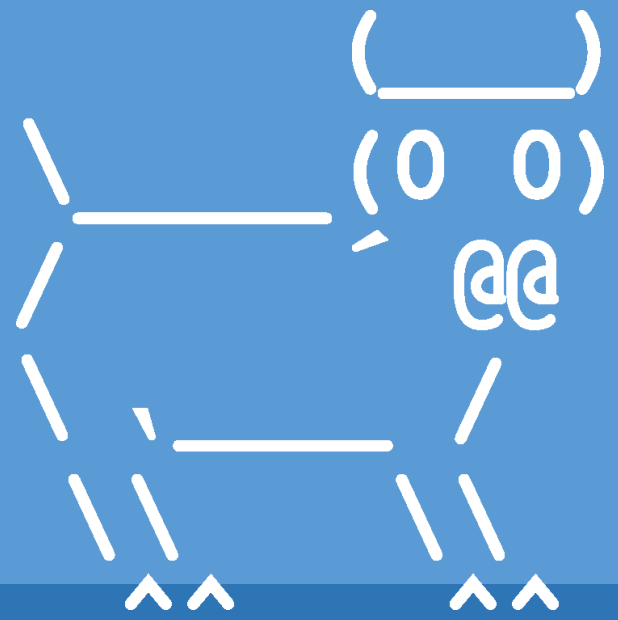
Characters



Introduction to Computer Science

Iris Howley

TODAY'S LESSON

Unicode

(Characters have numerical representations)

Sorting Characters

```
>>> word = 'Hello, hi!'

>>> sorted(word)
[' ', '!', ',', 'H', 'e', 'h', 'i', 'l', 'l', 'o']
```

Why does 'H' come before 'e'?

Why does ' ' come before any letters?

Sorting Characters

- Python uses Unicode code points to sort characters
- Each character is represented by a number
- Python uses this number to sort

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

www.LookupTables.com

Unicode Values

```
>>> ord(' ')
32
>>> ord('H')
72
>>> ord('d')
100
>>> ord('e')
101
>>> ord('l')
108
```

```
>>> chr(32)
' '
>>> chr(72)
'H'
>>> chr(100)
'd'
>>> chr(101)
'e'
>>> chr(108)
'l'
```

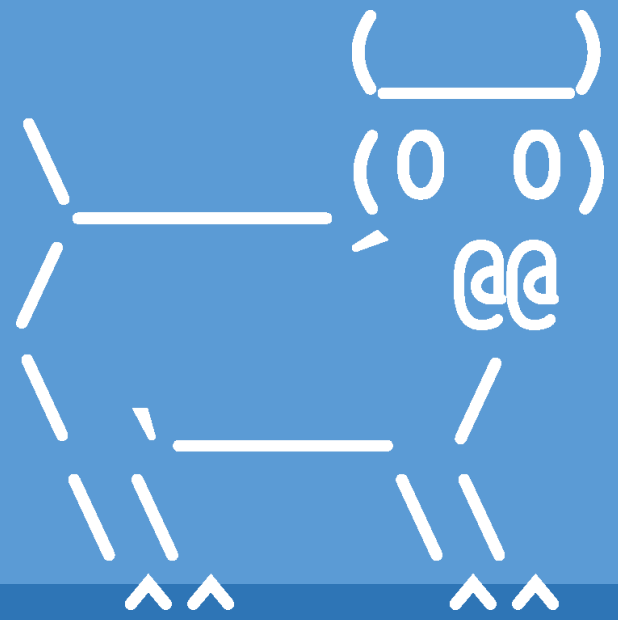
`ord('s')` converts to unicode

`chr(115)` converts back to a string

QUESTIONS?

Please contact me!

