STEAM CLOWN™ PRODUCTIONS

PYTHON - RUNNING THE PYTHON SHELL (RUNTIME INTERPRETER)

Last Updated: Saturday, January 19, 2019

STEAM CLOWN™
& Squeaky Hinge
PRODUCTIONS

Objective, Overview & Introduction

1. You will be able to run the Python Runtime Interpreter, and enter valid line of Python code. You will create Python functions and pass parameters
2. You will use the Python Runtime Interpreter, and screen capture your output
3. You will turn in your screen shots to show you have been able to run the code

STEAM CLOWN™ & Squeaky Hinge PRODUCTIONS

STEAM Clown™ Productions

Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)

See Appendix A,B,C, for Licensing & Attribution information

These slides are an adaption, to better target my SVCTE High School Mechatronics Engineering class, primarily from Dr. Charles R. Severance's Python for Everybody class https://www.py4e.com/ … but from other sources as well. See Appendix A

CC BY-NC-SA 4.0

https://creativecommons.org/licenses/by-nc-sa/4.0/
https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode

GNU Public License

Any included Programming Code Is licensed under the GNU General Public License v3.0

EUPL (European Union Public Licence) Code and Content is also licensed under the EUPL 1.2 or later

How Will You Be Measured

- Your mastery of entering code into the Python Runtime Interpreter will be demonstrated by screen capturing the code output and turning in your results
- The more examples the better your grade, but you should turn in at least 5 screen shots showing different code examples

New Words or Concepts…

- Python Runtime Interpreter
- Function
- Parameters
- Type
- ID
- Mutable
- Immutable

STEAM Clown™ & Squeaky Hinge Productions

OK, Remember Where to get Resource Support? Dr. Charles R. Severance

- We are going to use a few resources on the internet…
- Bookmark and remember a few sites…
  - SVCTE Mechatronics Python Resource link
    - Python Resources
- Python 4 Everybody - https://www.py4e.com/

