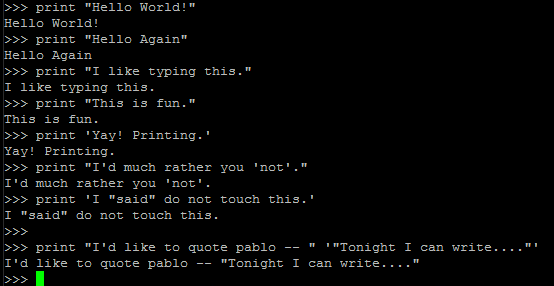
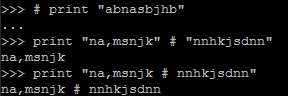
**EX 1: First Hello**

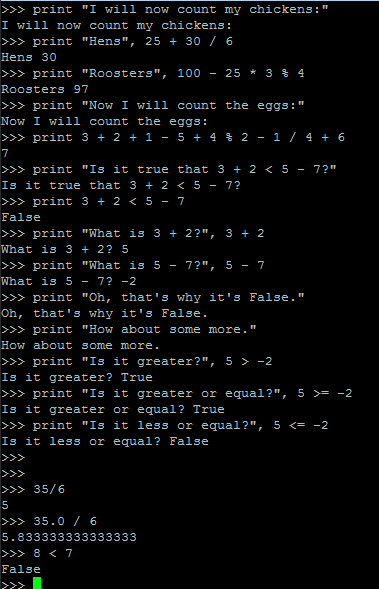


**Note : Wrap with “” to use ‘ and vv**

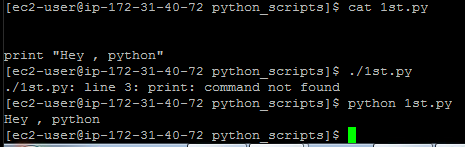
**Ex 1.1: Commenting out**

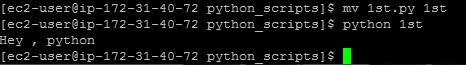


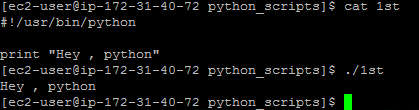
**Ex 2 : Numbers and Math**



**Ex 3 : Your first script**

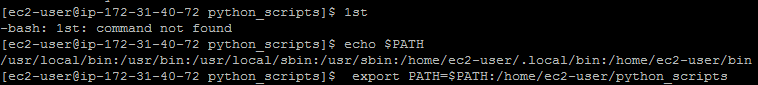






**Note : Update PATH variable with script location**

**Then you can run script directly like**





**Ex 4: variable**

cars **= 100**

space\_in\_a\_car **= 4.0**

drivers **= 30**

passengers **= 90**

cars\_not\_driven **=** cars **-** drivers

cars\_driven **=** drivers

carpool\_capacity **=** cars\_driven **\*** space\_in\_a\_car

average\_passengers\_per\_car **=** passengers **/** cars\_driven

**print** "There are"**,** cars**,** "cars available."

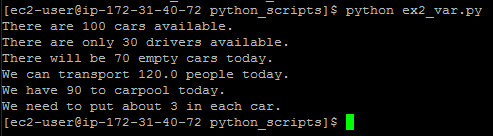
**print** "There are only"**,** drivers**,** "drivers available."

**print** "There will be"**,** cars\_not\_driven**,** "empty cars today."

**print** "We can transport"**,** carpool\_capacity**,** "people today."

**print** "We have"**,** passengers**,** "to carpool today."

**print** "We need to put about"**,** average\_passengers\_per\_car**,** "in each car."



**Note : Printing variables : Separate them with , from the quote**

**Ex 4.1 : More variables and printing**

my\_age **= 35** *# not a lie*

my\_height **= 74** *# inches*

my\_weight **= 180** *# lbs*

my\_eyes **=** 'Blue'

my\_teeth **=** 'White'

my\_hair **=** 'Brown'

**print** "Let's talk about %s." **%** my\_name

**print** "He's %d inches tall." **%** my\_height

**print** "He's %d pounds heavy." **%** my\_weight

**print** "Actually that's not too heavy."

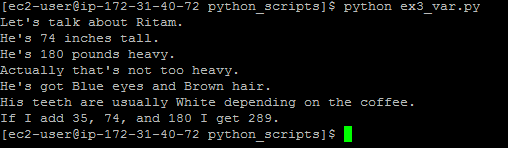
**print** "He's got %s eyes and %s hair." **% (**my\_eyes**,** my\_hair**)**