Ciphers



Introduction to Computer Science

Iris Howley

TODAY'S LESSON

Ciphers

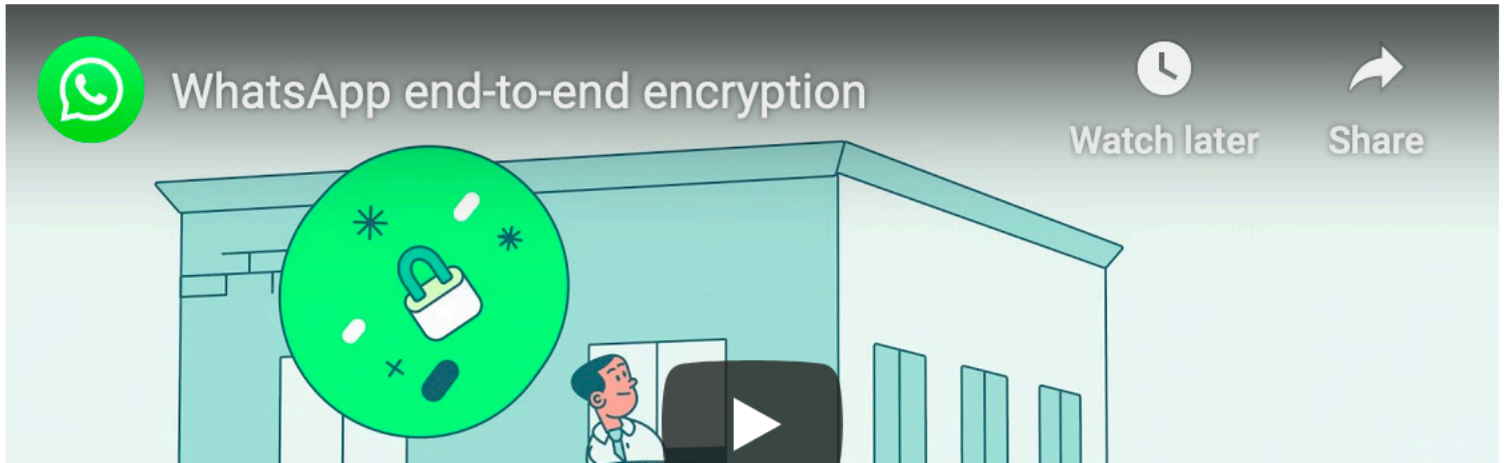
(Encrypting and decrypting data for security)

What is encryption?

FAQ

End-to-end encryption

Privacy and security is in our DNA, which is why we have end-to-end encryption. When end-to-end encrypted, your messages, photos, videos, voice messages, documents, status updates and calls are secured from falling into the wrong hands.



What is encryption?

"the process of converting messages in ordinary language, or other information into a secret coded form that cannot be interpreted without knowing the secret method for interpretation, called the key."