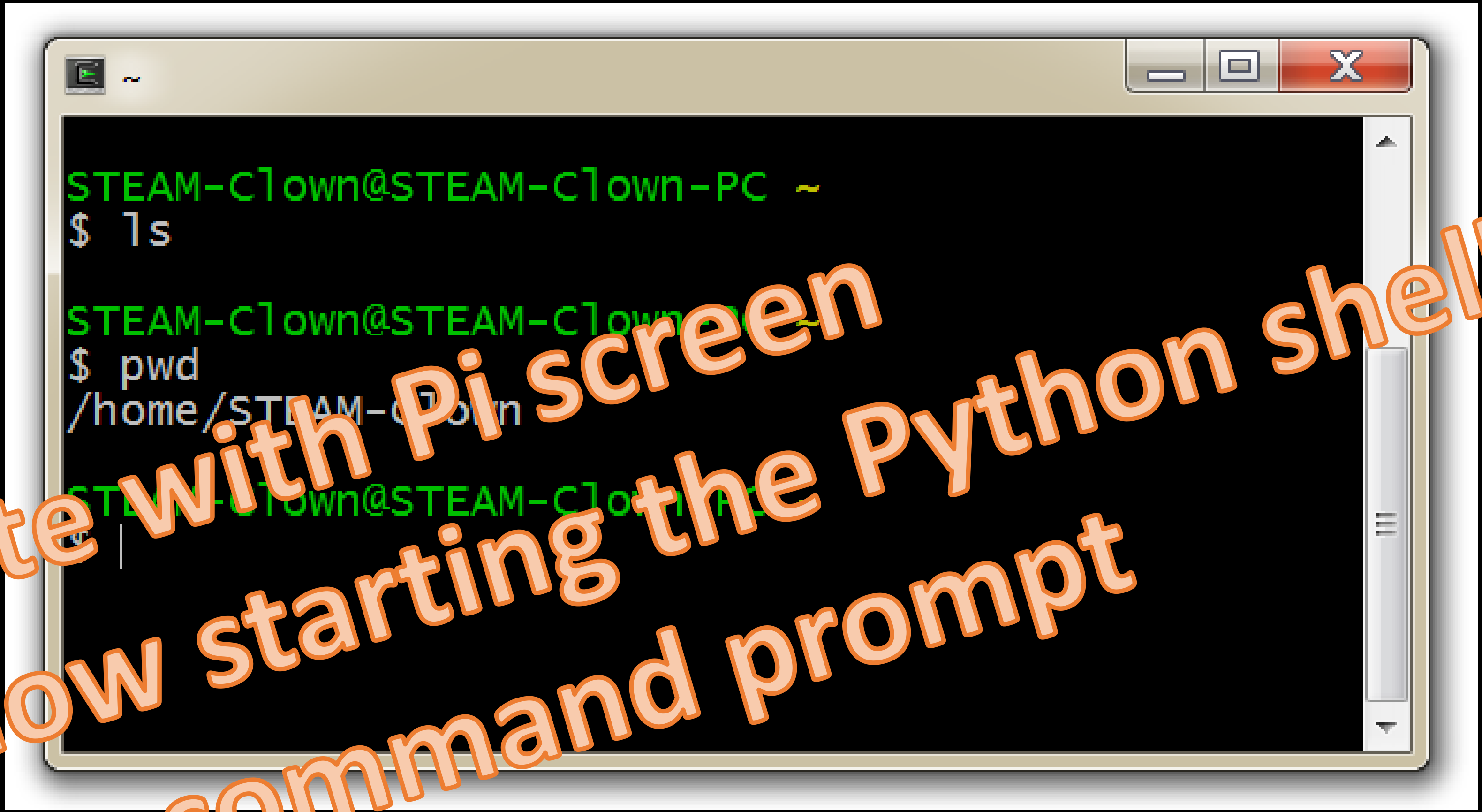
Interpreter vs Compiler

- Compiled language like C++
  - Write source code – Human readable
  - Then you compile the code to an executable (.exe)
  - Double click the .exe and it will run
- Interpreted language like Python
  - Run source code directly, because you are running with/through the Python interpreter
  - Actually running the Python program and passing it your source code
  - Not Stand alone. Requires the Python interpreter to be installed or accessed

Open a Cygwin Terminal

- Open a Cygwin terminal

```
$ ls
$ pwd
```

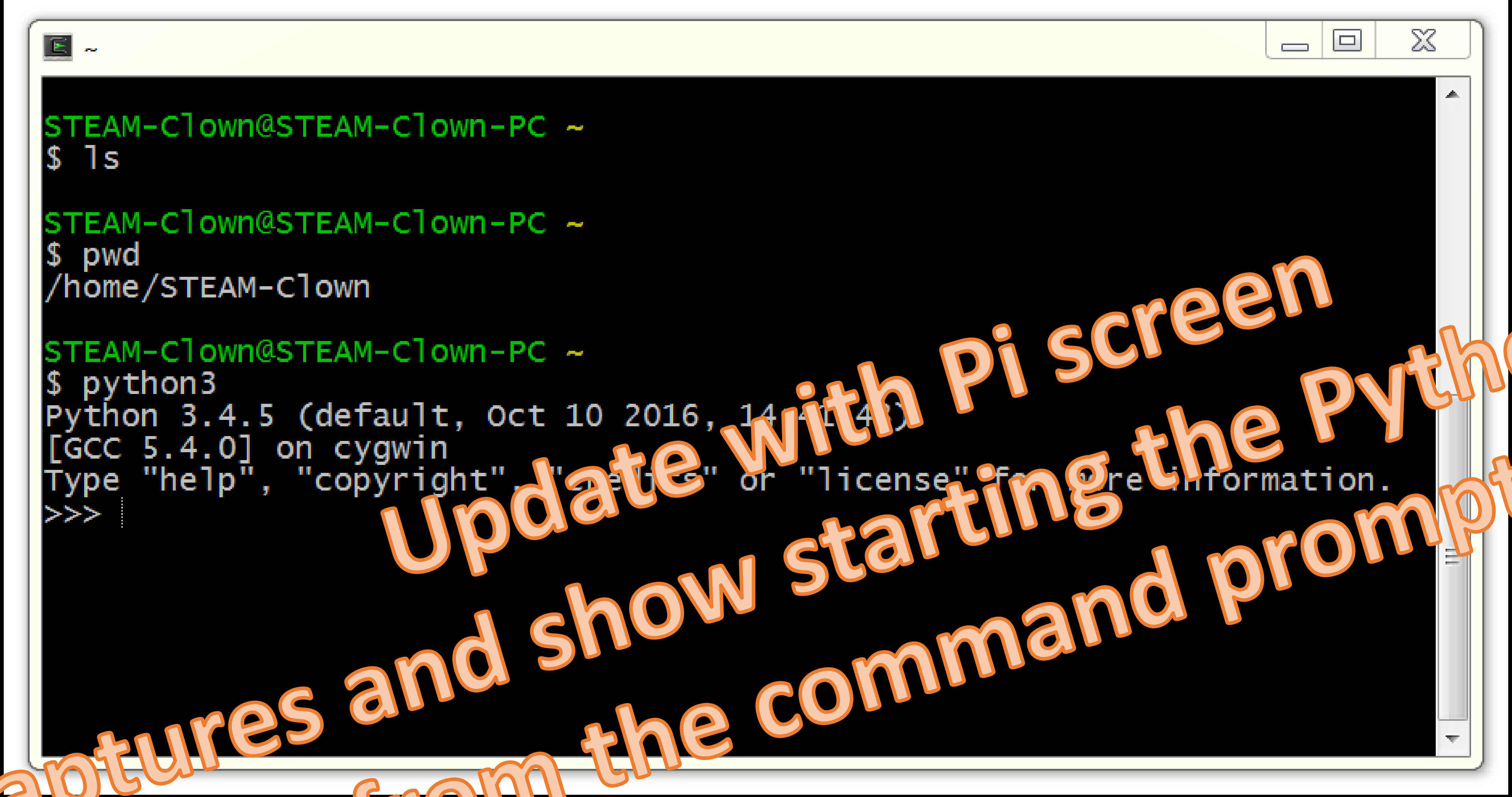


```
STEAM-Clown@STEAM-Clown-PC ~
$ ls

STEAM-Clown@STEAM-Clown-PC ~
$ pwd
/home/STEAM-Clown

STEAM-Clown@STEAM-Clown-PC ~
$ |
```

