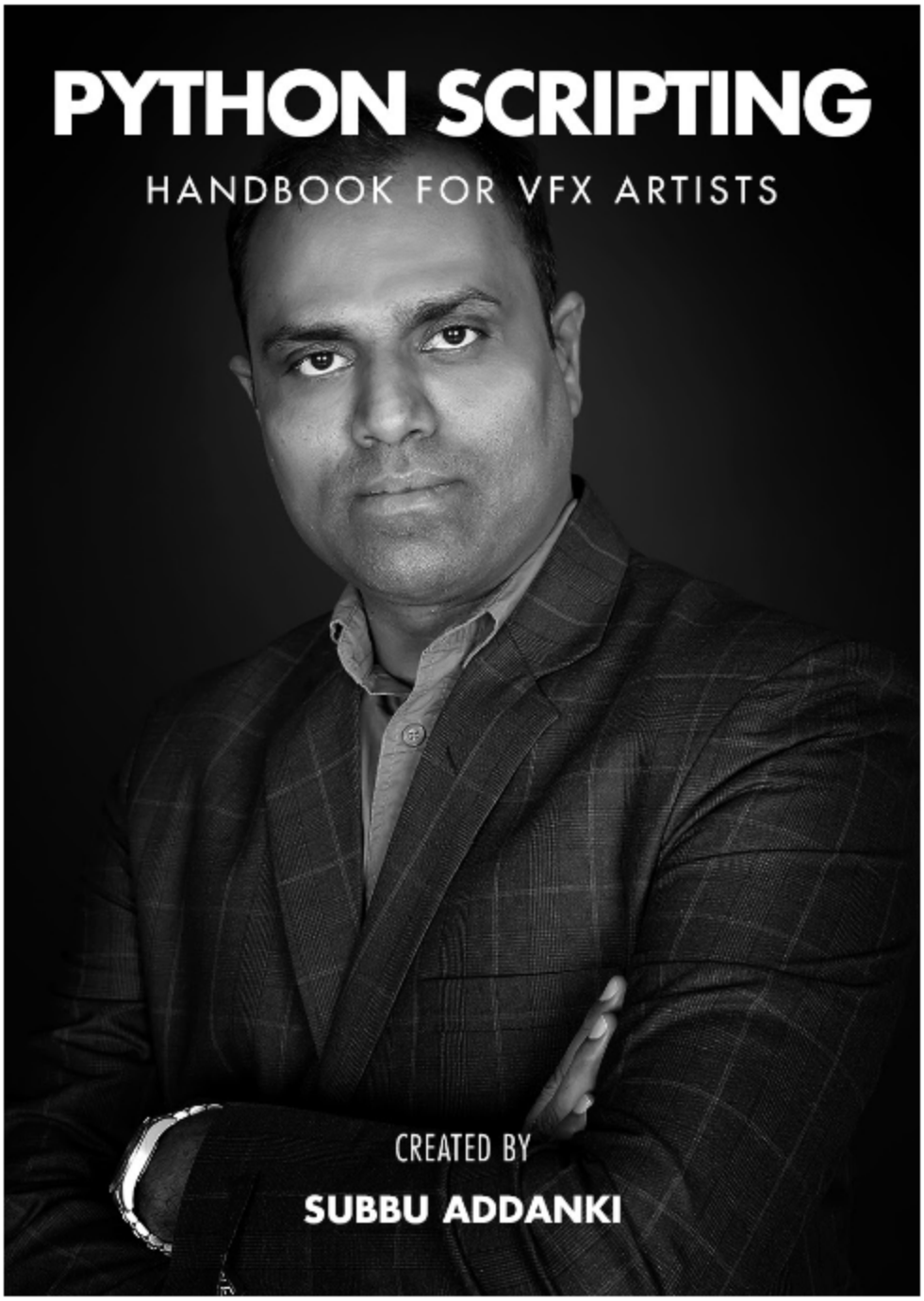
PYTHON SCRIPTING

HANDBOOK FOR VFX ARTISTS



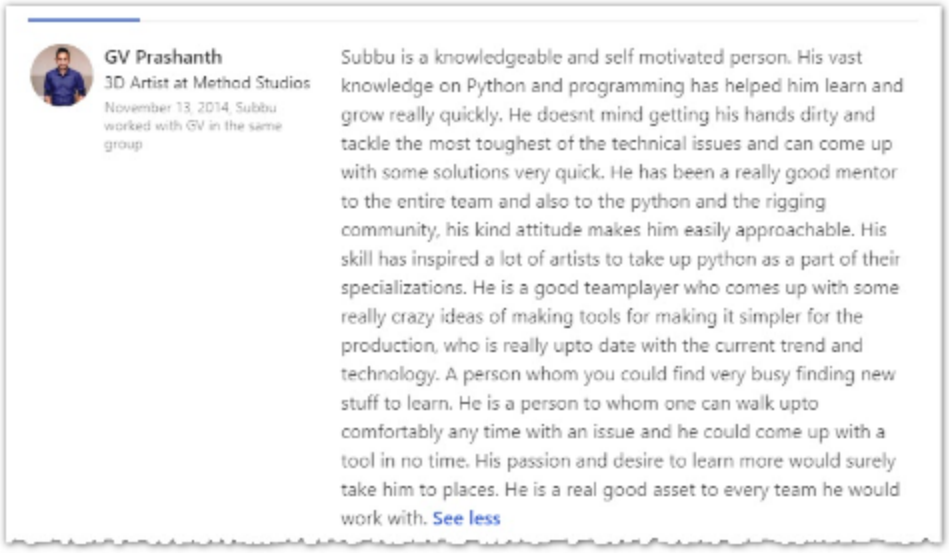
"Kindness is the language which the deaf can hear and the blind can see"
-Mark Twain

Dedicating this book to the people who can't see this beautiful world with their eyes. Donations can be made at: Sightsavers | Protecting sight and fighting for disability rights

About
The Author

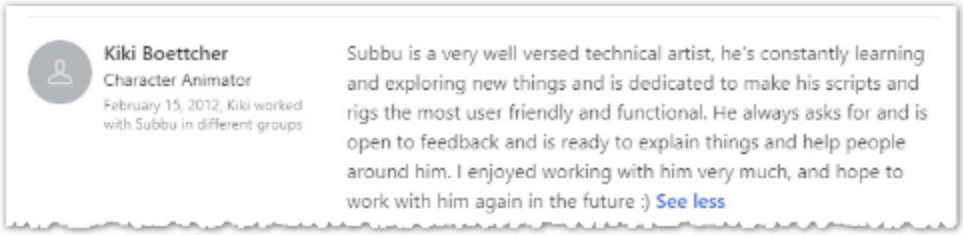
I am Subbu Addanki, Veteran Technical Director and humble founder of Boom Rig Systems. I create complex rigs with simple interfaces. I believe a good rig/ character is something that adapts to any scene. Be it a simple walk cycle or a complex action sequence with lots of cloth simulation and interactions.

All characters and tools are created with one simple goal in mind, "flood my inbox with thank you messages". Yes, you heard it right. I want to see my inbox filled with "Thank You" and "happy emojis" 24/7. This is the very fuel that drives me to the office every single day. Nothing is more rewarding than making fellow artists happy. The pain of trial and error, writing thousands of lines of code and waiting for code to compile is gone when I receive testimonials like this from fellow artists:



GV Prashanth
3D Artist at Method Studios
November 13, 2014, Subbu worked with GV in the same group

Subbu is a knowledgeable and self motivated person. His vast knowledge on Python and programming has helped him learn and grow really quickly. He doesnt mind getting his hands dirty and tackle the most toughest of the technical issues and can come up with some solutions very quick. He has been a really good mentor to the entire team and also to the python and the rigging community, his kind attitude makes him easily approachable. His skill has inspired a lot of artists to take up python as a part of their specializations. He is a good teamplayer who comes up with some really crazy ideas of making tools for making it simpler for the production, who is really upto date with the current trend and technology. A person whom you could find very busy finding new stuff to learn. He is a person to whom one can walk upto comfortably any time with an issue and he could come up with a tool in no time. His passion and desire to learn more would surely take him to places. He is a real good asset to every team he would work with. See less



Kiki Boettcher
Character Animator
February 15, 2012, Kiki worked with Subbu in different groups

Subbu is a very well versed technical artist, he's constantly learning and exploring new things and is dedicated to make his scripts and rigs the most user friendly and functional. He always asks for and is open to feedback and is ready to explain things and help people around him. I enjoyed working with him very much, and hope to work with him again in the future :) See less

And more about the author and his works:

http://www.animationinsider.com/2016/11/subbaiah-addanki/
http://www.3dvf.com/actualite-2983-subbu-addanki-rigging-skinning-scripting.html
https://theinterviewportal.com/2020/05/11/character-animator-rigger-interview/

About This Book

Learn how to create your tools to speed up your workflow with the power of Python Scripting. With the help of this guide, you can create your own tools in just 3 weeks. I will share my industry secrets and strategies, which I've used to create tools that have saved thousands of hours of production time. My tools have been downloaded over 10K times.

Learn Python Scripting

.. From Real Life Examples !!

If you are a CG Artist who has never written a line of code in your life, this book is the best place to start learning how to code for VFX. I wish I had this book when I first started learning python.

By the end of this book, you will have hands-on experience writing 10+ tools and 10+ mini python modules (step by step instructions, and video tutorials included).

I believe that learning python scripting principles using real-life situations as examples of coding scenarios makes learning easier. The goal of this book is to allow any CG Artist, able to understand and remember the programming principles for a lifetime. In my experience of previously teaching a Python course in person, when we approach learning this way we interactively connect on an emotional level with the information, so it's easy to understand and learn new things quickly.

So let's get started!

About
The Editor

Alexandra Papouchina, 3D Generalist, is passionate about exploring every area in the VFX and Animation pipeline.
As Editor, she is a great help for this book. It's my great pleasure to work with Alexandra for more than a year. We worked together for a couple of animation workshops, where she delivered great models for my rigs.
She had a great interest in learning Python Scripting. When I asked about editing this book, she came up with a positive response. I am really glad to have her as Editor for this book. She is helping me while correcting the sentence framing in this book without losing the original meaning, out of her busy schedule.

Alexandra is able to pick up new techniques and processes quickly. When I am trying to bring this book with every detail and to make the reader curious about upcoming pages, I want everything to be perfect. In that process, her skills will really help me produce this book to the next level.
Alexandra is a great artist, dependable and hardworking teammate. She is on time and ready to help. Alexandra consistently provides quality in everything she does.

More about the editor:
https://www.linkedin.com/in/alexandra-papouchina-a9681814a/

More Content
& For Latest News

Please Visit : www.pythonscripting.com
Or www.boomrigs.com

01. Introduction To Python
-The Journey Begins

02. Maya Script Editor

-Interpreter

- Maya Script Editor — *Titanic Ship*
- Interpreter Options — *Input & Output Zones*
- External Interpreters — *Easy To Use - Have Fun*

ActiveState, Eclipse & PyCharm, Etc.

- Mel, Python Commands & Arguments — *Mobile Shopping Options*
- Color Coding & Auto-Completion — *Coder's Life Made Easy*
- Nuke Scripting & Commands — *Commands Reference Library*
- Be A Player - For Your Happiness — *Rahul Dravid, Mr. Dependable*
- Conclusion - Script Editor / Interpreter — *Your Play Area*

03. Syntax

-Command Line Structure

- Just Learn One Line Code .. That's it !! — *Just For One Nail ??*
- PEP-8 Style Guide Of Python — *Neatly Packed Gift - All eyes are on It*
- Module Import — *Hey... He Called Me, Not You !! - Welcome Magician*
- Indentation — *Thanks Mr Python, Well Organized Outline*
- Naming Convention — *Easy To Read & Understand*
- Comments (Strings) — *Know What You Have Written- School Notes*

04. Strings
-For User Interaction

- Print To Screen — *Let Your Users Know*
- Strings Concatenation — *A Series Of Connected Railway Bogies*
- Type Casting — *Convert Type As You Need*
- String Quotes — *Convey Message Effectively -Product Description*

"Be careful--with quotations, you can damn anything."

— André Malraux

05. Variables
-Containers

- What is Variable — *Chinnu, Bannu - Nick Names = Long Names*
- Declare Variables — *Snack Boxes In Kitchen*
- Local & Global Variables — *It's mine & That's For Everyone*
- Variable Types — *Auto Detection/Recognize Automatically.*

06. Lists

-Sequence Of Elements

- Get The List Of Items — *Shopping List: List Of Items To Purchase*
- Methods Of List — *Shopping List: Edit List Of Items*

  [append, remove, insert, count, extend, reverse, sort, pop, index]
- List Operations — *Getting List Items: Indexing & Slicing*
- Tuples & Their Usage — *Items List Declared, Can't Change Now*
- Conclusion — *Can't Live Without Lists Now*

07. Operators

-Basic Math

Now, Time To Visit Our School Days Again

- Addition — *(+, [Add Anything -Names, Numbers, Lists])*
- Subtract, Multiply, Divide — *(-, *, / , [Use As Needed])*
- Modulus — *(%, [Get Remainder For Cycle Anim, etc])*
- Exponent — *(**, [A Power Operator])*
- Comparison — *(==, !=, <=, >=, >, < .[Test Us Again])*
- Logical — *(and, or, not, [Use Logical Skills])*
- Assignment — *(+=, -=, /=, *=, **= , [Mostly Used In Loops])*
- Conclusion — *(Your Helpers In Day To Day Coding Life)*

08. Conditional Statements

-Decision Makers

- Decisions: Left | Right, Up | Down

  Google Map : Route Via Flyover

- Practical Decisions — *One-Way Traffic Dead End - Car Example*
- True | False — *And More : 1 | 0, [a,] | [], {a:1, } | {}, Universal Truths*
- Operators — *and, or, in, is, not - Life Examples*
- If-elif-else Statement — *Time To Make A Decision*
- Types Of Decisions — *Real Life Challenging Decisions*
- Simple & Complex Decisions — *10th Passed? What Next ??*
- Type Based Decisions — *Beer, Whisky, Rum & Gin Etc*
- User-Based Decisions — *Rigger, Animator & Lighter Etc*
- Survey Based Decisions — *Recent Survey - Python Classes Per Week*
- Decisions Based On Fixed Options — *Shopping - Limited Options*
- Order & Stock-Based Decisions — *Viewers Vs Available Movie Tickets*
- Multiple Situations — *House Purchase & Marriages*
- Conclusion — *Make More Decisions & Learn More*

"It is in your moments of decision that your destiny is shaped"

- Tony Robbins

09. Loop Statements

-Repetitive Statements

- Loops In General — *Wow.. What A Great Giant Wheel*
- for loop — *Be In A Queue, Everyone Get It*
- while loop — *Love You Python, Till I Collapse*
- Nested Loops — *Everyone refers/goes to all persons in the group*
- break statement — *Time to Come Out Of 'Q' & Let others behind you*
- continue statement — *Everyone in 'Q' gets it, except you !!*
- Danger Zone — *No Savings ?? Save Today For A Better Tomorrow :)*
- Conclusion — *Use This Guy Wisely & Effectively !!*

10. Dictionaries

-Mapping Of Items

- Mapping Of Key, Value Pair — *Time To Switch On The Lights*

  Artist Details: Age, DOJ, Role Etc
- Methods Of Dictionary — *Use Keys: Age, DOJ, Role Etc To Get Details*

  keys, items, has_key, get, copy, clear, update, fromkeys, setdefault
- Dictionary Operations/ Uses — *One Point ➢ To Another One*
- Conclusion

11. Functions

-Mapping Of Items

- Name For List Of Works — *Mom don't repeat the list All the times...*
- Function Uses — *Artist Details: Age, DOJ, Role Etc*
- Commands & Arguments — *Mobile Shopping Options*
- Conclusion — *Easy to use & Easy to remember !!*

12. Modules

-Mapping Of Items

- Modules — *Your library of code which you develop*
- Supporting Modules — *Build the strong base*
- Conclusion — *Gateway to become master chef for your tools.*

13. Files I/O

-Mapping Of Items

- Files — *Your library of personal modules*
- Usage Of Files — *Export and import as needed*
- Conclusion — *You had all the weapons now.. Ready to do an awesome tool*

14. Exceptions

-Mapping Of Items

- Exceptions (*Life Examples: WIP*)
- Practical Use Of Exceptions
- Conclusion

15. Regular Expressions

-Algorithm To Search Strings

- Regular Expressions (*Life Examples: WIP*)
- Practical Use Of Regular Expressions
- Conclusion

16. Metaclass Programming

-A Master Class For Classes

- MetaClass Programming (*Life Examples: WIP*)
- Practical Use Of MetaClass Programming
- Conclusion

17. Python 2.0 to 3.0

-To The Next Level...