Simple Encryption: Caesar Ciphers

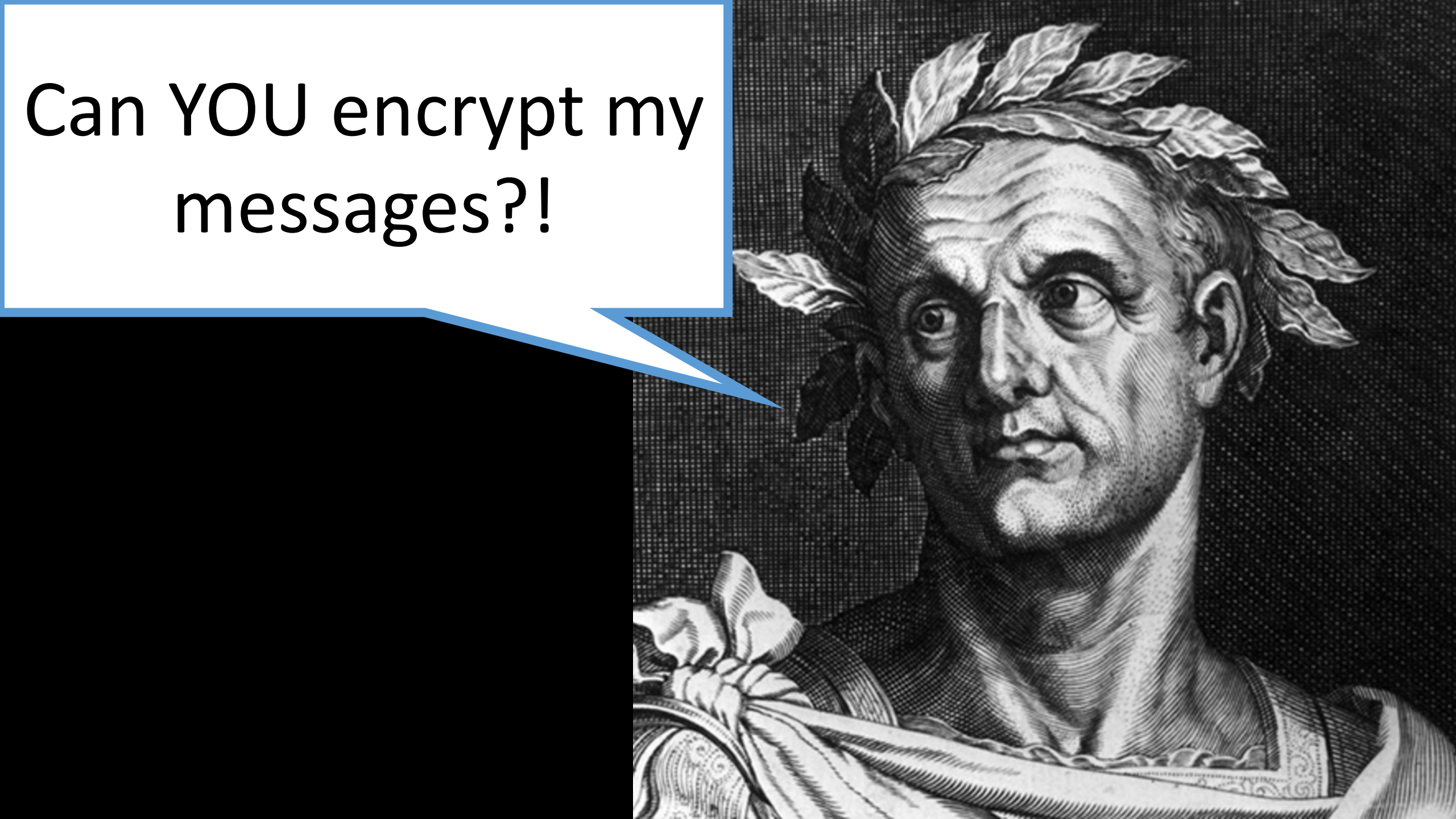
I send private communication via secret messages, where I shift each letter of the alphabet by a certain number of characters.

Simple Encryption: Caesar Ciphers

With a shift of 1, 'a' would become 'b', 'b' would be 'c', all the way to 'z' which would wrap around and become 'a'.

Simple Encryption: Caesar Ciphers

With a shift of 2, 'a' would become 'c', 'b' would be 'd', all the way to 'y' which would wrap around and become 'a'.


Can YOU encrypt my messages?!

How do we rotate a character by 1?

- a → b, b →c, …, y→z, z → a, etc

Any ideas?

```
>>> rotated = ord('a') + 1
>>> chr(rotated)
'b'
```

Rotating Letters

- `chr(ord('b') + 1)`
  - `'c'`
- `chr(ord('z') + 1)`
  - `'{'`

What happened here?

How do we wrap around back to 'a' after 'z'?

v	w	x	y	z	{
118	119	120	121	122	123

Rotating Letters

a	…	v	w	x	y	z	{
97	…	118	119	120	121	122	123

- What we want is for 123 to wrap around back to 97…
- Let's solve a simpler problem:

a	…	v	w	x	y	z	{
0	…	21	22	23	24	25	26

- Any number over 25 should wrap back around to 0, 1, 2, etc.
- What might we use to do that?

Rotating Letters

a	…	v	w	x	y	z	{
0	…	21	22	23	24	25	26

- 24 _____ = 24
- 25 _____ = 25
- 26 _____ = 0
- 27 _____ = 1
- 28 _____ = 2

- 24 % 26 = 24
- 25 % 26 = 25
- 26 % 26 = 0
- 27 % 26 = 1
- 28 % 26 = 2

Rotating Letters

a	…	v	w	x	y	z	{
0	…	21	22	23	24	25	26

- Our formula is: rotatedValue = value % 26
- But that's for 0+, how do we convert to 97+?

a	…	v	w	x	y	z	{
97	…	118	119	120	121	122	123

- Subtract 97 to adjust to 'a' starting at 0:
  - rotatedValue = (originalValue - 97) % 26
- And then add 97 back so we can convert to real unicode value
  - rotatedValue = 97 + (originalValue - 97) % 26

Rotating Numbers

- `rotatedValue = 97 + (originalValue - 97) % 26`
- Generalizes to:
  - `rotatedVal = ord('a') + (ord(givChar)-ord('a')) % 26`
- Convert from Unicode to character:
  - `rotatedChr = chr(ord('a') + (ord(givChar)-ord('a')) % 26)`

- …Are we missing anything?