Update with Pi screen captures and show starting the Python shell from the command prompt

STEAM CLOWN™ & Squeaky Hinge PRODUCTIONS

START PYTHON3



```
STEAM-Clown@STEAM-Clown-PC ~
$ ls

STEAM-Clown@STEAM-Clown-PC ~
$ pwd
/home/STEAM-Clown

STEAM-Clown@STEAM-Clown-PC ~
$ python3
Python 3.4.5 (default, Oct 10 2016, 14:41:48)
[GCC 5.4.0] on cygwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Update with Pi screen captures and show starting the Python shell from the command prompt

STEAM CLOWN™ & Squeaky Hinge PRODUCTIONS

Remember… Hello World?

- The “>>>” is a Python *prompt* indicating that Python is ready for us to give it a command. These commands are called *statements*

```
>>> print("Hello World“)
Hello World
>>> print(2+3)
5
>>> print("2+3=", 2+3)
2+3= 5
>>>
```

STEAM CLOWN™ & Squeaky Hinge PRODUCTIONS

PYTHON DOES IT’S BEST TO INTERPRET

- At The “>>>” *prompt,* Python will make a lot of assumptions about what you are trying to do or infer
- Here it is assuming integer math

```
>>> 2+3
5
>>> 2+3*3
11
>>> (2+3)*3
15
```

STEAM CLOWN™
& Squeaky Hinge
PRODUCTIONS

Types of Results as We Do Math

- Integer
- Float
- Long

```
>>> 2+3
5
>>> 2+3.14
5.15
>>> 10**100
```

Reserved Words

- You cannot use reserved words as variable names / identifiers