- Python 2.0 to 3.0 (For Maya 2022) (*Life Examples: WIP*)
- Practical Use Of Python 3.0
- Conclusion

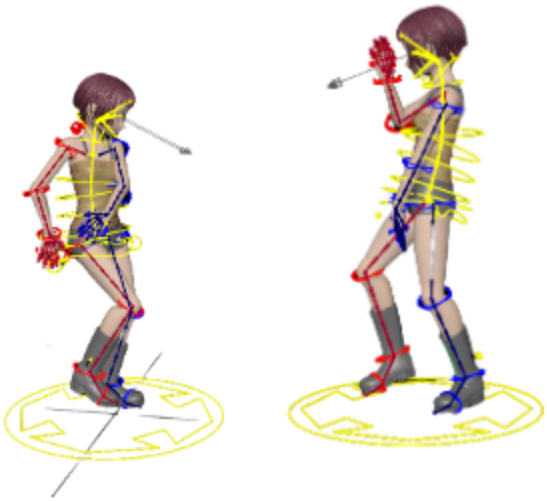
18. Practical Exercises

-Practice Makes A Man Perfect..

- One Decision But Hundred reasons — *Use The Power Of 6th Sense*
- Make Quick Decisions — *Decisions Vs Time Vs Many Reasons*
- Conclusion — *It's a never ending, happy & challenging journey*
- What's Next — *A Journey Towards Machine Learning & Deep Learning*

01. Introduction To Python

-The Journey Begins



Lists & Functions (01A)

Parents >>> Our First Programmers

Lists & Functions From Real Life

One can say that parents are our first programmers who taught us about programming concepts like Lists and Functions etc. At some point in our childhood, they taught us how to do some household work on our own.

Mother >> Our First Teacher

REAL LIFE SCENARIO / EXAMPLE

Ravi's parents told him to go to the vegetable market to get a few items. He is given a paper slip having a list of items to purchase. By the time he arrived at the market, his mother called him to say that one of the items in the list was not needed and she asked him to remove that item from the list. He has crossed off that item with a pen. And the same way, when later his father asked him to add an item to the list, he might have added that extra item at the end of the list.

Explained Real Life Example Here Presented Using Python:

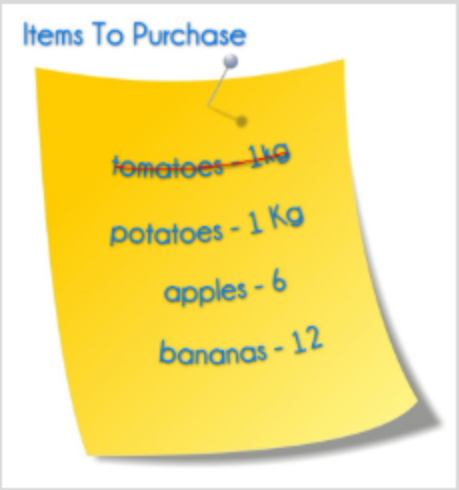
Vegetable Market >>>
We Learned Lists Here

```
>>> itemsToPurchase = ['tomatoes', 'potatoes', 'apples', 'bananas']

# To remove 'tomatoes' from list in Python Scripting :
>>> itemsToPurchase.remove('tomatoes')

>>> print itemsToPurchase
 [ 'potatoes', 'apples', 'bananas']
```

 # In real life, mother asked him to remove the item 'tomatoes' #
 # Now our list on paper looks something like this

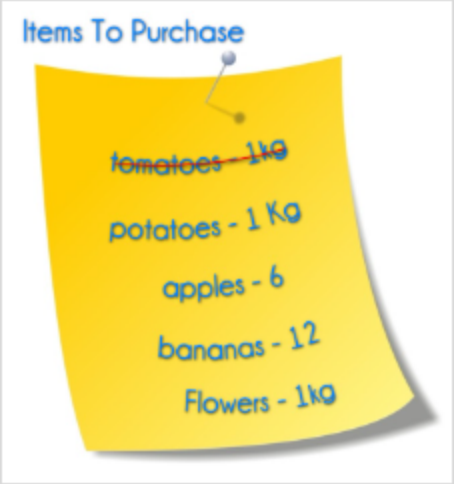


 # Now, Father asked him to add 'flowers' to the list and guided him where to purchase the same ..
 # In Python Scripting :

```
>>> itemsToPurchase.append('Flowers')

>>> print itemsToPurchase
 [ 'potatoes', 'apples', 'bananas', 'Flowers']
```

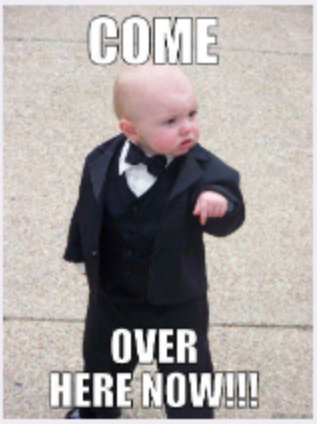
 # Now our list on paper looks something like this



Loops, If-else, Functions, Exceptions (LIFE) (01B)

Idhar Aaiye /

(Come Here)



If you learn a few concepts in Python Scripting like "Loops, If-else conditional statements, Functions and Exceptions (LIFE) ", you can write a lot of tools, even though you are just beginning to write scripts. And Thinking creatively about using that gives you a lot of confidence to move forward in writing more tools.

Same as with communicating in any language: Once a man named Ramesh used the phrase 'Idhar Aaiye (Come Here)' in the Hindi language to tell my friend Sandeep to walk over to be physically closer to him (Hindi - One of the main Indian languages). Sandeep was

surprised at Ramesh's new knowledge of Hindi and asked if Ramesh truly believes that he knows the language. To which Ramesh replied with confidence: 'Yeah I learned Hindi'.

Sandeep asked 'What will you say in Hindi for "Go There"', to which Ramesh replied 'First I will go there, then I will ask the other person to 'Idhar Aaiye (Come Here)'. Sandeep started laughing in an uncontrolled way knowing that Ramesh can deal with a lot of stuff by just learning two words 'Idhar Aaiye'.

So, like Ramesh, try to approach Python with creativity and a sense of curiosity - like a game or learning a new musical instrument!

if one can learn the main concepts in Python Scripting like "***Loops, If-else conditional statements, Functions and Exceptions (LIFE)***", he/she can write a lot of tools in the beginning days of writing some scripting stuff

Way To Learn Python (OIC)

So, like Ramesh, try to approach Python with creativity and a sense of curiosity - like a game or learning a new musical instrument!

REAL LIFE
SCENARIO / EXAMPLE

Another scenario: think about this like learning Chess

When someone first shows us only a chessboard and tells us that this game is very addictive and interesting, if we are new to this game it's likely that we won't believe this statement...

Wow .. I Liked It ..

So the next day, he might start to show us the nicely carved chess pieces and teach us their roles to spark our curiosity about the game. And once we learn the basic rules and strategy, we get more and more engaged and soon start enjoying the game.

Then that's it.. we're hooked. Soon as we get more experience learning the game, the addiction to keep playing sets in.

I guarantee this is what is going to happen, even while Learning Python Scripting. :)

And What about Guitar?

Hey! I Did A Nice Tune

During the initial days of learning guitar, we have only pains and no major gains. The moment we start doing some nice tunes, everyone likes it and we want to do more tunes. The more we do, the more we want to play the guitar.

Soon, we will find that it's going to be addictive and highly enjoyable.

This is what is going to happen even with Python Scripting also.

We have to type a lot. Only finger pains in earlier days. Once we have done one nice UI with great functionality, We soon find we are addicted to Python Scripting. Hahaha, you will agree to it soon.

Quick Examples (01D)

Readymade Code

Quick UI Example - Maya

When I started writing tools in python over 14 years ago, I couldn't find a single video tutorial on python scripting at the time. The best resources I found at that time were Maya Quick Python examples which are available in Technical documentation from Maya help. And here goes

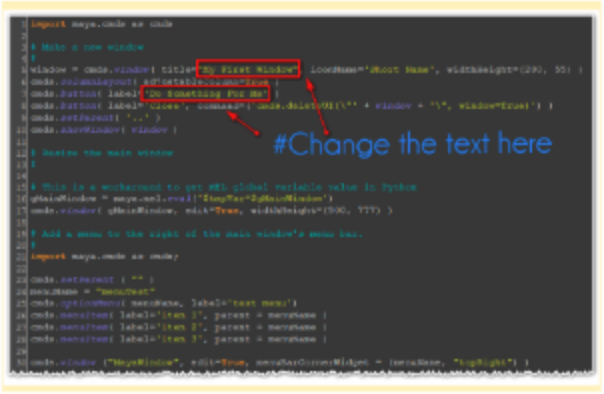
the link:

http://help.autodesk.com/view/MAYAUL/2019/ENU/

Grab a few lines of code from python command examples. For Example, Get the code for the window command and paste it in Maya Python script editor as shown below:

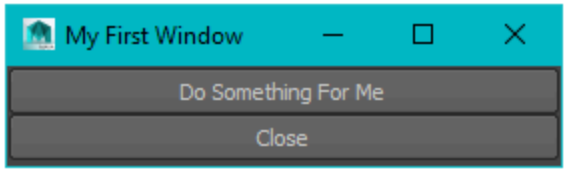
Can I Get ..

A Quick Example?



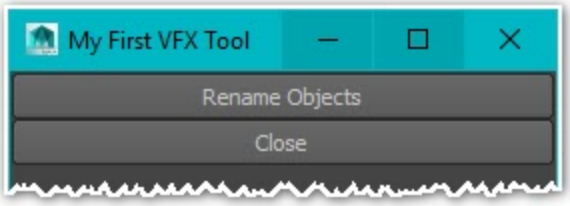
Ref Image 01

Once you run the above code using ***'Ctrl + Enter'*** from the keyboard or '***Enter***' from the Numeric pad, you can see below window with the title '***My First Window***' and button '***Do Something For Me***'



Just by changing the text in the script editor as shown in above Ref

Image 01, window title to '***My First VFX Tool***' and button name to '***Rename Objects***', you can get this UI.



Now all you need is to write a bit of code for the button '***Rename Objects***'. That's it. You can start using your first UI.

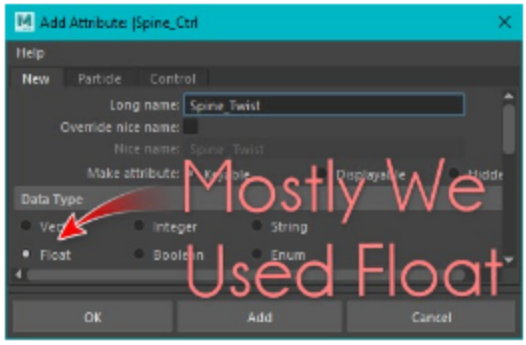
Maya Commands (01E)

May(a) - I Help You ..

All you guys need is here if you are writing Python tools for Maya. After visiting the Maya help page (by pressing F1)

After that, visit Technical Documentation -> Python Commands

Once in a while, every one of us used this function 'Add Attribute' from Maya ***Maya Main menu -> Modify -> Add Attribute***:

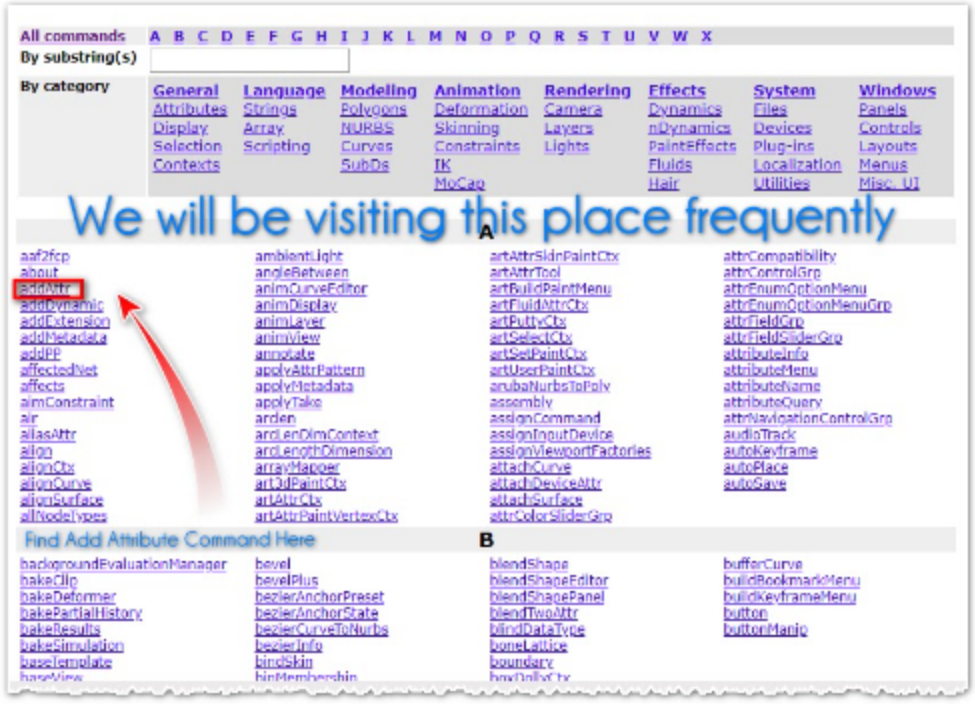


Where Can I Find You?

