STEAM CLOWN™ PRODUCTIONS

PYTHON-FUNCTIONS

Objective, Overview & Introduction

- Functions are a way to re-use code or access code some one else has created
- Take a brief look at how Python implements the 'store and use later' programming pattern

STEAM CLOWN™ & Squeaky Hinge PRODUCTIONS

WHAT YOU WILL KNOW...

- Prior Knowledge & Certifications
  - You should have a basic understanding of Python language structures

- What You Will Know & Be Able To Do
  - You will be able to implement function code that can be use over and over again
  - You will learn how to call functions with and without parameters

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

STEAM CLOWN™ & Squeaky Hinge PRODUCTIONS

How will you be measured

- Individual Students will submit working code
- Students, as individuals or teams will present orally how they solved the coding challenge, and depth of understanding will be graded
- Success will be determined by how well your code runs as checked by the instructor after you have turned in your Lastname-Firstname-ProgramName.py text files

STEAM CLOWN™ & Squeaky Hinge PRODUCTIONS

NEW WORDS...

- Function
- Subroutine
- Parameter

WHERE CAN I RUN MY PYTHON CODE?

- The main way we will implement Python code will be by running it on a Raspberry Pi, using the Linux command terminal shell, or the Idle3 Python interpreter
- If you don't have a Raspberry Pi, or if you don't have Python installed, there are a few Python interpreters online. This lets you try code with out having to install Python on your own PC or physically have a Raspberry Pi or other hardware. Here are a few. If you find a better one, please let me know
  - Python 3 On-Line Python Interpreter - Tutorials Point
    - Python 2.7 On-Line Python Interpreter - Tutorials Point
  - Python Interpreter - Online GDB
  - Python Shell - Python.org

I GOT THIS… CAN I JUMP AHEAD?

- Jump Ahead and do the labs, save them and turn them in (show me and turn in later)
- Still need something to do? Try writing your own program or try this Extra Credit <linktolab> (show me and turn in later)

STEAM CLOWN™ & Squeaky Hinge PRODUCTIONS

RESOURCES & MATERIALS NEEDED

- PY4E Chapter 4 - Functions

STEAM CLOWN™ PRODUCTIONS

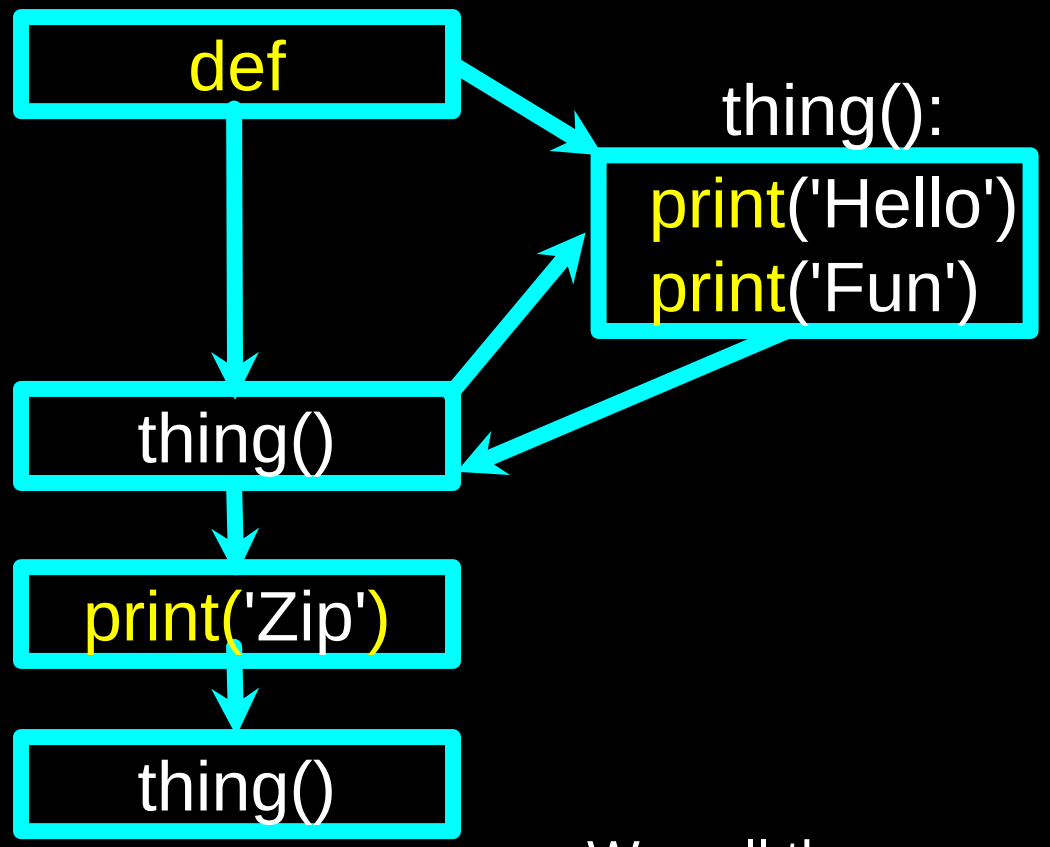
MOSTLY DR. CHARLES R. SEVERANCE'S SLIDES

