CMPT 165
INTRODUCTION TO THE INTERNET AND THE WORLD WIDE WEB

Unit 7
Introduction to Programming

SFU

Learning Objectives

In this unit you will learn the following.

- **Explain** what programming is and what Python is.
- **Learn** Python basics.
- **Create** simple Python programs.

Topics

1. Computer Programming
2. Python for Programming

Lectures 1 & 2

3. Python Basics…
4. …more Python Basics
5. …and even more Python Basics

Lectures 2, 3, & 4

6. Demos and Examples
7. Code Tracing

Lectures 5 & 6

Computer Science

def. is the scientific and practical approach to computation and its applications (a.k.a *algorithms*).

- Theory - *is it possible to compute…*
- Analysis - *how well did that compute…*
- Design - *how can we compute…*
- Optimization - *how fast can we compute…*
- Implementation - *let's create a program to compute…*
- Application - *let's run program to compute this data…*

Computer *def.* a general purpose programmable information processor with input and output.

von Neumann Machine (1943)

Computer Program

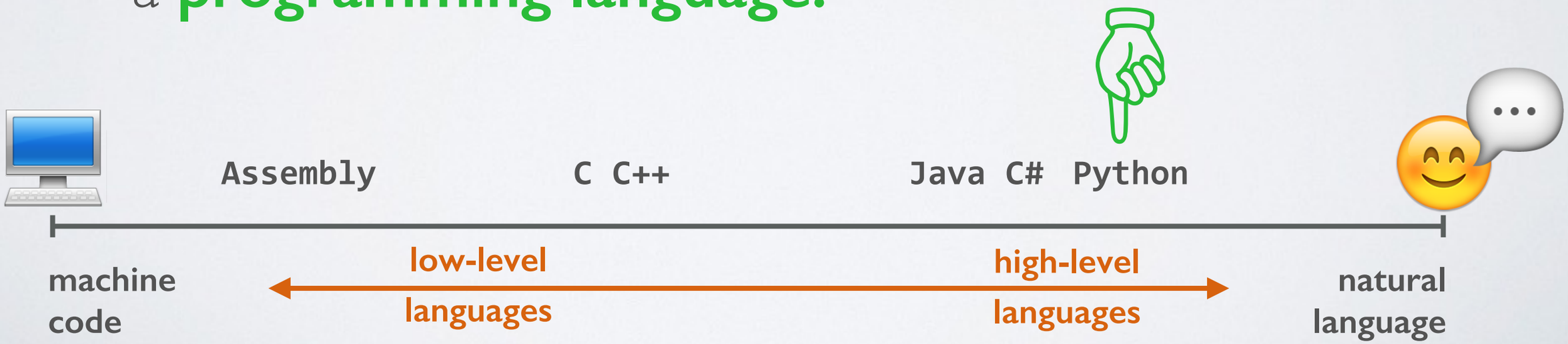
- *def.* a list of instructions a computer follows to perform a given task.
- **Programming** is the act of creating that list of instructions.
  - You need to tell the computer what you want it to do.
  - Sometimes this requires specifying a lot of details.
  - Instructions have to be specified in a way the computer can understand (a.k.a. machine code)

```
A4 00 F5 34 22 BA BB 75
00 9F 67 6E EE 80 EF 21
```

Programming Languages

```
A4 00 F5 34 22 BA BB 75
00 9F 67 6E EE 80 EF 21
```

- We **do not** understand machine code.
- Computers **do not** understand natural language.
- **Compromise:** so we create something in between
  - a **programming language:**



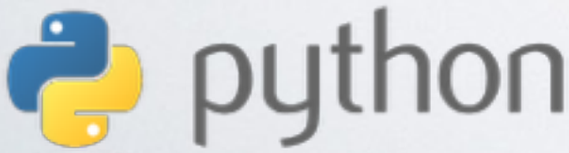
Programming Approaches

Structured Programming

- Came about the 1960s
- Programming broken down into functions and modules
- Clear structure, easy to test, debug, and modify
- Design more complex program

Object Oriented Programming

- Object *def.* entity characterized by state and behaviours.
- State is the data and behaviours are methods/functions.



can do both, we will only do structured.