"Find Me Here .. !!"

http://help.autodesk.com/view/MAYAUL/2019/ENU/?guid=__CommandsPython_index_html

- addAttr



This command '***addAttr***' does the same thing as this UI does ..

(***Maya Main Menu -> Modify -> Add Attribute***)

These Keywords Are ..

Important Key Words (OIF)

---

and - A Keyword, which is mostly used in conditional statements.

For Ex: 'If Ram ***and*** Gopal comes together, Then they can make this movie'

```
#_ Code starts here !! <Code theme: rainbow>
ram_came=True
gopal_came=False

if ram_came and gopal_came:
        print 'Both are here'
else:
        print 'One of the two or Both are missing'
"""
Result (Printed) :
One of the two or Both are missing
"""
```

---

or - Either you can come to the party ***or*** your friend but not your children

```
#_ Code starts here !! <Code theme: rainbow>
ram_came=True
gopal_came=False

if ram_came or gopal_came:
        print 'One of the two or both are here'
else:
        print 'Both are missing'
```

```
"""
Result (Printed) :
One of the two or both are here
"""
```

---

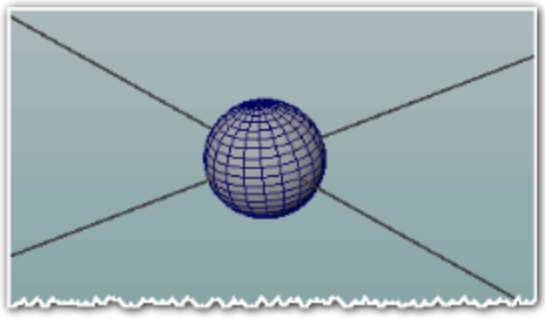
from - A keyword that helps to import modules ***from*** packages/modules

as - Venkat is also called ***as*** Venki. It is used as an alias name for the imported module in the import statement

import - A keyword which helps to ***import*** modules from other modules

```
#_ Code starts here !!  <Code theme: Pojoaque>
from maya import cmds
cmds.polySphere()
import maya.cmds as cmds
cmds.polySphere()
```

Polygon sphere just got created in the Maya scene



---

assert - Test that condition, and raise error immediately if the condition is false.

```
#_ Code starts here - assert statement <Code theme:
```

CODING

TIME

```
atelier-forest-light>
a ='100'
b =100
assert a==b, 'Error.. Hmm both are not the same'
print 'both are the same'
"""
Result (Printed):
# Error: Error.. Hmm both are not the same
# Traceback (most recent call last):
#   File "<maya console>", line 3, in <module>
# AssertionError: Error.. Hmm both are not the same #
"""
a =100
b =100
assert a==b, 'Error.. Hmm both are not the same'
print 'both are the same'
 """
Result (Printed) :
both are the same
 """
```

---

while - ***while*** you are allowed to study upto specific class, you can go to The school, after that you have to study from home

continue - ***continue*** to study the next class by skipping remaining part at said class

break - You got what you want, Time to exit and ***break*** the loop (here Studying Classes). Don't look at remaining items in 'Q' or Loop

```
#_ while loop & break statement !!  <Code theme: atelier-cave-light>
studyingClass =1
skipAtClass =5
stopAtClass =7
while studyingClass <=10:
        #_ Print which class is being studied
        if studyingClass == skipAtClass:
                studyingClass += 1
                continue

        print ('I am studying {} Class'.format(studyingClass))

        if studyingClass == stopAtClass:
                break

        #_ Increment by 1
        studyingClass += 1
print 'I stopped my studies at class ' + str(stopAtClass)
 """
Result (Printed) :
I am studying 1 Class
I am studying 2 Class
I am studying 3 Class
I am studying 4 Class
I am studying 6 Class
I am studying 7 Class
I stopped my studies at class 7
 """
```

C O D I N G

T I M E

class - Hey I belongs to this rigging Artist ***class***

```
#_ Code starts here !!  <Code theme - foundation>
class Artist:
        def __init__(self, name, age):
                self.name =name
                self.age =age
        def myName(self):
                return self.name
        def myAge(self):
                return self.age
        def myRole(self):
                return 'Rigger'
artist =Artist('Kiran', 35)
print artist.myName()
print artist.myAge()
print artist.myRole()
"""
Result (Printed) :
Kiran
35
Rigger
"""
```

---

def - Give (***define***) a name for set of actions, works or commands

```
#_ Code starts here !!  <Code theme: gruvbox-dark>
#_ Define Function
import maya.cmds as cmds
def printSelectedObjects():
```

```
        objList =cmds.ls(sl=1)
        if objList:
                for obj in objList:
                        print obj

printSelectedObjects()
"""
Result (Printed) :
pSphere1
pCube1
pCylinder1
pCone1
"""
```

---

del - Delete this variable, I don't need this anymore

```
#_ Code starts here !!  <Code theme: monokai-sublime>
#declare a variable named 'myVariable'
myVariable ='Hey dude, how are you\n'
print myVariable
"""Returns: Hey dude, how are you"""
del(myVariable)
print myVariable
"""

# Error: name 'myVariable' is not defined
# Traceback (most recent call last):
#   File "<maya console>", line 5, in <module>
# NameError: name 'myVariable' is not defined #
"""
```

What If I Do This ?

C O D I N G

T I M E

if **-** ***If*** you donate to [sightsavers.org](), then you can get a copy of this book. And monthly updates of this book also :)

elif **-** or else ***(elif)*** you have to give your free time to provide drawings to for the development of this book to get a copy of this book :)

else **-** ***else***, please wait till the year 2022, to get a free copy of it :)

```
#_ if-elif-else loop !! <Code theme : atelier-dune-light>
donatedVal = 'service'
if donatedVal == 'paid':
    print ('Show the payment receipt & Get the copy of the book')
elif donatedVal == 'service':
    print ('As drawings are provided, get the copy of the book')
else:
    print ('Please wait till 2021, to get the free copy of the book')
 """
Result (Printed) :
As drawings are provided, get the copy of the book
 """
```

---

try **-** ***Try*** completing the pending homework during this weekend

except **-** otherwise ***(except)*** expect that there will be a punishment

else - or ***(else)*** you need to leave the classroom

finally **-** remember that ***(finally)*** "leave the habit of keeping work pending"

```
#_ Code starts here !! <Code Theme : Railscasts>
try:
```

```
    print ('\n')
    print ("try: adding string to " + 2)
except:
    print ("added string to " + str(2))
    print ('"except:" block executed here')
else:
    print ('"try:" & "else:" blocks are executed here')
finally:
    print ('"Finally: block always executed"')
"""
Result (Printed):
added string to 2
"except:" block executed here
Finally: block always executed
"""
```

---

exec **-** ***exec***ute "Do what your father ordered"

```
#_ Code starts here !! <Code Theme: vs2015>
exec ('print "Father said to do homework today\nYeah... I did it"')
"""
Result (Printed) :
Father said to do homework today
Yeah... I did it
"""
```

All of you guys need to do this, but one guy at a time..

---

for **-** A keyword at the start of the ***for*** loop.

```
#_ Code starts here !! <Code Theme: solarized-light>
```

```
exec ('print "Father said to do homework today\nYeah... I did it"')
"""
Result (Printed) :
Father said to do homework today
Yeah... I did it
"""
```

---

global - Michael Jackson is ***globally*** known to everyone as he is a great dancer

This guy is known globally ..

```
#_ Code starts here !! <Code Theme: foundation>
#This variable is globally available when initialization of
#variable starts with keyword global.
global myName
myName ='Subbu'

def testIf_HeIsThere():
        print '{} is here'.format(myName)
testIf_HeIsThere()
"""
Result (Printed) :
Subbu is here
"""
```

---

in - Test if Anil is there ***in*** this group of artists or not

```
#_ Code starts here !! <Code Theme: xcode>
artistList =['Subbu', 'Venkat', 'Sam', 'Jan', 'Alex']
if 'Anil' in artistList:
```

Learn These

30+ Key words ..

```
        print 'Anil is in Artist group'
else:
        print 'Anil is not in Artist group'
"""
Result (Printed) :
Anil is not in Artist group
"""
```

---

is **-** That day, you were saying this about that guy, he ***is*** the same guy?

```
#_ Code starts here !! <Code Theme: tomorrow-night-eighties>
#Test  objects, which are on both sides of the keyword 'is', whether
they
#are the same
riggingArtist ='Subbu'
rigger ='Subbu'

if riggingArtist is rigger:
        print 'riggingArtist and rigger both are the same'
else:
        print 'Both are not the same'
"""
Result (Printed) :
riggingArtist and rigger both are the same
"""
```

---

lambda **-** Create function objects during run-time while executing the code

```
#_ Code starts here !! <Code Theme: tomorrow>
#_ Both the functions return the same in this case
whatIsHe =lambda name, role: name + ' is :' + role
whatIsHe('Jan', 'VFX Artist')

def whoIsThat(name, role):
        return name + ' is :' + role
whoIsThat('Jan', 'VFX Artist')
"""
Result (Printed) :
Jan is :VFX Artist
Jan is :VFX Artist
"""
```

---

not **-** You had the permission to come into the theatre but ***not*** your friend

```
#_ Code starts here !! <Code Theme: solarized-light>
exec ('print "Father said to do homework today\nYeah... I did it"')
"""
Result (Printed) :
Father said to do homework today
"""
```

---

pass **-** I am just ***pass****ing* through that lane but actually not doing anything

```
#_ Code starts here !! <Code Theme: sunburst>
```

I am just passing it on..

```
#_ Here nothing is happening. 'pass' is just placeholder for future code, #which you are going to write
def do_nothing_here():
        pass
do_nothing_here()
"""
Result (Printed Nothing Here) :
"""
```

---

print - Instead of just explaining it, can you give me the printout, so that I can read it

```
#_ Code starts here !! <Code Theme: tomorrow-night-eighties>
printRequested =True
if printRequested:
        print 'Yes.. print-out is provided'
else:
        print 'No print-out is made'
"""
Result (Printed) :
Yes.. print-out is provided
"""
```

---

raise - As some of your colleagues are harassing you, just ***raise*** a complaint

```
#_ Code starts here !! <Code Theme: xt256>
rigDelivered = False
```

```
if not rigDelivered:
  raise Exception("Sorry, I am yet to receive the rig")
"""
Result (Printed) :
# Error: Sorry, I am yet to receive the rig
# Traceback (most recent call last):
#   File "<maya console>", line 4, in <module>
# Exception: Sorry, I am yet to receive the rig #
"""
```

---

return **-** ***return*** whatever you got it as soon as you got something

You have got

Something in return ..

```
#_ Code starts here !! <Code Theme: tomorrow-night-blue>
global cash
cash = '100 USD'
def returnMoney(cash):
        return 'Returned {}'.format(cash)
returnMoney(cash)
"""
# Result: 'Returned 100 USD' #
"""
```

---

with **-** ***with*** your information, I can check this file one more time

```
#_ Code starts here !! <Code Theme: solarized-dark>
#_ Save a file C:/Users/<userName>/Documents/readMe.txt' with a
couple
#_ of lines in it, then run below code
from __future__ import with_statement
```

C O D I N G

T I M E

```
filePath ='C:/Users/Om/Documents/readMe.txt'
with open(filePath, 'r') as myFile:
        numLines = sum(1 for _ in myFile)
        print 'No of line in readMe.txt : {}'.format(numLines)
#_ Print each line from the file
with open(filePath, 'r') as myFile:
        for line in myFile:
                print line
"""
# Result:
No of line in readMe.txt : 2
Line No: 1
Line No: 2
"""
```

Yield - The woods do not ***yield*** another such a gem.

```
#_ Code starts here !! <Code Theme: tomorrow-night-bright>
#_ Use yield in place of return when you need performance
def return123():
        for num in range(1, 4):
                yield num
numbers =return123()
print type(numbers)
for num in numbers:
        print num
"""
# Result: '
1
```

```
2
3
"""
```

Color Coding - Maya Script Editor (01G)

Coding Is Colorful ..

When I first started writing python code, Maya 8.5 and below versions of Maya didn't have color coding. Those days are a bit boring to write code. When color coding was first implemented in Maya 2008/09, I found that coding is beautiful. We can see a few of those here.. Script Editor : Maya 2019

```python
from asNode import *

class as_eCtrlMain:
    def __init__(self):

        """To Support writing main tools while there are rep

    def as_eCtrl(self):
        """To Support writing main tools while there are rep
        MGlobal.displayInfo('eCtrl activated...!')

    def applyCtrlColor (self, ctrlList=None, colorNum=None
        '''
        Purpose : Change control colors based on prefix, co

        Args:[**shortArgs : ctrlList =cl, colorNum =cn, LPr
        -----
        ctrlList : ctrl(str) | ctrlList(strList) #_ ctrlLis
        colorNum : colorVal(int)                 #_ if numbe
        LPrefix : [prefix(str), colorVal(int)]   #_ Left si
        '''

        if not ctrlList:
            ctrlList =nselected()
            if not ctrlList: return None
```

Here goes all Keywords in this color

```python
from asNode import asNode as asN
from asNode import *

from pymel.all import *

class as_eCtrlMain:
    def __init__(self):

        """

    def as_eCtrl(self):
        """To Support writing main tools while there are repetitive tasks
        MGlobal.displayInfo('eCtrl activated...!')

    def applyCtrlColor (self, ctrlList=None, colorNum=None, LPrefix=None,
        '''
        if shortArgs:
            ctrlList =shortArgs['cl'] if 'cl' in shortArgs else ctrlList
            colorNum =shortArgs['cn'] if 'cn' in shortArgs else colorNum
            LPrefix =shortArgs['lp'] if 'lp' in shortArgs else LPrefix
            RPrefix =shortArgs['rp'] if 'rp' in shortArgs else colorNum
            CPrefix =shortArgs['cp'] if 'cp' in shortArgs else CPrefix

        if not ctrlList:
            ctrlList =nselected()
            if not ctrlList: return None

        ctrlList =[ctrlList] if type(ctrlList) != list else ctrlList
        ctrlList =map(asNode, ctrlList)
```

Eclipse Interpreter For Maya

Department-Wise Tools (0li)

Dude, Waiting For The Model ..

Modeling ..

In a typical Modeling pipeline, we use a lot of tools and do lots of tests.

I will be sharing here some useful links for free tools from various other developers which are available on highend3d.com for VFX & Animation pipeline. Some of the tools we use in the VFX pipeline for modeling are for Symmetry Testing, Extracting Blend Shapes, Wrapping Blend Shapes from old mesh to new mesh with the same topology etc.

abSymMesh 1.9.1 for Maya (Maya script)

https://www.highend3d.com/maya/script/absymmesh-for-maya

Tool Description:

A useful little script for building symmetrical and asymmetrical blend shapes. Check for symmetry, mirror and flip polygon geometry, mirror selection, and much more. ...ok, not much more, but it is pretty useful.