Rotating Numbers

- We need to rotate by 'n':
  - `rotCh = chr(ord('a')+(((ord(givChar)-ord('a'))+ n)%26))`

- …Are we missing anything?

- What about…

A	B	…	Y	Z	[	\
65	66	…	89	90	91	92

- `start = ord('A')`
- `rotCh =chr(start+(((ord(givChar)-start)+ n)%26))`

Crypt.py

```
def encrypt(astr, shift):
    """ Encrypt code, similar to p.81 of textbook """
    result = ''
    for letter in astr:
        if letter.isalpha():
            if letter.isupper():
                start = ord('A')
            elif letter.islower():
                start = ord('a')

            rotateVal = (ord(letter) - start + shift) % 26 + start
            result += chr(rotateVal)
        else:
            result += letter
    return result
```

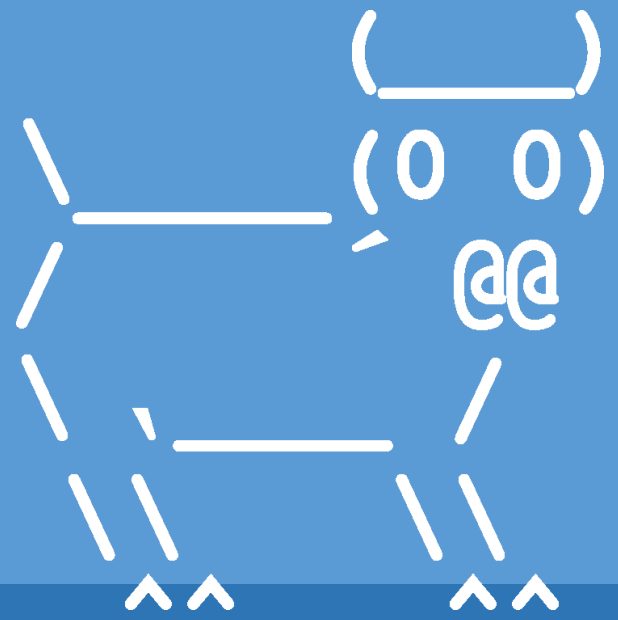
```
if __name__ == '__main__':
    #print("shift by 1", encrypt('abcd',
1))
    #print("shift by 2", encrypt('abcd',
2))


    orig = "Hello, my name is Iris!"
    enc = encrypt(orig, 15)
    dec = encrypt(enc, -15)
    print("encrypted: ", enc)
    print("decrypted: ", dec)
```

See crypt.py

QUESTIONS?

Please contact me!

Lab 8



Introduction to Computer Science

Iris Howley

CONNECT TO CAMPUS VIA VPN BEFORE `git clone` FROM GITLAB AND BEFORE PUSHING TO GITLAB

https://oit.williams.edu/help-guides/wifi-and-wired-connections/vpn/

TODAY'S LESSON

Building our own ciphers

(An overview of Lab 8)

WHAT IS A CIPHER?

See Lecture 21: Ciphers

WHAT IS ENCRYPTION?

See Lecture 21: Ciphers

HOW TO ROTATE LETTERS?

See Lecture 21: Ciphers

Class Diagram: Lab08

`Message` represents an object with a `text` attribute

`Message`

(and some other properties & methods),
See Lab document + starter code)

What's an attribute?

See Lecture 16 & 17

What's a method?

