```
# -*- coding: utf-8 -*-
"""
Topic:Python Strings
Summary By: Varpe K. M.
Reference : https://www.geeksforgeeks.org/
Reference: https://www.w3schools.com/python/python_strings.asp

String Literals

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

'hello' is the same as "hello".

You can display a string literal with the print() function:
"""
print("Hello")
print('Hello')
"""
Assign String to a Variable

Assigning a string to a variable is done with the variable name
followed by an equal sign and the string:
Example
"""
a = "Hello"
print(a)
"""
Multiline Strings

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

You can use three double quotes:
"""
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
print(a)

"""
Or three single quotes:
Example
"""
a = '''Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.'''
print(a)
"""
Note: in the result, the line breaks are inserted
at the same position as in the code.
"""
"""
Strings are Arrays

Like many other popular programming languages, strings in Python are arrays of bytes representir

However, Python does not have a character data type, a single character is simply a string with
```

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

Get the character at position 1 (remember that the first character has the position 0):
"""
a = "Hello, World!"
print(a[1])

"""
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 string.
Example

Get the characters from position 2 to position 5 (not included):
"""
b = "Hello, World!"
print(b[::-1])
print(b[1::-1])
print(b[-3::-1])
print(b[2:5])
print(b[2:])
print(b[:5])
print(b[::1])
print(b[::2])
print(b[::3])
print(b[3:7:1])
print(b[::-2])


"""
Negative Indexing
Use negative indexes to start the slice from the end of the string:
Example
Get the characters from position 5 to position 1,
starting the count from the end of the string:
"""
b = "Hello, World!"
print(b[-5:-2])
print(b[-6:-1])
print(b[::-1])
print(b[::-2])
print(b[len(b)::-1])

print(b[::]) #Hello, World!
print(b[::1]) #Hello, World!
print(b[::2]) #Hlo ol!
print(b[::3])
print(b[::-1])
print(b[:-2])
print(b[-2:])
"""
String Length
To get the length of a string, use the len() function.
Example
The len() function returns the length of a string:
"""
a = "Hello, World!"
```

```
print(len(a))
"""
String Methods

Python has a set of built-in methods that you can use on strings.
Example
The strip() method removes any whitespace from the beginning or the end:
"""
a = " Hello, World! "
print(a.strip()) # returns "Hello, World!"
"""
The lower() method returns the string in lower case:
"""
a = "Hello, World!"
print(a.lower())
"""
The upper() method returns the string in upper case:
"""
a = "Hello, World!"
print(a.upper())
"""
The replace() method replaces a string with another string:
"""
a = "Hello, World!"
print(a.replace("H", "J"))
"""
The split() method splits the string into substrings
if it finds instances of the separator:
"""
a = "Hello,Wor,ld!"
print(a.split(",")) # returns ['Hello', ' World!']
"""
Check String
To check if a certain phrase or character is present in a string,
we can use the keywords in or not in.
Example
Check if the phrase "ain" is present in the following text:
"""
txt = "The rain in Spain stays mainly in the plain"
x = "ain" in txt
print(x)
"""
Check if the phrase "ain" is NOT present in the following text:
txt = "The rain in Spain stays mainly in the plain"
"""
x = "ain" not in txt
print(x)
"""
String Concatenation

To concatenate, or combine, two strings you can use the + operator.
Example
Merge variable a with variable b into variable c:
"""
a = "Hello"
b = "World"
c = a + b
print(c)

"""
Example
```

```
To add a space between them, add a " ":
"""
a = "Hello"
b = "World"
c = a + " " + b
print(c)
"""
String Format
As we learned in the Python Variables chapter,
we cannot combine strings and numbers like this:
Example
"""
age = 36
y=str(age)
txt = "My name is John, I am " + y
print(txt)
"""
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:
Example
Use the format() method to insert numbers into strings:
"""
age = 36
txt = "My name is John, and I am {}"
print(txt.format(age))
"""
The format() method takes unlimited number of arguments, and
 are placed into the respective placeholders:
Example
"""
quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
print(myorder.format(quantity, itemno, price))
"""
You can use index numbers {0} to be sure the arguments are placed
 in the correct placeholders:
Example
"""
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))
"""
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:
Example

You will get an error if you use double quotes
inside a string that is
```

```
surrounded by double quotes:

txt = "We are the so-called "Vitians" from the VIT."
To fix this problem, use the escape character \":
Example
The escape character allows you to use double quotes
when you normally would not be allowed:
"""
txt = "We are the so-called \"Vitians\" from the VIT."
print(txt)
"""
\r Carriage Return
"""
txt = "Hello\rWorld!"
print(txt)
print("You have NOT used Carriage Return")
print("You have used Carriage \r Return")
print("You have used Carriage\rReturn")

