Python Strings

String

- some characters inside quotes

String

- some characters inside quotes

  - “Hello World!”
  - 'Hello World!'

String

- some characters inside quotes

What is a Character ?

One of these things

String

- some characters inside quotes

  - “Hello World!”
  - 'Hello World!'

Operations on Strings

- newUser = 'Thomas'
- greeting = 'Hello'
- greeting = greeting + newUser

- result = '25' + '2'

- 'Fun' * 3 → 'FunFunFun'

You Try It

Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license" for more information.

```
>> newUser = 'Thomas'
>> greeting = 'Hello'
>> greeting = greeting + newUser
>> greeting
>> result = '25' + '2'
>> result
```

String Length

- newUser = 'Thomas'
- greeting = 'Hello'
- len(newUser)
- len(greeting)

Collection Data Types

- Primitive data types
  - int
  - float
  - bool
- Collection data types
  - Strings
  - Lists

Collection Data Types

- Can work with Collection Data types as
  - Single entity (the whole)

Collection Data Types

- Can work with Collection Data types as
  - Single entity (the whole)

  - userName = 'thomas'
  - newUser = userName

Collection Data Types

- Can work with Collection Data types as
  - Single entity (the whole)
  - Individual elements

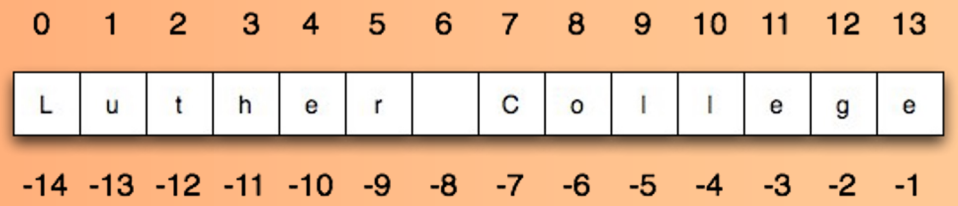
Collection Data Types

- Can work with Collection Data types as
  - Single entity (the whole)
  - Individual elements

  - userName = 'thomas'
  - firstChar = userName[0]

Index Operator

- Selects a single character from a string
- Python uses square brackets to enclose the index
- characters are accessed by their position or index value
- positions are named from right to left using negative numbers where -1 is the rightmost index



Strings

- sequential collections of characters

  - userName = 'thomas'
  - userName[0] 't'
  - userName[1] 'h'
  - userName[2] 'o'
  - userName[3] 'm'
  - userName[4] 'a'
  - userName[5] 's'

Strings

- sequential collections of characters

  - userName = 'thomas'
  - userName[0] 't'
  - userName[1] 'h'
  - userName[2] 'o'
  - userName[3] 'm'
  - userName[4] 'a'
  - userName[5] 's'

Element Numbering starts at 0

Strings are Immutable

- Not exactly
- What 'they' mean is
  - inputLine = 'Klaatu barada nikto'
  - inputLine[3] = 'o'

  - inputLine = "I die, repair me, do not retaliate."

  - line = 'one simply'
  - line = line + “doesn't walk into”

Empty String

- sequence of zero characters
- ‘’ or “”
  - (two single or two double quotes with nothing in between)

Empty String

- sequence of zero characters
- ‘’ or “”
  - (two single or two double quotes with nothing in between)

This is one of those weird computer things
that you probably wouldn't even think of doing in real life
but come in real handy in the computer world

Strings are Objects

- Strings have Attributes
  - Collection of characters
- Strings have methods
  - lower()

```
>>> ss = "Hello World"
>>> ss.lower()
'hello world'
>>> ss
'Hello World'
>>>
```

Strings are Objects

- Strings have Attributes
  - Collection of characters
- Strings have methods
  - lower()

```
1 ss = "Hello, World"
2 print(ss.upper())
3
4 tt = ss.lower()
5 print(tt)
6
```

```
HELLO, WORLD
hello, world
```

Methods & Dot Notation

yertle.right(90) right is a method

. “dot notation”

connects name of object to

name of method

String Methods

upper	none	Returns a string in all uppercase
lower	none	Returns a string in all lowercase
capitalize	none	Returns a string with first character capitalized, the rest lower
strip	none	Returns a string with the leading and trailing whitespace removed
lstrip	none	Returns a string with the leading whitespace removed
rstrip	none	Returns a string with the trailing whitespace removed
count	item	Returns the number of occurrences of item
replace	old, new	Replaces all occurrences of old substring with new
center	width	Returns a string centered in a field of width spaces
ljust	width	Returns a string left justified in a field of width spaces
rjust	width	Returns a string right justified in a field of width spaces
finditem		Returns the leftmost index where the substring item is found
rfind	item	Returns the rightmost index where the substring item is found
index	item	Like find except causes a runtime error if item is not found
rindex	item	Like rfind except causes a runtime error if item is not found

Length

- string = “The spam tastes delicious”
- stringLength = len(string)
- print(string[stringLength])