Functions

Chapter 4

Python for Everybody
www.py4e.com

Stored (and reused) Steps



Program:

```
def thing():
    print('Hello')
    print('Fun')

thing()
print('Zip')
thing()
```

Output:

Hello
Fun
Zip
Hello
Fun

We call these reusable pieces of code “functions”

Python Functions

- There are two kinds of functions in Python.

  - Built-in functions that are provided as part of Python - print(), input(), type(), float(), int() ...

  - Functions that we define ourselves and then use

- We treat the built-in function names as “new” reserved words
  (i.e., we avoid them as variable names)

Function Definition

- In Python a function is some reusable code that takes arguments(s) as input, does some computation, and then returns a result or results
- We define a function using the def reserved word
- We call/invoke the function by using the function name, parentheses, and arguments in an expression

Argument

big = max('Hello world')

Assignment

'w'

Result

```
>>> big = max('Hello world')
>>> print(big)
w
>>> tiny = min('Hello world')
>>> print(tiny)

>>>
```

WHY IS W "BIGGER" THAN THAN H

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

```
>>> big = max('Hello world')
>>> print(big)
w
>>> tiny = min('Hello world')
>>> print(tiny)

>>>
```

Space

STEAM CLOWN™ & Squeaky Hinge PRODUCTIONS

ORD() &CHAR() FUNCTIONS

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

```
>>> big = max('Hello world')
>>> print(big)
w
>>> print(ord("w")
119
>>> print(chr(119)
w
>>> print(ord("H")
72
>>> print(chr(72)
H
>>> print(ord(" ")
32
>>> print(chr(32)

>>>
```

Space

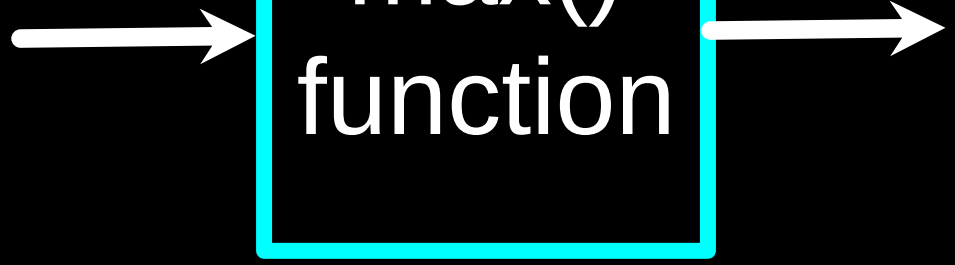
STEAM CLOWN™ & Squeaky Hinge PRODUCTIONS

Max Function

```
>>> big = max('Hello world')
>>> print(big)
w
```

A function is some stored code that we use. A function takes some input and produces an output.

