Sequence types:

- Sequence types aggregate individual items into ordered collections of items.
- 4 common: strings lists tuples and ranges
- They share terminology: it is reasonable to talk about the length of any item of type string, list, tuple or range in exactly the same manner
- Strings are ordered sequences of characters(in Python, characters are just strings of length 1) enclosed in either single or double quotes
- Strings contain some normal characters , but also contain some unusual unprintable characters or characters from different alphabets.
- 'This is a test', 14 characters in it
- 'a b c'
- 'I don't know' #error "I don';t know" #no error
- ' ' does not equal "
- '\t' tab
- '\n' newlone
- ' \r ' carriage return Returning carriage and goes newline
- '\' ' embedded quote
- ' \ \ ' backward slash
- ***There is a difference between a value and its printed representation:***
- >>> "this\tthat"
- 'This\tthat' REPL provides value
- print("this\tthat"")
- This that
-

STrings have own set of operations:

```
>>> 'test'*3
'testtesttest'
>>> "this is" + ' a test'
'this is a test'
>>> 4*'A'+3*'b'
'AAAAbbb'
```

- The operator ***in*** are specific to strings(and sequences in general).

```
>>> 'pin' in "Happiness"
True
>>> 'S' in 'seltzer'
False
>>> '' in "anything"
True
>>> '7' not in 'foobar'
True
```

-
-

Tuple:

- Python allows you to aggregate items (of any type) into a collection called a tuple
- Tuples are (almost always) enclosed in parentheses, and items are separated by commas (every non-empty tuple requires at least one comma; why?).

```
(1, 3, 5)
(0.01, True, False, "my house")
()                  # empty tuple of 0 elements
(1)                 # not what you think! why?
(1,)                # probably what you meant!
```

-
- Yeah

```
>>> "string"*2
'stringstring'
>>> (1, "string"*2, 3)
(1, 'stringstring', 3)
>>> 1 in (1, "string"*2, 3)
True                        # why? 1 is an element or subsequence of the tuple
>>> 'ing' in (1, "string"*2, 3)
False                       # why not? "ing" is not an element or subsequence of the tuple
>>> 'ing' in ("string"*2)
True                        # huh? "stringstring" is not a tuple: "ing" is in "stringstring"
>>> 'ing' in ("string"*2,)
```

- False # why not? "ing" is not an element or subsequence of the tuple

```
>> (3, 5) in (1, 3, 5, 7, 9)
'rue
```

-