User Comments:

You can't be a modeler in Maya without this.

Rigging.

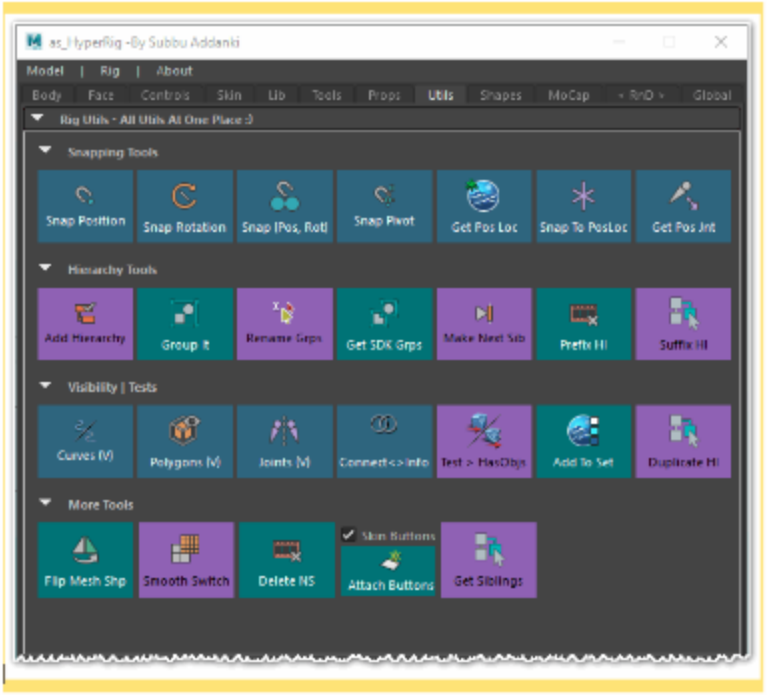
Throughout this book, most of the rigging concepts will be explained along with python programming concepts. Thereby, it creates the

Rigging.. That's My Favorite One.

ground for better understanding the pipeline requirements and how to come up with procedural solutions :)

Some of the tools we use in the VFX pipeline for Rigging are mainly modular auto rigging systems (Body & Facial Rigging Tools) and then skinning related tools. Then comes supporting tools like sticky / cluster-based deformer controls, Helper tools for control creation, import and export tools and a lot of tools like this ..

as_HyperRig - Modular Rigging System



We can categorize utility tools like Snap Tools, Hierarchy Tools and Toggle Tools and More Tools etc.

Here goes one of my free tools:

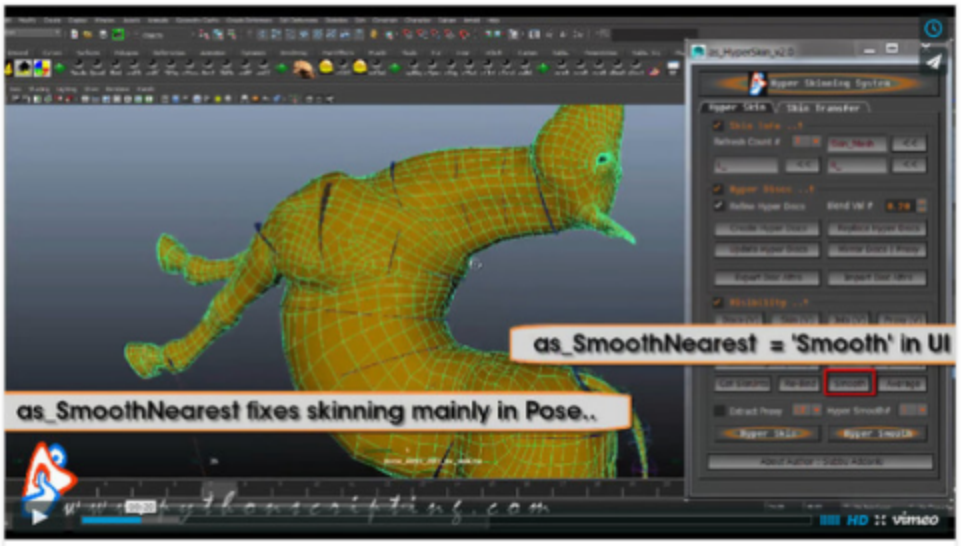
as_SmoothNearest (A magic feature from Hyper Skinning System)

https://www.highend3d.com/maya/script/free-as_smoothnearest-a-magic-feature-from-hyper-skinning-system-for-maya

Simple Python Tool, But Looks Like Magic ..

Tool Description:

Smooth Projected Vertices, Before You Know It. A Magic Feature From Advanced Hyper Skinning System (Auto Skinning For Maya)



as_SmoothNearest (A magic feature from Hyper Skinning System) for Maya

highend3d.com

And visit here for more.
https://pythonscripting.com

Texturing.

"Wow..Nice Textures" I Would Love To Hear This..

WIP
Coming Soon

Animation.

Guys just think about animation in simple terms. Playing every frame one by one within the given time frame. Without any break, let the flow go on.

I Love Only Swimming & Animation

Just think about a fish, which loves to swim in the water. You just got a crazy idea that this swimming animation should be automated. In the beginning, just don't think about animation principles that can be addressed in this automation.

When your thoughts are allowed freely to think about it. There is nothing to lose. Now slowly start thinking about automating it.

- It travels along the path and the path may not be straight. (Just imagine you are drawing NURBS curve along the path imagined by you
- Now, fish is translating along this path which means its position is changing with time along the path
- Now, give some wave motion to the body and tail. Just think about sine waves. (Here it may not go with exact animation principles)
- Once some automation is done in this direction, one can see some results. It may not be satisfactory in the beginning. But some results will come in that direction.
- Soon you may realize that study some of the animation principles might need to be studied and some RnD in python modules etc
- The amount of automation done at this point in time may not meet the requirements of VFX quality. But it may be useful at

crowd simulation level or basic gaming quality level.
- In the coming months and years, this swimming tool can be taken to the next level.

Why am I saying all this? Good question. Many times I heard that Programming is not for animators.. Keep going and keep thinking without any limitations in the beginning.

Some time ago, when the movie Ironman came out, we might have thought it's just a fancy visualization. But I noticed in one of the recent LinkedIn posts that that technology is really available. I have seen someone <name of that person can be given if possible> flying just like Ironman in the real world.

Scripting In Python - Awesome Journey (01j)

Not A Destination, A Journey Of Automation

Python Scripting

Loved This Ride ..

& I Want More Rides !!

Don't ever think "I will be done with Python Scripting soon" .. To start tool development in Python is in your hands. To stop it.. is not possible. Hahaha, **Many Python tool developers know it** and not only me
During the initial days of my tools development (i.e, 14 years ago), I thought I may soon not have much work in tools development. I came to eventually know that this is a foolish thought. .

Yeah.. I moved forward easily in every stage as the scripting language of tool development is Python. And the Beauty and Simplicity of Python always made tool development enjoyable. I might have failed while getting new Ideas but not implementing those ideas with Python

Scripting. For Example, while I started the Skinning Semi-Automation tool, a couple of my colleagues told me that it's impossible. But the simplicity of the Python language helped me to make it possible, where Hyper Skinning System was written by me. If I use C++ or Mel, This task would have been more complicated than I thought. It will take almost double the time to complete the tool.
Ideas are like if someone wants to do beautiful painting but he/she needs one of the thin brushes to complete the major part of the job. Here the thin brush is like Python Scripting. Python comes with many supporting modules. For example, NumPy module for numerical calculations etc

Python is really beginner-friendly and even friendly for advanced coders too, while C++ is a much more complicated and low-level language. C++ has more syntax rules and other programming conventions, while Python is just like imitating the English language.

Python Language always gave me enough tools to accomplish my RnD goals, which I initiated. One example is my ***Advanced Hyper Skinning System*** (A Semi-Auto Skinning System) tool- it took 4 years of my personal time to develop.

I hope you guys too enjoy this upcoming long ride of tools development in Python. Before moving forward, Just a few more thoughts. I know it's enough to introduce Python Scripting in VFX, but just 3 more thoughts.

Speed Typing, Building a Strong Base and Conclusion :)

Speed Typing - Useful Skill (01K)

So Flexible.. So Fast..

My Speed >> 30 WPM ..

Who wants a journey with a lot of breaks. With one or two breaks in our journey. It's so joyful.

Same way with tool development. The first and foremost thing to enjoy the tool development ride is .. 'Speed Typing Is Must.. !!'

You had a lot of curiosity to implement your ideas and wanted to see the fast results. Then why are you waiting? Try to get Typing Speed at least 30 WPM

Let's Just imagine, It's like someone is watching a nice movie. And he/she is forced to take 10 to 15 breaks due to personal work. Let's say each break lasts for 5 min. That means break time is almost equal to movie time. So, when he completes watching a nice movie?

Strong Base !! (01L)

My Base Is So Strong..
-Python Tool.

Sometimes, when we learn a new skill, we are in too much of a hurry to present it. This happens even while developing tools with Python Scripting.

REAL LIFE SCENARIO / EXAMPLE

Recently I came across a situation in India. For annual day celebrations, the school management decided to train and use the students. Due to lack of time, they built the performance stage in a hurry.

Just imagine what happened. When the time comes, and when the show is going on - the stage collapses! They lost all their joy and finally, it ended with tragedy. The original plan was to enjoy the show but the end is the tragedy.

Building supporting modules in Python is like building a strong base for main tools. If this is not good enough, tools will fail very frequently and when in need, just like that accident in annual day celebrations.

Conclusion (01M)

Python Scripting is Highly Addictive, Be Cautious !!

Get ready to become a Tools Developer !!

That's all I want to say !!

It's just like how kids are addicted to CAKEs

(Umm... Yummy... Python Scripting... So Tasty.)

'Buss... Jyada Ho Gaya', My Friend Sandeep Grover Is Shouting

(In Hindi)...

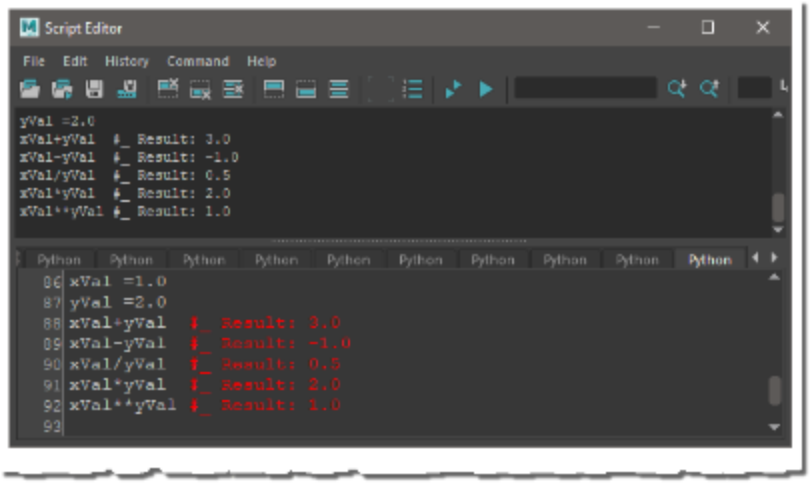
OK, Guys...

Let's Start Our Python Scripting Journey...

02.

Maya Script Editor

Interpreters



Let's Learn Maya Python Scripting ...

The Great Titanic Ship !! (02A)

Hey Guys, Welcome back to Chapter 02 of Learn Python Scripting From Life...

Maya Script Editor comes with nice features. Mainly...

- Auto Completion
- Color Coding
- Auto Indentation

No One Knows ..

When It Crashes ..

```
import pymel.core as pm
import maya.cmds as mc

def confirmAction(messageTxt='Confirm Action?'):
    '''
    Sends a given message through confirmDialog window
    '''
    confirm = pm.confirmDialog(title='Message ..!', message=messageTxt, button=['Yes', 'No'], defaultButton='Yes')
    if confirm == 'Yes':
        return True
    else:
        return trueVal

def message(messageTxt='Confirm Action?'):
    '''
    Sends a given message through confirmDialog window
    '''
    pm.confirmDialog(title='Message ..!', message=messageTxt, button=['Yes'], defaultButton='Yes')
```

But, it's always recommended to use external editors/interpreters.

The Titanic movie is well known and almost everyone has seen it. When I asked the participants of '**Learn Python Scripting From Life**' sessions, "What you remembered from the movie". Almost everyone replied, "Ship Crashed Due To Accident'. During the first half of the movie, there was a lot of fun and entertainment. But everyone remembered the fatal

accident.

The same thing will happen with Maya native script editors. ***When it crashes no one knows...*** When we start enjoying the results of our outputs from the script editor and forget saving the code, and then just imagine if Maya crashes, we will lose all the code written.

For small tests and all, Maya native script editors work fine. But when you guys are going to write a big chunk of code, Using Maya script editor is not a good idea. I suggest the usage of external editors/interpreters like ActiveState, Eclipse & PyCharm, Etc.

Interpreter Options

Input & Output Zones

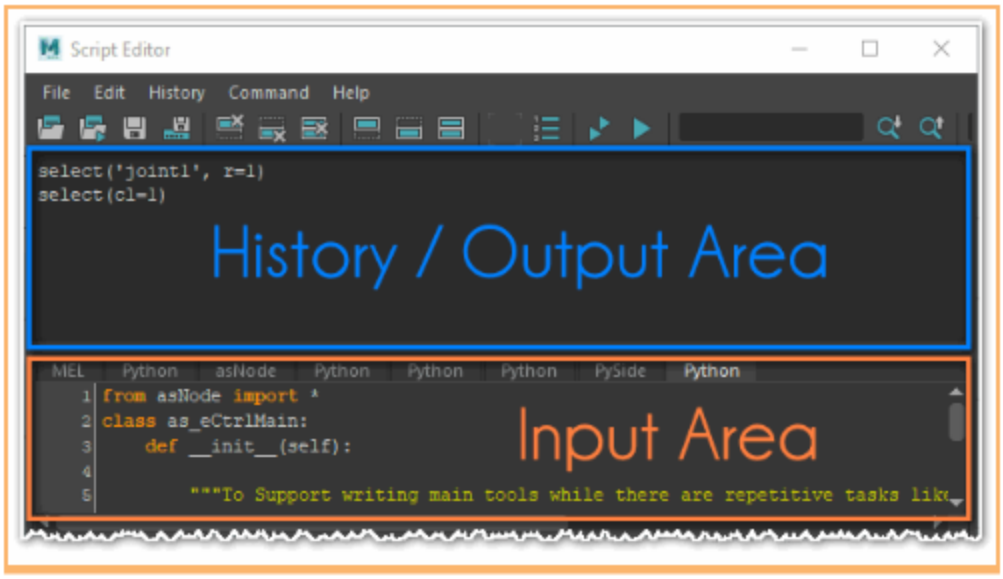
It's good to know parts or sections of Maya Script Editor. Major divisions of script editor are Input area and History (Output) area.

