In [ ]:

```
Strings

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

'hello' is the same as "hello".

You can display a string literal with the print() function:
```

In [1]:

```
#You can use double or single quotes:

print("Hello")
print('Hello')
```

```
Hello
Hello
```

In [2]:

```
a = "Hello"
print(a)
```

```
Hello
```

In [3]:

```
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tem
]por incididunt
ut labore et dolore magna aliqua."""
print(a)
```

```
Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.
```

In [4]:

```
a = '''Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.'''
print(a)
```

```
Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.
```

In [3]:

```
a = "Hello, World!"
print(a[1])
```

```
e
```

In [3]:

```
for x in "banana":
  print(x)
```

```
b
a
n
a
n
a
```

In [7]:

```
a = "Hello, World!"
print(len(a))
```

```
13
```

In [2]:

```
txt = "The best things in life are free!"
print("free" in txt)
```

```
True
```

In [9]:

```
txt = "The best things in life are free!"
if "free" in txt:
  print("Yes, 'free' is present.")
```

```
Yes, 'free' is present.
```

In [10]:

```
txt = "The best things in life are free!"
print("expensive" not in txt)
```

```
True
```

In [11]:

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

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

In [ ]:

```
Slicing
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 st
```

In [12]:

```
b = "Hello, World!"
print(b[2:5])
```

```
llo
```

In [5]:

```
b = "Hello, World!"
print(b[:5])
```

```
Hello
```

In [14]:

```
b = "Hello, World!"
print(b[2:])
```

```
llo, World!
```

In [15]:

```
b = "Hello, World!"
print(b[-5:-2])
```

```
orl
```

In [16]:

```
a = "Hello, World!"
print(a.upper())
```

```
HELLO, WORLD!
```

In [17]:

```
a = "Hello, World!"
print(a.lower())
```

```
hello, world!
```

In [3]:

```
a = "     Hello, World!     "
print(a.strip())
```

```
Hello, World!
```

In [19]:

```
a = "Hello, World!"
print(a.replace("H", "J"))
```

```
Jello, World!
```

In [4]:

```
a = "He$llo World"
b = a.split("$")
print(b)
```

```
['He', 'llo World']
```

In [13]:

```
a = "Hello"
b = "World"
c = a + b
print(c)
```

```
HelloWorld
```

In [22]:

```
a = "Hello"
b = "World"
c = a + " " + b
print(c)
```

```
Hello World
```

In [23]:

```
age = 36
txt = "My name is John, and I am {}"
print(txt.format(age))
```

```
My name is John, and I am 36
```

In [24]:

```
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.
```

In [6]:

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want to pay {2} dollars for {0} pieces of item {1}."
print(myorder.format(quantity, itemno, price))
```

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

In [26]:

```
txt = "We are the so-called \"Vikings\" from the north."
print(txt)
```

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

In [ ]:

```
Escape Characters
```

In [ ]:

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

In [27]:

```
#A backslash followed by an 'x' and a hex number represents a hex value:
txt = "\x48\x65\x6c\x6c\x6f"
print(txt)
```

```
Hello
```

In [28]:

```
#A backslash followed by three integers will result in a octal value:
txt = "\110\145\154\154\157"
print(txt)
```

```
Hello
```

In [29]:

```
#This example erases one character (backspace):
txt = "Hello \bWorld!"
print(txt)
```

```
HelloWorld!
```

In [30]:

```
txt = "Hello\tWorld!"
print(txt)
```

```
Hello   World!
```

In [31]:

```
txt = "Hello\rWorld!"
print(txt)
```

```
World!
```

In [32]:

```
txt = "Hello\nWorld!"
print(txt)
```

```
Hello
World!
```

In [33]:

```
txt = "This will insert one \\ (backslash)."
print(txt)
```

```
This will insert one \ (backslash).
```

In [34]:

```
txt = 'It\'s alright.'
print(txt)
```

```
It's alright.
```

In [ ]:

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

In [ ]:

```
Python String capitalize() Method
```

In [35]:

```
txt = "hello, and welcome to my world."

x = txt.capitalize()

print (x)
```

```
Hello, and welcome to my world.
```

In [7]:

```
txt = "36 is my age."

x = txt.capitalize()

print (x)
```

```
36 is my age.
```

In [ ]:

```
Python String casefold() Method
```

In [5]:

```
txt = "Hello, And Welcome To My World!"

x = txt.casefold()

print(x)
```

```
hello, and welcome to my world!
```

In [ ]:

