APS106

UNIVERSITY OF TORONTO

Cryptography and Cryptanalysis.

Week 4 | Lecture 1 (4.1.1)

While waiting for class to start:

Download and open the Jupyter Notebook (.ipynb) for Lecture 4.1.1

You may also use this lecture's JupyterHub link instead (although opening it locally is encouraged).

Upcoming:

- Reflection 4 released Friday @ 11 AM
- Lab 4 released this Friday @ 12 PM
- PRA (Lab) on Friday @ 2PM this week
- **Midterm** - May 31 in lecture @ 9:10 AM

if nothing else, write #cleancode

Cryptanalysis was a defining factor in WWII

Intersection of history and technology

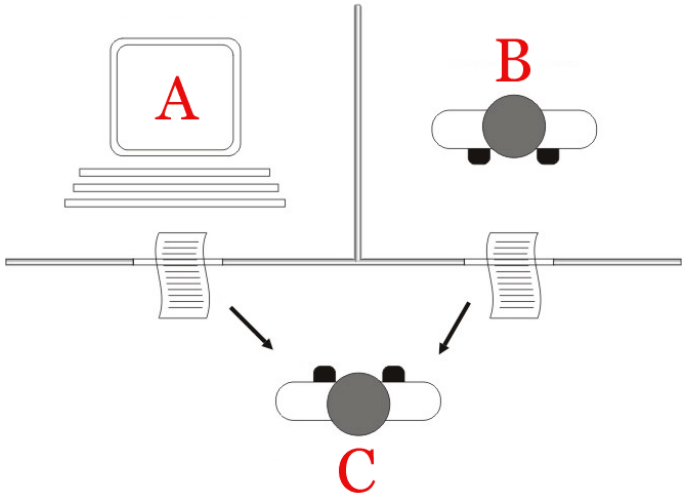
- Alan Turing - The “Father” of Computer Science and Artificial Intelligence
- Pioneered the technology to decrypt Nazi Germany's secret communications during World War II
- Decryption machines were called “Turing machines”
- Prosecuted for ‘gross indecency’ for being homosexual in 1952, died in 1954

Alan Turing circa 1951

Workers of Bletchley Park, centre of Allied Code-Breaking circa 1938
8,000 women, 75% of the workforce