The input area is the place where we write the code and execute the same.

Once we execute the code, we can see the output/results of executed code in the Output area. And there are a few other buttons available for saving the code and loading the saved code, clear history etc.

Input Something ..

& Get Something Out ..



External Interpreters For Maya (02B)

ActiveState Editor >>>

Wow ..

Lovely Interpreter / Editor ..

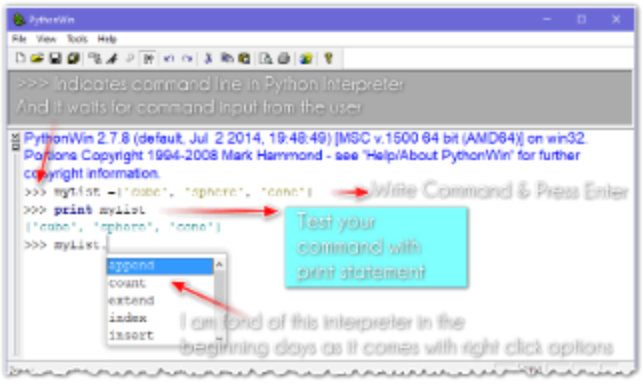
In the beginning days of my tools development, I used to depend on 'PythonWin' editor / interpreter. I am still using it for quickly testing single line general python code.

ActivePython 2.7 can be downloaded from the link below. ActivePython can't be linked to Maya

https://www.activestate.com/products/python/downloads/

Try to learn the following 3 things from this interpreter ..

- Write your command & Press Enter
- Test your command with "*print statement*"
- I was fond of this interpreter in the beginning days as it comes with right-click options. Try to append anything to the list by using right-click options. How to do it?
- Just type "myList" followed by the period or dot and wait for right-click options to appear
- Now append 'ball' to the "myList" by using this command: *myList.append('ball')*
- Now "*print myList*" again to see what myList contains. Now, myList is appended with 'ball' at the end.



Win Python..

With PythonWin

">>>" Indicates command line in PythonWin Interpreter

And it waits for command input from the user

Explaining about Eclipse and PyCharm editors is a bit heavy for those who just started learning Python for VFX.

Hey, Ec-lips-e.. You are always on my lips

I just want the readers here just to touch base with Interpreters.

Eclipse Editor >>>

This is my all-time favourite External Script Editor. Code can be executed directly from Eclipse editor to Maya with a few simple steps. Eclipse supports auto-completion for Maya Python, Maya Python API & PyMel commands.

One can find the below link to setup Eclipse IDE for Maya Python:

https://www.youtube.com/watch?v=FXs-cJAHZBI

PyCharm Editor >>>

As the name indicates, PyCharm is really fascinating to many developers these days. One major difficulty with PyCharm is to creating 100s of pointers is not an easy job, whereas in Eclipse, Just starting a line with "#---" creates a pointer and that's simple. I requested the feature in the PyCharm community, but till now I didn't see any action on that part. Other than that everything seems to be fine with PyCharm.

Where to download?

Please check this below link for community free edition

https://www.jetbrains.com/pycharm/download/#section=windows

To Connect Autodesk Maya and Pycharm IDE using MayaCharm

https://www.youtube.com/watch?v=DbIx3ds3Y4E

So, what PyCharm is saying about their product is here:

Be More Productive

Save time while PyCharm takes care of the routine. Focus on the bigger things and embrace the keyboard-centric approach to get the most of PyCharm's many productivity features.

Get Smart Assistance

PyCharm knows everything about your code. Rely on it for intelligent code completion, on-the-fly error checking and quick-fixes, easy project navigation, and much more.

Maya Commands & Arguments (02C)

Would Love To See ..

More Options ..

Just imagine a case, where we want to purchase a mobile with few specific options. Everyone wants basic mobile functions like incoming

and outgoing calls. The next immediate requirement is to use the internet on mobile.

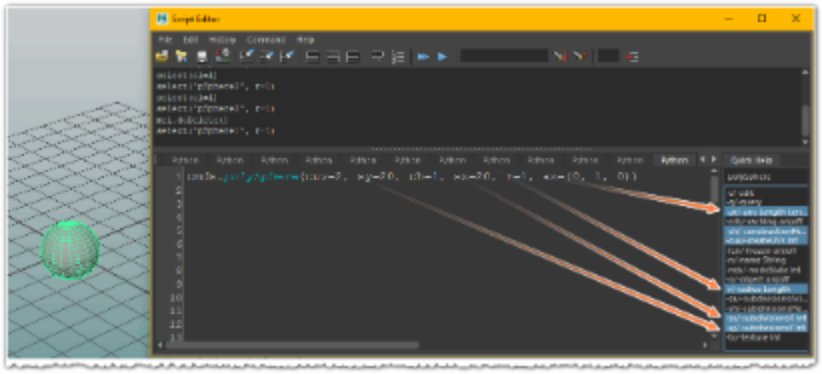
And then, what are the options a customer can think about these days might be the quality of the camera. Then comes, front camera and/or rear camera.

Now, Let's review the same in one of the python commands:

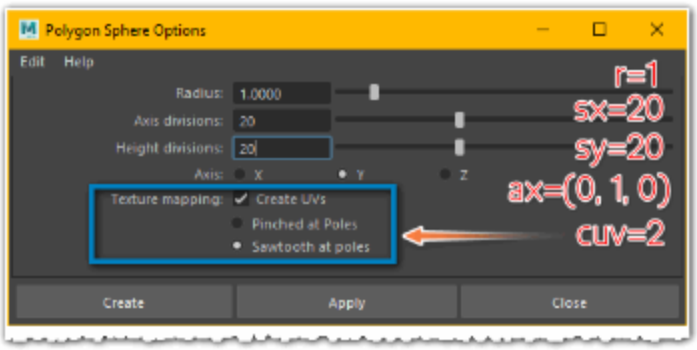
Python version of the '*polySphere*' command here:

```
import maya.cmds as cmds
cmds.polySphere(cuv=2, sy=20, ch=1, sx=20, r=1, ax=(0, 1, 0))
```

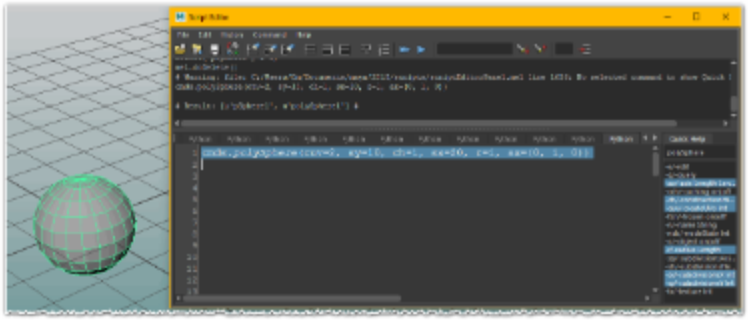
Now, let's review this command options :



Args/Options: cuv=2 (createUVs, Int), sy=20 (subdivisionsY, Int), ch=1 (channelHistory, ON), sx=20 (subdivisionsX, Int), r=1 (radius, Length), ax=(0, 1, 0)) (axis [Length, Length, Length])



Now Let's run the same command with argument/ option: sy=10. The result can be seen as below (It's nothing but the option: Height divisions =10 in the UI)



Python Interpreters (02D)

Color Coding

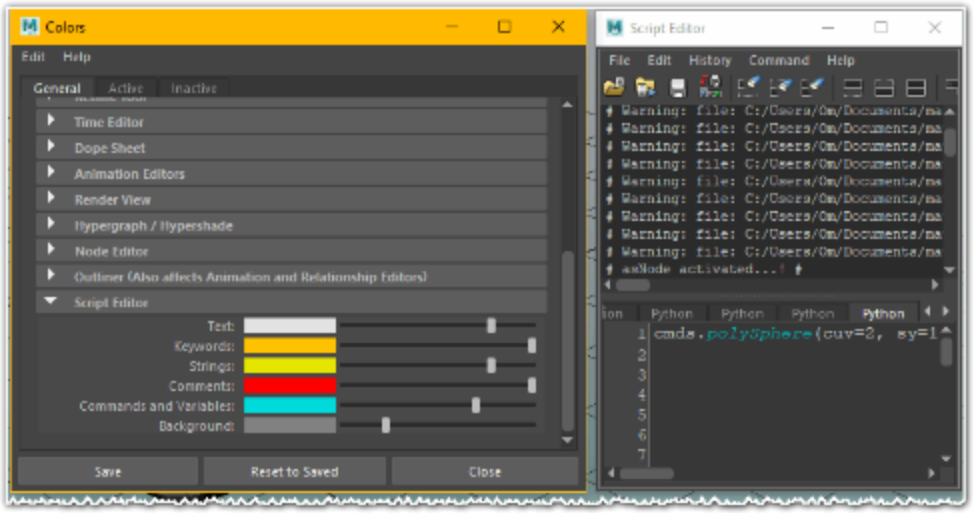
Every Color ..

Means .. Some Thing ..

Let's see one real-life example: traffic lights in India. Red indicates 'Stop', Orange indicates 'Prepare to stop', green means 'Go'. We are habituated to know when specific light glows and what it means,

though there is no text written there.

Similarly, Give your own colours to better identify your code quickly just by looking at colours. In this case, I am using green colour for '**Commands and Variables**'. Coder's life is beautiful when colour coding is with us :)

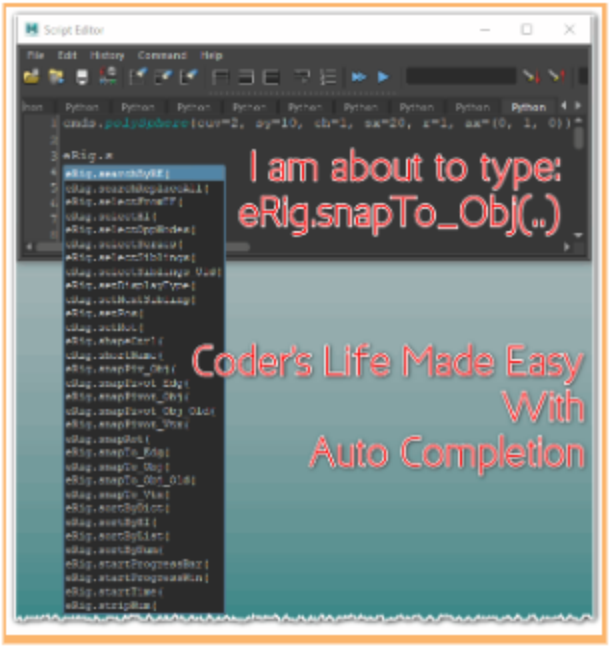


Auto Completion

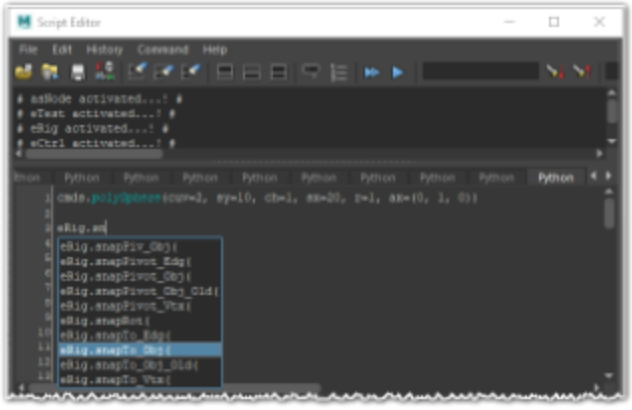
When we heard "Auto-Completion", there were a few things to remember. Suppose, we want to type 'Please let me know' in WhatsApp chat on mobiles. By the time we type 'P' or 'Pl', we can see these suggestions will pop up on mobile keypad like 'Please' | 'Please let'. Then we press that suggestion, instead of typing 'Please let me know' completely.

Auto completion in Maya python is almost the same. Please check the below example. Some time ago, I wrote a python module called "eRig". This module provides most of the common functions like snapTo_Obj,

snapPivot_Obj etc

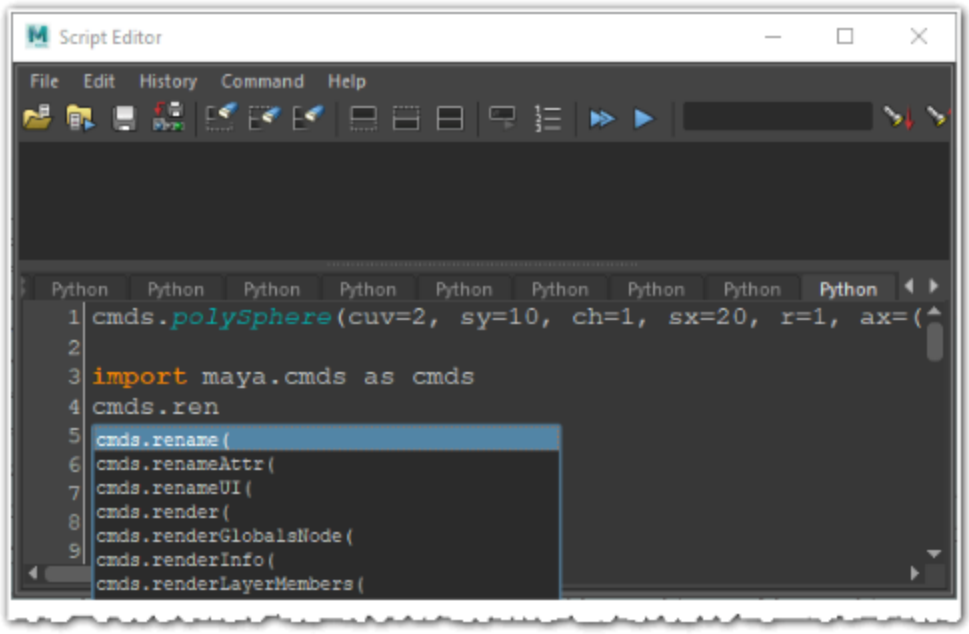


Try the above example, when I type "eRig.s", we can see a popup menu appear next to the letter 's'. And by the time I type "eRig.sn" we can see this pop-up menu is filtered and showing a few methods only. Now I can select the method, "eRig.snapTo_Obj(" and press the tab button. With this, our typing speed while coding will be awesome.



Now try this below code... By the time you type, "cmds.ren", you can see the auto completion pop-up menu

```
>>> import maya.cmds as cmds
>>> cmds.ren
```



Be A Player (Developer) (02E)

Rahul Dravid, Mr. Dependable

Be A Python Player -For Your Happiness

Just Try It .. You Will Be An Awesome Player ..

In the Indian cricket team, Rahul Dravid is well known as Mr Dependable. In the beginning days, you will be writing the code for your happiness. In that process, you might develop some awesome simple tools. These tools might be useful in the long run for many projects.

