String in python

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

In [1]:
```
#Strings in python are surrounded by either single quotation marks, or double quotation marks.

#'Muhammad Nasir' is the same as "Muhammad Nasir".
print("Muhammad Nasir")
#
print('Muhammad Nasir')
```

```
Muhammad Nasir
Muhammad Nasir
```

Multiline Strings

In [2]:
```
#You can assign a multiline string to a variable by using three quotes single / double:
firstStr = """This
is multi line string quoted in double quotes"""
secondStr = '''This
is multi line string quoted in single quotes'''
print(firstStr)
print(secondStr)
```

```
This
is multi line string quoted in double quotes
This
is multi line string quoted in single quotes
```

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 [3]:
```
name = "Muhammad Nasir"
print(name[0])
#we can also use for in loop to iterate through each character in string
for char in name:
    print(char)
```

```
M
M
u
h
a
m
m
a
d

N
a
s
i
r
```

In [4]:
```
#since string are stored in array we can calculate their length using len() function as,
print(len(name))
```

```
14
```

In [5]:
```
#check if string / part of string exist in given string
name = "Islamia University Of Bahawalpur"
print("Islamia" in name)
```

```
True
```

In [6]:
```
#check if string / part of string exist in given string or not
name = "Islamia University Of Bahawalpur"
print("islamia" in name) #it will return false, because in variable name Islamia exist but not islamia
```

```
False
```

In [7]:
```
#check if string / part of string does not exist in given string
print("Islamia" not in name)
```

```
False
```

In [8]:
```
#use of in with if
if "Islamia" in name:
    print("Islamia exsists")
else:
    print("Islamia does not exists in given text")
```

```
Islamia exsists
```

String Slicing

Slicing String Slicing a string in Python means extracting a substring from the original string by specifying the start and end index. keep in mind the end index is not included in returned substring for example 0:5 mean return substring from index 0 to 5(4 index will be included but not 5)

In [9]:
```
#string containing digit from 0-5
x = "012345"
#printing strings from 0 to 5 where included index are 0,1,2,3,4 and 5 will
#not be included
print(x[0:5])
```

```
01234
```

In [10]:
```
#Slicing from start
#we can slice a string from start to givenNumberofIndex-1 as,
myString = "0123456789"
print(myString[:9])
#above line will print string which include 012345678 but not 9
#one more thing here is that when we don't give start index
#it consider it to start from starting index which always will be 0 index
```

```
012345678
```

In [11]:
```
#Slicing from end
#we can slice a string till the end of string from specific start point
myString = "0123456789"
print(myString[5:])
#above line will print string which include 56789
#one more thing here is that when we don't give end index
#it consider it to go till end of string when end index in range is not provided
```

```
56789
```

Negative Slicing

```
Negative slicing in strings refers to extracting a portion of a string by using negative indices.
It starts counting from the end of the string, with -1 being the index of the last character, -2
being the second to last, and so on. This allows you to select a range of characters from the end
of the string by specifying the start and end index of the slice using negative numbers.
```

In [12]:
```
#let suppoe we want to print last 5 character we can do as,
print(myString[-5:])
#as we don't give any end index it will go till end of string and start
#from 5th element from last as -1 index refer to first element so -5 refers to 5th element from end
```

```
56789
```

In [13]:
```
#slice a range of sub-string using negative slicing
print(myString[-5:-3])
#it will print 5th and 4th element from end as the ending index which here is -3 is not included
```

```
56
```

In [14]:
```
#negative slicing with another example
myString2 = "Islamia University of Bahawalpur"
#now we don't know from where the index of B of Bahawalpur start so simply we
#can calculate its character no as they are 10 so we print a range as [-10:] and it will print Bahawalpur
print(myString2[-10:])
```

```
Bahawalpur
```

String helping methods

```
Python has some built in method for string data type which we can use to perform
different operations on string which are,
```