'Hello world'
(a string)



'w'
(a string)

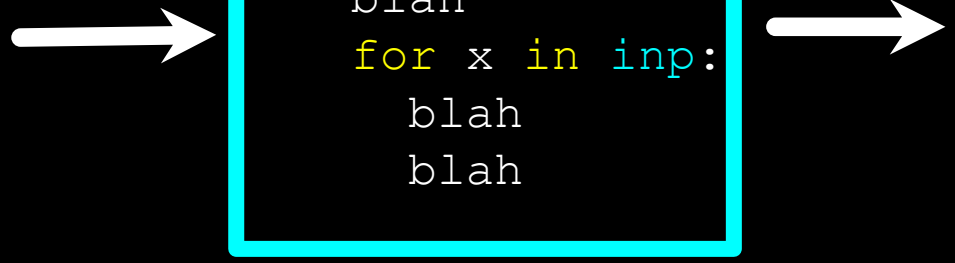
Guido wrote this code

Max Function

```
>>> big = max('Hello world')
>>> print(big)
w
```

A function is some stored code that we use. A function takes some input and produces an output.

'Hello world'
(a string)



'w'
(a string)

Guido wrote this code

Type Conversions

- When you put an integer and floating point in an expression, the integer is implicitly converted to a float
- You can control this with the built-in functions int() and float()

```
>>> print(float(99) / 100)
0.99
>>> i = 42
>>> type(i)
<class 'int'>
>>> f = float(i)
>>> print(f)
42.0
>>> type(f)
<class 'float'>
>>> print(1 + 2 * float(3) / 4 - 5)
-2.5
>>>
```

String Conversions

- You can also use int() and float() to convert between strings and integers
- You will get an error if the string does not contain numeric characters

```
>>> sval = '123'
>>> type(sval)
<class 'str'>
>>> print(sval + 1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate 'str'
and 'int'
>>> ival = int(sval)
>>> type(ival)
<class 'int'>
>>> print(ival + 1)
124
>>> nsv = 'hello bob'
>>> niv = int(nsv)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int()
```

Functions of Our Own…

Building our Own Functions

- We create a new function using the def keyword followed by optional parameters in parentheses
- We indent the body of the function
- This defines the function but does not execute the body of the function

```
def print_lyrics():
    print("I'm a lumberjack, and I'm okay.")
    print('I sleep all night and I work all day.')
```

LAB #1 - FUNCTIONS

- Enter this code into a new program

```
x = 5
print('Hello')

def print_lyrics():
    print("I'm a lumberjack, and I'm okay.")
    print('I sleep all night and I work all day.')

print('Yo’)

x = x + 2
print(x)
```

Hello
Yo
7

Definitions and Uses

- Once we have defined a function, we can call (or invoke) it as many times as we like
- This is the store and reuse pattern

LAB #2 - FUNCTIONS

- Edit to invoke the `print_lyrics` function

```
def print_lyrics():
    print("I'm a lumberjack, and I'm okay.")
    print('I sleep all night and I work all day.')

x = 5
print('Hello')
print('Yo')
print_lyrics()
x = x + 2
print(x)
```

```
Hello
Yo
I'm a lumberjack, and I'm okay.
I sleep all night and I work all day.
7
```

Arguments

- An argument is a value we pass into the function as its input when we call the function
- We use arguments so we can direct the function to do different kinds of work when we call it at different times
- We put the arguments in parentheses after the name of the function

```
big = max('Hello world')
```

Argument

Parameters

A parameter is a variable which we use in the function definition. It is a “handle” that allows the code in the function to access the arguments for a particular function invocation.

```
def greet(lang):
    if lang == 'es':
        print('Hola')
    elif lang == 'fr':
        print('Bonjour')
    else:
        print('Hello')

def main():
    greet('en')
    greet('fr')

main()
```

```
Hello
Bonjour
```

LAB #3 - FUNCTIONS

- Enter this code
- Edit the main function to call the "es" option
- Edit to the current languages, then add an new language in the greet() function
  - like Portuguese, or Quechua

```
def greet(lang):
    if lang == 'es':
        print('Hola')
    elif lang == 'fr':
        print('Bonjour')
    else:
        print('Hello')

def main():
    greet('en')
    greet('fr')

main()
```

STEAM CLOWN™ & Squeaky Hinge PRODUCTIONS

Return Values

Often a function will take its arguments, do some computation, and return a value to be used as the value of the function call in the calling expression. The return keyword is used for this.

```
def greet():
    return "Hello"

print(greet(), "Glenn")
print(greet(), "Sally")
```

```
Hello Glenn
Hello Sally
```