There might be some difficulty in writing code in the earlier days. Once you have written some small useful few lines of code, you can start

enjoying this journey of automation. When another code written by you is run, and you can see the UI (User Interface) with a couple of buttons and a few options like checkboxes, and it's admired by your team, you are Mr Dependable for your team soon as your team members start expecting more tools from you.

Conclusion (02F)

WoW..

It's My Play Zone ..

Your typing speed will increase soon and a few more modules will be developed. Soon, you will realize that you can remember so many Maya Python commands just like how you can remember your friends' names.

As tools are saving some amount of your and your team's time and as you are receiving little appreciation for your tools, you want to write more and more...

As the journey continues like "Tools Developer", soon you will realize that this editor is **"Your Play Area"** now.

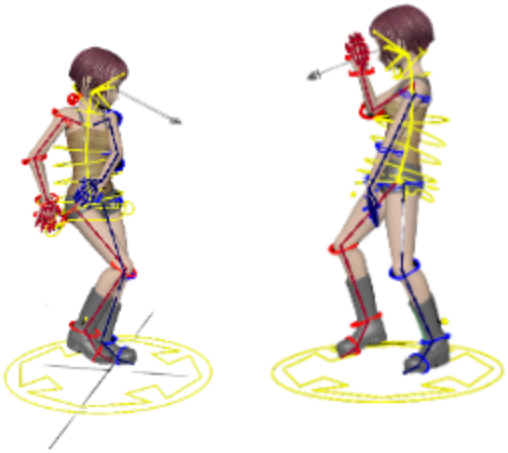
03. Syntax

-Command Line Structure

Syntax

Command Line Structure

Command Line Structure



"Syntax"

"""The syntax in the Python Scripting is the set of rules that defines how a Python program will be written and interpreted"""

```
cmds.polySphere (cuv=2, sy=20, ch=1, sx=20, r=1, ax=(0, 1, 0))
```

03 Mel Syntax In Maya

```
polySphere -r 1 -sx 20 -ax 0 1 0 -h 1;
# Result: 'pSphere1' in scene
```

Let's Learn Syntax ...

Just Learn One Line Of Code (03A)

Just For One Nail ??

Just Learn One Line Code .. That's it !!

It's Just One Nail..

When someone say that, "Just learn one line of code, and then you can do lots of stuff in the tools development process", what do you think, "Is it really possible?"

REAL LIFE SCENARIO / EXAMPLE

While I was watching a movie few years ago, there was some interesting scene.. Two groups named Eagles (Students) & Bulls (Rowdies) play a game called Rugby.. To save their college play ground

Rugby is a 15-a-side team sport. The object of the game is to ground the ball behind the opponent's try line, into what is called the in-goal area.

5 students, who support the Eagles team, from the audience clap whenever their team gains one point. There was a frustrated man next to these guys and for every point he

shouts, "*Just for one point?*" to discourage the 5 guys.. Finally these 5 guys fedup with this frustrated man. Once this man woke up from the seat to clap when the Bulls team made a point.. One of these 5 guys kept a nail on his seat and this man sat back.. You can imagine what happened.. Hahaha.. This man cries a lot.. There were so many laughs in the theater..

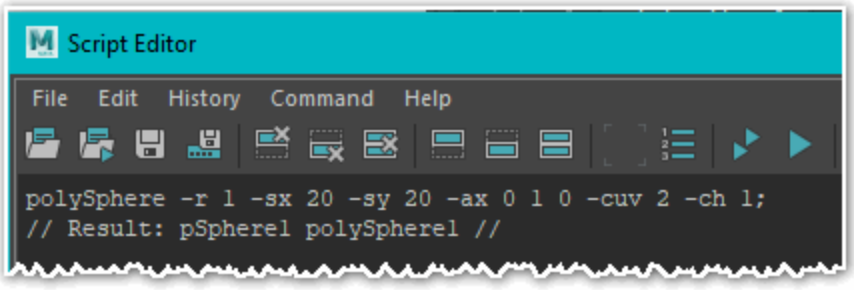
Follow...
This Center Line

Now let's see how it works and how simple it is. *Let's start with one line of Mel (Maya Embedded Language) code here...*

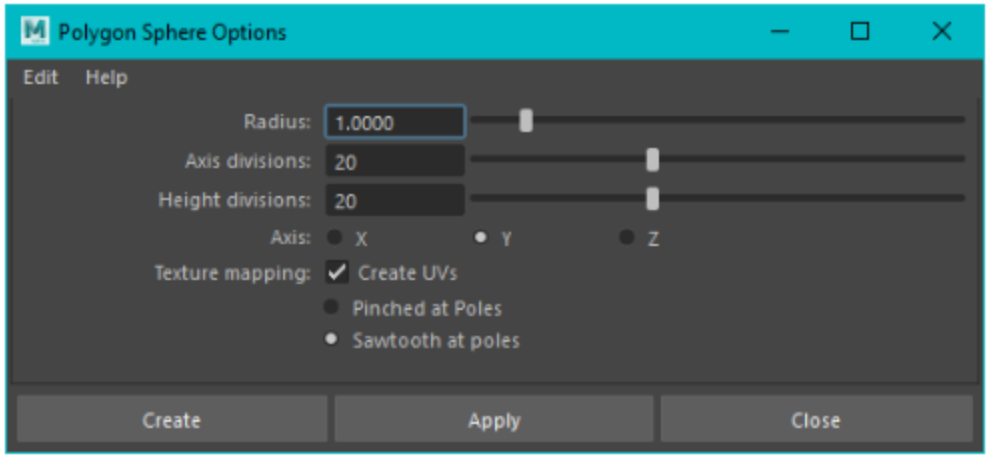
When we create a sphere from UI in Maya, we can find the following command in the history of the script editor:

Mel Version of the command :

polySphere -r 1 -sx 20 -sy 20 -ax 0 1 0 -cuv 2 -ch 1;



UI version of the same command :



Python version of the same command :

```
import maya.cmds as cmds

cmds.polySphere(cuv=2, sy=20, ch=1, sx=20, r=1, ax=(0, 1, 0))
```

Synopsis for polySphere command from cmds library is given below

command (Python)

polySphere

MEL version

In categories: Modeling, Polygons

No frames

Go to: Synopsis. Return value. Related. Flags. Python examples.

Synopsis

```
polySphere([axis=[linear, linear, linear]], [caching=boolean],
[constructionHistory=boolean], [createUVs=int], [name=string], [nodeState=int],
[object=boolean], [radius=linear], [subdivisionsAxis=int],
[subdivisionsHeight=int], [subdivisionsX=int], [subdivisionsY=int],
[texture=int])
```

Note: Strings representing object names and arguments must be separated by commas. This is not depicted in the synopsis.

Start...

One Line...

Custom Command.

Just Start Learning With One Line Of Code... This is how it works

- Mel Version of the command :
  - polySphere -r 1 -sx 20 -sy 20 -ax 0 1 0 -cuv 2 -ch 1;
  - "-r 1" : 'Radius of the sphere'
  - "-sx 20" : 'Number of subdivisions in the X direction'
  - "-ax 0 1 0" : 'The primitive axis used for the sphere'
  - "-ch 1" : 'Turn the construction history on or off'
  - "-cuv 2" : 'This flag allows a specific UV mechanism
- Python version of the same command :
  - cmds.polySphere(cuv=2, sy=20, ch=1, sx=20, r=1, ax=[0, 1, 0])
- Here is another one line command 'applyShader'
  - ***applyShader***(ml=[], sn='aiStandard', cv=[1, 0, 0], n='as_Shd')
  - Now you might have tried searching this command in Maya

    commands help page (as given above) & couldn't find it
  - But it's a custom command/python module (You can learn

    more about this in the chapter 11: Functions)
- Now it can be one of many methods of a module/ method of main tool:
  - eSpec.***applyShader***(...)
  - It's something like
    - cmds.polySphere(...)

applyShader...

- In this custom command's case, eSpec is a class. Format is as given below. Find more content about this in the chapter 13: Classes
- class eSpc():

```
def __init__(self):
    pass
def applyShader(self):
    pass
```

  - Now ***applyShader*** is just one of many methods in the eSpec module. Let's say eSpec is supporting module for main tool and it represents all specific functions related to VFX pipeline/ Rigging pipeline etc

- Now finally this ***applyShader*** can be used in one of the main tools.. For Example
  - EasyBird.applyShader(...)
  - Here, EasyBird can be an Advanced Auto Bird Rigging Tool
- Let's summarize here.. Every Time it's one line code.. In this process, one can define his/her custom command/ function/ class as per the needs
  - cmds.polySphere(...)

- eSpec.applyShader(...)
- EasyBird.applyShader(...)

Python Coding Style Guide (03B)

Neatly Packed Gift - All eyes are on It

PEP-8 Style Guide Of Python

Wowwww...
It's a Nice Gift.

That was your birthday. At the end of the day, your eyes are on one of the gift packs. The reason why it has got your attention is it's neatly packed with a gift wrapper with nice colors.

When your code is so clean and easy to read and well documented with proper comments, and when you revisit your code after a few years, you love the way you have written this code. Not only you, but your team members also love the way you have written the code

Pep-8 style code is industry standard. This document gives coding conventions for the Python code comprising the standard library in the main Python distribution. Please grab some useful information from the link below...

https://www.python.org/dev/peps/pep-0008/

Module Import (03C)

REAL LIFE SCENARIO / EXAMPLE

This is ...
Well Integrated Feature.

The magic show is going on in an auditorium. The host is welcoming the magician 'Antony' by saying this, "Welcome Mr Antony on to the stage". Since the host didn't mention "Magician Antony", some other one with the name "Antony" came to the dais, while the actual Magician was busy talking to his girl friend :)

Why is this happening? If the host can mention the name "Magician Antony" while inviting him, another Antony couldn't have come onto the dais. Here the name "Magician" acts as a namespace or identifier for actual Antony, who is supposed to come onto the dais.

Just ...
Snapped It Right !!

In the earlier example, when I call the function "eRig.snapTo_Obj(src, dest)", its function is to match the position of the source object to the destination object.

There might be another module like 'eShape.snapTo_Obj(src, dest)". This function might be doing the task "snapping vertex or shape to the target". Here the function name "snapTo_Obj" is the same in both the modules "eRig" and "eShape" but the functionality is not the same.

So, be careful who you are calling and which function you want. Let's see more about this function later in practical examples.

Well ...
It's Easy To Read... &
Easy To Understand...

Indentation (03D)

Indentation is nothing but tab spacing in python. It gives clarity about in which space the command is going to execute and many times it's related to the variable declaration. The variable declared in global space might be different from other variables declared in local space.

Most of the time, indentation is given at conditional statements where the body of the code starts for that conditional statement. And 'for.. loop' and 'while.. poop' are good examples to demonstrate this.

Hello ...
My Name Is ???

Naming Convention (03E)

Just imagine this, you are traveling from one country to another by flight and you need to pick your luggage once you reach the destination airport.

Waiting to pick the baggage from the conveyor belt in the airport. On the name tag, you have written your name 'Venkat'. Now you are seeing 2 different bags with the same tag names 'Venkat' only. You can identify the baggage by type of suitcase or something else. It's ok, but it's a different story.

Just imagine if you give your full name 'Venkat_Balaji', it's easy to recognise. In the above example, it's not a big issue. But all this is about avoiding error proneness.

And while declaring variables, please try to avoid very long names. You want to declare a variable for storing information about the candidate

Good Declaration :

```
#_ Code
candidateName = 'Venkat'
candidate_name = 'Venkat'
```

Bad Declaration :

```
#_ Code
#_ Can't read easily here
storinginformationaboutcandidate = 'Venkat'
```

Easy-To-Read Declaration :

```
#_ Code
#_ Readable but variable name is too lengthy.
storing_information_about_candidate = 'Venkat'
```

I Really ... Loved What I Have Written

Comments (Strings) (03F)

If we remember our school days, at least once everyone of us should have experienced this...

We want to write notes but the teacher is going a bit faster while explaining things. In a hurry, we also should have written notes quickly. After a month's time, if we see the notes which we only have written, few things can't be understood.

Comments are those which are useful while we want to edit our own

code after some time in future. If we forget what we have written, it will be difficult to edit the code later.

To write a comment in our code, start any line with *'#'*. When we run the below code, this line "***#_ Creating box control here***" won't' do any action. It's just only for information purposes and it won't raise any error while code is run.

Let's see the example below. In this case, we are creating a box shape control. Write a comment "***#_ Creating box control here***"

```
#_ Code starts here..
#--------------------
Import maya.cmds as mc
#_ Creating box control here
boxCtrl =mc.curve(n='as_BoxCtrl',
p=[(-0.5, 0.5, 0.5), (0.5, 0.5, 0.5), (0.5, -0.5,
0.5), (-0.5, -0.5, 0.5), (-0.5, 0.5, 0.5), (-0.5, 0.5,
-0.5), (0.5, 0.5, -0.5), (0.5, -0.5, -0.5), (-0.5,
-0.5, -0.5), (-0.5, 0.5, -0.5), (-0.5, -0.5, -0.5),
(-0.5, -0.5, 0.5), (0.5, -0.5, 0.5), (0.5, -0.5,
-0.5), (0.5, 0.5, -0.5), (0.5, 0.5, 0.5)],k=[0, 1, 2,
3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15],d=1)
```

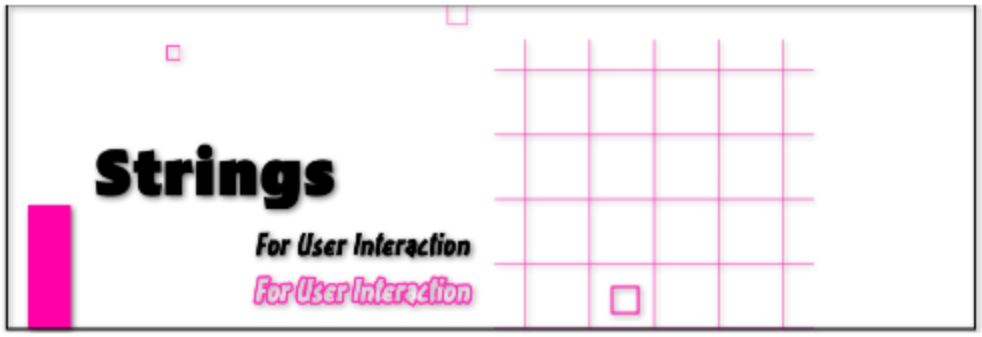
By writing these comments and when we come back to revisit this code later, we know that this particular piece of code is to generate box shape control. When the above code is run from Maya script editor, Below box shaped control will be created.