Intersection of history and technology

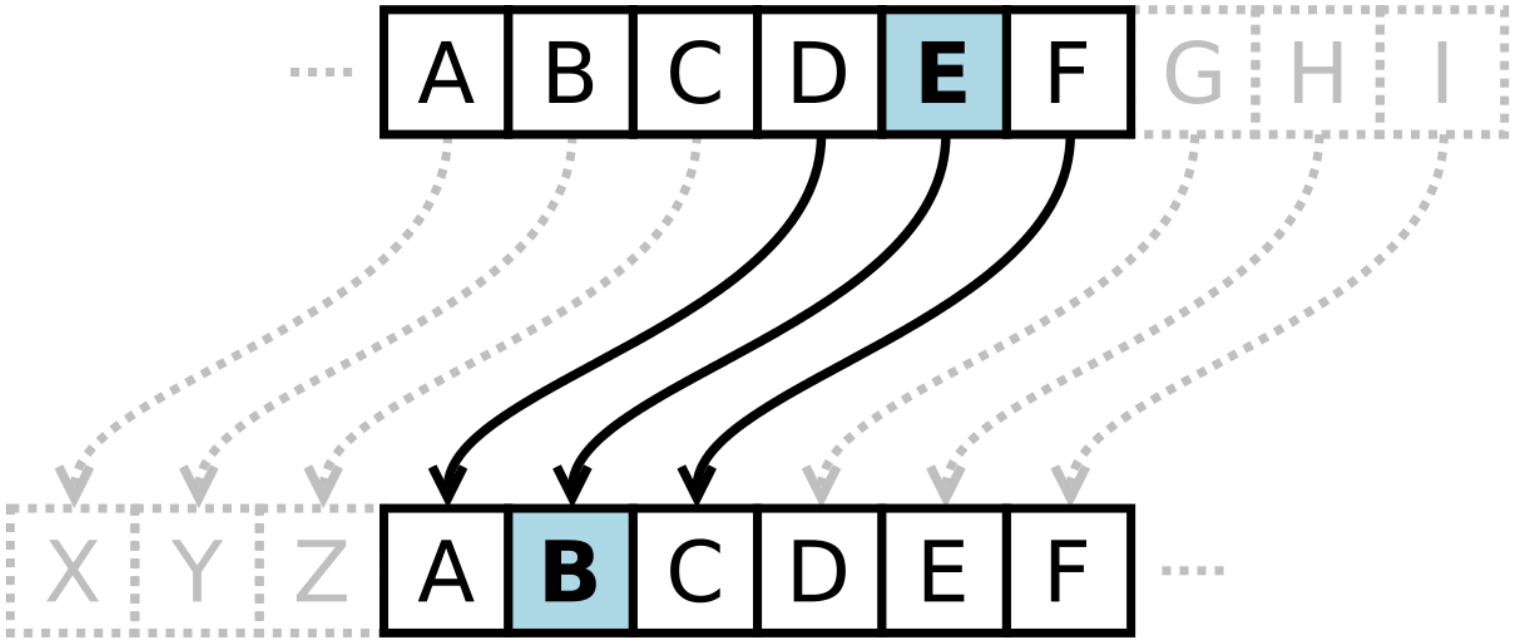
- Turing Award
  - "Nobel Prize of Computing" recipients
  - 2016 - Tim Berners-Lee, British computer scientist and inventor of the World Wide Web
  - 2018 - Geoffrey Hinton, British-Canadian computer scientist and inventor of deep neural networks (at the University of Toronto!)
  - 2022 - Robert Metcalfe, American Engineer and inventor of Ethernet
  - 2023 - Avi Wigderson, Israeli computer scientist, contribute to research on randomness
- Turing Test
  - The test of a machine's ability to be indistinguishable from a human
  - 5 supercomputers had passed as of 2023
  - Studies coming out now showing LLMs could be reaching 50% pass rate!
- Modern uses of cryptography and ciphers
  - VPN (Virtual Private Networks) – encrypts your IP address and who you are!
  - https (the s stands for secure!) – encrypts your data being sent to other websites!
  - Cryptocurrency (you control your private cipher key!)
  - Email and messaging service encryption (iMessage, Outlook, etc.)



The Caesar Cipher was used in ancient Rome

- Invented by Julius Caesar ~100 BCE
- Also known as the Caesar Shift
- Used in ancient Rome for secure military communications
- Served as the foundation for the development of more sophisticated encryption methods