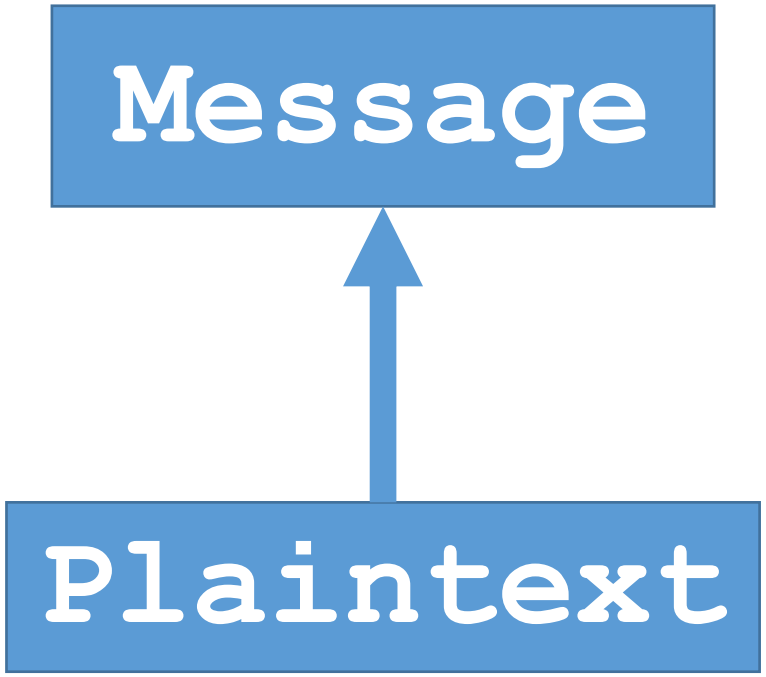
See Lecture 16 & 17

What's a property?

See Lecture 16 & 17

Class Diagram: Lab08

`Message` is `Plaintext's` super-class



`Plaintext` is a sub-class of `Message`

What's super/sub-classes?

See Lecture 19 & 20

`Plaintext` represents a plain text message that can be encrypted.

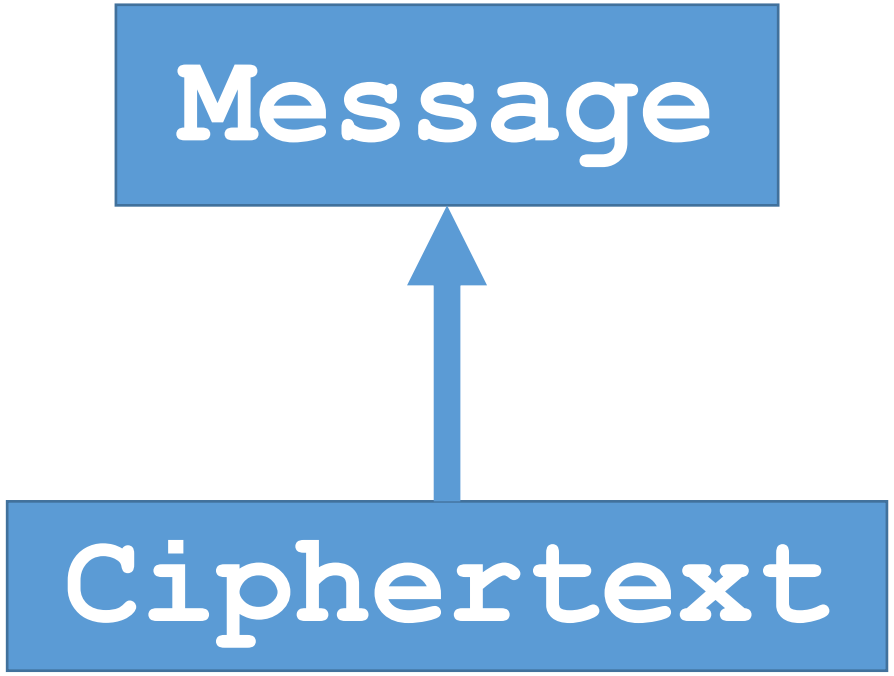
We can call `Message`'s initializer from `Plaintext` using:

```
super().__init__(text)
```

See Lecture 20 on Inheritance & Methods

Class Diagram: Lab08

`Message` is `Ciphertext`'s super-class



`Ciphertext` is a sub-class of `Message`

`Ciphertext` represents an encrypted text message that can be decrypted.

We can call `Message`'s initializer from `Ciphertext` similar to how we do so with `Plaintext`.