In [15]:
```
#upper() a method which will convert all lower case character in string into uppercase characters as,
myVal = "web design and frameworks"
myValyUpper = myVal.upper()
#lets print both
print(myVal)
print(myValyUpper)
```

```
web design and frameworks
WEB DESIGN AND FRAMEWORKS
```

In [16]:
```
#lower() a method which converts lowercase
myValLower = myVal.lower()
print(myValLower)
```

```
web design and frameworks
```

In [17]:
```
#remove leading (starting) and trailing (ending) spaces using strip() method
myStr = "       H A B I B I      "
#in above string we have some space at start, in between of characters and at end
#the strip() method will remove all spaces which are at start and end of string but it will not rmeove the space
#between string so let see,
myStrStrip = myStr.strip()
print(myStrStrip)
```

```
H A B I B I
```

as shown above the strip() method removes all the leading(start) and trailing (end) spaces from text

Replacing a substring with another substring

```
we have a replace() method using which we can replace all occurence of on substring
with another substring. let see an example,
```

In [18]:
```
originalString = "I am an employee of Microsoft. I love working in Microsoft. Microsoft is my Love"
#lets replace the word MicroSoft with Google
modifiedString = originalString.replace("Microsoft","Google")
#lets print the modified string
print(modifiedString)
```

```
I am an employee of Google. I love working in Google. Google is my Love
```

```
as shown above the substring Microsoft replaced with Google, this is how it workd,
signature of method are,
replace(oldSubString,newSubString) and it will return modified string
```

Breaking sting in list of tokens

we can break a string in different tokens by providing a seperator to method split() and it will return a list of substring. let see an example,

In [19]:
```
userStatment = "4 + 5 * 99 * 47 + 998"
#now we want to create token of above statment as the some of compilers do
listOfTokens = userStatment.split(" ")
#lets print the list returns by split method
print(listOfTokens)
```

```
['4', '+', '5', '*', '99', '*', '47', '+', '998']
```

as shown above the split method converts the given strings into list of tokens of string.

```
Now we can combine a list of strings or any iteratable in a single string. the variable listOfTokens
refers to a list of strings so we can combine them into single string using join methods as,
```

In [20]:
```
#an empty string
myString = "";
myString = myString.join(listOfTokens)
print("type of listOfTokens {}".format(type(listOfTokens)))
print("type of listOfTokens {}".format(type(myString)))
```

```
type of listOfTokens <class 'list'>
type of listOfTokens <class 'str'>
```

Formatting a String

we can format a string using format() method which will place the values passed as arguments in placeholders. placeholders in strings are defined using curly braces {}.

In [21]:
```
myInfo = "Hi, My Name is {}. I am {} years old.My Registration Id is {}".format("Muhammad Nasir",23,"SP20M2BB030")
print(myInfo)
```

```
Hi, My Name is Muhammad Nasir. I am 23 years old.My Registration Id is SP20M2BB030
```

Finding a specific subString

there are two methods which we can use to find the lowest (the first occurence) of given substring from a string these methods are, str.find(subString) str.index(subString) the key difference in both above methods is that the first find() method will not give error instead it will return -1 if the substring is not found in given string and the second method index() will show an error in case substring is not found

In [24]:
```
myString = "Pakistan"
print(myString.find("k"))
myString = "Pakistan"
print(myString.index("k"))
```

```
2
2
```

In [25]:
```
#let see with a substring which is not in myString
print(myString.find("z"))
print(myString.index("z"))
```

```
-1

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[25], line 3
      1 #let see with a substring which is not in myString
      2 print(myString.find("z"))
----> 3 print(myString.index("z"))

ValueError: substring not found
```

In [32]:
```
#we can also check if a string contains non alphabetic characters like space, any kind of special symbol
#or digit, if the string contains only uppercase / lowercase letter without any space, special symbol or
#charcter this method will return true otherwise false in any case
myString = "Habibi"
```

```
myString2 = "Habibi with some space :)"
myString3 = "Habibi with 4 Habibian :)"
print(myString.isalpha())
print(myString2.isalpha())
print(myString3.isalpha())
```

```
True
False
False
```

In [ ]: