Strings:

```
In Python, a string is a sequence of characters enclosed within either single quotes ('), double quotes ("), or triple quotes (''' or """). Strings are immutable, meaning once they are created, their contents cannot be changed.
```

In [1]:
```
print("Hello")
print('Hello')
```

```
Hello
Hello
```

```
Assigning a string to a variable is done with the variable name followed by an equal sign and the string:
```

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

```
Hello
```

Multiline Strings

```
You can assign a multiline string to a variable by using three quotes:
```

In [3]:
```
s="""Computer refers to a programmable electronic device designed to perform mathematical cal
```

In [5]:
```
s
```

Out[5]:
```
'Computer refers to a programmable electronic device designed to perform mathematical calculatio
ns, process data, and execute sequences of instructions or programs. '
```

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 [1]:
```
a = "Hello, World!"
print(a[1])
print(a[4])
```

```
e
o
```

Slicing Strings

```
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.
```

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

```
llo
```

```
Slice From the Start:
```

```
    By leaving out the start index, the range will start at the first character:
Example:
    Get the characters from the start to position 5 (not included):
```

In [3]:
```
b = "Hello, World!"
print(b[:5])
```

```
Hello
```

```
Slice To the End:
    By leaving out the end index, the range will go to the end:
Example:
    Get the characters from position 2, and all the way to the end:
```

In [4]:
```
b = "Hello, World!"
print(b[2:])
```

```
llo, World!
```

```
Negative Indexing:
    Use negative indexes to start the slice from the end of the string:
Example:
Get the characters:
    From: "o" in "World!" (position -5), but not included: "d" in "World!" (position -2):
```

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

```
orl
```

Strings Methods:

In [1]:
```
#Converts the first character to upper case
txt = "hello, and welcome to my world."
txt.capitalize()
```

Out[1]: 'Hello, and welcome to my world.'

In [63]:
```
#Converts string into lower case
txt = "Hello, And Welcome To My World!"

txt.casefold()

print(x)

```

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

In [5]:
```
#Returns a centered string
txt = "banana"
txt.center(20, '#')
print(x)
```

```
#######banana#######
```

In [6]:
```
txt = "banana"
x = txt.center(20,"*")
print(x)
```

```
*******banana*******
```

In [8]:
```
#Returns the number of times a specified value occurs in a string
```

In [64]:
```
txt = "I love apples, apple are my favorite fruit"
txt.count("apple")
```

Out[64]: 2

In [65]:
```
#Returns an encoded version of the string
txt = "My name is Zubåir"
x = txt.encode()
print(x)
```

```
b'My name is Zub\xc3\xa5ir'
```

In [66]:
```
x="zubair"
x.encode()
```

Out[66]: b'zubair'

In [11]:
```
txt = "My name is Ståle AAAAAAAAAAAAAA"
x = txt.encode()
print(x)
```

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

In [67]:
```
#Returns true if the string ends with the specified value
txt = "Hello, welcome to my world."
x = txt.endswith("d")
print(x)
```

```
False
```

In [69]:
```
#Sets the tab size of the string
txt = "H\te\tl\tl\to"
x =  txt.expandtabs(12)
print(x)
```

```
H           e           l           l           o
```

In [14]:
```
txt = "Hello \t to \t all\t my\t students"
x =  txt.expandtabs(12)
print(x)
```

```
Hello        to          all         my          students
```

In [71]:
```
#Searches the string for a specified value and returns the position of where it was found
txt = "Hello, welcome to my world."
x = txt.find("l")
print(x)
```

```
2
```

In [73]:
```
txt = "Hello, welcome to my world."
x = txt.index("l")
print(x)
```

```
2
```

In [8]:
```
txt = "Hello, welcome to my world."

x = txt.index("apple")

print(x)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [8], in <cell line: 3>()
      1 txt = "Hello, welcome to my world."
----> 3 x = txt.index("apple")
      5 print(x)

ValueError: substring not found
```

In [9]:
```
txt = "Hello, welcome to my world."

x = txt.find("apple")

print(x)
```

```
-1
```

```
What is difference between find () and index () in the string?
The index() method is similar to the find() method for strings. The only difference is that
find() method returns -1 if the substring is not found, whereas index() throws an exception.
```

In [19]:
```
txt = "Hello, welcome to my world."
x = txt.rfind("world")
print(x)
```

```
21
```

In [20]:
```
#Formats specified values in a string
txt = "For only {price:.2f} dollars!"
print(txt.format(price = 49))
```

```
For only 49.00 dollars!
```