```
Python String center() Method
```

In [38]:

```
txt = "banana"

x = txt.center(20)

print(x)
```

```
        banana
```

In [39]:

```
txt = "banana"

x = txt.center(20, "O")

print(x)
```

```
OOOOOOObananaOOOOOOO
```

In [ ]:

```
Python String count() Method
```

In [40]:

```
txt = "I love apples, apple are my favorite fruit"

x = txt.count("apple")

print(x)
```

```
2
```

In [41]:

```
txt = "I love apples, apple are my favorite fruit"

x = txt.count("apple", 10, 24)

print(x)
```

```
1
```

In [ ]:

```
Python String encode() Method
```

In [42]:

```
txt = "My name is Ståle"

x = txt.encode()

print(x)
```

```
b'My name is St\xc3\xa5le'
```

In [43]:

```
txt = "My name is Ståle"

print(txt.encode(encoding="ascii",errors="backslashreplace"))
print(txt.encode(encoding="ascii",errors="ignore"))
print(txt.encode(encoding="ascii",errors="namereplace"))
print(txt.encode(encoding="ascii",errors="replace"))
print(txt.encode(encoding="ascii",errors="xmlcharrefreplace"))
```

```
b'My name is St\\xe5le'
b'My name is Stle'
b'My name is St\\N{LATIN SMALL LETTER A WITH RING ABOVE}le'
b'My name is St?le'
b'My name is St&#229;le'
```

In [ ]:

```
Python String endswith() Method
```

In [44]:

```
txt = "Hello, welcome to my world."

x = txt.endswith(".")

print(x)
```

```
True
```

In [45]:

```
txt = "Hello, welcome to my world."

x = txt.endswith("my world.")

print(x)
```

```
True
```

In [ ]:

```
Python String expandtabs() Method
```

In [46]:

```
txt = "H\te\tl\tl\to"

x =  txt.expandtabs(2)

print(x)
```

```
H e l l o
```

In [47]:

```
txt = "H\te\tl\tl\to"

print(txt)
print(txt.expandtabs())
print(txt.expandtabs(2))
print(txt.expandtabs(4))
print(txt.expandtabs(10))
```

```
H	e	l	l	o
H       e       l       l       o
H e l l o
H   e   l   l   o
H         e         l         l         o
```

In [ ]:

```
Python String find() Method
```

In [48]:

```
txt = "Hello, welcome to my world."

x = txt.find("welcome")

print(x)
```

```
7
```

In [49]:

```
txt = "Hello, welcome to my world."

x = txt.find("e")

print(x)
```

```
1
```

In [50]:

```
txt = "Hello, welcome to my world."

x = txt.find("e", 5, 10)

print(x)
```

```
8
```

In [51]:

```
txt = "Hello, welcome to my world."

print(txt.find("q"))
print(txt.index("q"))
```

```
-1

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-51-233e888b7cf2> in <module>
      2
      3 print(txt.find("q"))
----> 4 print(txt.index("q"))

ValueError: substring not found
```

In [ ]:

```
Python String format() Method
```

In [53]:

```
txt = "For only {price:.2f} dollars!"
print(txt.format(price = 49))
```

```
For only 49.00 dollars!
```

In [ ]:

```
Formatting Types
Inside the placeholders you can add a formatting type to format the result:

:<      Left aligns the result (within the available space)
:>      Right aligns the result (within the available space)
:^      Center aligns the result (within the available space)
:=      Places the sign to the left most position
:+      Use a plus sign to indicate if the result is positive or negative
:-      Use a minus sign for negative values only
:       Use a space to insert an extra space before positive numbers (and a minus sign befo
:,      Use a comma as a thousand separator
:_      Use a underscore as a thousand separator
:b      Binary format
:c      Converts the value into the corresponding unicode character
:d      Decimal format
:e      Scientific format, with a lower case e
:E      Scientific format, with an upper case E
:f      Fix point number format
:F      Fix point number format, in uppercase format (show inf and nan as INF and NAN)
:g      General format
:G      General format (using a upper case E for scientific notations)
:o      Octal format
:x      Hex format, lower case
:X      Hex format, upper case
:n      Number format
:%      Percentage format
```

In [ ]:

```
Python String index() Method
```

In [54]:

```
txt = "Hello, welcome to my world."

x = txt.index("welcome")

print(x)
```

```
7
```

In [55]:

```
txt = "Hello, welcome to my world."