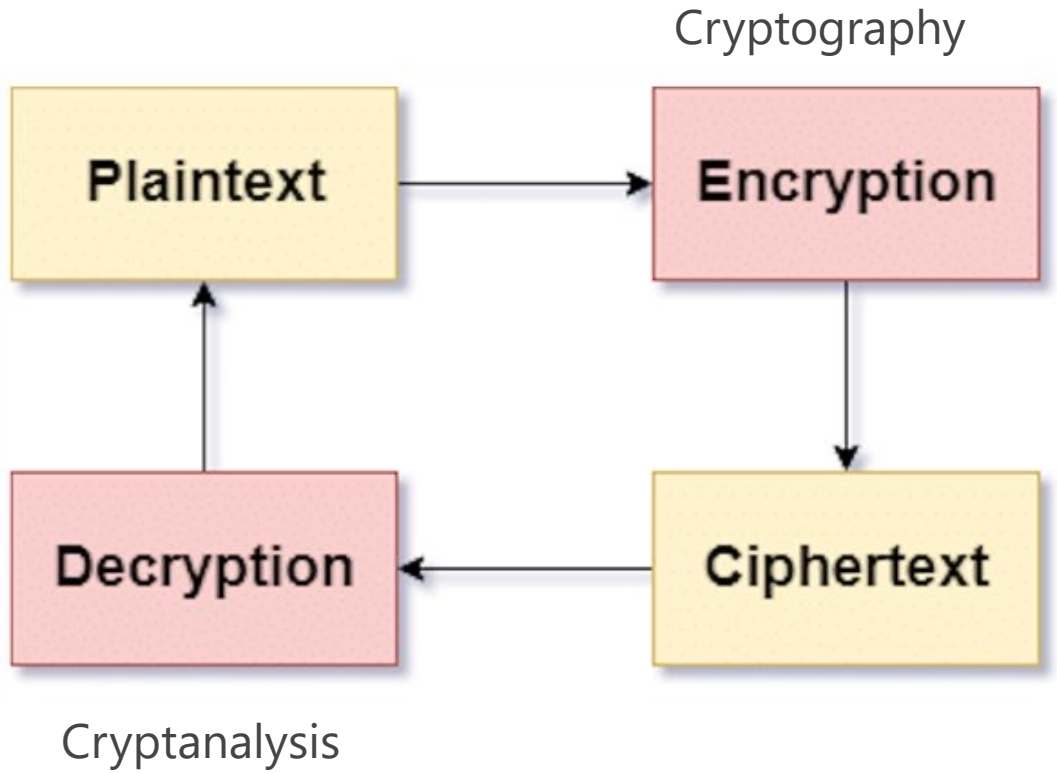
Also known as the Caesar Shift



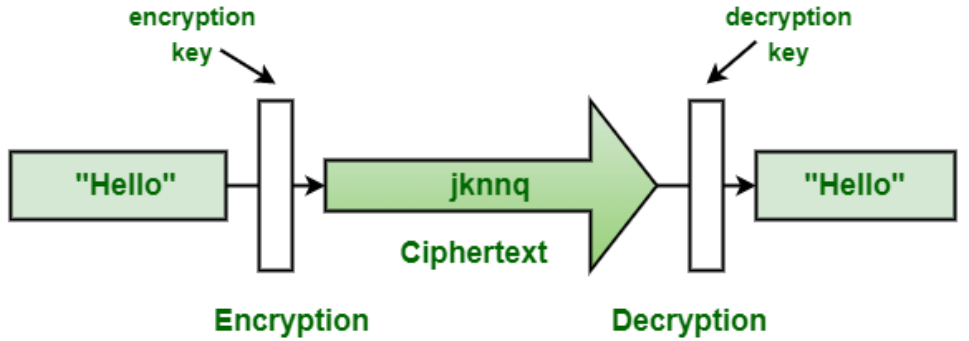
- **Goal:** Write a Python program that enables us to encrypt a message that can only be read by someone with the secret key.
- The key will represent how much we shift each letter.
- Also known as a Caesar Shift

The cipher key matters!