Markup vs. Programming

- We have used HTML to markup content.
- Use **markup** to create structure for content.
- Use **programming** to perform tasks.
  - Take and input and produce an output.
  - Process and manipulate input data.
  - Create a list of instructions to process data.
- Programs use markup to understand content for processing.
- HTML markup is not programming.
- Rendering HTML in its visual form is programming.

Why learn to program?

- So you probably never make a program before…
- And you might ask yourself *why start now?* 😏
- Some might reply *because it is fun!* 😀
- Your instructor would reply *because you have to!!!* 😡
- A real reason: *because is helps solve really hard problems, fast. Problems that would take us a long time to solve.*
- An example: what are the first 10,000 prime numbers?
  - A computer can do this easily in under 2 seconds.
  - A human? …days? …weeks?

Why Python?

…why not C++, C#, or Java?

- Python is simple to learn and use.
  - You can solve problems quickly with little overhead.
    - No need to worry about: memory allocation, numerical limits, the structure of data types, etc.
- Python is a good first programming language to learn.
- Python is well suited to web programming.

- The focus is not to making you into a programmer, rather for you to understand the general concept of programming.

Python 2.7.x

Now would be a good time to install Python!

Computers ❤ Binary

- **Decimal** is base-10, symbols `0, 1, 2, 3, 4, 5, 6, 7, 8, 9`
- **Binary** is base-2, symbols `0, 1` (or OFF, ON)
  - e.g. 10100010101010111111000 (can be very long)
- **Hex** is base-16 (2^4) (shorter then binary!),
  - symbols `0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F`
- Computers love whole numbers.
- Rational numbers need special computation
  - e.g. **`4.5`** or **½**
- What about characters — our alphabet?

Characters are Numbers

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	Y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Characters are Numbers

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b

The letter **A** (uppercase) has the

integer value **65** (hex **0x41**)

where as lowercase **a** has the

integer value **97** (hex **0x61**).

Source: www.LookupTables.com

Characters are Numbers

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b

The letter **A** (uppercase) has the

integer value **65** (hex **0x41**)

where as lowercase **a** has the

integer value **97** (hex **0x61**).

* Is this correct? `'A' + 0x20 = 'a'`

Source: www.LookupTables.com

Characters are Numbers

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b

The letter **A** (uppercase) has the

integer value **65** (hex **0x41**)

where as lowercase **a** has the

integer value **97** (hex **0x61**).

* Is this correct? **'A' + 32 = 'a'**

We need 2 Python functions: **ord()** and **chr()**

ord() turns a character into an integer.

chr() turns an integer into a character.

Source: www.LookupTables.com

Characters are Numbers

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b

* Is this correct? `'A' + 32 = 'a'`

We need 2 Python functions: **`ord()`** and **`chr()`**

`ord()` turns a character (chr) into an integer/ordinal.

`chr()` turns an integer/ordinal into a character (chr).

In Python 2.7.x IDLE execute:

```
chr(ord('A') + 32)
```

Is the question on the first line correct?

Source: www.LookupTables.com

QUESTIONS?

Program Execution

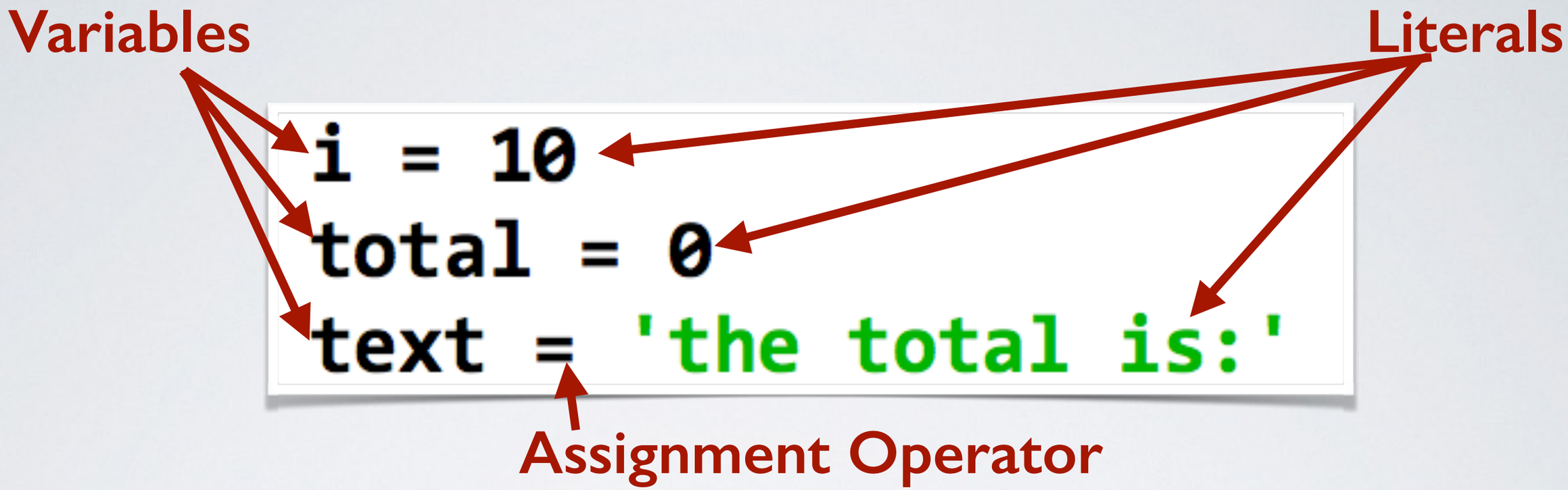
def. performing the instructions of a computer program.

