C11 String Objects

1. common to call method on literal string because string is first class object in Python

```python
print("Hello, World.".upper())
```

2. able to subclass the string

```python
class MyString(str):
    def __str__(self):
        return self[::-1]

s = MyString("Hello, World.")
print(s)
```

3. casefold similar to lower but it remove all distinction even in unicode

```python
print("Hello, World.".casefold())
```

4. string is immutable

```python
s1 = "Hello, World."
s2 = s1.upper()

print(id(s1))
print(id(s2))
```

5. concatenate the string by using plus sign

```python
s1 = "Hello, World."
s2 = "Hello, Everyone."

print(s1 + s2)
```

6. concatenate the string by using single quote with space

```python
s1 = 'Hello, World.' ' Hello, Everyone.'
print(s1)
```

7. format the string with .format() and {}(placeholder)

```
x = 42
y = 73
#will take the arguments with the order (without any specification)
print("The number is {} {}".format(x, y))
#or.. named the placeholder with the variable
print("The number is {a} {b}".format(a = x, b = y))
#or.. named the placeholder by using positional arguments (which start at one)
print("The number is {1} {0}".format(x, y))
#formating instruction which preceded by the colon(:)
                    #right justify with 5 spaces
print("The number is {0: < 5} {1: > 5}".format(x, y))
                                #left justify with 5 spaces

#format the string with ,
x = 42 * 747 * 1000
print("The number is {:,}".format(x))

#format the decimal numbers, f stands for default decimal places
print("The number is {:.2f}".format(x))

#format the string to octact(o), hexadecimal(x), binary(b)
print("The number is {:b}".format(x))

#used f string to replace .format() (starting from Python 3.6 and above)
print(f"The number is {x}")
```

8. split and join the string

```
s = 'This is a long string with a bunch of words in it.'
print(s.split())
l = s.split()
s2 = ':'.join(l)
print(s2)
```