See Lecture 19 & 20

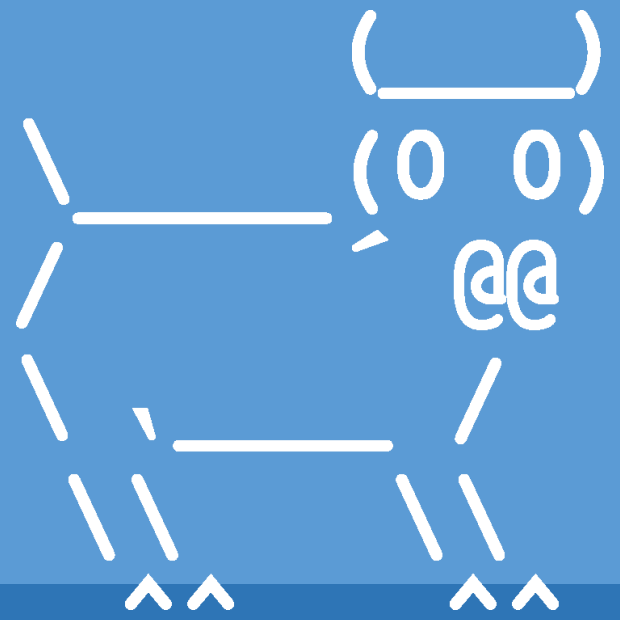
Lab 08: Helper Methods

- These helper functions are useful and fully implemented:
  - `canonical`
  - `loadWords`
  - `isWord`
  - `loadStory`
  - Use them! Don't make redundant code!
- `rotateLetter` is a helper function that you need to implement
  - See Lecture 21!
  - Look at page 81 of the textbook!

QUESTIONS?

Please contact me!