```
False   class   return  is      finally
None    if      for     lambda  continue
True    def     from    while   nonlocal
and     del     global  not     with
as      elif    try     or      yield
assert  else    import  pass
break   except  in      raise
```

Parts of Speech…

`x = 2` ← Assignment statement

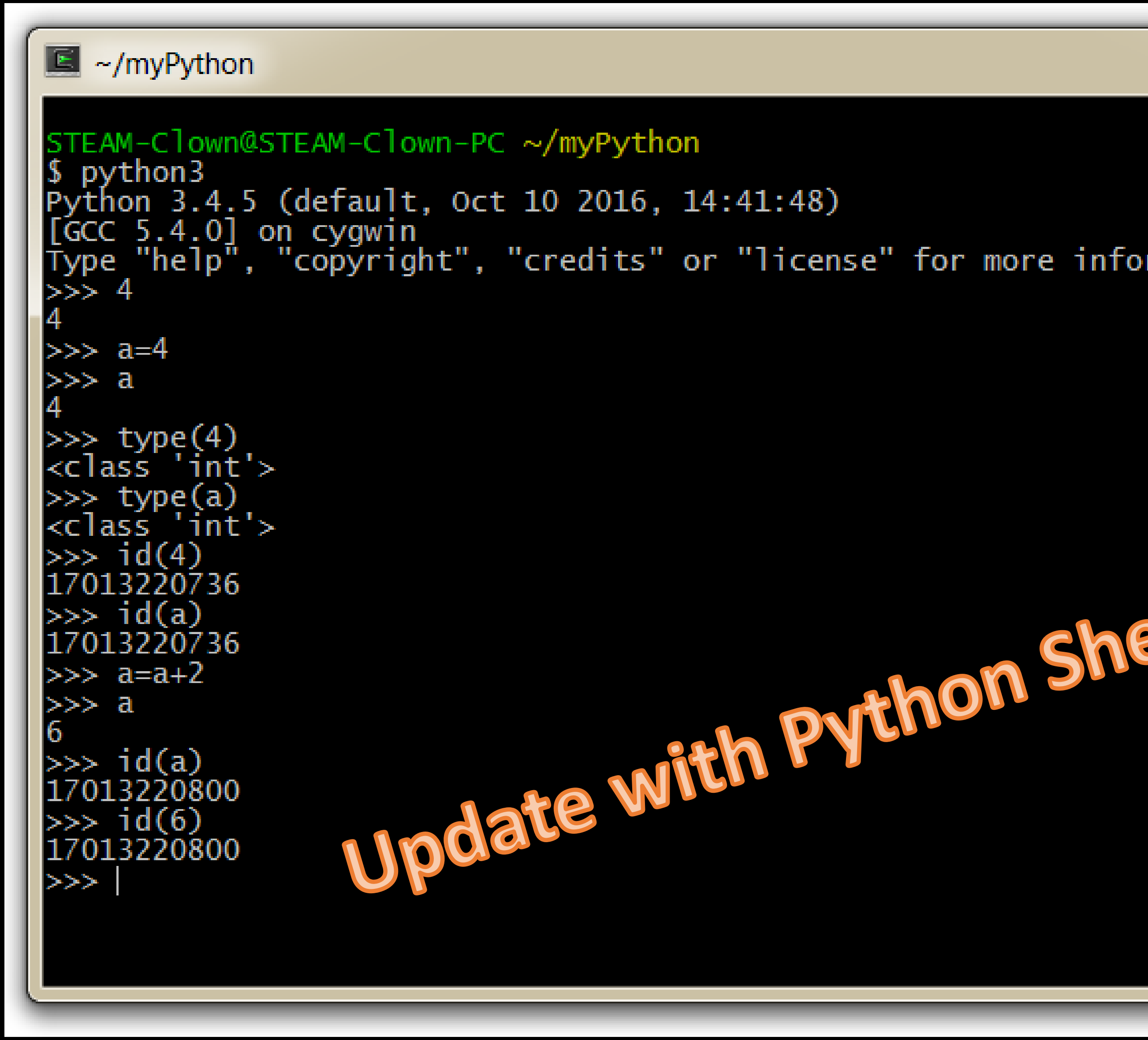
`x = x + 2` ← Assignment with expression

`print(x)` ← Print statement

Variable Operator Constant Function

Objects

- Everything in *Python is an object that has:*
  - an *identity (id)*
  - a *type*
  - a *value (mutable or immutable)*



```
STEAM-Clown@STEAM-Clown-PC ~/myPython
$ python3
Python 3.4.5 (default, Oct 10 2016, 14:41:48)
[GCC 5.4.0] on cygwin
Type "help", "copyright", "credits" or "license" for more info
>>> 4
4
>>> a=4
>>> a
4
>>> type(4)
<class 'int'>
>>> type(a)
<class 'int'>
>>> id(4)
17013220736
>>> id(a)
17013220736
>>> a=a+2
>>> a
6
>>> id(a)
17013220800
>>> id(6)
17013220800
>>> |
```

Update with Python She

STEAM CLOWN™ & Squeaky Hinge PRODUCTIONS

VALUE

- **Mutable**: When you alter the item, the id is still
- the same. Variables, Dictionary, List
  - a=a+1

- **Immutable**: String, Integer, Tuple
  - 5
  - 3.14
  - “Hello”, “X”, “C”
  - “4”

STEAM CLOWN™ & Squeaky Hinge PRODUCTIONS

LAB#1.1

- Do some math

- Make some variable assignments