Cryptography and Cryptanalysis work against one another

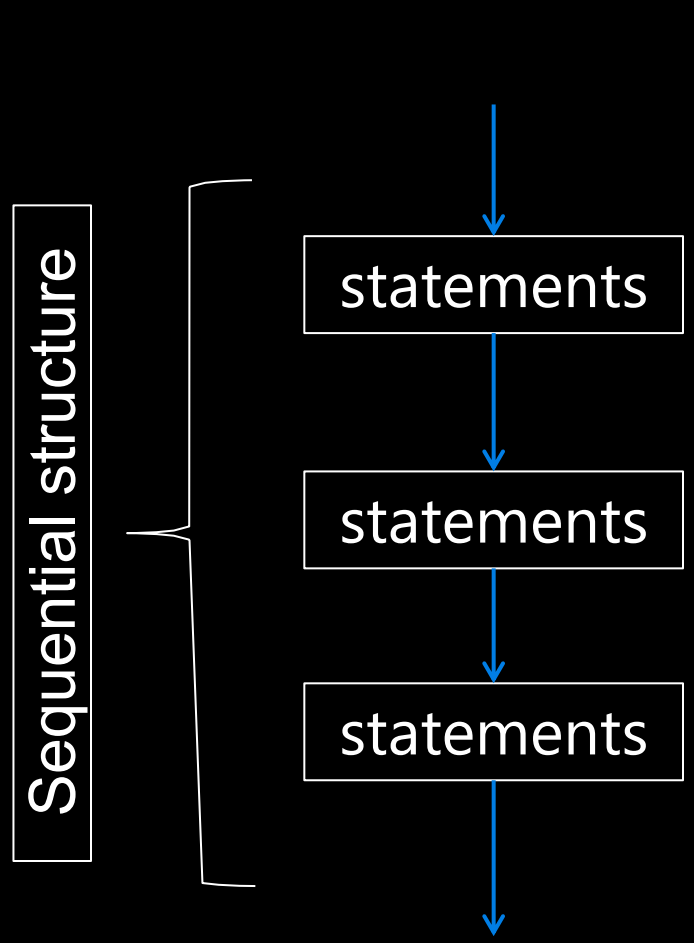


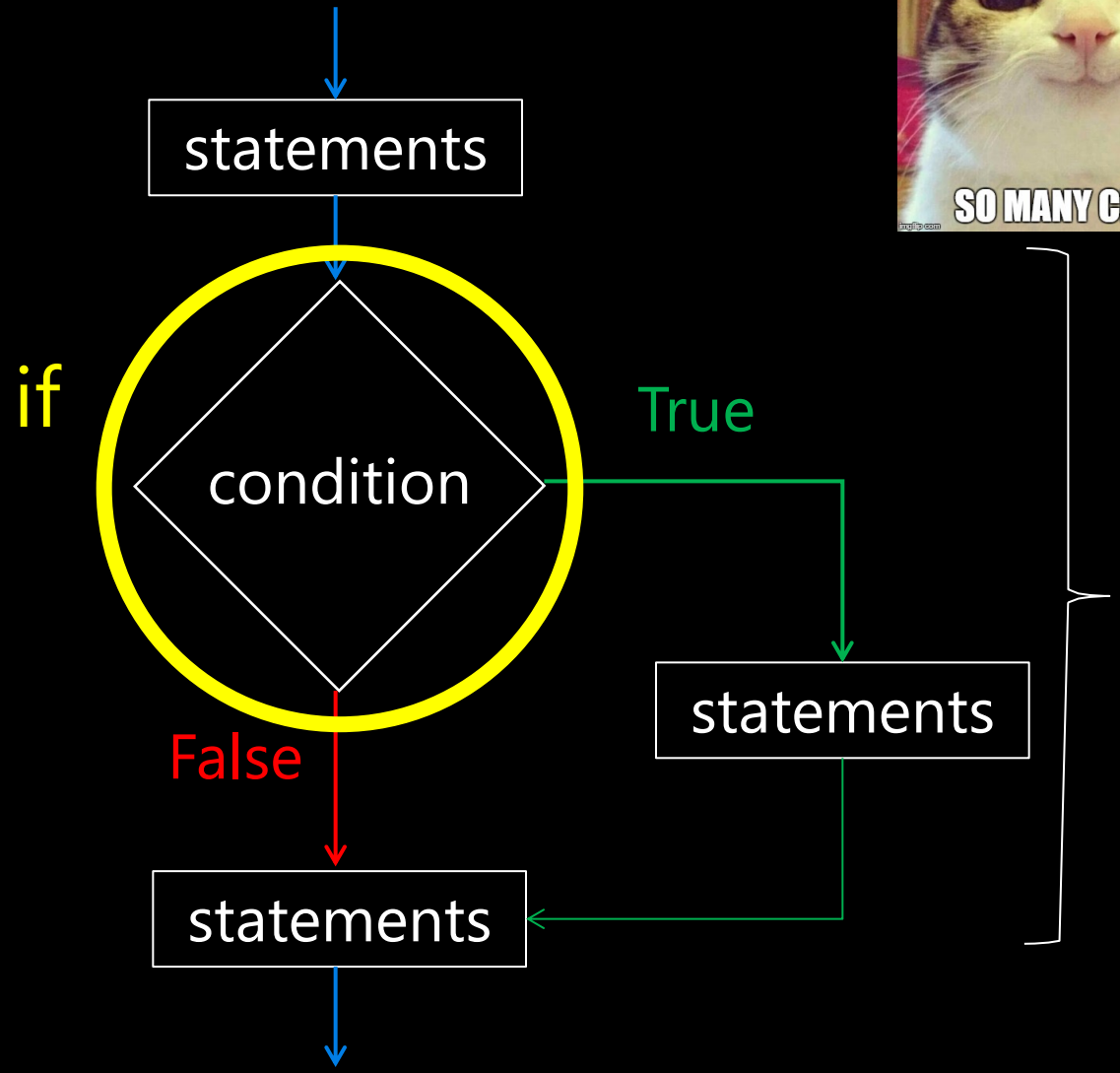
- To create our own cryptography code, we can take advantage of:
  - user input,
  - loops,
  - conditional statements,
  - ASCII codes,
  - string methods,
  - comparison operators