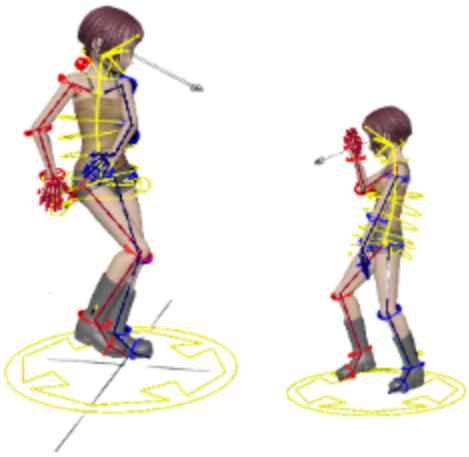
Tips & Tricks

Use Triple Quotes
For Comments
If No of lines
goes beyond one
or two lines.
For Ex:
'''
Creating Box Ctrl
Change The Shape
If Needed
'''

04. Strings

-For User Interaction





"Strings"

"""Strings are arrays of bytes which represents Unicode characters
In other terms, A string is a sequence of Unicode characters with single or double or triple quotes"""

04

```
aStr = 'I am in string'
>>>aStr.title()
# Result: 'I Am In String'
```

Let's Learn Strings ...

Print To Screen (04A)

Let Your Users Know

Print To Screen

"Be careful--with quotations, you can damn anything."

— Andre Malraux

REAL LIFE SCENARIO / EXAMPLE

Let's consider this example..

All of us are using mobile phones. We have seen many cases here like when we do a financial transaction, we immediately get a message on our mobile screen that a particular transaction has been done.

Srinivas is playing a game where he needs to select some option every time on the screen, so that he can play the game as he wishes.

Jan is playing a nice game, whenever he is done with particular level of the game, he gets a popup message on the screen that

he did really well at that level of the game and he can go to next level of the game

Manas went to the ATM to withdraw some money. When he tried to withdraw some money, a pop-up message came that funds were not available in the account. He was a bit sad and returned home.

Most employees are very happy on the month end to see a sudden pop-up message that you received salary for the month of ...

Yeah... all these are nothing but some programme is printing to your screen. This screen can be either a mobile screen or ATM screen or desktop screen.

It's the same with tool development, where you are the backend programmer, but one of your tools users can be anyone of these above guys. During runtime, they may need to make some decisions. During the tools development we need to consider all these real-life requirements. In the below example, if... else... statement is used. We can learn more about it in the chaper 08 : Conditional Statements

```
#_ Code starts here <Code theme: atelier-cave-light>
balanceIn_Account =1000
```

```
amountToBe_Withdrawn = 1500
if balanceIn_Account < amountToBe_Withdrawn:
   print "Hey man, you don't have enough funds in account"
else:
   print "{} amount withdrawn'.format(amountToBe_Withdrawn)
"""

#Printed:
Hey man, you don't have enough funds in account
"""
```

Strings Concatenation (04B)

It's safe now..

Bogies are all connected properly ..

Please get into the train ..

REAL LIFE
SCENARIO / EXAMPLE

Everyone of us loves the train journey at some point of time. When we see the train from a long distance, it looks like Entire train looks like a single body like a snake. But when we are getting closer we can notice that each compartment/bogie is connected with one another with linkages. If those linkages are not there, the next compartment won't follow the front one.

In this case, how these are connected, the same way when different strings are connected with '+' that's called string concatenation.

So, what's this string actually? Any letters, numbers, or words in a sentence are called strings when these are enclosed in single/double quote marks.

For Example, Let's consider this to understand why we need this kind of string operation.

Ram and Laxman are traveling in a train. In the middle of the journey one of his friends is joining. For now that's a surprise. Once his friend joined them, then only then came to know his/her name.

Coding Time...

```
#_ Code starts here !! < Code theme: vs>
friend_03 ="Krish"
print ('Ram and Laxman are thinking about who is that friend joining them')

#_ During the middle of the journey, they came to know that Krish is joining them
#_ Let's write above print statement like this
print ('Ram and Laxman came to know that ' + friend_03 + " joined them")
#In the second print statement, we can notice that '+' is acting as linkage which
# joined the train compartments
"""
#Result (Printed):
```

```
Ram and Laxman are thinking about who is that friend joining them
Ram and Laxman came to know that Krish joined them
"""
```

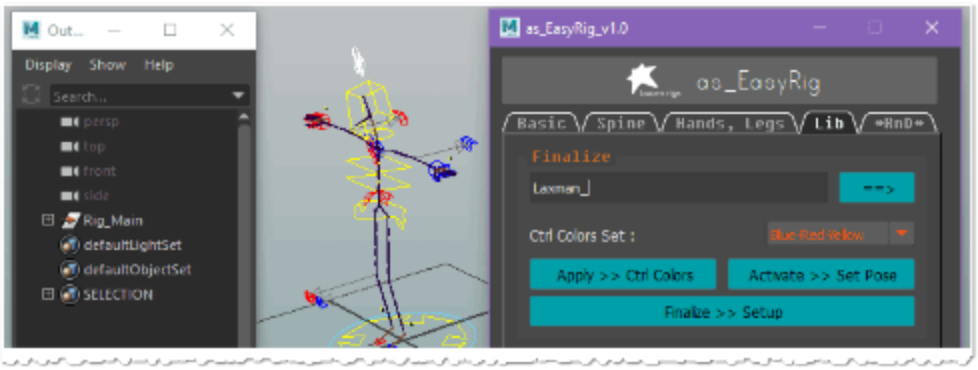
Let's see where it's needed in a typical rigging pipeline.

Suppose a rigger has completed the setup using one of the auto rigging systems available. And he is yet to finalize the rig with the character prefix 'Laxman_'. In this image we can see the outliner before finalizing the rig

Nice..

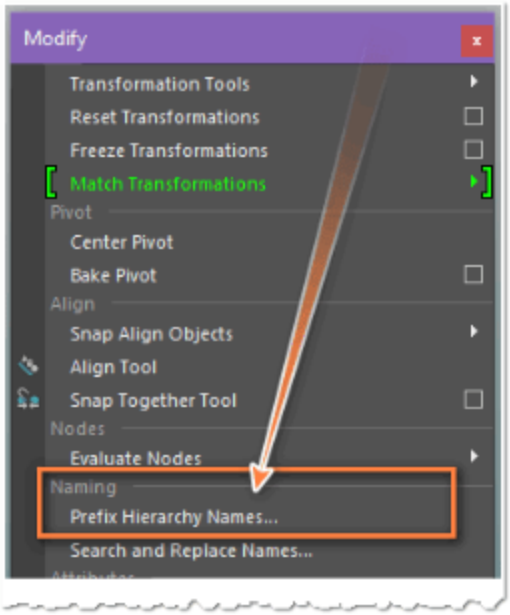
I am just rigged by Subbu.

My name is Laxman ..

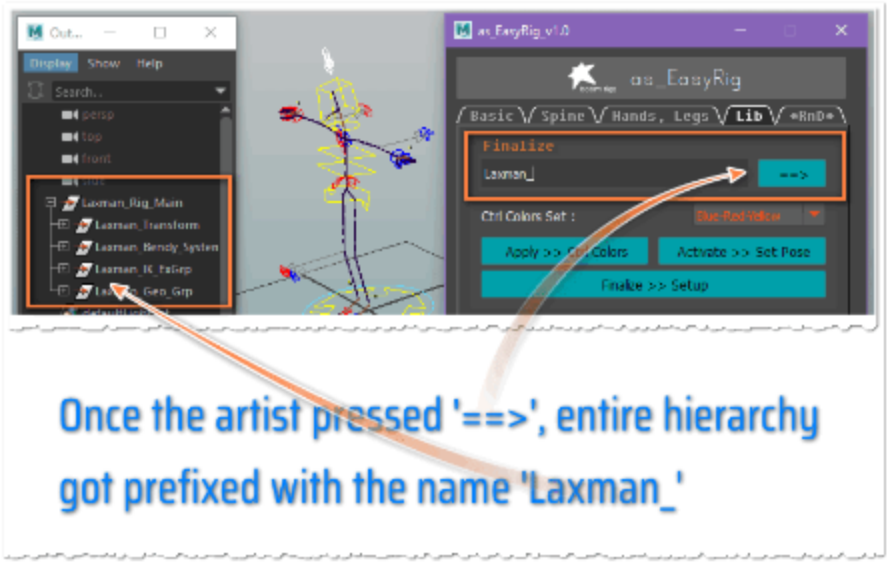


During runtime the end user of your tools (one of the riggers) will decide which character prefix to be given to the specific rig. Initially this name is not known. This tool can be used for any human characters. But the name of the character can be given only when he comes to know about that character name.

That means the end user of your tools will make some decisions during runtime. As a developer, this kind of facility can be provided from the UI itself. This is just an example.



Even Though, it can be done using '**Maya -> Modify -> *Prefix Hierarchy Names***' from Maya native commands, in this case when artist executes this button '**==>**', it can prefix all and it can do some other functions like 'Removing unwanted nodes from the scene or coloring the controls etc'



Type Casting (04C)

Convert Type As You Needed ...

Type Casting

REAL LIFE SCENARIO / EXAMPLE

Hehe hey.. hey...

I am the tiger now

Don't dare come towards me ..

Mithun is a well known artist for his voice modulations while singing. One day he went to one of the famous director's office to give a test. He just got down from the cab and was about to enter the office. He just started coughing a little. He blamed himself for having a cool drink an hour ago before the test with his friend, whom he met after a long time.
Somehow he managed to sing a song that day. But he was not that happy. So, he attended the test once again after a week's time. Director was pretty happy with his voice and for his command of the song. And the Director requested to sing the same song with different modulations to reach the expectations of different groups of audience. And Mithun did it well.

Type Casting

Type Casting is nothing but the conversion of one variable data type to another variable data type based on the various needs

In this case, the lyrics of the song are the same in 3 different situations, but modulation changed based on various requests

from different groups of the people.

Type Casting is the method which is used to convert one variable data type to another variable data type based on the various requirements.

Let's consider the previous example one more time. Before starting the journey, Ram purchased 2 tickets for both of his friends. How do we write the same in Python

```
#_ Code starts here < Code Theme: tomorrow-night-blue>
numTickets =2
print 'Ram purchased ' + numTickets + ' tickets'
"""
Result:
# Error: cannot concatenate 'str' and 'int' objects
# Traceback (most recent call last):
#   File "<maya console>", line 2, in <module>
# TypeError: cannot concatenate 'str' and 'int' objects #
"""


#With type conversion using str command
#str is the command used to convert other possible types to string
print 'Ram purchased ' + str(numTickets) + ' tickets'
'''
Result:
Ram purchased 2 tickets
'''
```

In the above case, Number 2 is converted to string '2', so that it can be added to another string.

In the below example, we use the re module. Let's learn more about re module in the Regular Expressions section. In this below Maya Python example, we can learn extracting number from a vertex name

```
#_ Code starts here !! <Code Theme : github-gist>
import re
'''
To extract the number from end of the object name & convert to
string using Type Casting function : str()
Usage:
------
obj.vtx[105]  # Returns 105
obj.e[206]    # Returns 206
'''
objName ="obj.vtx[105]"
testObj =re.search('([0-9]+)$', objName)
if testObj:
        num =int(testObj.group())
        numStr =str(num)
Print numStr
'''
#Returns : 105
'''
```

String Quotes (04D)

For a second, just ignore what the string quote is. The moment we hear about the quote, we remember that there is some message and it's with the single quotes or double quotes.

At some point of time in life we like atleast couple of quotes. What else these real-life quotes say is that there is something which has certain importance and it conveys some message.

In the programming language also these are very useful in many places.

They came very next to definition, where some explanation will be there about what that function is all about.

Time to confirm .. Something..

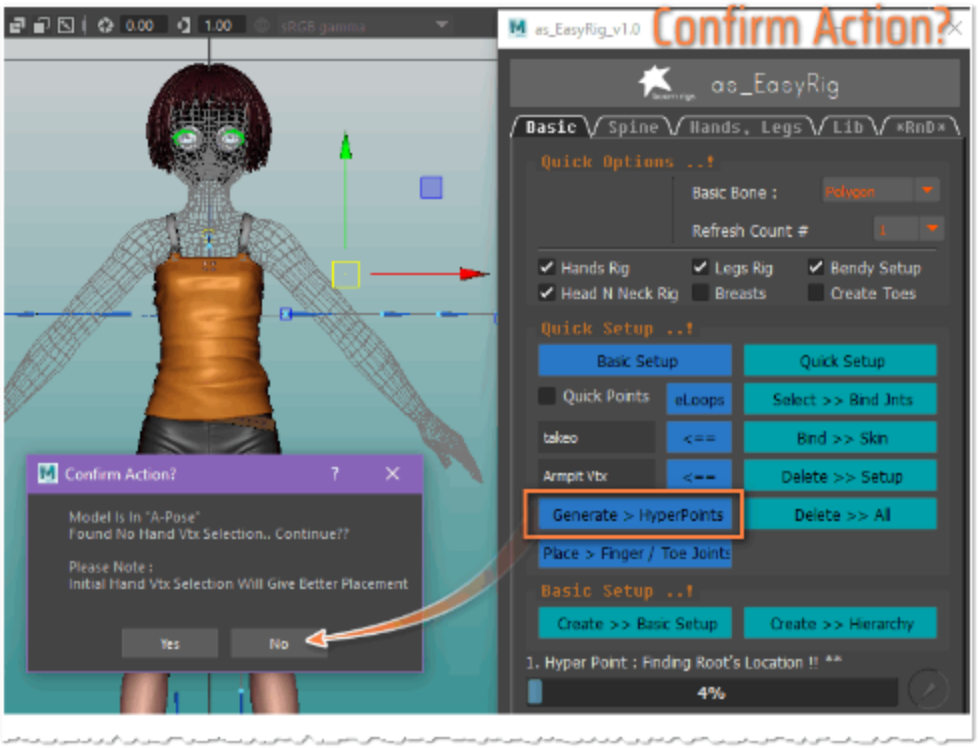
Some important information will be written on how to use this function.

By the way, whatever is written within these quotes, it may be for the actual execution part of

the tool's functionality or it may be useful for the author of the tool or for the end user of the tool. Sometimes, it's mainly for conveying error messages or to share some information about the functionality of the tool.

And sometimes, it will be used for asking the user to choose the option during the runtime of the code... It's something like when we try to get the money from the ATM, and if we see the error message like there are not enough funds to withdraw. This error message helps to understand the fact that there is not enough money to be withdrawn as per the requested money. After seeing that pop-up message, the user will choose the option to cancel the transaction.

Let's see one example in Maya, "Confirm Action ?"



In the above tool, the button '***Generate > Hyper Points***' is pressed, it's doing auto joints placement. That means, the tool is trying to find joint placement automatically to the nearest possible placement.

In that process, the tool has identified that character model in A-Pose. And it tries to tell the user that since the model is in A-Pose, by selecting a few hand vertices can give faster action in the process. So, the end user of your tool can take necessary action based on the suggestion given by the tool. In this case it's nothing but a string, which just pops-up here when the situation arises. So, make use of strings as much as you can. Right string (message) at the right time helps a lot..

Hold On ..

I have some news ..

