Python Strings

Python strings are immutable:

```python
s = 'abc'
s[2] = 'd'
s = 'abd'
s = s[:len(s)-1] + 'd'
```

These don't change the string ‘abc’

they reassign the string variable `s`

Accessor and Slices

```python
s = 'A diamond necklace played the pawn'
print(s[5])          # prints m
print(s[len(s)-1])    # prints n
print(s[-1])          # prints n
print(s[-2])          # prints w
print(s[10:18])       # prints necklace
t = s[19:]             # t is 'played the pawn'
u = s[:9]             # u is 'A diamond'
print(s[-4:])          # prints pawn
v = s[:16]            # v is 'A diamond necklace'
```

Methods

Python provides a broad array of string operations.

Because strings are immutable, the operations do not modify the string.

Most methods fall into one of the following categories:

- Return a new string
- Return a Boolean
- Return an int
- Return a tuple or list
Methods that return a string

- `s.capitalize()`
- `s.lower()`
- `s.upper()`
- `s.join(t)`
- `s.replace(old, new)`
- `s.strip()`
- `s.rstrip()`

Upper and lower case

- `s.lower()`
  returns an all lower case conversion of `s`
- `s.upper()`
  returns an all upper case conversion of `s`
- `s.capitalize()`
  returns `s` with first character capitalized

Examples: upper and lower case

```
s = 'Hand in hand some drummed along'
s.lower()                   # returns 'hand in hand some drummed along'
s.upper()                   # returns 'HAND IN HAND SOME DRUMMED ALONG'
s.lower().capitalize()      # returns 'Hand in hand some drummed along'
'123'.capitalize()          # returns '123'
```

Join

- `s.join(t)`
  `t` is an iterable of strings
  returns a concatenation of strings in `t` with string `s` as a separator
**Examples: join**

```python
s = ''
t = ['this', 'is', 'a', 'test']
s.join(t)  # t is an iterable of strings
          # returns 'this is a test'
''.join(t)
          returns 'this,is,a,test'
' la '.join(t)
          returns 'this la is la a la test'
```

**Replace**

```python
s.replace(old, new)
old is a string
new is a string
returns a string in which occurrences of old
have been replaced with new
```

**Examples: replace**

```python
s = 'this is a test'
s.replace('is', 'was')
          returns 'this was a test'
          # unintended consequences
s.replace(' is', ' was')
          returns 'this was a test'
          # d'oh!
```

**Strip**

```python
s.strip()
          returns s without leading and trailing whitespace
s.rstrip()
          returns s without trailing whitespace
s.strip(t)
          returns s without leading and trailing chars in t
s.rstrip(t)
          returns s without trailing chars in t
```
Examples: strip

s = 'Early one mornin the sun was shinin \n'
s.strip()
  returns 'Early one mornin the sun was shinin'
s.rstrip()  
  returns 'Early one mornin the sun was shinin'

Methods that return a Boolean

• s.startswith(prefix)
• s.endswith(suffix)
• s.isalnum()
• s.isalpha()
• s.isdigit()
• s.isupper()
• s.islower()
• s.isspace()
• s.istitle()

Examples: startswith & endswith

s = 'She loves you, yeah, yeah, yeah'
s.startswith('She lo
')  
  returns True
s.startswith('she')  
  returns False
s.lower().startswith('she')  
  returns True
s.endswith('ah, yeah')  
  returns True
Examples: `isalnum`, `isalpha`, & `isdigit`

```python
s = 'Waiting for the break of day'
t = '25 or 6 to 4'
u = '314159'
s.isalnum()  # returns True
s.isalpha()  # returns True
t.isalnum()  # returns True
```

Examples: `islower` & `isupper`

```python
s = 'Waiting for the break of day'
t = '25 OR 6 TO 4'
u = '314159'
s.islower()  # returns False
s.upper().isupper()  # returns True
t.isupper()  # returns True
```
Examples: isspace & istitle

```python
s = ' "t "n'
t = 'Let Him Run Wild'

s.isspace()  # returns True

istitle()
  returns True

t.upper().istitle()
  returns False
```