Some friendly reminders…

RECAP: Making Choices





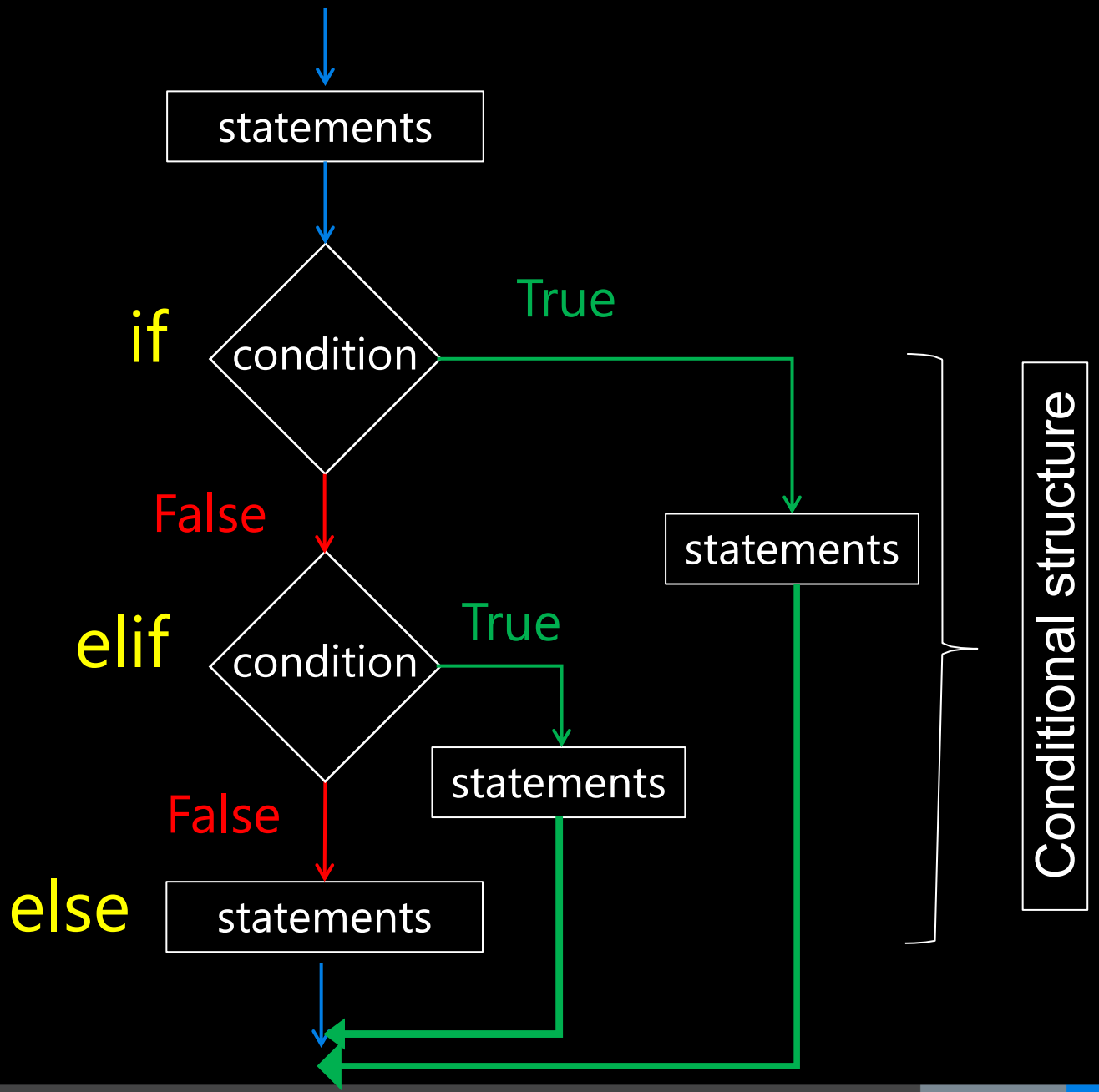
Adding the elif (else if) statement

- The most general form of the if conditional statement is:

```
if condition1:
    body1
elif condition2:
    body2
elif conditionN:
    bodyN
else:
    other_body
```

- Note the colons (:) and the indents!
- ONLY 1 body will be executed.
  - if statement is True, execute body1, exits if structure
  - if statement is False, continue to elif statement
  - elif statement is True, execute elif body, exits if structure
  - elif statement is False, continue to next elif statement
  - All if's and elif's are False, execute else statement

Making Choices



Function Definitions

```
def function_body(parameters):
```

1. """DOCSTRING""" (optional)

2. Code the does the thing

3. return [expression]
   The return statement is options and if it is not included, it's the same as writing return None

Input

- Python has a built-in function named **input** for reading text from the user.
- The general form of a **input** function call:

```
input(argument)
```

- The argument is the text you want displayed to the user.
  - *"What is your name?"*
- The value returned by the **input**function is always a string.

String Comparisons

- Boolean comparisons can also be applied to strings, whether single characters or sets of characters
- Compare two strings by their **dictionary order**, comparing letter by letter

Description	Operator	Example	Result of example
equality	==	'cat' == 'cat'	True
inequality	!=	'cat' != 'Cat'	True
less than	<	'A' < 'a'	True
greater than	>	'a' > 'A'	True