"""
\b Backspace
"""
txt = "Hello\bWorld!"
print(txt)
"""
\f page break
"""
txt = "\f Hello World!"
print(txt)
print("You have used Form Feed")
txt = "\f Hello on Page 2!"
print(txt)
print("On Page 2")
print("You have used Form Feed")
"""
\ooo   Octal value
"""
#A backslash followed by three integers will
#result in a octal value:
txt = "\110\145\154\154\157"
print(txt)
"""
\xhh   Hex value
"""
#A backslash followed by an 'x' and
#a hex number represents a hex value:
txt = "\x48\x65\x6c\x6c\x6f"
print(txt)

str="sAmPLE sTrING"
"""
capitalize() Converts the first character to upper case
"""
print(str.capitalize())
"""
casefold()   Converts string into lower case
"""
print(str.casefold())
"""
center()  Returns a centered string
"""
```

```
print(str.center(50))
"""
count()   Returns the number of times a specified
value occurs in a string
"""
print(str.count("s"))
"""
encode() Returns an encoded version of the
string
"""
print(str.encode())
"""
endswith()   Returns true if the string ends with the
specified value
"""
# Python code shows the working of
# .endswith() function

text = "geeks for geeks."

# start parameter: 10
result = text.endswith('geeks.', 10)
print(result)

# Both start and end is provided
# start: 10, end: 15
# Returns False
result = text.endswith('geeks', 10, 15)
print(result)

# returns True
result = text.endswith('geeks', 10, 14)
print(result)
"""
expandtabs() Sets the tab size of
the string
"""
print(str.expandtabs())
"""
find() Searches the string for a specified value and
 returns the position of where it was found
"""

word = 'VIT for Scholars'

# returns first occurrence of Substring
result = word.find('VIT')
print ("Substring 'VIT' found at index:", result )

result = word.find('for')
print ("Substring 'for ' found at index:", result )

# How to use find()
if (word.find('power') != -1):
    print ("Contains given substring ")
else:
    print ("Doesn't contains given substring")

if (word.find('Scholars') != -1):
    print ("Contains given substring ")
else:
```

```
    print ("Doesn't contains given substring")

"""
format()  Formats specified values in a string
"""

# Python3 program to demonstarte
# the str.format() method

# using format option in a simple string
print ("{}, A computer science portal for geeks.".format("GeeksforGeeks"))

# using format option for a
# value stored in a variable
str = "This article is written in {}"
print (str.format("Python"))

# formatting a string using a numeric constant
print ("Hello, I am {} years old !".format(18))
################################################

# Python program demonstrating Index error

# Number of placeholders are four but
# there are only three values passed

# parameters in format function.
my_string = "{2}, is a {1} science {0} portal"

print (my_string.format("ProgrammingTech", "computer","engineering"))
################################################

# Python program using multiple place
# holders to demonstrate str.format() method

# Multiple placeholders in format() function
my_string = "{}, is a {} science portal for {}"
print (my_string.format("GeeksforGeeks", "computer", "geeks"))

# different datatypes can be used in formatting
print ("Hi ! My name is {} and I am {} years old"
                              .format("User", 19))

# The values passed as parameters
# are replaced in order of their entry
print ("This is {3} {2} {1} {0}"
       .format("one", "two", "three", "four"))

"""
format_map() Formats specified values in a string
Parameters :

input_dict : Takes a single parameter which is input dictionary.

Returns :

Returns key's values of the input dictionary.
"""

# input stored in variable a.
a = {'x':'John', 'y':'Wick'}
```

```
# Use of format_map() function
print("{x}'s last name is {y}".format_map(a))
################################################

# Input dictionary
profession = { 'name':['Barry', 'Bruce'],
               'profession':['Engineer', 'Doctor'],
               'age':[30, 31] }

# Use of format_map() function
print('{name[0]} is an {profession[0]} and he'
      ' is {age[0]} years old.'.format_map(profession))

print('{name[1]} is a {profession[1]} and he'
      ' is {age[1]} years old.'.format_map(profession))
################################################

# Python code showing practical
# use of format_map() function
def chk_msg(n):

    # input stored in variable a.
    a = {'name':"George", 'mesg':n}

    # use of format_map() function
    print('{name} has {mesg} new messages'.format_map(a))


chk_msg(10)

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

# Python code to demonstrate the working of
# index()

# initializing target string
ch = "geeksforgeeks"

# initializing argument string
ch1 = "geeks"

# using index() to find position of "geeks"
# starting from 2nd index
# prints 8
pos = ch.index(ch1,2)

print ("The first position of geeks after 2nd index : ",end="")
print (pos)
################################################

# Python code to demonstrate the exception of
# index()

# initializing target string
ch = "geeksforgeeks"

# initializing argument string
```

```
ch1 = "gfg"

