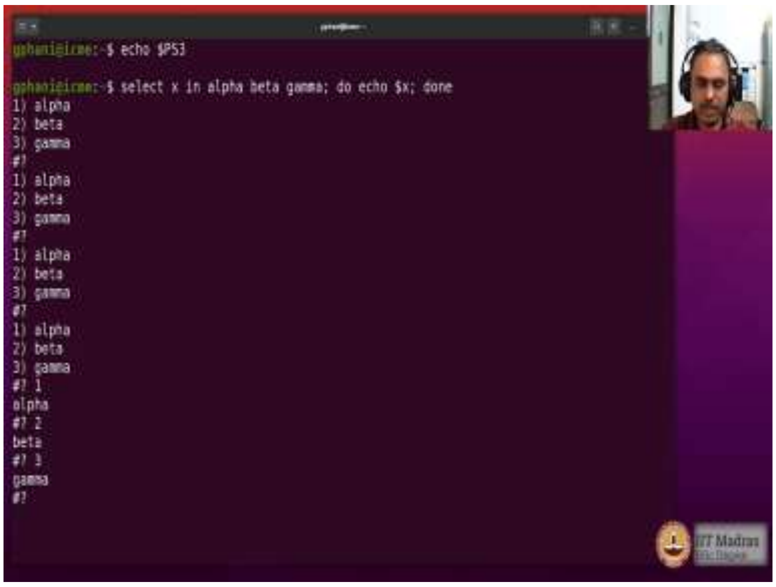
So, you can see that the number in front of the colon is increasing, which means that this particular field is enabling us to keep track of how many of the command are we typing in, in the current instance of the bash. So, of course, if we have messed up the prompt string, and we would like to revert it back, all that we need to do is to source the bash rc and you get back the prompt, because the prompt is defined in the bash rc.

So, where is it defined? You can see here. So, in this paragraph, you have the bash prompt defined and therefore whenever you have any problem, you can just look at this particular part, make a copy of it. So, that you do not lose it, you can actually comment it out and then change it to what you want. So, that whenever you log in the prompt is the way that you like.

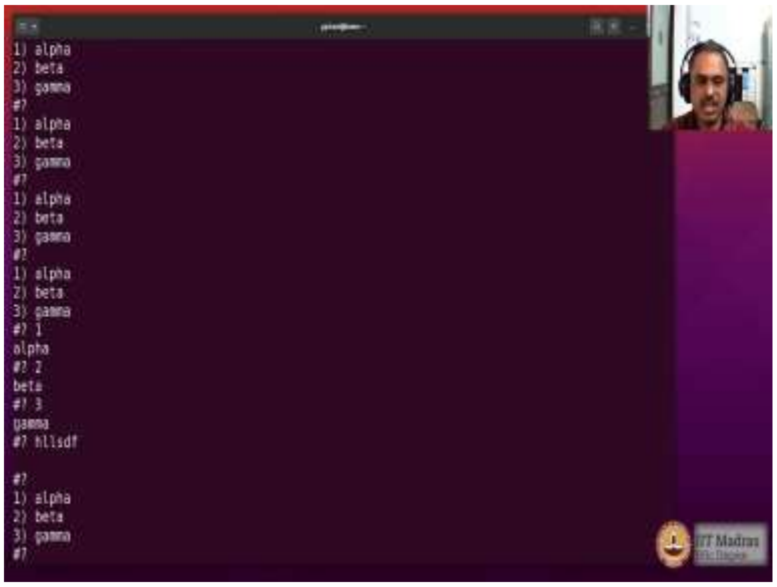
(Refer Slide Time: 8:52)



