# STRINGS AND LEN

## STRINGS

- Letters and numbers in a string are called elements.
- Strings are immutable.

## LEN

- Built-in Python function or method
- Counters the number of elements in a string, list, set or dictionary.

```
greet = "Hello World"
len(greet)
```

```
11
```

In [18]:

```
cat = "meow" ; dog = "woof" ; parrot = 'hello'
print(cat, dog, parrot)
print()
print(cat, dog, parrot, sep = ",")
print()
print(cat, dog, parrot, sep = " - ")
print()
print(cat, dog, parrot, end = "!!!!")
```

```
meow woof hello

meow,woof,hello

meow - woof - hello

meow woof hello!!!!
```

In [316]:

```
day = "GOOD DAY"
night = "good night"

len(day)
```

Out[316]:

```
8
```

In [224]:

```
# TAB LIST ALL METHODS AND FUNCTIONS
day.lower()
```

```
night.upper()
night.capitalize()
```

Out[224]:

```
'Good night'
```

In [210]:

```
# CONCATENATION
lang = "C#"
"This is a cool " + lang + " course!"
```

Out[210]:

```
'This is a cool C# course!'
```

In [212]:

```
num = 20
"lecture " + str(num + num) + " is on strings"
```

Out[212]:

```
'lecture 40 is on strings'
```

In [225]:

```
"20" + "50"

type(eval)

type(eval("20"))

eval("20") + eval("50")

eval("20 * 100")
```

Out[225]:

```
2000
```

In [227]:

```
check = "a a a b  b b B c"
```

In [231]:

```
check.count("B")


cosmos = ""
```

Out[231]:

```
1
```

In [317]:

```
messy = """PLEASE #@ UP!!! #@ THIS ---- MESSY ---- DOCSTRING
WHICH //CAN HAVE// MULTPLE LINES
OF STRING//!!!"""



[illegible]
```

```
messy.replace("#@", "").replace("!!!", "").replace("----", "").replace("//"
, " ").replace("\n", " ").lower()
```

Out[317]:

```
'please  up  this  messy  docstring which  can have  multple lines of
string '
```

In [327]:

```
pet = "cat"

# INDEXING STARTS AT ZERO
pet[0]
pet[1]
pet[2]

pet[0] = "b"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-327-2c0d307e6a7e> in <module>()
      6 pet[2]
      7
----> 8 pet[0] = "b"

TypeError: 'str' object does not support item assignment
```

In [250]:

```
print(night)

# INDEXING STARTS AT ZERO FOR ELEMENTS IN A SEQUENCE
night[0]
night[3]
# SLICING
night[5:]
```

```
good night
```

Out[250]:

```
'night'
```

In [267]:

```
code = "P-Y-T-H-O-N-I-C"

print(code)
# STRIDE
code[0::2]

code[1::2]

words = "I saw a cat jump over the moon and into the clouds"
words.split()[3:8]
```

```
P-Y-T-H-O-N-I-C
```

Out[267]:

```
['cat', 'jump', 'over', 'the', 'moon']
```

In [280]:

```
print(words.split())
print()
print(len(words.split()))
```

```
['I', 'saw', 'a', 'cat', 'jump', 'over', 'the', 'moon', 'and', 'into', 'the
', 'clouds']

12
```

In [279]:

```
words.split()[3:12:2]
```

Out[279]:

```
['cat', 'over', 'moon', 'into', 'clouds']
```

In [277]:

```
words.split()[3:8]
```

Out[277]:

```
['cat', 'jump', 'over', 'the', 'moon']
```

In [286]:

```
words.split()[::-2]
```

Out[286]:

```
['clouds', 'into', 'moon', 'over', 'cat', 'saw']
```

In [288]:

```
sc1 = slice(3, 12, 2)
```

In [290]:

```
words.split()[sc1]
```

Out[290]:

```
['cat', 'over', 'moon', 'into', 'clouds']
```

In [300]:

```
new_words = "I climbed a mountain and fly past clouds into sky"

len(new_words.split())

len(new_words.split())


new_words.split()[sc1]
```

Out[300]:

```
['mountain', 'fly', 'clouds', 'sky']
```

In [199]:

```
sky = "I SAW A STAR FALL FROM HEAVEN"
```

```
len(sky)
```

Out[199]:

```
29
```

In [200]:

```
sky[28]
```

Out[200]:

```
'N'
```

In [134]:

```
print(sky)

sky[2:5] + sky[22:]

sky[8:12]
```

```
I SAW A STAR FALL FROM HEAVEN
```

Out[134]:

```
'STAR'
```

In [132]:

```
sky[8:12]
```

Out[132]:

```
'STAR'
```

In [137]:

```
sky[8:12][::-1]
```

Out[137]:

```
'RATS'
```

In [138]:

```
sky[-1]
```

Out[138]:

```
'N'
```

In [252]:

```
# NEGATIVE INDEX COUNTS BACKWARDS WITH -1, INDEXIN STARTS AT ZERO, 0
sky[-6:]
```

Out[252]:

```
'HEAVEN'
```

In [149]:

```
eval("10" + "40")

eval("10 * 40")
```

Out[149]:

```
400
```

In [144]:

```
calc = "The final value is 40 + 37"
eval(calc[19:])
```

Out[144]:

```
77
```

In [145]:

```
calc
```

Out[145]:

```
'The final value is 40 + 37'
```

In [57]:

```
doc = """This -is a #DOCSTRING for -multiple
lines of #string to
print out -and#
can be formatted""".replace("\n", " ").replace("-", " ").replace("#", " ")
doc
```

Out[57]:

```
'This  is a  DOCSTRING for  multiple  lines of  string to  print out  and 
can be formatted'
```

In [60]:

```
doc.upper()
doc.lower()
```

Out[60]:

```
'this  is a  docstring for  multiple  lines of  string to  print out  and 
can be formatted'
```

In [63]:

```
"docstring" in doc.split()
```

Out[63]:

```
False
```

In [65]:

```
doc_list = doc.split()
```

In [66]:

```
" ".join(doc_list)
```

Out[66]:

```
'This is a DOCSTRING for multiple lines of string to print out and can be f
ormatted'
```

In [67]:

```
bill = "This total price for the pizza and chips is $25. How will you pay?"
```

In [71]:

```
bill.split("$")
```

Out[71]:

```
['This total price for the pizza and chips is ', '25. How will you pay?']
```