PYTHON STRINGS

STRINGS

Strings in python are surrounded by either single quotation marks, or double quotation marks.

"HELLO" IS THE SAME 'HALLO'

In [3]:
```
print("hello")
print('hello')
```

```
hello
hello
```

Strings are Arrays

Like many other popular programming languages, strings in Python are arrays of bytes representing unicode characters.

However, Python does not have a character data type, a single character is simply a string with a length of 1.

Square brackets can be used to access elements of the string.

In [4]:
```
#ExAMPLE:
a="valavala umesh"
print(a[2])
```

```
l
```

In [5]:
```
print(a[5])
```

```
a
```

In [6]:
```
print(a[5])
```

```
a
```

looping through a string

Since strings are arrays, we can loop through the characters in a string, with a for loop.

In [7]:
```
#EXAMPLE:
for x in "valavala umesh":
    print(x)
```

```
v
a
l
a
v
a
l
a

u
m
e
s
h
```

STRING LENGTH

TH LEN() FUNCTION IS USED TO LENGTH OF THE STRING

In [9]:
```
name="umesh"
```

In [10]:
```
print(len(name))
```

```
5
```

CHECK STRING

To check if a certain phrase or character is present in a string, we can use the keyword in.

In [13]:
```
txt="my name is umesh"
```

```
print("umesh" in txt)
```

```
True
```

In [14]:
```
txt="my name is umesh"
print("ramesh" in txt)
```

```
False
```

Use it in an if statement:

In [15]:
```
txt="my name is umesh"
if "umesh"in txt:
    print("yes")
```

```
yes
```

In [16]:
```
txt="my name is umesh"
if "umesh"in txt:
    print("yes")
else:
    print("no")
```

```
yes
```

In [17]:
```
txt="my name is umesh"
if "ramesh"in txt:
    print("yes")
else:
    print("no")
```

```
no
```

CHECK IF NOT

To check if a certain phrase or character is NOT present in a string, we can use the keyword not in.

Check if "expensive" is NOT present in the following text:

In [19]:
```
txt = "The best things in life are free!"
print("expensive" not in txt)
```

```
True
```

Use it in an if statement:

In [20]:
```
txt = "The best things in life are free!"
if "expensive" not in txt:
  print("No, 'expensive' is NOT present.")
```

```
No, 'expensive' is NOT present.
```

SLICING IN PYTHON

You can return a range of characters by using the slice syntax.

Specify the start index and the end index, separated by a colon, to return a part of the string.

Get the characters from position 2 to position 5 (not included):

In [21]:
```
#EXAMPLE
a="umesh"
print(a[2:5])
```

```
esh
```

Get the characters from the start to position 5 (not included):

In [22]:
```
a="ramesh"
print(a[:5])
```

```
rames
```

Get the characters from position 2, and all the way to the end:

In [23]:
```
a="suresh"
print(a[2:])
```

```
resh
```

Negative Indexing

Get the characters:

From: "o" in "World!" (position -5)

To, but not included: "d" in "World!" (position -2):

In [24]:
```
b="hello umesh"
print(b[-5:-2])
```

```
ume
```

In [1]:
```
b="hello umesh"
print(b[-7:-4])
```

```
o u
```

PYTHON BUIT IN FUNCTIONS

UPPER CASE: THE UPPER() METHODS RETURNS THE STRING IN UPPER CASE

In [3]:
```
name="umesh"
print(name.upper())
```

```
UMESH
```

LOWER CASE: THE UPPER() METHODS RETURNS THE STRING IN LOWER CASE

In [5]:
```
name="UMESH"
print(name.lower())
```

```
umesh
```

Remove Whitespace

Whitespace is the space before and/or after the actual text, and very often you want to remove this space.

The strip() method removes any whitespace from the beginning or the end:

In [10]:
```
a = " umesh, valavala "
print(a.strip())
```

```
umesh, valavala
```

Replace String:

The replace() method replaces a string with another string:

In [12]:
```
name="umesh valavala"
print(name.replace("umesh","suresh"))
```

```
suresh valavala
```

Split String: The split() method returns a list where the text between the specified separator becomes the list items.

The split() method splits the string into substrings if it finds instances

In [14]:
```
name="umesh suresh ramesh"
print(name.split())
```

```
['umesh', 'suresh', 'ramesh']
```

Python - String Concatenation

String Concatenation To concatenate, or combine, two strings you can use the + operator.

Merge variable a with variable b into variable c:

In [16]:
```
a="umesh"
b="valavala"
c=a+b
print(c)
```

```
umeshvalavala
```

To add a space between them, add a " ":

In [18]:
```
a="umesh"
b="valavala"
c=a+" "+b
print(c)
```

umesh valavala

PYTHON STRING FORMATS

String Format As we learned in the Python Variables chapter, we cannot combine strings and numbers like this:

In [22]:
```
age = 36
txt = "My name is John, I am " + age
print(txt)# IT COMES AN ERROR BECAUSE THE STRING AND NUMERIC VALUE CVANNOT CONCATINATE
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [22], in <cell line: 2>()
      1 age = 36
----> 2 txt = "My name is John, I am " + age
      3 print(txt)

TypeError: can only concatenate str (not "int") to str
```

IT COMES AN ERROR BECAUSE THE STRING AND NUMERIC VALUE CVANNOT CONCATINATE

But we can combine strings and numbers by using the format() method!