```
>>>i=90
>>>print i*(i+10)
9000
```

- Can you run more than 1 line of code?

Python Function

- You can mix types in evaluations
- You can create structures like an `If` statement

```
>>> x = 34 - 23
>>> y = "Hello“
>>> z = 3.45
>>> if z == 3.45 or y == "Hello":
...       x = x + 1
...       y = y + " World“

>>>
```

```
>>> print(x)
???
>>> print(y)
???
```

Python Function

- Usually we want to execute several statements together that solve a common problem. One way to do this is to use a *function*

```
>>> def hello():
        print("Hello")
        print("Computers are
Fun")



>>>
```

STEAM CLOWN™ & Squeaky Hinge PRODUCTIONS

Parsing a Function

```
>>> def hello():
        print("Hello")
        print("Computers are Fun")

>>>
```

- The first line tells Python we are defining a new function called hello
- The following lines are indented to show that they are part of the hello function
- The blank line (hit enter twice) lets Python know the definition is finished

STEAM CLOWN™
& Squeaky Hinge
PRODUCTIONS

Running a Python Function

```
>>> def hello():
        print("Hello")
        print("Computers are Fun")

>>>
```

- Nothing has happened yet! We’ve defined the function, but we haven’t told Python to run the function!
- A function is invoked by typing its name

```
>>> hello()
Hello
Computers are Fun
>>>
```

Passing Parameters or Variables

- What’s the deal with the ()’s?
- Commands can have changeable parts called parameters that are placed between the ()’s

```
>>> def greet(name):
        print("Hello",name)
        print ("I’m Sorry",name,"I'm afraid I can't do
that")

>>>
```

https://www.youtube.com/watch?v=ARJ8cAGm6JE

STEAM CLOWN & Squeaky Hinge PRODUCTIONS

Calling a Function with a Parameter