- len(string) – 1 Last element of string

String Slicing

```
singers = "Peter, Paul, and Mary"
print(singers[0:5])
print(singers[7:11])
print(singers[17:21])
print(singers[:5])
print(singers[4:])
```

String Slicing

```
fruit = "banana"
print(fruit[:3])
print(fruit[3:])
```

String Comparison

```
word = "banana"
if word == "banana":
    print("Yes, we have bananas!")
else:
    print("Yes, we have NO bananas!")
```

String Comparison

```
word = "zebra"
if word < "banana":
    print("Your word, " + word + ", comes before banana.")
elif word > "banana":
    print("Your word, " + word + ", comes after banana.")
else:
    print("Yes, we have no bananas!")
```

String Comparison

```
print("apple" < "banana")


print("apple" == "Apple")
print("apple" < "Apple")
print("apple" < "applepie")
print("apple" < "Applepie")
```

ASCII Table

Hex	Dec	Char		Hex	Dec	Char	Hex	Dec	Char	Hex	Dec	Char
0x00	0	NULL	null	0x20	32	Space	0x40	64	@	0x60	96	`
0x01	1	SOH	Start of heading	0x21	33	!	0x41	65	A	0x61	97	a
0x02	2	STX	Start of text	0x22	34	"	0x42	66	B	0x62	98	b
0x03	3	ETX	End of text	0x23	35	#	0x43	67	C	0x63	99	c
0x04	4	EOT	End of transmission	0x24	36	$	0x44	68	D	0x64	100	d
0x05	5	ENQ	Enquiry	0x25	37	%	0x45	69	E	0x65	101	e
0x06	6	ACK	Acknowledge	0x26	38	&	0x46	70	F	0x66	102	f
0x07	7	BELL	Bell	0x27	39	'	0x47	71	G	0x67	103	g
0x08	8	BS	Backspace	0x28	40	(	0x48	72	H	0x68	104	h
0x09	9	TAB	Horizontal tab	0x29	41	)	0x49	73	I	0x69	105	i
0x0A	10	LF	New line	0x2A	42	*	0x4A	74	J	0x6A	106	j
0x0B	11	VT	Vertical tab	0x2B	43	+	0x4B	75	K	0x6B	107	k
0x0C	12	FF	Form Feed	0x2C	44	,	0x4C	76	L	0x6C	108	l
0x0D	13	CR	Carriage return	0x2D	45	-	0x4D	77	M	0x6D	109	m
0x0E	14	SO	Shift out	0x2E	46	.	0x4E	78	N	0x6E	110	n
0x0F	15	SI	Shift in	0x2F	47	/	0x4F	79	O	0x6F	111	o
0x10	16	DLE	Data link escape	0x30	48	0	0x50	80	P	0x70	112	p
0x11	17	DC1	Device control 1	0x31	49	1	0x51	81	Q	0x71	113	q
0x12	18	DC2	Device control 2	0x32	50	2	0x52	82	R	0x72	114	r
0x13	19	DC3	Device control 3	0x33	51	3	0x53	83	S	0x73	115	s
0x14	20	DC4	Device control 4	0x34	52	4	0x54	84	T	0x74	116	t
0x15	21	NAK	Negative ack	0x35	53	5	0x55	85	U	0x75	117	u
0x16	22	SYN	Synchronous idle	0x36	54	6	0x56	86	V	0x76	118	v
0x17	23	ETB	End transmission block	0x37	55	7	0x57	87	W	0x77	119	w
0x18	24	CAN	Cancel	0x38	56	8	0x58	88	X	0x78	120	x
0x19	25	EM	End of medium	0x39	57	9	0x59	89	Y	0x79	121	y
0x1A	26	SUB	Substitute	0x3A	58	:	0x5A	90	Z	0x7A	122	z
0x1B	27	FSC	Escape	0x3B	59	;	0x5B	91	[	0x7B	123	{
0x1C	28	FS	File separator	0x3C	60	<	0x5C	92	\	0x7C	124	|
0x1D	29	GS	Group separator	0x3D	61	=	0x5D	93	]	0x7D	125	}
0x1E	30	RS	Record separator	0x3E	62	>	0x5E	94	^	0x7E	126	~
0x1F	31	US	Unit separator	0x3F	63	?	0x5F	95	_	0x7F	127	DEL

Strings ↔ ASCII

- ord('A')
- chr(65)

Traversal

- s = “She cannae take any more”

```
for aChar in s:
    print(aChar)
```

Traversal

```
s = “She cannae take any more”
for index in range(len(s)):
    print(s[index])
```

Traversal

```
s = “She cannae take any more”
position = 0
while position < len(s):
    print(s[position])
    position = position + 1     # position =+ 1
```

in and not in Operators

- print('p' in 'apple')
- print('i' in 'apple')
- print('ap' in 'apple')
- print('pa' in 'apple')
- print('a' in 'a')
- print('apple' in 'apple')
- print('' in 'a')
- print('' in 'apple')
- print('x' not in 'apple')