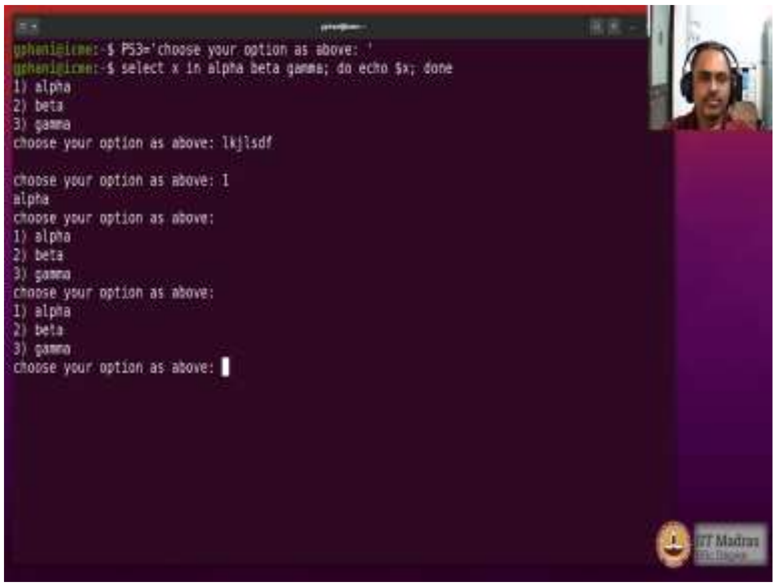
```
gphani@icme:~$ echo $PS2
>
gphani@icme:~$ echo 'hello
>
> world'
hello

world
gphani@icme:~$ PS2='Hey \u, close the string: '
gphani@icme:~$
gphani@icme:~$ echo 'hello
Hey gphani, close the string:     '
Hey gphani, close the string:
Hey gphani, close the string: world'
hello


world
gphani@icme:~$
gphani@icme:~$
gphani@icme:~$
gphani@icme:~$
gphani@icme:~$
```



```
gphani@icme:~$ echo $PS3

gphani@icme:~$ select x in alpha beta gamma; do echo $x; done
1) alpha
2) beta
3) gamma
#?
1) alpha
2) beta
3) gamma
#?
1) alpha
2) beta
3) gamma
#?
1) alpha
2) beta
3) gamma
#? 1
alpha
#? 2
beta
#? 3
gamma
#?
```



```
1) alpha
2) beta
3) gamma
#?
1) alpha
2) beta
3) gamma
#?
1) alpha
2) beta
3) gamma
#?
1) alpha
2) beta
3) gamma
#? 1
alpha
#? 2
beta
#? 3
gamma
#? hllsdf

#?
1) alpha
2) beta
3) gamma
#?
```



```
gphani@icme:~$ PS3='choose your option as above: '
gphani@icme:~$ select x in alpha beta gamma; do echo $x; done
1) alpha
2) beta
3) gamma
choose your option as above: lkjlsdf

choose your option as above: 1
alpha
choose your option as above:
1) alpha
2) beta
3) gamma
choose your option as above:
1) alpha
2) beta
3) gamma
choose your option as above: 
```

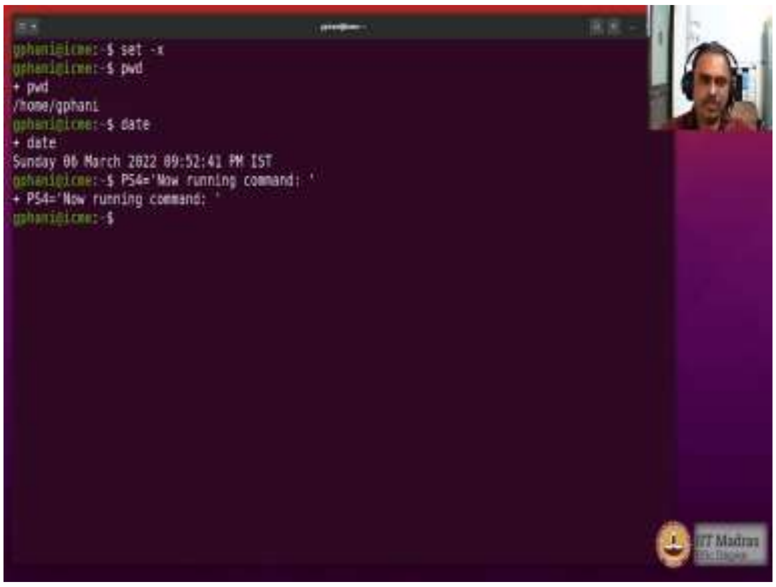
Now the second command prompt is here. So, that is just a greater than symbol. So, if you say like echo hello and I must close the code symbol, I do not close it. So, you now see that the second prompt has come, which is basically asking me to close the quotation symbol. So, I now close it and the command is complete. So, I get back the primary prompt. Now I would like to change the secondary prompt to something else.