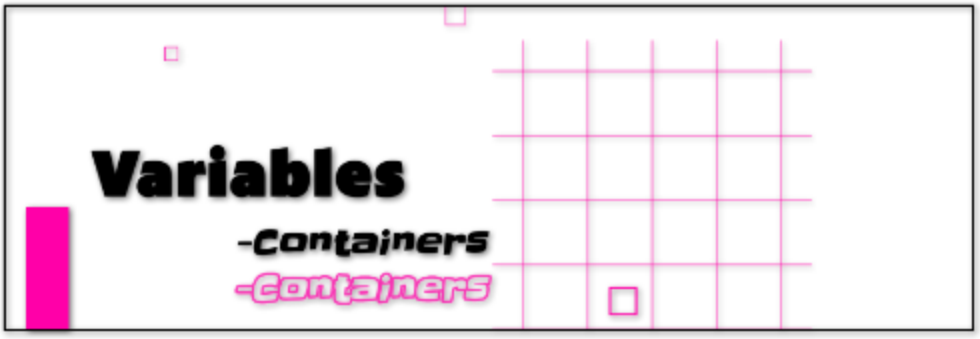
If we can use these strings effectively, that makes a bigger difference during the long run. Consider this need like we have written some code during the hurry and without giving any notes what it does. And we want to revisit this code after a couple of years. When we look at that old code, we may not understand well why that particular function is all about or what the specific variable is doing in that particular function.

Triple quotes (“””..blah blah...”””) can be very much useful to explain what that function is all about. Double and single quotes can be used to print something to the console or to output some information to the end user of the tool.

In some of the cases it’s really useful to process the information which is available in the form of strings.

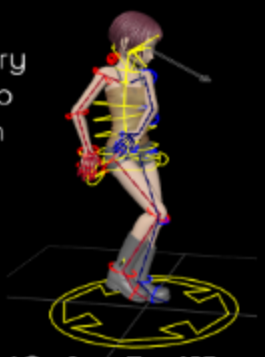
05. Variables

-Containers



aVar="It`s variable"
aVar.title() > "It'S Variable"

1) Variable is a name that refers to a memory location and is used to contain a value. When we query / run this variable, it can return what it contains



asVar =['Spine01', 'SpineEnd']

asVar.remove('SpineEnd')

print (asVar [1:])

And More

Let's Learn....

Variables.

What Is Variable (05A)

Hi Chinnu ...
Let's have some Tea .

Chinnu & Bannu - NickNames

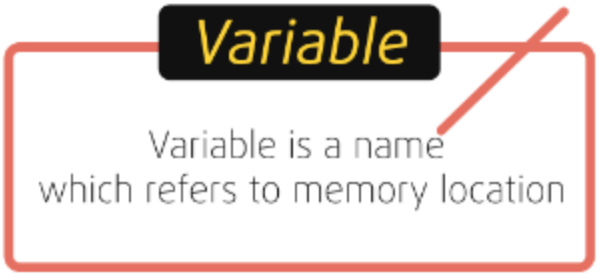
What Is Variable

REAL LIFE
SCENARIO / EXAMPLE

During the Corona pandemic times (2020-2022), Many of us worked from home (WFH). During this time at least some of us might have tried something new in the kitchen like making Tea or some tasty dish etc..

Suppose one of us has tried making Tea in the kitchen. At some point in time, he is looking for sugar in a specified container (bottle). Let's say, what if there is salt in place of the sugar in that container. Yeah... sometimes it's possible.

In this case, Container is nothing but a Variable. Even Though it's there for sugar, salt also can be kept in that container. Whatever this container (let's say, 'Sugar Variable') contains, it returns the same when someone is looking into that container...



Variable is a name that refers to a memory location and is used to contain a value. When we query / run this variable, it can return what it contains

Declare Variables (05B)

Can you ...
Get Me That Sugar Candy From That Bottle Please.

Let's declare couple of variables and check it once in Maya Python

```
#_ Code starts here !! Code Theme <'Theme: Foundation>
snackBox ='Sweet Box'
snackBoxContains ='Gulab Jamun' # Initial content
print snackBoxContains
#'Gulab Jamun'
snackBoxContains ='Salt' # Content changed
print snackBoxContains
#'Salt'
```

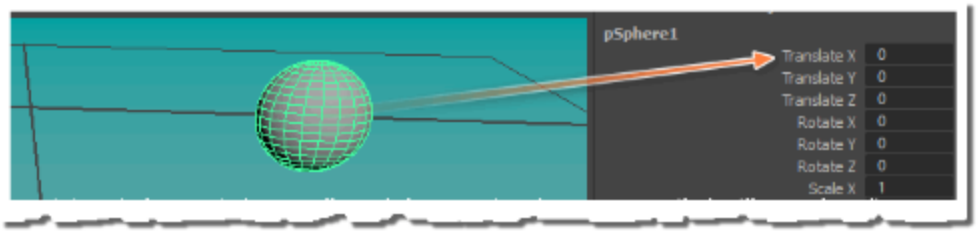
Let's write another variable with a Maya Python example which is 'To freeze any object which has some transformations'. Create a sphere in

Add Me ...
Some Value

Maya scene with the name 'pSphere1' and move it in translateX for 3 or 5 units


```
#_ Code starts here <Code Theme :
#_ "To freeze transforms of moved object" ...
#_ Let's Create one 'Sphere' and move in translateX for 5 Units
#-Import maya.cmds library as mc
import maya.cmds as mc
#-Declare the variable named obj for selected object 'pSphere1'
obj ='pSphere1'
#Move the sphere in X for 3 units
mc.setAttr(obj + ".translateX", 3)
#- Select the object
mc.select(obj, r=1)
#- Freeze it
mc.makeIdentity(apply=True)
mc.select(cl=1)
```

After running the above code, we can see that translateX of 'pSphere1' is freezed... As shown in the below image.



If we can define the above code as a function, it looks like the one below.

Instead of writing a few lines of code every time, we can call all the lines with one name 'freeze'. It's like instead of remembering 5 lines of code, we can remember the same with one variable name.

This function 'freeze' can be a method of any particular class.

We can learn more in 'Chapter 11: Functions' & 'Chapter 13: Classes'.

Freeze Me...

```
#_ Code starts here !!  <'Code Theme:  darcula>
def freeze(obj, **kwargs):
        '''To freeze transformations of any given object'''
        if not kwargs:
                kwargs ={'t': 1, 'r':1, 's':1}

        mc.select(obj, r=1)
        mc.makeIdentity(apply=True, **kwargs)
        mc.select(cl=1)
#-Declare the variable named obj for selected object 'pSphere1'
obj ='pSphere1'
freeze(obj)
```

Local & Global Variables (05C)

Hey ...
How do you know my Personal Number ...

REAL LIFE SCENARIO / EXAMPLE

One day, Radha got a call from one of her friends. And she was surprised that, "how come my friend knew my personal mobile number which I never shared with anyone".

Someone might have tried this... These days most of the mobiles come with 2 sim slots. One sim card for internal / limited to family members use. And another one for internal, external and for social relations like friends, colleagues and for generic needs.

CODING

Let's check this below example, Initially the globalSim variable is declared outside of the function localFunc. Even though it's assigned with another value within the localFunc, Finally when it's run, it returns (prints) the same which is assigned in the beginning. When localFunc is run, it returns/prints local value

TIME

```
#_ Code starts here !! <'Code Theme: Foundation>
globalSim ='Sim Card For Generic (Global)'
print globalSim
# Result: 'Sim Card Generic (Global)' #
def localFunc():
        #print globalSim
        globalSim ='Sim Card For Family in localFunc'
        print globalSim

localFunc()
# Result: 'Sim Card For Family in localFunc' #Local
print globalSim
# Result: 'Sim Card For Generic (Global)' #
```

Hey...
What Type Is This?

Variable Types (05D)

Unlike in C++, Python Variable type is detected automatically on the data type of it's value. There is no need to declare the data type for variables specifically. Every value in the Python variable had a data type.

Value for any selected variable can be changed any time. And data type for that variable can be updated automatically. Different data types are

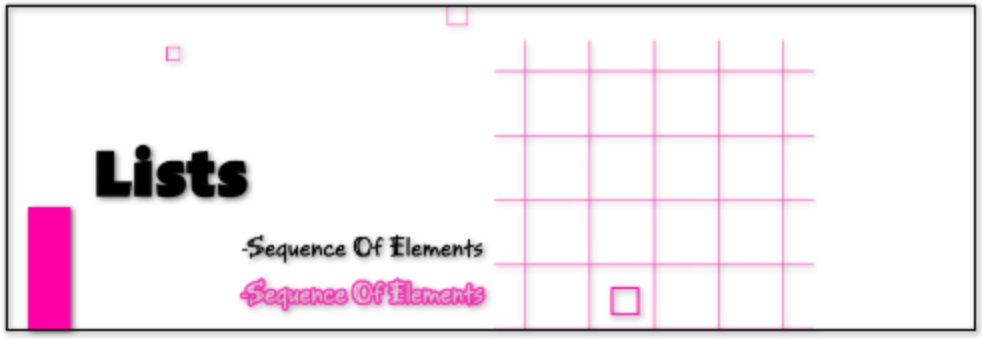
-Numbers, Strings, Lists, Integers, Floats, Dictionaries & Sets etc.

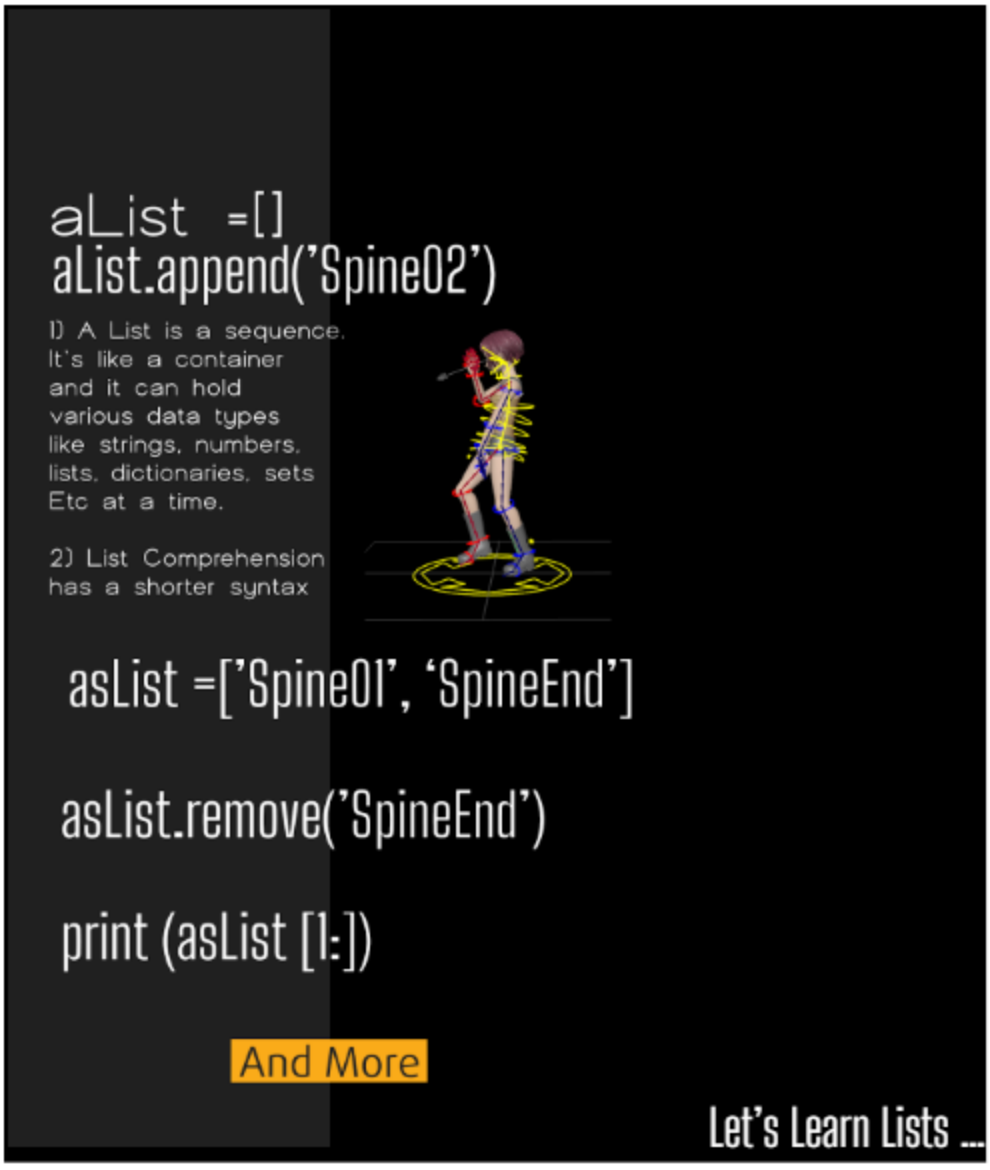
type() Command can be used to find out the data type (class) of that variable. Data types are nothing but actual classes like strings / integers and variables are instances of that classes


```
#_ Code starts here !! <Theme : gruvbox-dark>
#_ Declaring variable for the first time
#////////////////////////////////////////////////
vtxNumOrName ='Head.vtx[256]'
print(vtxNumOrName)
# Result: 'Head.vtx[256]' #
print(type(vtxNumOrName))
# <type 'str'>
#_ Assigning number to same variable later
#////////////////////////////////////////////////////
vtxNumOrName =256
print(vtxNumOrName)
# Result: 256 #
print(type(vtxNumOrName))
# <type 'int'>
```

06. Lists

-Sequence Of Elements





Get The List Of Items (06A)

The List Of Items... Is Ready Dude !! What To Do Next ??

Shopping List: List Of Items To Purchase

Get The List Of Items

One can say that parents are our first programmers who taught us about programming concepts like Lists. At some point in our childhood, they taught us how to do some household work, like preparing a list of items to purchase, on our own.

This is something we already discussed in the first chapter : '*Introduction To Python*'

Now let's make a list of items from the Maya scene and do some operations on the same.. We can take an example of controls/control shapes from one of rigs..

WIP
Coming Soon

Methods Of List (06B)

My Methods Are... So Much Useful -Python List

Shopping List: Edit List Of Items

Methods Of List

A python list class has the following methods. Python lists are the most useful data types which help us to work with multiple and different elements (like numbers, strings etc) at a time.

[append, remove, insert, count, extend, reverse, sort, pop, index]

01.append

06B | Methods Of List

Now, Let's see one example of creating an empty list and appending items.

Method description & Syntax:

- *Add an item to the end of the list. Equivalent to a[len(a):] = [x]*
- *list.append(x)*

Provided here Simple Example and curve CVs Example using Maya Python. For Loop is used in the 2nd example. More about 'for loop' is in chapter 08: Conditional Statements. The method 'append' is the most commonly used one in Maya Python. The 2nd example can be done using some advanced methods like Maya Python API and python's native list comprehensions also.