For simplicity, we can assume that the computer execute each line of code in a program from top to bottom, from outwards to inwards to outwards.

```
i = 10
total = 0
text = 'the total is:'

for j in range(i):
    total = total + j
    print text, j

print
print 'the end'
```

```
the total is: 0
the total is: 1
the total is: 2
the total is: 3
the total is: 4
the total is: 5
the total is: 6
the total is: 7
the total is: 8
the total is: 9

the end
>>>
```

Literals & Variables



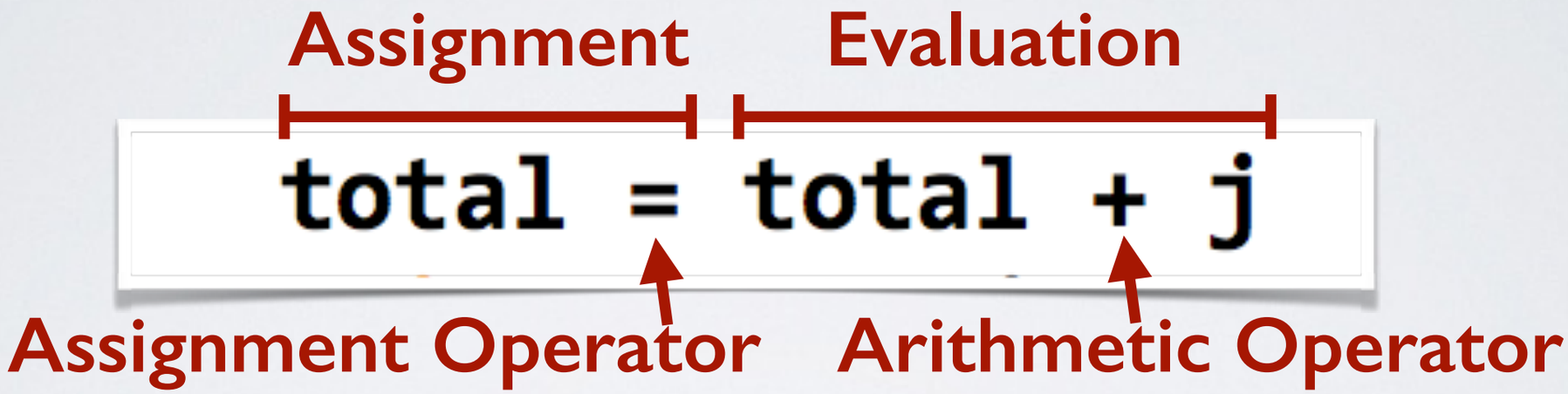
Literals are fixed values.

For example: boolean, integer, float, character.

Variables (for simplicity) are a labelled place in computer memory that store literals.

Assignment & Evaluation

For simplicity, each line of the program is executed sequentially.



Evaluation performs the instructions on the right side of the Assignment Operator which results in a literal.

Assignment takes the results from evaluation and stores them in a variable on the left side of the Assignment Operator.