Return Value

- A “fruitful” function is one that produces a result (or return value)
- The return statement ends the function execution and “sends back” the result of the function

```
def greet(lang):
    if lang == 'es':
        return'Hola'
    elif lang == 'fr':
        return'Bonjour'
    else:
        return'Hello'

def main():
    print(greet('en'),'Glenn')
    print(greet('fr'),'Sabine')
    print(greet('es'),'Carlos')

main()
```

LAB #4 - FUNCTIONS

- Edit your lab #3 greet function to return a value

```
def greet(lang):
    if lang == 'es':
        return'Hola'
    elif lang == 'fr':
        return'Bonjour'
    else:
        return'Hello'

def main():
    print(greet('en'),'Glenn')
    print(greet('fr'),'Sabine')
    print(greet('es'),'Carlos')

main()
```

STEAM CLOWN™ & Squeaky Hinge PRODUCTIONS

Arguments, Parameters, and Results

```
>>> big = max('Hello world')
>>> print(big)
w
```

Parameter

```
def max(inp):
    blah
    blah
    for x in inp:
      blah
      blah
    return 'w'
```

'Hello world'

Argument

'w'

Result

Multiple Parameters / Arguments

- We can define more than one parameter in the function definition
- We simply add more arguments when we call the function
- We match the number and order of arguments and parameters

```
def addtwo(a, b):
    added = a + b
    return added

x = addtwo(3, 5)
print(x)
```

```
8
```

Void (non-fruitful) Functions

- When a function does not return a value, we call it a “void” function
- Functions that return values are “fruitful” functions
- Void functions are “not fruitful”

To function or not to function...

- Organize your code into “paragraphs” - capture a complete thought and “name it”
- Don’t repeat yourself - make it work once and then reuse it
- If something gets too long or complex, break it up into logical chunks and put those chunks in functions
- Make a library of common stuff that you do over and over - perhaps share this with your friends...

Summary

- Functions
- Built-In Functions
- Type conversion (int, float)
- String conversions
- Parameters
- Arguments
- Results (fruitful functions)
- Void (non-fruitful) functions
- Why use functions?

LAB #5

- Rewrite your pay computation with time-and-a-half for overtime and create a function called computepay which takes two parameters (hours and rate).

- Enter Hours: 45
- Enter Rate: 10

- Pay: 475.0

STEAM CLOWN™ & Squeaky Hinge PRODUCTIONS

ACKNOWLEDGEMENTS / CONTRIBUTIONS

Initial Development: Charles Severance, University of Michigan School of Information

… Insert new Contributors and Translators here

...

ASSESSMENT

- Assessment Type(s):
  - ✓Demonstrations
  - ✓Interviews
  - ✓Journals
  - ✓Observations
  - ✓Labs
  - ✓Projects
  - ✓Portfolios
  - ✓Rubrics
  - ✓Surveys
  - ✓Teacher-Made Test
  - ✓Writing Samples

STEAM CLOWN™ & Squeaky Hinge PRODUCTIONS

REFERENCE SLIDES

Learning Domain, CTE Standards and Stuff Like That…

- Learning Domain

  [ ] cognitive [ ] affective[ ] psychomotor
  - What are some cognitive skills required for success in your pathway?
  - What are some affective skills required for success in your pathway?
  - What are some psychomotor skills required for success in your pathway?
- Time:
  - Lecture
  - Lab
- Standards
  - CTE
  - CCSS
  - NCSS

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

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.

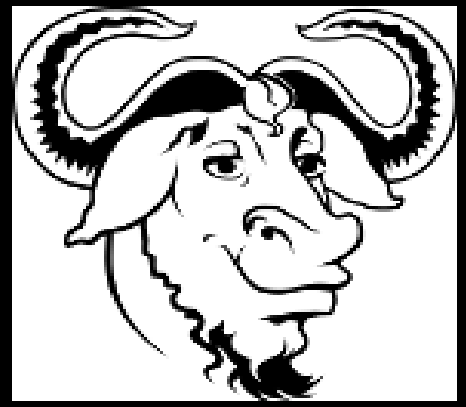
No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

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

STEAM CLOWN™ & Squeaky Hinge PRODUCTIONS

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

Please maintain this slide with any modifications you make

STEAM CLOWN™
& Squeaky Hinge
PRODUCTIONS