# using index() to find position of "gfg"
# raises error
pos = ch.index(ch1)

print ("The first position of gfg is : ",end="")
print (pos)
"""
This function is used to extract the suffix or
prefix length after or before the target word.
"""
"""
isalnum()
Returns True if all characters in the string are
alphanumeric
Parameter: isalnum() method takes no parameters

Return:

    True: If all the characters are alphanumeric
    False: If one or more characters are not alphanumeric
"""

# Python program to demonstrate the use of
# isalnum() method

# here a,b and c are characters and 1,2 and 3
# are numbers
string = "abc123"
print(string.isalnum())

# here a,b and c are characters and 1,2 and 3
# are numbers but space is not a alphanumeric
# character
string = "abc 123"
print(string.isalnum())
"""
isalpha() Returns True if all characters in the string are
in the alphabet
string.isalpha()
Parameters:
isalpha() does not take any parameters
Returns :
1.True- If all characters in the string are alphabet.
2.False- If the string contains 1 or more non-alphabets.
"""

# Python code for implementation of isalpha()

# checking for alphabets
string = 'VIT'
print(string.isalpha())


string = 'VIT2020'
print(string.isalpha())

# checking if space is an alphabet
string = 'VIT PUNE'
print( string.isalpha())
```

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

# Python3 program to demonstrate the use
# of isdecimal()
s = "12345"
print(s.isdecimal())

# contains alphabets
s = "12geeks34"
print(s.isdecimal())

s = "110000"
print(s.isdecimal())

# contains numbers and spaces
s = "12 34"
print(s.isdecimal())
"""
isdigit() Returns True if all characters in the string are
digits
Parameters:
isdigit() does not take any parameters
Returns :
1.True- If all characters in the string are digits.
2.False- If the string contains 1 or more non-digits.
"""

# Python code for implementation of isdigit()

# checking for digit
string = '15460'
print(string.isdigit())


string = '154vit60'
print(string.isdigit())

"""
isidentifier()  Returns True if the string is an identifier
Parameters:
    The method does not take any parameters

Return Value:

    The method can return one of the two values:
    True: When the string is a valid identifier.
    False: When the string is not a valid identifier.
"""
# Python code to illustrate the working of isidentifier()

# String with spaces
string = "Geeks for Geeks"
print(string.isidentifier())

# A Perfect identifier
string = "GeeksforGeeks"
print(string.isidentifier())
```

```
# Empty string
string = "_"
print(string.isidentifier())

# Alphanumerical string
string = "Geeks0for0Geeks"
print(string.isidentifier())

# Beginning with an integer
string = "54Geeks0for0Geeks"
print(string.isidentifier())
"""
islower() Returns True if all characters in the string are
lower case
string.islower()
Parameters:
islower() does not take any parameters
Returns :
1.True- If all characters in the string are lower.
2.False- If the string contains 1 or more non-lowercase characters.
"""

# Python code for implementation of isupper()

# checking for lowercase characters
string = 'geeksforgeeks'
print(string.islower())

string = 'GeeksforGeeks'
print(string.islower())
"""
isnumeric()  Returns True if all characters in the string are
numeric
"""
txt = "565543"

x = txt.isnumeric()

print(x)

txt = "ABCDE"

x = txt.isnumeric()

print(x)

a = "\u0030" #unicode for 0
b = "\u00B2" #unicode for &sup2; superscript 2
c = "10km2"

print(a.isnumeric())
print(b.isnumeric())
print(c.isnumeric())
"""
isprintable()Returns True if all characters in the string
are printable
"""
txt = "Hello! Are you #1?"

x = txt.isprintable()
```

```
print(x)

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

x = txt.isprintable()

print(x)
"""
isspace()
Returns True if all characters in the string are
whitespaces
"""
txt = "   "

x = txt.isspace()

print(x)

txt = "   s   "

x = txt.isspace()

print(x)
"""
istitle()
Returns True if the string follows the rules of
a title
"""
txt = "Hello, And Welcome To My World!"

x = txt.istitle()

print(x)

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())

"""
isupper() Returns True if all characters in the string are
upper case
string.isupper()
Parameters:
isupper() does not take any parameters
Returns :
1.True- If all characters in the string are uppercase.
2.False- If the string contains 1 or more non-uppercase characters.
"""

# Python code for implementation of isupper()

# checking for uppercase characters
string = 'GEEKSFORGEEKS'
print(string.isupper())
```

```
string = 'GeeksforGeeks'
print(string.isupper())

"""
join() Joins the elements of an iterable to the end of the
string
"""
myTuple = ("John", "Peter", "Vicky")

x = "#".join(myTuple)

print(x)

myDict = {"name": "John", "country": "Norway"}
mySeparator = "TEST"