**print** "His teeth are usually %s depending on the coffee." **%** my\_teeth

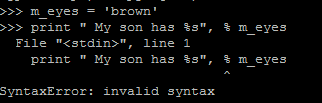
*# this line is tricky, try to get it exactly right*

**print** "If I add %d, %d, and %d I get %d." **% (**

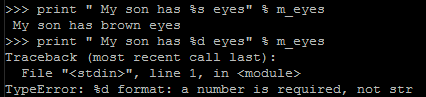
my\_age**,** my\_height**,** my\_weight**,** my\_age **+** my\_height **+** my\_weight**)**



**Extrass:**



Don’t use , before %



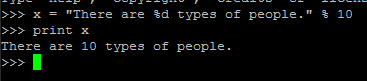
%s 🡪 string %d 🡪 Number

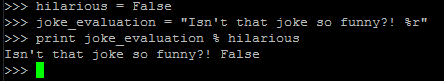


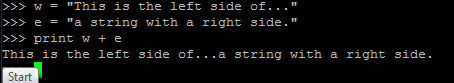
Use the **%r** for debugging, since it displays the **"raw" data of the variable**, but the others are used for displaying to

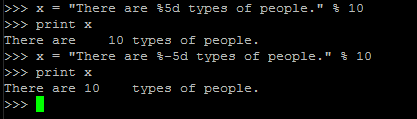
users.

**Ex 4.3 More printing examples**







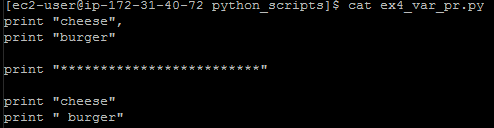


**%5d : puts 5 spaces before**

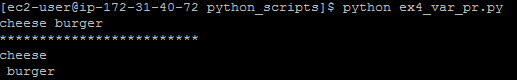
**%-5d: puts 5 spaces after**

**EX 4.4: use of , between 2 print func**

**Scr:**



**o/p:**



**Ex 4.5: Formatter**

formatter **=** "%r %r %r %r"

**print** formatter **% (1, 2, 3, 4)**

**print** formatter **% (**"one"**,** "two"**,** "three"**,** "four"**)**

**print** formatter **% (**True**,** False**,** False**,** True**)**

**print** formatter **% (**formatter**,** formatter**,** formatter**,** formatter**)**

**print** formatter **% (**

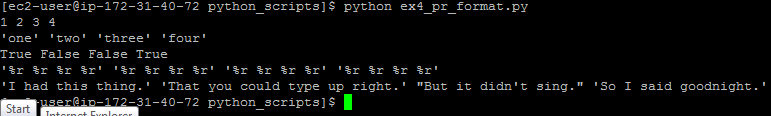
"I had this thing."**,**

"That you could type up right."**,**

"But it didn't sing."**,**

"So I said goodnight."

**)**



**Ex 4.6: 3 double quotes and newline etc**

**Scr:**

*# Here's some new strange stuff, remember type it exactly.*

**days = "Mon Tue Wed Thu Fri Sat \tSun"**

**months = "\nJan\nFeb\nMar\nApr\nMay\nJun\nJul\nAug"**

**print "Here are the days: ", days**

**print "Here are the months: ", months**

**print """**

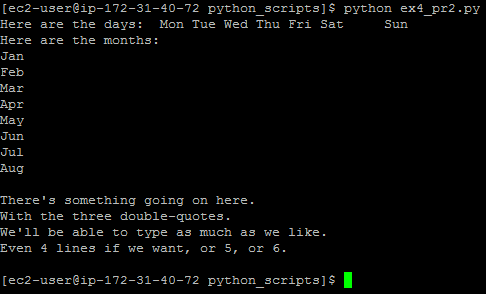
**There's something going on here.**

**With the three double-quotes.**

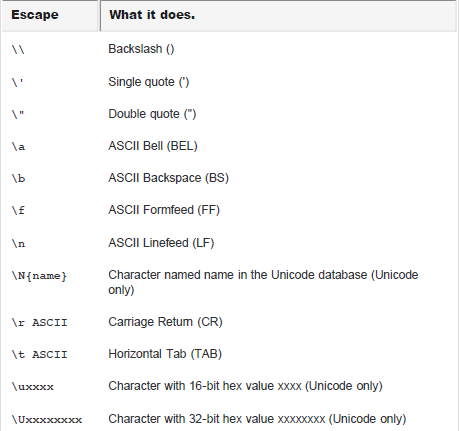
**We'll be able to type as much as we like.**

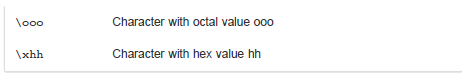
**Even**4 lines if we want, or 5, or 6.