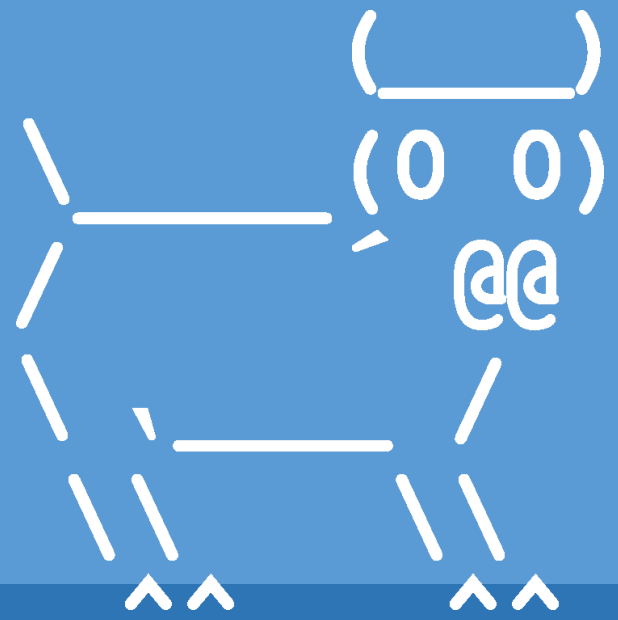
Lab 8: Starter Code



Introduction to Computer Science

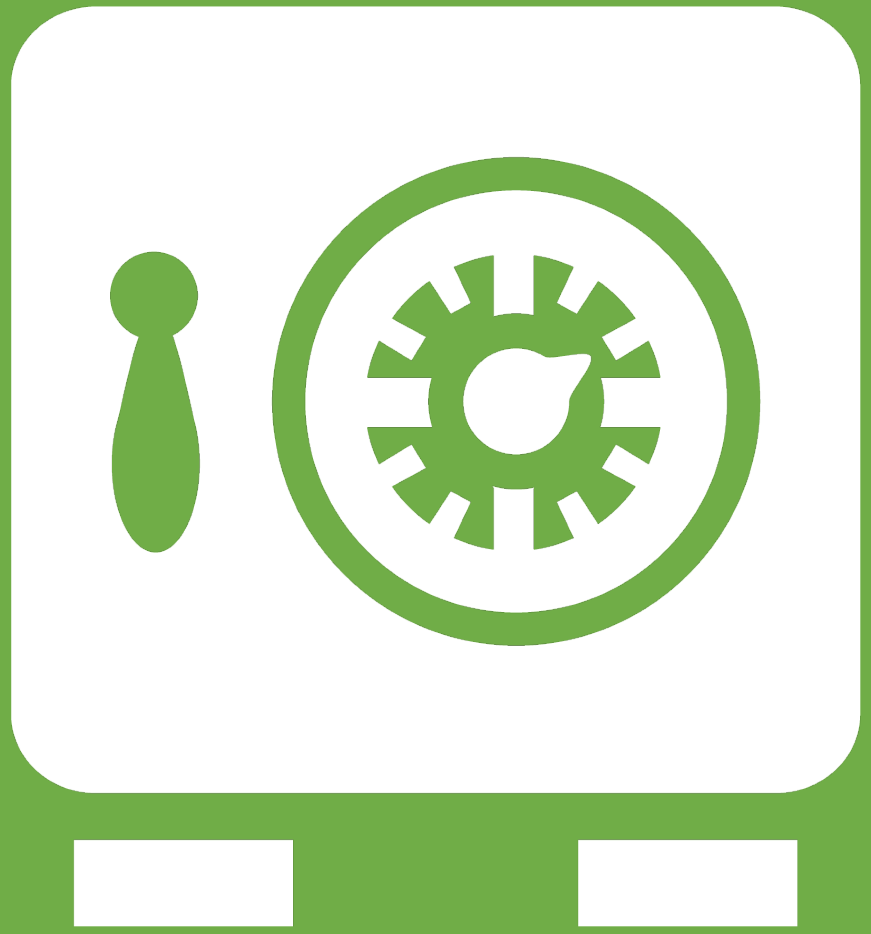
Iris Howley

Lab 8:
PDF Document



Introduction to Computer Science

Iris Howley



Leftover Slides

TODAY'S LESSON

`super()`

(Calling super methods implicitly)

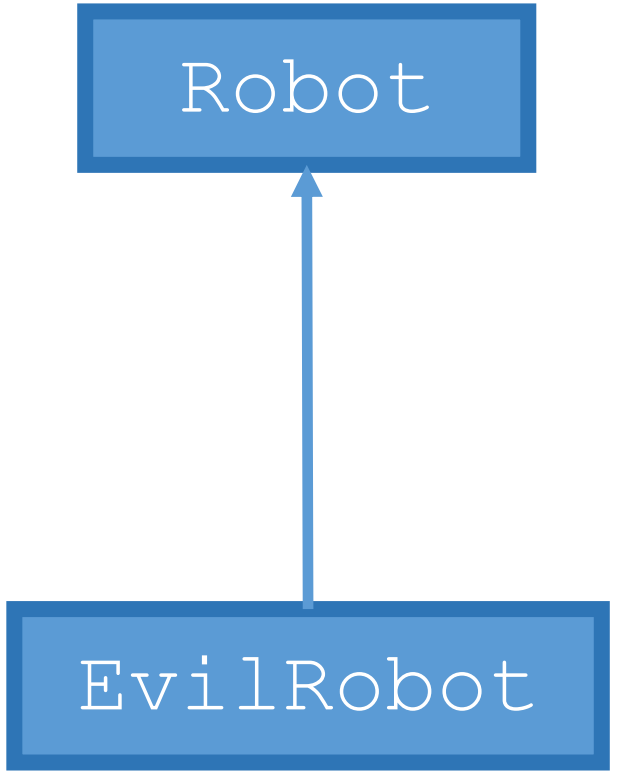
Our Simple Robot Class

```
class Robot:
    __slots__ = ['name']
    def __init__(self, nm):
        self.name = nm

class EvilRobot(Robot):
    __slots__ = ['mission']
    def __init__(self, nm, misn):
        Robot.__init__(self, nm)
        self.mission = misn
```

Our Simple Robot Class

```
class Robot:
    __slots__ = ['name']
    def __init__(self, nm):
        self.name = nm

class EvilRobot(Robot):
    __slots__ = ['mission']
    def __init__(self, nm, misn):
        Robot.__init__(self, nm)
        self.mission = misn
```