less than or equal	<=	'a' <= 'a'	True
greater than or equal	>=	'a' >= 'A'	True

Strings as Integers (ASCII Encoding)

- Each character in a string is actually represented by integers following the ASCII encoding
  - American Standard Code for Information Interchange
- All uppercase letters come before all lowercase letters
  - Uppercase "Z" is less than lowercase "a"

Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char
32	[space]	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	[backspace]

Strings as Integers (ASCII Encoding)

- When you compare two strings, what you are really doing is comparing their numerical representations
- For example in ASCII the characters ‘a’ and ‘w’ are encoded as 97 and 119, respectively
  - The comparison ‘a’ > ‘w’ would translate to 97 > 119, evaluating to False

0-9	A-Z	a-z
48-57	65-90	97-122

ASCII Encoding

- To obtain the ASCII (integer) representation, we use the built-in function ord

```
>>> ord('a')
97
```

- To convert from ASCII integer representation back to a string, we use the built-in function chr
  - Think "character" (i.e. what is this integer's character)

```
>>> chr(97)
'a'
```

APS106

UNIVERSITY OF TORONTO

Cryptography and Cryptanalysis.

Week 4 | Lecture 1 (4.1.1)

While waiting for class to start:

Download and open the Jupyter Notebook (.ipynb) for Lecture 4.1.1

You may also use this lecture's JupyterHub link instead (although opening it locally is encouraged).

Upcoming:

- Reflection 4 released Friday @ 11 AM
- Lab 4 released this Friday @ 12 PM
- PRA (Lab) on Friday @ 2PM this week
- **Midterm** - May 31 in lecture @ 9:10 AM

if nothing else, write #cleancode