And here I can use the field names that we have learned just now. So, I can say, hey, and then user name close the string. So, now what happens is that when I tried to do the same command, the prompt would now look very different. So, you see the prompt is saying hey buddy close the string. And I close it now and now you see that the prompt is back to the primary prompt the secondary prompt alone we have configured where the user name field is used here and rest of it is what I like to show. So, you can customize as you like.

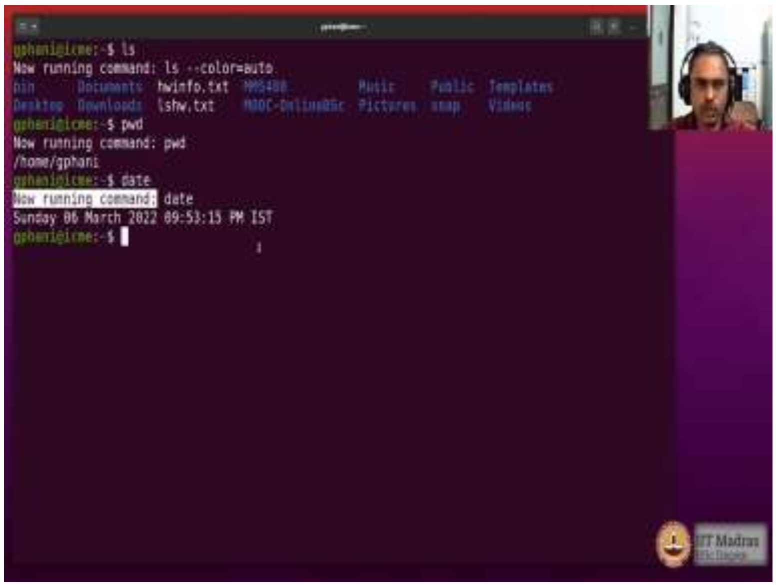
Now the ps3 is for the select command. So, let us look at that first of all. So, it does not display, however, it is already configured. And that is something that you could look at it here, select x in alpha, beta, gamma; do echo dollar x done. So, now this is the select command in bash syntax. And I am now supposed to select one of the three options given alpha, beta gamma, and the values are 123.

And you see the prompt is actually hash question mark. So, I can choose 1, and then it gives me the value that I have chosen, I can choose 2 it gives me the value I have chosen, and if I happen to choose something else, it does not allow that. So, I can come out with ctrl c, and I can customize the ps3. ps3 is equal to I will say choose choose your option as above. So, that is what I want to display. So, let us go ahead and run this command again. And you see that the prompt has changed. So, you can customize the ps3 also and come out.