x = mySeparator.join(myDict)

print(x)

myList = ['A','B','C','D']
mySeparator = "Iam"

x = mySeparator.join(myList)

print(x)

mySet = {'1','2','3','4'}
mySeparator = "Iam"

x = mySeparator.join(mySet)

print(x)

"""
ljust()   Returns a left justified version of the string
"""
txt = "apple"

x = txt.ljust(20)

print(x, "is my favorite fruit.")

txt = "orange"

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

print(x)

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

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

1 - everything before the "match"
2 - the "match"
3 - everything after the "match"
"""
txt = "I could eat apples all day"
```

```
x = txt.partition("apples")

print(x)

txt = "I could eat bananas all day"

x = txt.partition("apples")

print(x)

"""
rfind()   Searches the string for a specified value and
returns the last position of where it was found
"""
txt = "Mi casa, su casa."

x = txt.rfind("casa")

print(x)

txt = "Hello, welcome to my world."

x = txt.rfind("e")

print(x)
txt = "Hello, welcome to my world."

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

print(x)

txt = "Hello, welcome to my world."

print(txt.rfind("q"))
#print(txt.rindex("q"))
"""
rindex()  Searches the string for a specified value and
returns the last position of where it was found
"""
txt = "Mi casa, su casa."

x = txt.rindex("casa")

print(x)

txt = "Hello, welcome to my world."

x = txt.rindex("e")

print(x)
txt = "Hello, welcome to my world."

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

print(x)
txt = "Hello, welcome to my world."

print(txt.rfind("q"))
print(txt.rindex("q"))
"""
rjust()   Returns a right justified version of the string
```

```
Return a 20 characters long, right justified version of the word "apple
"""
txt = "apple"

x = txt.rjust(20)

print(x, "is my favorite fruit.")
#Using the letter "O" as the padding character:
txt = "banana"

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

print(x)
"""
rpartition() Returns a tuple where the string is parted
into three parts
The rpartition() method searches for the last occurrence of a specified string, and splits the s

The first element contains the part before the specified string.

The second element contains the specified string.

The third element contains the part after the string.
"""
txt = "I could eat apple all day, apples are my favorite fruit"

x = txt.rpartition("apples")

print(x)

txt = "I could eat bananas all day, bananas are my favorite fruit"

x = txt.rpartition("apples")

print(x)
"""
rsplit()  Splits the string at the specified separator, and
returns a list
Split a string into a list, using comma, followed by a space (, ) as the separator:
"""
txt = "apple, banana, cherry"

x = txt.rsplit(", ")

print(x)

txt = "apple, banana, cherry"

# setting the maxsplit parameter to 1, will return a list with 2 elements!
x = txt.rsplit(", ", 1)

print(x)
"""
rstrip()  Returns a right trim version of the string
Remove spaces to the right of the string:
"""
txt = "     banana     "

x = txt.rstrip()

print("of all fruits", x, "is my favorite")
```

```
txt = "banana,,,,,tssqqqww....."

x = txt.rstrip(",.tsqw")

print(x)

"""
splitlines() Splits the string at line breaks and
returns a list
"""
txt = "Thank you for the music\nWelcome to the jungle"

x = txt.splitlines()

print(x)

txt = "Thank you for the music\nWelcome to the jungle"

x = txt.splitlines(True)

print(x)
"""
startswith() Returns true if the string starts with
the specified value
Check if the string starts with "Hello":
"""
txt = "Hello, welcome to my world."

x = txt.startswith("Hello")

print(x)
#Check if position 7 to 20 starts with the characters "wel":
txt = "Hello, welcome to my world."

x = txt.startswith("wel", 7, 20)

print(x)

"""
swapcase()   Swaps cases, lower case becomes upper case and
vice versa
Make the lower case letters upper case and the upper case letters lower case:
"""
txt = "Hello My Name Is PETER"

x = txt.swapcase()

print(x)
"""
title()   Converts the first character of each word to upper case
Make the first letter in each word upper case:
"""
txt = "Welcome to my world"

x = txt.title()

print(x)

txt = "Welcome to my 2nd world"
```

```
x = txt.title()

print(x)
txt = "hello b2b2b2 and 3g3g3g"

x = txt.title()

print(x)
"""
upper()   Converts a string into upper case
"""
print(str.
"""
zfill()   Fills the string with a specified number of 0
values at the beginning
The zfill() returns a copy of the string with '0' filled
 to the left. The length of the returned string depends
 on the width provided.
 Ref:https://www.programiz.com/python-programming/methods/string/zfill
"""
print(str.zfill(15))
text = "program is fun"
print(text.zfill(15))
print(text.zfill(20))
print(text.zfill(10))
"""
Note: All string methods returns new values.
They do not change the original string.
"""
```