STRINGS

Strings are Immutable, Indexed

In [1]:
```
b = "Hello, World!"
print(b[2:5])
print(b[:5])
print(b[2:])
print(b[-5:-2]) # hence, [relative left : relative right] in any case.
```

```
llo
Hello
llo, World!
orl
```

In [2]:
```
x="sharan"
y="Jaiswal"
print("My name is "+x)
print(x*3)
print(x+y)
```

```
My name is sharan
sharansharansharan
sharanJaiswal
```

In [3]:

```
# str.method(parameter) or string.method(parameter)

txt1 = "36 is my age."
txt2 = "hello, and welcome to my world."
x = txt1.capitalize()
y = txt2.capitalize()   # if first character in the str is alphabet, then only it'll work. Otherwise, it'll return t
print (x)
print (y)

txt = "Welcome to my world hello b2b2b2 and 3g3g3g"
x = txt.title() # converts everyfirst alphabet, after the non-alphabet character, to opposite case.
print(x)

mystr = 'außen'
print(mystr.casefold())
mystr = 'außen'
print(mystr.lower())    # the German lowercase letter 'ß' is equivalent to 'ss'. Since it is already lowercase, the

txt = "Hello My3433 --=_(*%% hdf o) Name Is PETERaußen"
x = txt.swapcase()
print(x)

txt = "banana"
x = txt.center(20)  # string.center(length, [character]). Whichever among len(str) or length will be max, that lengt
print(x)

txt = "I love apples, apple are jdfapplekdslk my favorite fruit."
x = txt.count("apple")  # string.count(value, [start], [end]) ..... default value is 0
print(x)

x = txt.endswith(".")
print(x)    # string.endswith(value, [start], [end]) ....default is False
txt1 = "hello"
x = txt1.startswith("hello", 0, 3)
print(x)    # the whole match should be present within the limits

txt = "Hello, welcome to my world."
print(txt.find("q"))    # finds first|last occurence only. Default, if not found, is -1. string.[r]find(value, [star
# print(txt.[r]index("q")) # raises exception if value not found. Hence dont use. Similar as find(), but without ecx

# True/False --> string.isalnum() isalpha() isdigit() isidentifier()(a-z0-9_) islower|upper()(only checks for alphab
```

```
print(a.zfill(10))
print(b.zfill(10))
print(c.zfill(10))
```

```
36 is my age.
Hello, and welcome to my world.
Welcome To My World Hello B2B2B2 And 3G3G3G
aussen
außen
hELLO mY3433 --=_(*%% HDF O) nAME iS peterAUSSEN
       banana
3
True
False
-1
I like apples
00000hello
welcome to the jungle
000010.000
```

In [4]:

```
# string_separator.join(iterable) Takes all items in an iterable and joins them into one string. Separator must be s

myTuple = ("John", "Peter", "Vicky")
x = "#".join(myTuple)
print(x)

myDict = {"name": "John", "country": "Norway"}
mySeparator = "TEST"
x = mySeparator.join(myDict)
print(x)
```

```
John#Peter#Vicky
nameTESTcountry
```

In [5]:
```
txt = "I could eat bananas all day"
x = txt.partition("bananas")
print(x)
x = txt.partition("apple")
print(x)

"""
Search for the first match of the word "bananas", and return a tuple with three elements:

1 - everything before the "match", if match not found then whole input str
2 - the "match", if match not found then empty str
3 - everything after the "match", if match not found then empty str
==================================
if 'rpartition' is used, then the first match from the last is looked for. Rest all will be same.
"""
```

```
('I could eat ', 'bananas', ' all day')
('I could eat bananas all day', '', '')
```

Out[5]:
```
'\nSearch for the first match of the word "bananas", and return a tuple with three elements:\n\n1 - everything befor
e the "match", if match not found then whole input str\n2 - the "match", if match not found then empty str\n3 - ever
ything after the "match", if match not found then empty str\n==================================\nif \'rpartition\'
is used, then the first match from the last is looked for. Rest all will be same.\n'
```

In [6]:
```
txt = "banana"
x = txt.ljust(20)   # string.l|rjust(length, [character]) default value of'character' us " "(space). Whichever among
print(x, "is my favorite fruit.")


txt = ",,,,,ssaaww.....banana"
x = txt.lstrip(",.asw") # string.lstrip([characters]) space is the default leading CHARACTER-SET to remove. there is
print(x)
```