The format() method takes the passed arguments, formats them, and places them in the string where the placeholders {} are:

In [25]:
```
name="umesh"
age=20
details="my name is {},and my age is {}"
print(details.format(name,age))
```

```
my name is umesh,and my age is 20
```

In [26]:
```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
print(myorder.format(quantity, itemno, price))
```

```
I want 3 pieces of item 567 for 49.95 dollars.
```

You can use index numbers {0} to be sure the arguments are placed in the correct placeholders:

Python - Escape Characters

Escape Character To insert characters that are illegal in a string, use an escape character.

An escape character is a backslash \ followed by the character you want to insert.

An example of an illegal character is a double quote inside a string that is surrounded by double quotes

In [1]:
```
txt = "We are the so-called \"Vikings\" from the north."
```

In [2]:
```
print(txt)
```

```
We are the so-called "Vikings" from the north.
```

In [3]:
```
txt = "We are the so-called "Vikings" from the north."
#it is in valid
```

```
  Input In [3]
    txt = "We are the so-called "Vikings" from the north."
                                ^
SyntaxError: invalid syntax
```

Escape Characters Other escape characters used in Python:

Code Result \' -Single Quote
\ -Backslash
\n -New Line
\r -Carriage Return
\t -Tab
\b -Backspace
\f -Form Feed
\ooo -Octal value
\xhh -Hex value

Python - String Methods

Python has a set of built-in methods that you can use on strings.

Note: All string methods return new values. They do not change the original string.

In [6]:
```
#capitalize():
#converts the first character into upper caase
a="umesh"
n=(a.capitalize())
print(n)
```

```
Umesh
```

In [8]:
```
#casefold():
#converts string into lower case
name="umesh"
print(name.casefold())
```

```
umesh
```

In [14]:
```
#center()
#returns a centered string.
name="umEsh"
print(name.center(20))
```

```
       umEsh
```

In [25]:
```
#count()
#Returns the number of times a specified value occurs in a string
name="s.suresh"
print(name.count("s"))
```

```
3
```

In [28]:
```
#encode()
#Returns an encoded version of the string
name="rakesh"
print(name.encode())
```

```
b'rakesh'
```

In [30]:
```
#endswith()
#Returns true if the string ends with the specified value
name="suresh"
print(name.endswith("h"))
```

```
True
```

In [34]:
```
#expandtabs()
#Sets the tab size of the string
txt = "H\te\tl\tl\to"

x =  txt.expandtabs(2)

print(x)
```

```
H e l l o
```

In [36]:
```
#find()
#Searches the string for a specified value and returns the position of where it was found
name="umesh"
print(name.find("e"))
```

```
2
```

In [37]:
```
#format()
#Formats specified values in a string
name="laxmi"
print(name.format())
```

```
laxmi
```

Method Description capitalize() Converts the first character to upper case

casefold() Converts string into lower case

center() Returns a centered string

count() Returns the number of times a specified value occurs in a string

encode() Returns an encoded version of the string

endswith() Returns true if the string ends with the specified value

expandtabs() Sets the tab size of the strin

find() Searches the string for a specified value and returns the position of where it was found

format() Formats specified values in a string

format_map() Formats specified values in a string

index() Searches the string for a specified value and returns the position of where it was found

isalnum() Returns True if all characters in the string are alphanumeric

isalpha() Returns True if all characters in the string are in the alphabet

isdecimal() Returns True if all characters in the string are decimals

isdigit() Returns True if all characters in the string are digits

isidentifier() Returns True if the string is an identifier

islower() Returns True if all characters in the string are lower case

isnumeric() Returns True if all characters in the string are numeric

isprintable() Returns True if all characters in the string are printable

isspace() Returns True if all characters in the string are whitespacs

istitle() Returns True if the string follows the rules of a title

isupper() Returns True if all characters in the string are upper case

join() Joins the elements of an iterable to the end of the string

ljust() Returns a left justified version of the string

lower() Converts a string into lower case

lstrip() Returns a left trim version of the string

maketrans() Returns a translation table to be used in translations

partition() Returns a tuple where the string is parted into three parts

replace() Returns a string where a specified value is replaced with a specified value

rfind() Searches the string for a specified value and returns the last position of where it was found

rindex() Searches the string for a specified value and returns the last position of where it was found

rjust() Returns a right justified version of the string

rpartition() Returns a tuple where the string is parted into three parts

rsplit() Splits the string at the specified separator, and returns a list

rstrip() Returns a right trim version of the string

split() Splits the string at the specified separator, and returns a list

splitlines() Splits the string at line breaks and returns a list

startswith() Returns true if the string starts with the specified value

strip() Returns a trimmed version of the string

swapcase() Swaps cases, lower case becomes upper case and vice versa

title() Converts the first character of each word to upper case

translate() Returns a translated string

upper() Converts a string into upper case

zfill() Fills the string with a specified number of 0 values at the beginning

python booleans

Booleans represent one of two values: True or False.

Boolean Values In programming you often need to know if an expression is True or False.

You can evaluate any expression in Python, and get one of two answers, True or False.

When you compare two values, the expression is evaluated and Python returns the Boolean answer:

In [39]:
```
print(10>60)
```

```
False
```

In [40]:
```
print(20<90)
```

```
True
```

In [41]:
```
print(100==100)
```

```
True
```

In [44]:
```
a="umesh"
b="suresh"
if a==b:
    print("the names are  correct")
else:
        print("the given names are not correct")
```

```
the given names are not correct
```

In [ ]:
```

```