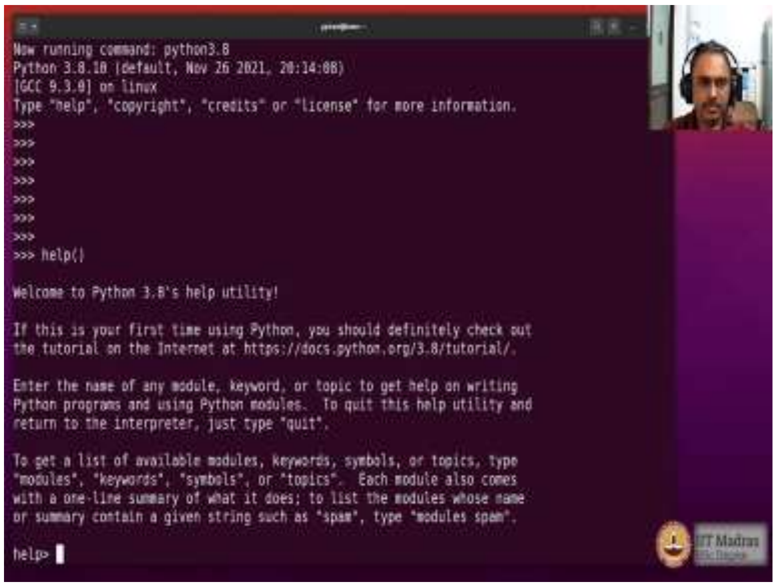
(Refer Slide Time: 11:56)



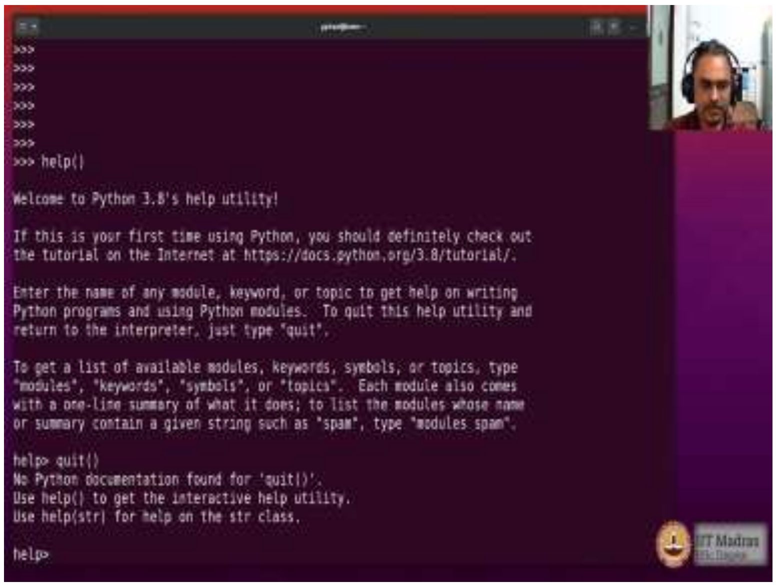
```
gphani@icme:~$ set -x
gphani@icme:~$ pwd
+ pwd
/home/gphani
gphani@icme:~$ date
+ date
Sunday 06 March 2022 09:52:41 PM IST
gphani@icme:~$ PS4='Now running command: '
+ PS4='Now running command: '
gphani@icme:~$
```



```
gphani@icme:~$ ls
Now running command: ls --color=auto
bin      Documents  hwinfo.txt  MM5480          Music     Public  Templates
Desktop  Downloads  lshw.txt    MOOC-OnlineBSc  Pictures  snap    Videos
gphani@icme:~$ pwd
Now running command: pwd
/home/gphani
gphani@icme:~$ date
Now running command: date
Sunday 06 March 2022 09:53:15 PM IST
gphani@icme:~$ 
```



```
Now running command: python3.8
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>> help()

Welcome to Python 3.8's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at https://docs.python.org/3.8/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics".  Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help> 
```