GO TO THE

PYTHON BASICS SLIDES

Python Dictionary

- data that is organized in key-value pairs:
  `dict = {key: value, …}`

```
>>> person = {'name': 'John Doe', 'age': 32}
>>> person['name']
'John Doe'
>>> person['age']
32
>>> 'age' in person
True
>>> 'gender' in person
False
>>>
```

Dictionary Lists

- And we can have lists of dictionaries:

```
>>> person = [{'name': 'John Doe', 'age': 32},
              {'name': 'Jane Doe', 'age': 76}]
>>> person[0]['name']
'John Doe'
>>> person[1]['name']
'Jane Doe'
>>>
```

```
>>> for p in person:
        print "%s is %d years old!" % (p['name'], p['age'])


John Doe is 32 years old!
Jane Doe is 76 years old!
>>>
```

QUESTIONS?

DEMOS & EXAMPLES

Coding: Max

Define a function `max()` that takes two numbers as arguments and returns the largest of them. Use the if-then-else construct available in Python.

It is true that Python has the `max()` function built in, but writing it yourself is nevertheless a good exercise.

Source: http://www.ling.gu.se/~lager/python_exercises.html

Coding: Max of 3

Define a function `max_of_three()` that takes three numbers as arguments and returns the largest of them.

Source: http://www.ling.gu.se/~lager/python_exercises.html

Coding: Max of *n*

Define a function `max_of_n()` that takes a list of numbers as an argument and returns the largest of them.

Source: http://www.ling.gu.se/~lager/python_exercises.html

Coding: Reverse

Define a function `reverse()` that computes the reversal of a string.

For example, `reverse("I am testing")` should return the string `"gnitset ma I"`.

Source: http://www.ling.gu.se/~lager/python_exercises.html

Coding: Repeat

Define a function `repeat_chars()` that takes an integer `n` and a character `c` and returns a string, `n` characters long.

For example, `repeat_chars(5,"x")` should return the string `"xxxxx"`.

Python is unusual in that you can actually write an expression `5 * "x"` that will evaluate to `"xxxxx"`. For the sake of the exercise you should ignore that the problem can be solved in this manner.

Source: http://www.ling.gu.se/~lager/python_exercises.html

Coding: Histogram

Define a procedure `histogram()` that takes a list of integers and prints a histogram to the screen. For example, `histogram([4, 9, 7])` should print the following:

```
****

*********

*******
```

Source: http://www.ling.gu.se/~lager/python_exercises.html

Coding: 99 Bottles Song

"99 Bottles of Beer" is a traditional song in the US and Canada. It is popular to sing on long trips, as it has a very repetitive format which is easy to memorize, and can take a long time to sing. The song's simple lyrics are as follows:

```
99 bottles of beer on the wall, 99 bottles of beer.
Take one down, pass it around, 98 bottles of beer on the wall.
```

The same verse is repeated, each time with one fewer bottle. The song is completed when the singer or singers reach zero.

Your task here is write a Python program capable of generating all the verses of the song.

Source: http://www.ling.gu.se/~lager/python_exercises.html

Code Tracing

How can determine how a program can run using pencil and paper, no computer?

```
1
2 #EXAMPLE 0 - Trace the Exection.
3 for i in range(5):
4     print i
5
```

View tracing.py

Code Trace

Line Number	range(5)	i	Output
2			
3-1	[0, 1, 2, 3, 4]		
3-1	[~~0~~, 1, 2, 3, 4]	0	
4-1	[~~0~~, 1, 2, 3, 4]	0	0
3-2	[~~0,1~~, 2, 3, 4]	1	
4-2	[~~0,1~~, 2, 3, 4]	1	1
3-3	[~~0,1,2~~, 3, 4]	2	
4-3	[~~0,1,2~~, 3, 4]	2	2
3-4	[~~0,1,2,3~~, 4]	3	
4-4	[~~0,1,2,3~~, 4]	3	3
3-5	[~~0,1,2,3,4~~]	4	
4-5	[~~0,1,2,3,4~~]	4	4
3-6	[]		
5			

Summary

- Learnt about computer programming.
- Learnt about the Python programming language.
- Performed a coding exercise.
- Experienced what it is like to program a specific project

Next Unit: look at HTML forms and CGI.

QUESTIONS?