"""

**o/p:** 

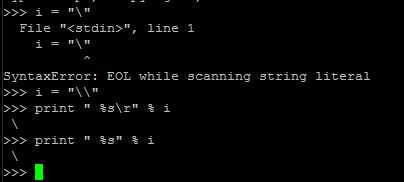
**Ex 4.7 : Escape Sequences**





**Escaping single or double quote**





**Ex 5: raw\_input and input func**

**print "How old are you?",**

**age = raw\_input()**

**print "How tall are you?",**

**height = raw\_input()**

**print "How much do you weigh?",**

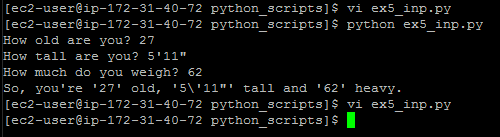
**weight = raw\_input()**

**print "So, you're %r old, %r tall and %r heavy." % (**

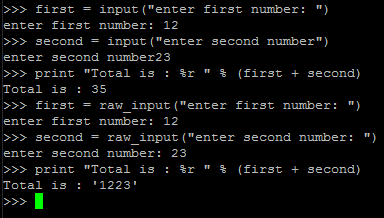
**age, height, weight)**

Getting input from

User.



**5.1 : Difference between input() raw\_input()**



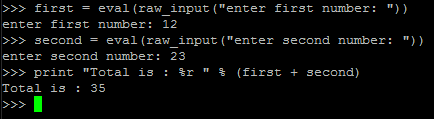
**Note: \***

**input evaluates user input and converts it to int , float or string**

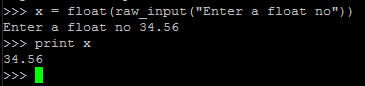
**Raw\_input returns string only**

**\*\* you can print message for user inside braces like –**

**5.2 : How to do math ops using raw\_input**



**Note: In place of eval we can also use int, float etc to convert the input string**



**Ex 6 : Parameters,**

**from sys import argv**

**script, first, second, third = argv**

**print "The script is called:", script**

**print "Your first variable is:", first**

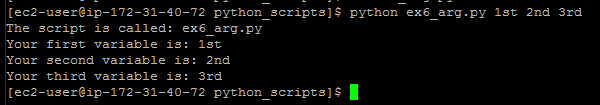
**print "Your second variable is:", second**

**print "Your third variable is:", third**

**Unpacking, Variables**

**Passing variables in script**

**Script 🡪**



**"import". This is how you add features to your script from the Python feature set.**

**Rather than give you all the features at once, Python asks you to say what you plan to use. This keeps your programs small.**

**argv is the "argument variable",**

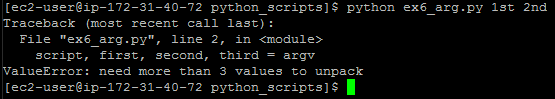
**This variable holds the arguments you pass to your Python script when you run it.**

**unpacks" argv so that, rather than holding all the arguments, it gets assigned to four variables you can work**

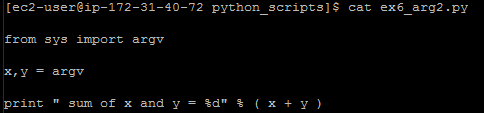
**with: script, first, second, and third.**

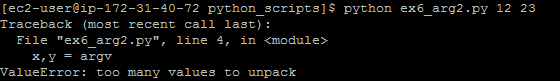
**Issues:**

* **Not enough Arguments**



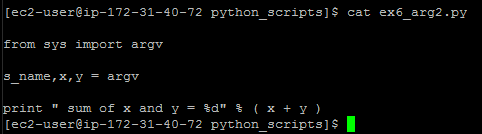
**\*\* Too many values**

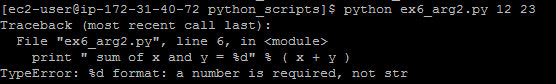
 **scr**

 **o/p**

**Note : script name is by default the 1st args**

**\*\*\***

 **scr**

 **o/p**

**Note: Argv takes the values as strings .**

**Convert them to int/float to use math operations.**

**Ex 7 : Reading Files**

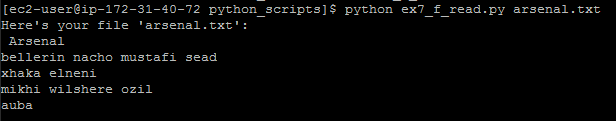
**from sys import argv**

**script, filename = argv**

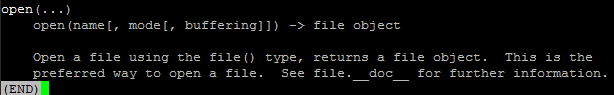
**txt = open(filename)**

**print "Here's your file %r:" % filename**

**print txt.read()**



**Note:**

****

**Read , Readline :**

**read(...)**

**| read([size]) -> read at most size bytes, returned as a string. If the size argument is negative or omitted, read until EOF is reached.**