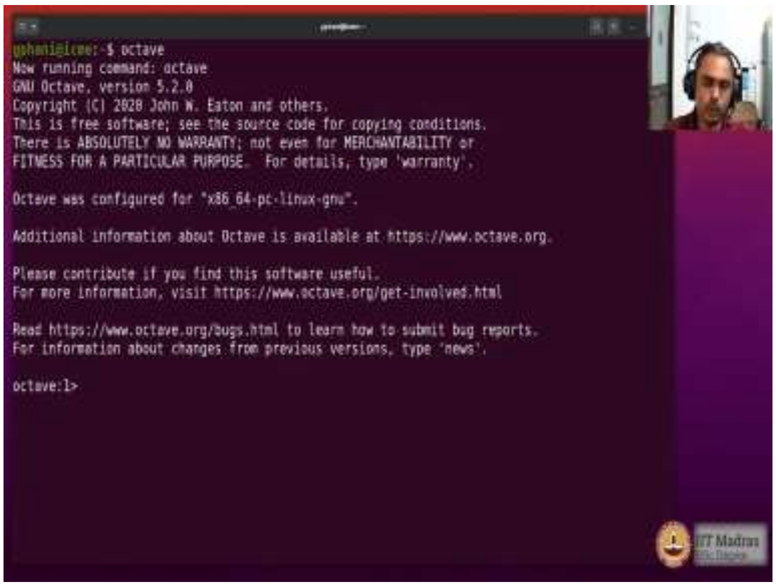
```
>>>
>>>
>>>
>>>
>>>
>>>
>>> help()

Welcome to Python 3.8's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at https://docs.python.org/3.8/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics".  Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help> quit()
No Python documentation found for 'quit()'.
Use help() to get the interactive help utility.
Use help(str) for help on the str class.

help>
```

Now, the ps4 as I mentioned to you is for displaying the command on the screen if we have put set minus x option for execution trace. So, I will put that option now. So, from now on, till I put the command set plus x what happens is that every command that I am running will be first printed out on the screen and then the command will be run. So, you see that plus pwd and then the command has been done and output is home gphani date.

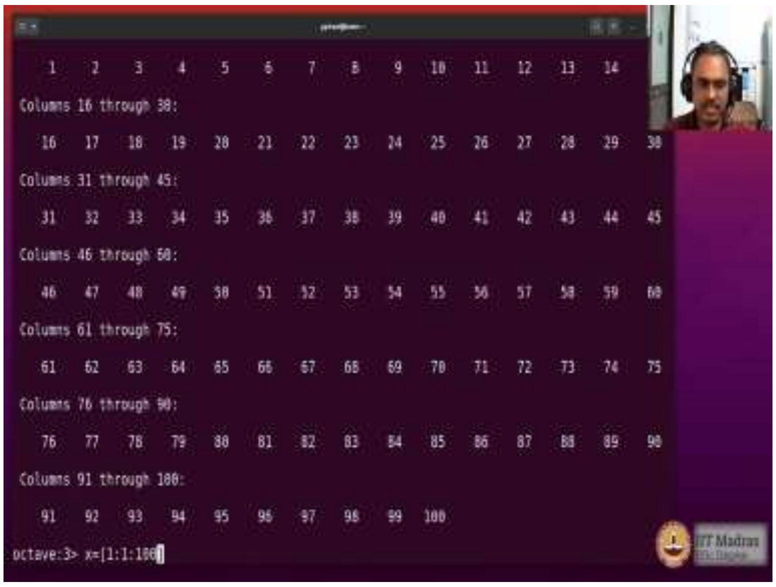
So, the command is again displayed plus date and then the date command output is coming up. So, the prompt string for this output is basically plus character. So, we can now change that ps4 is equal to and then I can give a string I can say now running command. So, I can now change it that way. So, if I now start typing you can see that the prompt has changed from plus symbol it has now changed to now run running command colon, which means that the ps4 has been customized.

So, all the four prompt strings in bash can be customized. Now, what are the other prompt strings that we came across? Python is one such thing. So, let us see what is the prompt string in the case of Python, and you can see it is basically three symbols of three greater than symbols side by side that is the prompt string. So, I come out of that.

