1. Strings are a sequence of characters. Unlike C where strings are nul terminated character arrays.

```
>>> s = 'this is a string' # this is how you define a string
>>> print(s)
this is a string
```

2.Indexing into a string returns a new string which contains the element at the desired index.

```
>>> s
'this is a string'
>>> s[0]
't'
>>> s[1]
'h'
>>> s[-1]
'g'
>>> s[-2]
'n'
```

The result of indexing is NOT a position in the string array (as it is in C).

```
>>> s
'this is a string'
>>> start = s[0]
>>> start
't'
```

---

3. String slicing: given a string s, s[p:q] returns a new string.
  s[p:q] contains elements of s from index p , upto but not including index q.

```
>>> s = 'this string is gonna get sliced'
>>> s[0:3]
'thi'
>>> s[0:4]
'this'
>>> s[2:] # all the string except the first 2 characters.
'is string is gonna get sliced'
>>> s[:-3] # all the string except the last 3 characters.
'this string is gonna get sli'
```

You can think of string slicing as the act of cutting a string along 2 vertical lines per below.

```
 +---+---+---+---+---+---+
```

```
 | P | y | t | h | o | n |
 +---+---+---+---+---+---+
 0   1   2   3   4   5   6
-6  -5  -4  -3  -2  -1
```

---

4. strings are IMMUTABLE (can not be changed).

```
>>> s = 'i can\'t change'
>>> s[0] = 'j'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

But how can we manipulate strings, answer: create new strings.

```
>>> s
"i can't change"
>>> t = 'j' + s[1:] # BTW, + operator in strings, concatenates ('a' + 'bc' = 'abc').
>>> t
"j can't change"
```

---

5. The built-in function len() returns the length of a string

```
>>> s = 'count this'
>>> len(s)
10
```