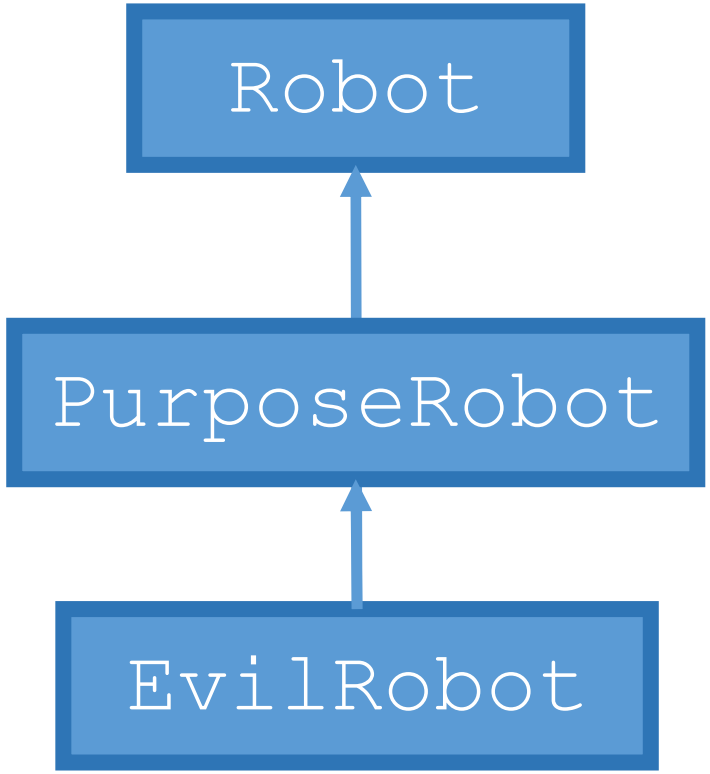
What happens if our class structure changes?



Our Simple Robot Class

```
class Robot:
    __slots__ = ['name']
    def __init__(self, nm):
        self.name = nm

class EvilRobot(PurposeRobot):
    __slots__ = ['mission']
    def __init__(self, nm, misn):
        Robot.__init__(self, nm)
        self.mission = misn
```

What happens if our class structure changes?



Calling Super Methods with `super()`

```
class Robot:
    __slots__ = ['name']
    def __init__(self, nm):
        self.name = nm

class EvilRobot(Robot):
    __slots__ = ['mission']
    def __init__(self, nm, misn):
        super().__init__(nm)
        self.mission = misn
```

`super().__init__(nm)`

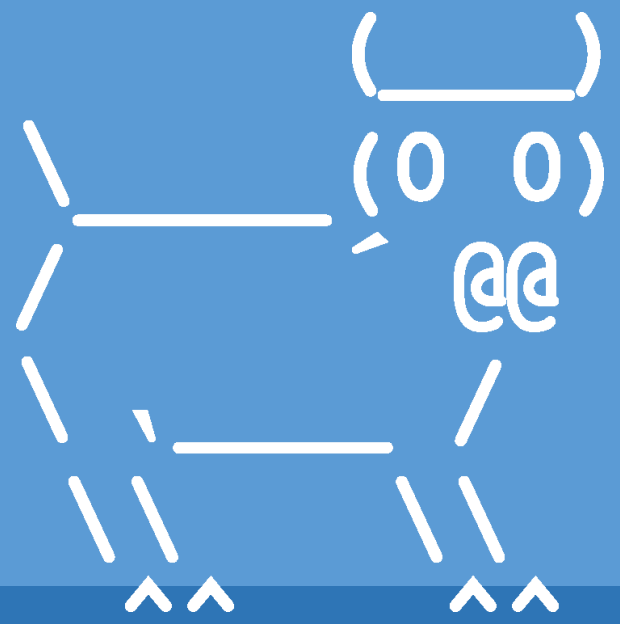
is similar to:

`Robot.__init__(self, nm)`

`super()` lets us call the super-class *implicitly*…

…and we no longer need to pass `self`

Ciphers Example



Introduction to Computer Science

Iris Howley

TODAY'S LESSON

Ciphers Example

(Using our cipher.py)

crypt.py

```
if __name__ == '__main__':
    with open('crypt.py', 'r') as fin:
        codeFile = fin.read()
    with open('encrypted', 'w') as fout:
        fout.write(encrypt(codeFile, 13))
    with open('encrypted', 'r') as fin:
        print(encrypt(fin.read(), -13))
```

Stdin

< for stdin

> for pushing output to a file

- python3 crypt.py < crypt.py > whatever
  - Encrypts stdin into `whatever` file
  - Rotates by 13 (by default, see program)
- python3 crypt.py < whatever > second.py
  - Rotates 'whatever' by 13, stores in second.py
  - What happens when you rotate by 13 twice?

| to run a second command

- python3 crypt.py < crypt.py | python3 crypt.py
  - Rotates by 13, then rotates by 13 again

QUESTIONS?

Please contact me!