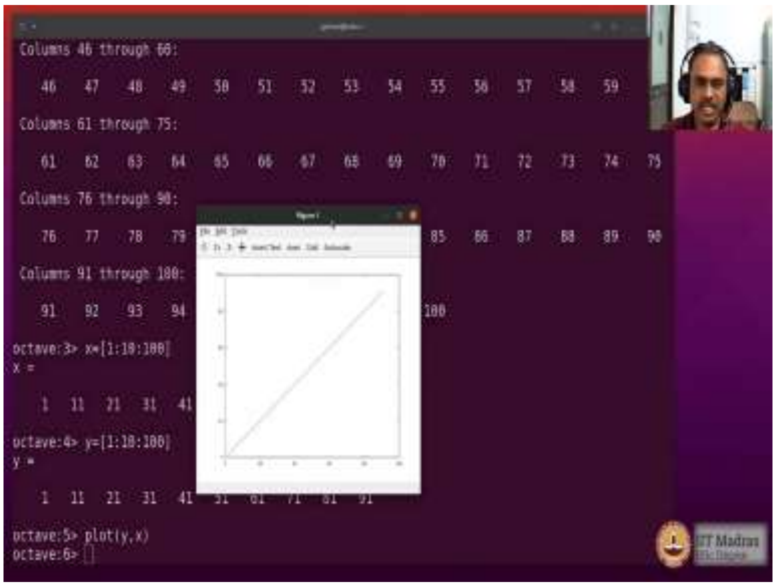
(Refer Slide Time: 13:41)



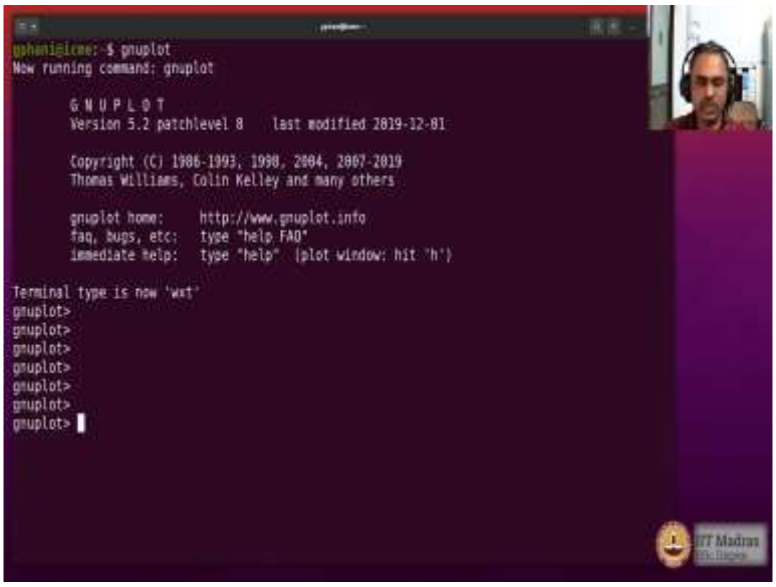
```
gphani@icme:~$ octave
Now running command: octave
GNU Octave, version 5.2.0
Copyright (C) 2020 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type 'warranty'.

Octave was configured for "x86_64-pc-linux-gnu".

Additional information about Octave is available at https://www.octave.org.

Please contribute if you find this software useful.
For more information, visit https://www.octave.org/get-involved.html

Read https://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type 'news'.

octave:1>
```