In [10]:
```
#Returns True if all characters in the string are alphanumeric
txt = "Company123"
x = txt.isalnum()
print(x)
```

```
True
```

In [ ]:
```
#Returns True if all characters in the string are in the alphabet
txt = "Company"
x = txt.isalpha()
print(x)
```

In [11]:
```
#Returns True if all characters in the string are ascii characters
txt = "Company123*/"
x = txt.isascii()
print(x)
```

```
True
```

In [27]:
```
#Returns True if all characters in the string are decimals
txt = "12349"
x = txt.isdecimal()
print(x)
```

```
True
```

In [30]:
```
#Returns True if all characters in the string are digits
txt = "508010"
x = txt.isdigit()
print(x)
```

```
True
```

In [33]:
```
#Returns True if the string is an identifier
txt = "Demo 12"
x = txt.isidentifier()
print(x)
```

```
False
```

In [ ]:
```
#Returns True if all characters in the string are lower case
txt = "hello world!"
x = txt.islower()
print(x)
```

In [34]:
```
#Returns True if all characters in the string are numeric
txt = "565543"
x = txt.isnumeric()
print(x)
```

```
True
```

In [3]:
```
#Returns True if all characters in the string are printable
txt = "Hello! Are you 1"
x = txt.isprintable()
print(x)
```

```
True
```

In [2]:
```
# Returns True if all characters in the string are whitespaces
txt = ""
x = txt.isspace()
print(x)
```

```
False
```

In [6]:
```
#Returns True if the string follows the rules of a title
txt = "Hello, And Welcome To My World!"
x = txt.istitle()
print(x)
```

```
True
```

In [9]:
```
#Returns True if all characters in the string are upper case
txt = "THIS IS NOW!"
x = txt.isupper()
print(x)
```

```
True
```

In [12]:
```
#Splits the string at the specified separator, and returns a list
txt='OVER THE NEXT THREE DECADES, THE GROUP DIVERSIFIED INTO AREAS INCLUDING FOOD PROCESSING,
x=txt.split()
print(x)
```

```
['OVER', 'THE', 'NEXT', 'THREE', 'DECADES,', 'THE', 'GROUP', 'DIVERSIFIED', 'INTO', 'AREAS', 'IN
CLUDING', 'FOOD', 'PROCESSING,', 'TEXTILES,', 'INSURANCE,', 'SECURITIES,', 'AND', 'RETAIL.']
```

In [13]:
```
#Joins the elements of an iterable to the end of the string
T = ("John", "Peter", "Vicky")
x = "i love U    ".join(T)
print(x)
```

```
Johni love U    Peteri love U    Vicky
```

In [39]:
```
#Returns a left justified version of the string
txt = "banana"
x = txt.ljust(20)
print(x, "is my favorite fruit.")
```

```
banana               is my favorite fruit.
```

In [43]:
```
#removes white spsces before and sfter the string
a="      zubair        "
a.strip()
print(a)
```

```
      zubair
```

In [13]:
```
#Returns a left trim version of the string
a.lstrip()
```

Out[13]:
```
'Hello, World!'
```

In [45]:
```
#Returns a right trim version of the string
a.rstrip()
```

Out[45]:
```
'      zubair'
```

In [14]:
```
#Returns a translated string
txt = "Hello Sam!"
t = str.maketrans("S", "P")#maketrans() Returns a translation table to be used in translations
print(txt.translate(t))
```

```
Hello Pam!
```

In [ ]:
```
#Returns a tuple where the string is parted into three parts
txt = "I could eat bananas all day"
x = txt.partition("bananas")
print(x)
```

In [47]:
```
#Returns a tuple where the string is parted into three parts
txt = "I could eat bananas all day, bananas are my favorite fruit"
x = txt.rpartition("bananas")
print(x)
```

```
('I could eat bananas all day, ', 'bananas', ' are my favorite fruit')
```

In [48]:
```
txt = "I could eat bananas all day, bananas are my favorite fruit"
x = txt.rpartition("I")
print(x)
```

```
('', 'I', ' could eat bananas all day, bananas are my favorite fruit')
```

In [49]:
```
#Returns a string where a specified value is replaced with a specified value
txt = "I like bananas"
x = txt.replace("bananas", "apples")
print(x)
```

```
I like apples
```

In [17]:
```
#Fills the string with a specified number of 0 values at the beginning
a="hello"
a.zfill(10)
```

Out[17]:
```
'00000hello'
```

In [16]:
```
c="120.000"
c.zfill(20)
```

Out[16]:
```
'0000000000000120.000'
```