manipulating strings

In [2]:
```
# `\'`  ---- Single quote
# `\"`  ---   Double quote
# `\t` --- Tab
# `\n` --- Newline (line break)
# `\\` --- Backslash
# `\b`---  Backspace
# `\ooo` ---  Octal value
# `\r`  ---  Carriage Return
```

In [4]:
```
print("Hello there!\nHow are you?\nI'm doing fine.")
```

```
Hello there!
How are you?
I'm doing fine.
```

Raw strings

In [7]:
```
A raw string entirely ignores all escape characters and prints any backslash that a
 in the string
# Raw strings are mostly used for regular expression definition
```

In [9]:
```
print(r"Hello there!\nHow are you?\nI\n'm doing fine.")
```

```
Hello there!\nHow are you?\nI\n'm doing fine.
```

multiline strings

In [12]:
```
print(
    """Dear Alice,

    Eve's cat has been arrested for catnapping,
    cat burglary, and extortion.

    sincerely,
    Bob"""
)
```

```
Dear Alice,

   Eve's cat has been arrested for catnapping,
   cat burglary, and extortion.

   sincerely,
   Bob
```

indexing and slicing strings

In [ ]:
```
Hello world
012345678910
```

indexing

In [17]:
```
spam='Hello world!'
```

In [19]:
```
spam[0]
```

Out[19]: 'H'

In [21]:
```
spam[4]
```

Out[21]: 'o'

In [23]:
```
spam[-1]
```

Out[23]: '!'

slicing

In [26]:
```
spam='Hello world!'
```

In [28]:
```
spam[0:5] #prints element from 0th place to 5th index place
```

Out[28]: 'Hello'

In [34]:
```
spam[:6] #prints element from 0 to 6th index place
```

Out[34]: 'Hello '

In [36]:
```
spam[6:-1] # prints element from 6th to -1(last butone) index place
```

Out[36]: 'world'

In [38]:
```
spam[:-1]
```

Out[38]: 'Hello world'

In [41]:
```
name=spam[0:5]
name
```

Out[41]: 'Hello'

the in and not in operators

In [48]:
```
'Hello' in 'Hello world'
```

Out[48]: True

In [50]:
```
'hello' in 'Hello world' #python is case sensitive
```

Out[50]: False

In [52]:
```
'streets' not in 'streets and roads'
```

Out[52]: False

upper(),lower() and title()

Transforms a string to upper, lower and title case:

In [56]:
```
wishes='good morning!'
wishes.upper()
```

Out[56]: 'GOOD MORNING!'

In [58]:
```
wishes.lower()
```

Out[58]: 'good morning!'

isupper() and islower() methods

Returns `True` or `False` after evaluating if a string is in upper or lower case:

In [66]:
```
name='addisson park'
name.islower()
```

Out[66]: True

In [68]:
```
name.isupper()
```

Out[68]: False

In [72]:
```
'HELLO'.isupper()
```

Out[72]: True

In [74]:
```
'abc12345'.islower()
```

Out[74]: True

In [76]:
```
'1234'.islower()
```

Out[76]: False

In [78]:
```
'1234'.isupper()
```

Out[78]: False

the isX string methods

In [ ]:
```
Method      Description
isalpha()   returns `True` if the string consists only of letters.
isalnum()   returns `True` if the string consists only of letters and numbers.
isdecimal() returns `True` if the string consists only of numbers.
isspace()   returns `True` if the string consists only of spaces, tabs, and new-li
istitle()   returns `True` if the string consists only of words that begin with an
             uppercase letter followed by only lowercase characters
```

In [81]:
```
'1234'.isdecimal()
```

Out[81]: True

In [83]:
```
'abc123'.isalnum()
```

Out[83]: True

startswith() and endswith()

In [88]:
```
'Hello world'.startswith('Hello')
```

Out[88]: True

In [90]:
```
'Hello world'.endswith('Hello')
```

Out[90]: False

In [94]:
```
'123abc'.startswith('12')
```

Out[94]: True

In [102…
```
'123abc'.endswith('bc')
'123abc'.startswith('ab')
```

Out[102… False

join() and split()

join() The `join()` method takes all the items in an iterable, like a list, dictionary, tuple or set, and joins them into a string. You can also specify a separator

In [116…
```
''.join(['my','name','is','dell']) # joins without space
```

Out[116… 'mynameisdell'

In [118…
```
'..'.join(['my','name','is','dell'])
```

Out[118… 'my..name..is..dell'

In [122…
```
' rrr '.join(['my','name','is','dell'])
```

Out[122… 'my rrr name rrr is rrr dell'

split()

The `split()` method splits a `string` into a `list` . By default, it will use whitespace to separate the items, but you can also set another character of choice

In [127…
```
'my name is dell'.split() #separates with space
```

Out[127… ['my', 'name', 'is', 'dell']

In [129…
```
'my name is dell'.split(ab)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[129], line 1
----> 1 'my name is dell'.split(ab)

NameError: name 'ab' is not defined
```

In [133…
```
'my,name,is,dell'.split(',')
```

Out[133… ['my', 'name', 'is', 'dell']

justifying text with rjust(),ljust() and center()

In [136…
```
'hello'.rjust(8) # prints with length 8 and string alligns to right
```

Out[136… '   hello'

In [138…
```
'hello'.ljust(9) # prints with length 9 and string alligns to left
```

Out[138… 'hello   '

In [144…
```
'hello'.center(10) # prints with length 10 and string alligns to center
```

Out[144… ' hello   '

An optional second argument to `rjust()` and `ljust()` will specify a fill character apart from a space character:

In [147…
```
'hello'.rjust(10,'*') #prints with length 10,element alligns to right and left part
```

Out[147… '*****hello'

In [149…
```
'hello'.ljust(10,'a')
```
Out[149…
```
'helloaaaaa'
```
In [151…
```
'hello'.center(14,'.')
```
Out[151…
```
'....hello.....'
```

removing whitespace with strip(), rstrip() and lstrip()

In [154…
```
spam='       hello  world     '  #removing the spaces in the string
spam.strip()
```
Out[154…
```
'hello  world'
```
In [156…
```
spam.lstrip() #removes space in the left
```
Out[156…
```
'hello  world     '
```
In [158…
```
spam.rstrip() #removes space in the right
```
Out[158…
```
'       hello  world'
```
In [160…
```
spam='spamspambaconspameggsspamspam'
spam.strip('spam') # strip() removes specified characters only from the brgining
#and end of the string, not from the middle
```
Out[160…
```
'baconspamegg'
```

the count method

Counts the number of occurrences of a given character or substring in the string it is applied to. Can be optionally provided start and end index.

In [163…
```
sentence='one sheep two sheep three sheep four'
sentence.count('sheep')
```
Out[163…
```
3
```
In [165…
```
sentence.count('e')
```
Out[165…
```
9
```
In [167…
```
 sentence.count('e', 6)
 # returns count of e after 'one sh' i.e 6 chars since beginning of string
```
Out[167…
```
8
```

replace method

In [ ]:
```
Replaces all occurences of a given substring with another substring. Can be option
provided a third argument to limit the number of replacements. Returns a new strin
```

In [170…
```
text="Hello, world!"
text.replace("world", "planet")
```

Out[170…
```
'Hello, planet!'
```

In [182…
```
fruits="dell,hp,lenovo,apple"
fruits.replace("apple","anrooth")
```

Out[182…
```
'dell,hp,lenovo,anrooth'
```

In [ ]:
```

```