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

In [7]:
```
txt = "#welcom#e    t#o# the jun#gle#"
x = txt.split()
print(x)
x = txt.split("#", 2)
print(x)
x = txt.split("#")
print(x)
x = txt.split(" ")   # string.split([separator, [maxsplit]]) default values are <whitespaces> and <-1>(which means a
print(x)
txt = "#"*6
x=txt.split("#")
print(x)
```

```
['#welcom#e', 't#o#', 'the', 'jun#gle#']
['', 'welcom', 'e    t#o# the jun#gle#']
['', 'welcom', 'e    t', 'o', ' the jun', 'gle', '']
['#welcom#e', '', '', '', 't#o#', 'the', 'jun#gle#']
['', '', '', '', '', '', '']
```

In [8]:
```
txt = "Thank you for the music\nWelcome to the jungle"
x = txt.splitlines()
print(x)
x = txt.splitlines(True)
print(x)
print(x[0])
print(x[1])
```

```
['Thank you for the music', 'Welcome to the jungle']
['Thank you for the music\n', 'Welcome to the jungle']
Thank you for the music

Welcome to the jungle
```

In [9]:

```
'''
string.translate(table) # Required. Either a dictionary, or a mapping table describing how to perform the replace
The translate() method returns a string where some specified characters are replaced with the character described in
Use the maketrans() method to create a mapping table.
If a character is not specified in the dictionary/table, the character will not be replaced.
If you use a dictionary, you must use ascii codes instead of characters.
'''

#use a dictionary with ascii codes to replace 83 (S) with 80 (P):
mydict = {83:  80}
txt = "Hello Sam!"
print(txt.translate(mydict))

txt = "Hello Sam!"
mytable = txt.maketrans("S", "P")
print(txt.translate(mytable))

txt = "Hi Sam!"
x = "mSa"
y = "eJo"
mytable = txt.maketrans(x, y)
print(txt.translate(mytable))

txt = "Good night Sam!"
x = "mSa"
y = "eJo"
z = "odnght"
mytable = txt.maketrans(x, y, z)
print(txt.translate(mytable))

txt = "Good night Sam!"
mydict = {109: 101, 83: 74, 97: 111, 111: None, 100: None, 110: None, 103: None, 104: None, 116: None}
print(txt.translate(mydict))

'''
-----======--------=========--------==========---------==========----------==========-------==========-----
'''

# string.maketrans(x, y, z)
# x---->Required. If only one parameter is specified, this has to be a dictionary describing how to perform the repl
# y---->        Optional. A string with the same length as parameter x. Each character in the first parameter will b
# z---->        Optional. A string describing which characters to remove from the original string.
```

```
# The maketrans() method returns a mapping table that can be used with the translate() method to replace specified c
txt = "Hi Sam!"
x = "mSa"
y = "eJo"
mytable = txt.maketrans(x, y)
print(txt.translate(mytable))

txt = "Good night Sam!"
x = "mSa"
y = "eJo"
z = "odnght"
mytable = txt.maketrans(x, y, z)
print(txt.translate(mytable))


txt = "Good night Sam!"
x = "mSa"
y = "eJo"
z = "odnght"
print(txt.maketrans(x, y, z))
```

```
Hello Pam!
Hello Pam!
Hi Joe!
G i Joe!
G i Joe!
Hello Pam!
Hi Joe!
G i Joe!
{109: 101, 83: 74, 97: 111, 111: None, 100: None, 110: None, 103: None, 104: None, 116: None}
```