**| Notice that when in non-blocking mode, less data than what was requested. may be returned, even if no size parameter was given.**

**| readline(...)**

**| readline([size]) -> next line from the file, as a string.**

**| Retain newline. A non-negative size argument limits the maximum**

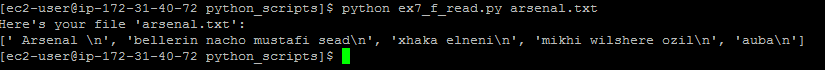
**| number of bytes to return (an incomplete line may be returned then).**

**| Return an empty string at EOF.**

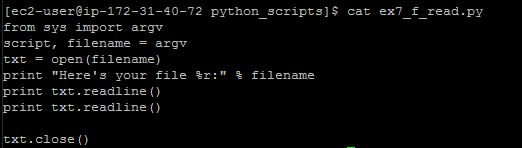
**| readlines(...)**

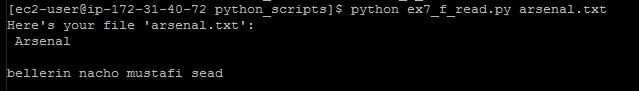
**| readlines([size]) -> list of strings, each a line from the file.**

* **Using readlines() in place of print txt.read()**



**\*\* Using readline()**

**Scr 🡪** 

**o/p**🡪 

**Have your script also do a close() on the txt and txt\_again variables. It's important to close files when you are done with them.**

**Ex 8 : Writing into a file**

**from sys import argv**

**from os.path import exists**

**script, from\_file, to\_file = argv**

**print "Copying from %s to %s" % (from\_file, to\_file)**

***# we could do these two on one line too, how?***

***#indata = open(from\_file).read()***

**in\_file = open(from\_file)**

**indata = in\_file.read()**

**print "The input file is %d bytes long" % len(indata)**

**print "Does the output file exist? %r" % exists(to\_file)**

**print "Ready, hit RETURN to continue, CTRL-C to abort."**

**raw\_input()**

**out\_file = open(to\_file, 'w')**

**out\_file.write(indata)**

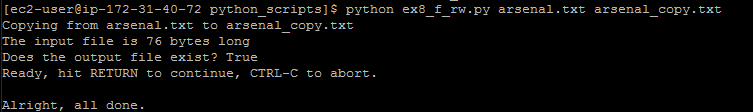
**print "Alright, all done."**

**out\_file.close()**

**in\_file.close()**

**In\_file**🡪 opens a file object for from\_file

We can also use like  
indata = open(from\_file).read()



**Note : indata = in\_file.read(20)**

**Allows only first 20 bytes of input file to be read.**

**Ex8.1 : Deleting a file content and writing again**

**from sys import argv**

**script, filename = argv**

**print "We're going to erase %r." % filename**

**print "If you don't want that, hit CTRL-C (^C)."**

**print "If you do want that, hit RETURN."**

**raw\_input("?")**

**print "Opening the file..."**

**target = open(filename, 'w')**

**print "Truncating the file. Goodbye!"**

**target.truncate()**

**print "Now I'm going to ask you for three lines."**

**line1 = raw\_input("line 1: ")**

**line2 = raw\_input("line 2: ")**

**line3 = raw\_input("line 3: ")**

**print "I'm going to write these to the file."**

**target.write(line1)**

**target.write("\n")**

**target.write(line2)**

**target.write("\n")**

**target.write(line3)**

**target.write("\n")**

**print "And finally, we close it."**

**target.close()**

**Truncates the file, ‘target’ holds**

**Remember:**

**1st , open a file object**

**To read🡪**

**F\_obj.read(size in byte)**

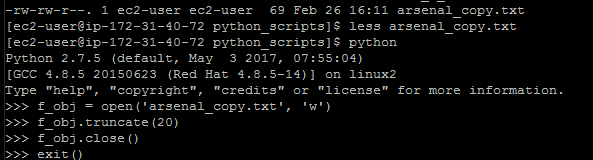
**Gives str o/p of file data**

**To Write 🡪**

**F\_obj.write(strings to be written)**

**To delete content 🡪**

**F\_obj.truncate()**



**Truncates the file to 20 bytes but makes it a binary file.**