Methods that return an int

- s.count(sub)
- s.find(sub)
- s.rfind(sub)
- s.index(sub)
- s.rindex(sub)

Count

```python
s.count(sub)  # sub is a string
  returns the number of occurrence of sub in s
```

Examples: count

```python
s = 'de do do do de da da da'
s.count('da')
  returns 3

s.count('da ')
  returns 2

s.count('do do')
  returns 1
```
Find

- `s.find(sub)`
  - sub is a string
  - returns index of first occurrence of sub in s or -1 if not found
- `s.rfind(sub)`
  - sub is a string
  - returns index of last occurrence of sub in s or -1 if not found

Examples: find and rfind

- `s = 'De do do do de da da da'`
- `s.find('do')`
  - returns 3
- `s.rfind('do')`
  - returns 9
- `s.find('de')`
  - returns 12
- `s.find('Do')`
  - returns -1

Index

- `s.index(sub)`
  - sub is a string
  - returns index of first occurrence of sub in s or gives an error if not found
- `s.rindex(sub)`
  - sub is a string
  - returns index of last occurrence of sub in s or gives an error if not found

Examples: index and rindex

- `s = 'De do do do de da da da'`
- `s.index('do')`
  - returns 3
- `s.rindex('do')`
  - returns 9
- `s.index('de')`
  - returns 12
- `s.index('Do')`
  - returns -1

Crash and burn – should have used find
Methods that return a tuple or list

- `s.partition(sep)`
- `s.rpartition(sep)`
- `s.split(sep)`
- `s.rsplit(sep)`
- `s.splitlines()`

Partition

- `s.partition(sep)`
  - first occurrence of `sep` partitions `s`
  - returns tuple: `(s_up_to_sep, sep, s_after_sep)`

- `s.rpartition(sep)`
  - last occurrence of `sep` partitions `s`
  - returns tuple: `(s_up_to_sep, sep, s_after_sep)`

Examples: partition

```python
s = 'de do do do de da da da'
s.partition('do')
returns ('de ', 'do', ' do do de da da da')
s.rpartition('do')
returns ('de do do ', 'do', ' de da da da')
s.partition('do do do')
returns ('de ', 'do do do', ' de da da da')
s.rpartition('do do do')
returns ('de ', 'do do do', ' de da da da')
```

Examples: partition

```python
s = 'Help me Rhonda help help me Rhonda'
s.partition('Rhonda')
returns ('Help me ', 'Rhonda', ' help help me Rhonda')
s.rpartition('Rhonda')
returns ('Help me Rhonda help help me ', 'Rhonda', '')
```
Split

s.split(sep [, max])
  each occurrence of sep partitions s, up to max times
  returns list of tokens
s.rsplit(sep [, max])
  each occurrence of sep partitions s, up to max times
  returns list of tokens
split and rsplit are same unless occurrences > max

Examples: split

s = 'de do do do de da da da'
s.split(' ') or s.split()
returns ['de', 'do', 'do', 'do', 'de', 'da', 'da', 'da']
s.rsplit(' ') or s.rsplit()
returns ['de', 'do', 'do', 'do', 'de', 'da', 'da', 'da']
s.split(' ', 3)
returns ['de', 'do', 'do', 'do de da da da']
s.rsplit(' ', 3)
returns ('de do do do', 'da', 'da', 'da')

Examples: split

s = '12,657,489,306'
t = 'de do do do de da da da'
s.split(',')
returns ['12', '657', '489', '306']
t.split('do')
returns ['de ', ' ', ' ', ' de da da da']
s.rsplit('do', 2)
returns ('de do ', ' ', ' de da da da')

Splitlines

s.splitlines()
  splits s at newlines; consumes the newlines
  returns list of lines
Examples: splitlines

```python
s = 'A diamond necklace played the pawn
    Hand in hand some drummed along
    To a handsome man and baton
    Bygone, bygone'

s.splitlines()
returns ['A diamond necklace played the pawn',
        'Hand in hand some drummed along',
        'To a handsome man and baton',
        'Bygone, bygone']
```