In [10]:

```
# string.format(value1, value2...)
# value1, value2,.... ----> Required. One or more values that should be formatted and inserted in the string. The va

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

# The placeholders can be identified using named indexes {price}, numbered indexes {0}, or even empty placeholders {
txt1 = "My name is {fname}, I'm {age}".format(fname = "John", age = 36)
txt2 = "My name is {0}, I'm {1}".format("John",36)
txt3 = "My name is {}, I'm {}".format("John",36)
print(txt1, txt2, txt3)

# insert the number 8 to set the available space for the value to 8 characters.
txt = "We have {:<8} chickens."
print(txt.format(49))   # Use "<" to left-align the value:
txt = "We have {:>8} chickens."
print(txt.format(49))   # Use "<" to right-align the value:
txt = "We have {:^8} chickens."
print(txt.format(49))   # Use "^" to center-align the value:
txt = "The temperature is {:=8} degrees celsius."
print(txt.format(-5))   # Use "=" to place the plus/minus sign at the left most position:
print("{:*^7}".format("cat"))
print("{:*^6}".format("cat"))
txt = "The temperature is between {:+} and {:+} degrees celsius."
print(txt.format(-3, 7))    # Use "+" to always indicate if the number is positive or negative:
txt = "The temperature is between {:-} and {:-} degrees celsius."
print(txt.format(-3, 7))    # Use "-" to always indicate if the number is negative (positive numbers are displayed w
txt = "The temperature is between {: } and {: } degrees celsius."
print(txt.format(-3, 7))    # Use " " (a space) to insert a space before positive numbers and a minus sign before ne
txt = "The universe is {:,} years old."
print(txt.format(13800000000))  # Use "," to add a comma as a thousand separator:
txt = "The universe is {:_} years old."
print(txt.format(13800000000))  # Use "_" to add a underscore character as a thousand separator:
txt = "The binary version of {0} is {0:b}"
print(txt.format(5))    # Use "b" to convert the number into binary format:
txt = "We have {:d} chickens."
print(txt.format(0b101))    # Use "d" to convert a number, in this case a binary number, into decimal number format:
txt = "We have {:e} chickens."
print(txt.format(5))    # Use "e" to convert a number into scientific number format (with a lower-case e):
txt = "We have {:E} chickens."
print(txt.format(5))    # Use "E" to convert a number into scientific number format (with an upper-case E):
txt = "The price is {:.2f} dollars."
```

```
x = float('inf')
txt = "The price is {:F} dollars."
print(txt.format(x))    # Use "F" to convert a number into a fixed point number, but display inf and nan as INF and
txt = "The price is {:f} dollars."
print(txt.format(x))    # same example, but with a lower case f:
txt = "The octal version of {0} is {0:o}"
print(txt.format(10))   # Use "o" to convert the number into octal format:
txt = "The Hexadecimal version of {0} is {0:x}"
print(txt.format(255))  # Use "x" to convert the number into Hex format:
txt = "The Hexadecimal version of {0} is {0:X}"
print(txt.format(255))  # Use "X" to convert the number into upper-case Hex format:
txt = "You scored {:%}"
print(txt.format(0.25)) # Use "%" to convert the number into a percentage format:
txt = "You scored {:.0%}"
print(txt.format(0.25)) # Or, without any decimals:
# :g for general format.
# :G for general format using upper case E
# :n for Number format. Same as 'd'. Except it uses current locale setting for number separator
# :% for Percentage. Multiples by 100 and puts % at the end.


# https://www.programiz.com/python-programming/methods/string/format
# https://www.programiz.com/python-programming/methods/string/format_map
```

```
For only 49.00 dollars!
My name is John, I'm 36 My name is John, I'm 36 My name is John, I'm 36
We have 49       chickens.
We have       49 chickens.
We have    49    chickens.
The temperature is -      5 degrees celsius.
**cat**
*cat**
The temperature is between -3 and +7 degrees celsius.
The temperature is between -3 and 7 degrees celsius.
The temperature is between -3 and  7 degrees celsius.
The universe is 13,800,000,000 years old.
```

```
The universe is 13_800_000_000 years old.
The binary version of 5 is 101
We have 5 chickens.
We have 5.000000e+00 chickens.
We have 5.000000E+00 chickens.
The price is 45.00 dollars.
Hello Adam, your balance is   230.235
Hello Adam, your balance is   230.235
The price is 45.000000 dollars.
The price is INF dollars.
The price is inf dollars.
The octal version of 10 is 12
The Hexadecimal version of 255 is ff
The Hexadecimal version of 255 is FF
You scored 25.000000%
You scored 25%
```

In [11]:

```
# Reverse a string
txt = "Hello World"[::-1]
print(txt)
```

```
dlroW olleH
```