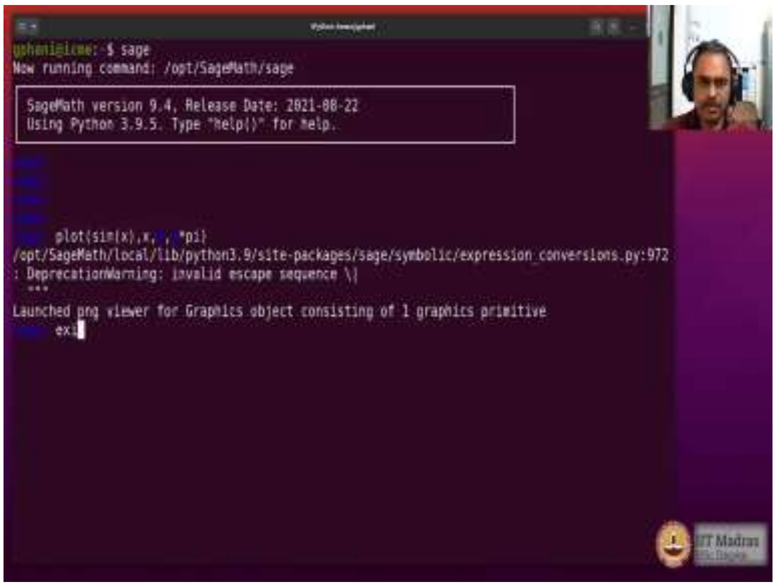
```
   1    2    3    4    5    6    7    8    9   10   11   12   13   14

 Columns 16 through 30:

   16   17   18   19   20   21   22   23   24   25   26   27   28   29   30

 Columns 31 through 45:

   31   32   33   34   35   36   37   38   39   40   41   42   43   44   45

 Columns 46 through 60:

   46   47   48   49   50   51   52   53   54   55   56   57   58   59   60

 Columns 61 through 75:

   61   62   63   64   65   66   67   68   69   70   71   72   73   74   75

 Columns 76 through 90:

   76   77   78   79   80   81   82   83   84   85   86   87   88   89   90

 Columns 91 through 100:

   91   92   93   94   95   96   97   98   99  100

octave:3> x=[1:1:100]
```



```
 Columns 46 through 60:

   46   47   48   49   50   51   52   53   54   55   56   57   58   59

 Columns 61 through 75:

   61   62   63   64   65   66   67   68   69   70   71   72   73   74   75

 Columns 76 through 90:

   76   77   78   79                                 85   86   87   88   89   90

 Columns 91 through 100:

   91   92   93   94                                100

octave:3> x=[1:10:100]
x =

   1   11   21   31   41

octave:4> y=[1:10:100]
y =

   1   11   21   31   41   51   61   71   81   91

octave:5> plot(y,x)
octave:6> 
```



```
gphani@icme:~$ gnuplot
Now running command: gnuplot

        G N U P L O T
        Version 5.2 patchlevel 8    last modified 2019-12-01

        Copyright (C) 1986-1993, 1998, 2004, 2007-2019
        Thomas Williams, Colin Kelley and many others

        gnuplot home:     http://www.gnuplot.info
        faq, bugs, etc:   type "help FAQ"
        immediate help:   type "help"  (plot window: hit 'h')

Terminal type is now 'wxt'
gnuplot>
gnuplot>
gnuplot>
gnuplot>
gnuplot>
gnuplot>
gnuplot> 
```



```
gphani@icme:~$ sage
Now running command: /opt/SageMath/sage
┌────────────────────────────────────────────────────────────────────┐
│ SageMath version 9.4, Release Date: 2021-08-22                     │
│ Using Python 3.9.5. Type "help()" for help.                        │
└────────────────────────────────────────────────────────────────────┘
sage:
sage:
sage:
sage:
sage: plot(sin(x),x,0,2*pi)
/opt/SageMath/local/lib/python3.9/site-packages/sage/symbolic/expression_conversions.py:972
: DeprecationWarning: invalid escape sequence \|
  """
Launched png viewer for Graphics object consisting of 1 graphics primitive
sage: exi
```

Octave is a tool I mentioned to you, which is a very powerful language with the syntax compatible with MATLAB. So, that is also available as a command line utility. So, you could do things that are quite complicated, creating an array from one to 100 creating array with the hops of 10. And making a plot which came on the other window. So, this is a language and the prompt further octave seems to be that it is a string octave followed by a colon and then the number which represents the command line sequence that we are entering.

So, there is another tool called gnuplot and you plot which is also having the command line. You can see that the prompt is gnuplot and you plot. Sage is a symbolic computational package is a language very powerful. And it is also available on command line as well as as a Jupyter notebook. The prompt is sage colon. So, the plot was coming on the other end. So, quite so, that is about the command prompts and how to customize in bash and also command prompts in other command line environments that we have seen below.