```
>>> greet("Terry")
Hello Terry
I’m Sorry, I can’t do that Terry
>>> greet("Paula")
Hello Paula
I’m Sorry, I can’t do that Paula >>>
```

- When we use parameters, we can customize the output of our function

LAB#1.2

- Create a Function
  - That takes a int variable and a string variable
  - Do some math and string concatenation
  - Print some results

  - Take a Screen Shot of your python Shell with out puts

The Magic of Python

- When we exit the Python prompt, the functions we’ve defined cease to exist!
- Programs are usually composed of functions, modules, or scripts that are saved on disk so that they can be used again and again
- A module file is a text file created in text editing software (saved as “plain text”) that contains function definitions.
- A programming environment is designed to help programmers write programs and usually includes automatic indenting, highlighting, etc

Why use the Python Run Time Interpreter?

- Great for testing some code really quick
- If you want to debug some code
- See if you have the syntax right

But you would not want to use it for complex or lots of lines of code…

STEAM Clown™ & Squeaky Hinge Productions

Summary

- int, float, long, string
- Functions
  - Making Code Modular
  - Passing Variable

- Next Steps? Running code outside the >>> Interpreter

STEAM CLOWN™
& Squeaky Hinge
PRODUCTIONS

REFERENCE SLIDES

STEAM CLOWN™
& Squeaky Hinge
PRODUCTIONS

APPENDIX

STEAM Clown™ Productions

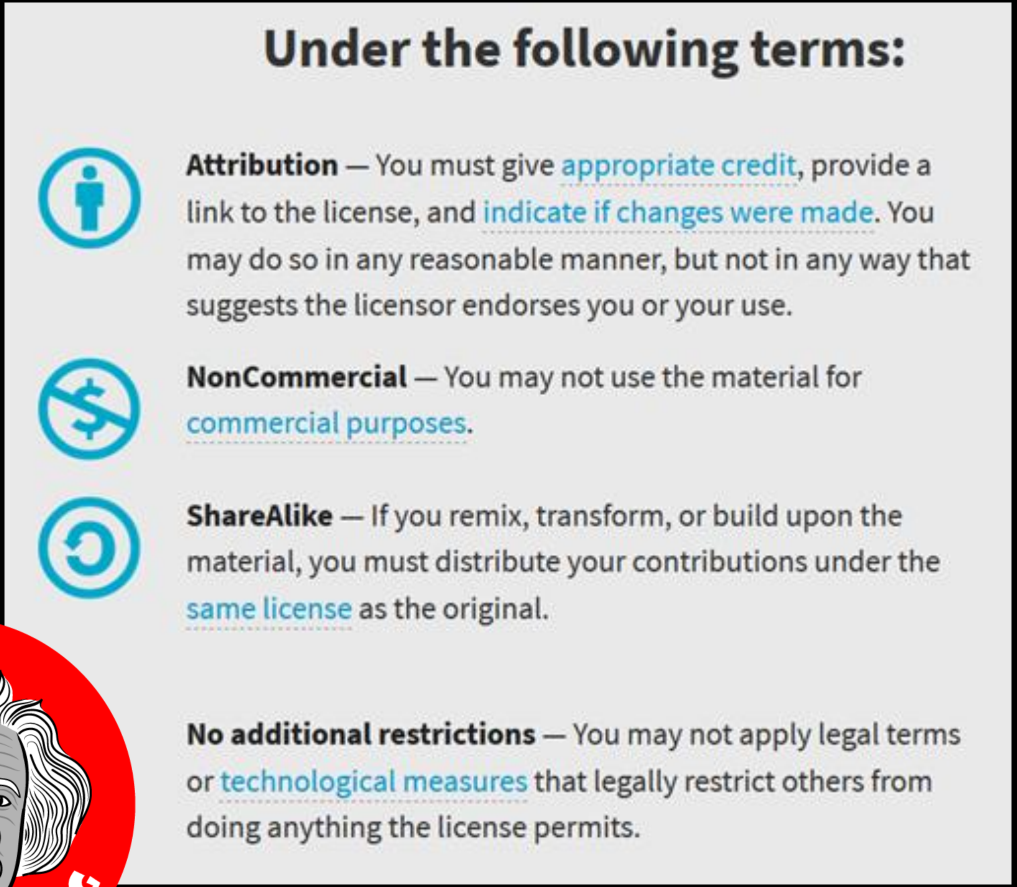
Can I get a copy of these slides? Yes, Probably…

Most presentation lecture slides can be found indexed on www.steamclown.org and maybe blogged about here on Jim The STEAM Clown's Blog, and on STEAM Clown's Mechatronics Engineering Google site, where you can search for the presentation title. While you are there, sign up for email updates

STEAM Clown™
& Squeaky Hinge
Productions

Appendix A: License & Attribution

- This interpretation is primarily the Intellectual Property of Jim Burnham, Top STEAM Clown, at STEAMClown.org
- This presentation and content is distributed under the Creative Commons License CC-BY-NC-SA 4.0
- My best attempt to properly attribute, or reference any other sources or work I have used are listed in Appendix C



Under the following terms:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

NonCommercial — You may not use the material for commercial purposes.

ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Please maintain this slide with any modifications you make

STEAM CLOWN™ & Squeaky Hinge PRODUCTIONS

Appendix B: Code License & Attribution

- This interpretation is primarily the Intellectual Property of Jim Burnham, Top STEAM Clown, at STEAMClown.org
- The programming code found in this presentation or linked to on my Github site is distributed under the:
  - GNU General Public License v3.0
  - European Union Public Licence EUPL 1.2 or later
- My best attempt to properly attribute, or reference any other sources or work I have used are listed in Appendix C

Please maintain this slide with any modifications you make

Appendix C: Primary Sources & Attribution for Material Used

- Charles R. Severance slides can be found on the https://www.py4e.com/ site are Copyright 2010 - Charles R. Severance (www.dr-chuck.com) of the University of Michigan School of Information and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.
  - Initial Development: Charles Severance, University of Michigan School of Information
  - Modifications and Adaptions by Jim Burnham, Top Clown @ www.steamclown.org
- Another great Python site is Barbara Saurette AKA mechanicalgirl and her Github site
- Additionally used some content from slide deck from Mr Ganesh Bhosale found https://github.com/gdbhosale/python-rpi/blob/master/python1.pdf

Please maintain this slide with any modifications you make

STEAM Clown™
& Squeaky Hinge
Productions