x = txt.index("e")

print(x)
```

```
1
```

In [56]:

```
txt = "Hello, welcome to my world."

x = txt.index("e", 5, 10)

print(x)
```

```
8
```

In [57]:

```
txt = "Hello, welcome to my world."

print(txt.find("q"))
print(txt.index("q"))
```

```
-1
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-57-233e888b7cf2> in <module>
      2
      3 print(txt.find("q"))
----> 4 print(txt.index("q"))

ValueError: substring not found
```

In [ ]:

```
Python String isalnum() Method
```

In [58]:

```
txt = "Company12"

x = txt.isalnum()

print(x)
```

```
True
```

In [59]:

```
txt = "Company 12"

x = txt.isalnum()

print(x)
```

```
False
```

In [ ]:

```
Python String isalpha() Method
```

In [60]:

```
txt = "CompanyX"

x = txt.isalpha()

print(x)
```

```
True
```

In [61]:

```
txt = "Company10"

x = txt.isalpha()

print(x)
```

```
False
```

In [ ]:

```
Python String isdecimal() Method
```

In [62]:

```
txt = "\u0033" #unicode for 3

x = txt.isdecimal()

print(x)
```

```
True
```

In [63]:

```
a = "\u0030" #unicode for 0
b = "\u0047" #unicode for G

print(a.isdecimal())
print(b.isdecimal())
```

```
True
False
```

In [ ]:

```
Python String isdigit() Method
```

In [64]:

```
txt = "50800"

x = txt.isdigit()

print(x)
```

```
True
```

In [65]:

```
a = "\u0030" #unicode for 0
b = "\u00B2" #unicode for ²

print(a.isdigit())
print(b.isdigit())
```

```
True
True
```

In [ ]:

```
Python String isidentifier() Method
```

In [66]:

```
txt = "Demo"

x = txt.isidentifier()

print(x)
```

```
True
```

In [67]:

```
a = "MyFolder"
b = "Demo002"
c = "2bring"
d = "my demo"

print(a.isidentifier())
print(b.isidentifier())
print(c.isidentifier())
print(d.isidentifier())
```

```
True
True
False
False
```

In [ ]:

```
Python String islower() Method
```

In [68]:

```
txt = "hello world!"

x = txt.islower()

print(x)
```

```
True
```

In [69]:

```
a = "Hello world!"
b = "hello 123"
c = "mynameisPeter"

print(a.islower())
print(b.islower())
print(c.islower())
```

```
False
True
False
```

In [ ]:

```
Python String isnumeric() Method
```

In [70]:

```
txt = "565543"

x = txt.isnumeric()

print(x)
```

```
True
```

In [71]:

```
a = "\u0030" #unicode for 0
b = "\u00B2" #unicode for ²
c = "10km2"
d = "-1"
e = "1.5"

print(a.isnumeric())
print(b.isnumeric())
print(c.isnumeric())
print(d.isnumeric())
print(e.isnumeric())
```

```
True
True
False
False
False
```

In [ ]:

```
Python String isprintable() Method
```

In [72]:

```
txt = "Hello! Are you #1?"

x = txt.isprintable()

print(x)
```

```
True
```

In [73]:

```
txt = "Hello!\nAre you #1?"

x = txt.isprintable()

print(x)
```

```
False
```

In [ ]:

```
Python String isspace() Method
```

In [74]:

```
txt = "    "

x = txt.isspace()

print(x)
```

```
True
```

In [75]:

```
txt = "   s   "

x = txt.isspace()

print(x)
```

```
False
```

In [ ]:

```
Python String istitle() Method
```

In [76]:

```
txt = "Hello, And Welcome To My World!"

x = txt.istitle()

print(x)
```

```
True
```

In [77]:

```
a = "HELLO, AND WELCOME TO MY WORLD"
b = "Hello"
c = "22 Names"
d = "This Is %'!?"

print(a.istitle())
print(b.istitle())
print(c.istitle())
print(d.istitle())
```

```
False
True
True
True
```

In [ ]:

```
Python String isupper() Method
```

In [78]:

```
txt = "THIS IS NOW!"

x = txt.isupper()

print(x)
```

```
True
```

In [79]:

```
a = "Hello World!"
b = "hello 123"
c = "MY NAME IS PETER"

print(a.isupper())
print(b.isupper())
print(c.isupper())
```

```
False
False
True
```

In [ ]:

```
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
rindex()    Searches the string for a specified value and returns the last position of wher
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
```