***Strings***

Strings in python are contiguous series of characters delimited by single or double quotes. Strings in python are immutable. String is a datatype in python.

**Creating strings**

* name = “wozti” # a string
* mychar = ‘A’ # a character
* name1 = str() # this will create empty string object
* name2 = str("WOZTI") # string object containing ‘WOZTI'
* String\_var = """ This document will help you to

explore all the concepts

of Python Strings!!! """

**Example:**

print('doesn\'t')

**Result:**

doesn't

**Example:**

print(' "Yes," they said.')

**Result:**

"Yes," they said.

**Example:**

print("\"Yes,\" they said.")

**Result:**

"Yes," they said.

**Example:**

print(' "Isn\'t," they said.')

**Result:**

"Isn't," they said.

**Example:**

print('C:\some\name')

**Result:**

C:\some

ame

/////////////////////////////

If you don’t want characters prefaced by \ to be interpreted as special characters, you can use raw strings by adding an r before the first quote:

//////////////////////////////////

**Example:**

print(r'C:\some\name')

**Result:**

C:\some\name

**Example:**

print("""

Usage: thingy [OPTIONS]

-h Display this usage message

-H hostname Hostname to connect to

""")

**Result:**

Usage: thingy [OPTIONS]

-h Display this usage message

-H hostname Hostname to connect to

/////////////////////////////////////

End of lines are automatically included in the string, but it’s possible to prevent this by adding a \ at the end of the line. The following example:

////////////////////////////////////

**Example:**

print("""\

Usage: thingy [OPTIONS]

-h Display this usage message

-H hostname Hostname to connect to

""")

**Result:**

Usage: thingy [OPTIONS]

-h Display this usage message

-H hostname Hostname to connect to

**Example:**

print(3 \* 'un' + 'ium')

**Result:**

unununium

**Example:**

print('py''thon')

**Result:**

python

**Example:**

a = 'py'

print(a+'thon')

**Result:**

Python

**Example:**

word = 'Python'

print(word[:2] + word[2:])

**Result:**

Python

**Example:**

word = 'Python'

print(word[:4] + word[4:])

**Result:**

Python

**Example:**

word = 'Python'

print(word[-2:])

**Result:**

on

**Example:**

word = 'Python'

print(word[4:42])

**Result:**

on

**Example:**

word = 'Python'

print(word[42:])

**Result:**

**Example:**

word = 'Python'

print('J' + word[1:])

**Result:**

Jython

**Example:**

word = 'Python'

print(word[:2] + 'py')

**Result:**

Pypy

**Example:**

a = "PYTHON IS AWESOME"

print("Lowercase string:", a.casefold())

a = "PYthon IS AWESOME"

print("Lowercase string:", a.casefold())

**Result:**

Lowercase string: python is awesome

Lowercase string: python is awesome

**Example:**

c = 'My name is ¦֎©༂' # unicode characters taken from web

print('The string is:', c)

a = c.encode()

print('The encoded version is:', a)

b = a.decode()

print('The decoded version is:', b)

**Result:**

The string is: My name is ¦֎©༂

The encoded version is: b'My name is \xc2\xa6\xd6\x8e\xc2\xa9\xe0\xbc\x82'

The encoded version is: My name is ¦֎©༂

**Example:**

# endswith() method can take start and end parameters.

a = "Python is easy to learn."

b = a.endswith('to learn')

print(b)

c = a.endswith('to learn.')

print(c)

d = a.endswith('Python is easy to learn.')

print(d)

**Result:**

False

True

True

**Example:**

print('a\tb\tc'.expandtabs(0))

print('a\tb\tc'.expandtabs(5))

print('a\tb\tc'.expandtabs(10))

**Result:**

abc

a b c

a b c

**Example:**

print("The sum of 1 + 2 is {}".format(1+2))

print("The sum of some numbers is {}, {}, {}".format(1+2, 4+5, 9+1))

print("you are {} at {}{}".format('good', 'python', 3))

print("you are {2} at {1}{0}".format(3, 'python', 'good'))

print("you are {b} at {1}{0}".format(3,'python',b = 'good'))

print("My salary is {0:f}".format(100))

print("My salary is {0:.0f}".format(10.4685))

print("My salary is {0:.0f}".format(10.7685))

print("My salary is {0:.2f}".format(10.7685))

**Result:**

The sum of 1 + 2 is 3

The sum of some numbers is 3, 9, 10

you are good at python3

you are good at python3

you are good at python3

My salary is 100.000000

My salary is 10

My salary is 11

My salary is 10.77

**Example:**

def unorganized(a, b):

for i in range (a, b):

print ( i, i\*\*2, i\*\*3, i\*\*4 )

def organized(a, b):

for i in range (a, b):

print("{:6d} {:6d} {:6d} {:6d}".format(i, i \*\* 2, i \*\* 3, i \*\* 4))

n1 = int(input("Enter lower range :-\n"))

n2 = int(input("Enter upper range :-\n"))

print("------Before Using Formatters-------")

unorganized(n1, n2)

print()

print("-------After Using Formatters---------")

print()

organized(n1, n2)

**Result:**

Enter lower range :-

2

Enter upper range :-

5

------Before Using Formatters-------

2 4 8 16

3 9 27 81

4 16 64 256

-------After Using Formatters---------

2 4 8 16

3 9 27 81

4 16 64 256

**Example:**

# format\_map is used to fetch values in a dictionary

a = {'x':'John', 'y':'Wick'}

print("{x}'s last name is {y}".format\_map(a))

**Result:**

John's last name is Wick

**Example:**

# The isdecimal() doesn't take any parameters.

s = "28212"

print(s.isdecimal())

# contains alphabets

s = "32ladk3"

print(s.isdecimal())

# contains spaces

s = "32l 3"

print(s.isdecimal())

# contains alphabets and spaces

s = "Mo3 nicaG el l22er"

print(s.isdecimal())

**Result:**

True

False

False

False

**Example:**

# String Containing digits and Numeric Characters

s = '23455'

print(s.isdecimal())

s = '²3455'

print(s.isdecimal())

s = '½'

print(s.isdecimal())

**Result:**

True

False

False

///////////////////////////////////

The isdigit() method returns True if all the characters are digits, otherwise False.

Exponents, like ², are also considered to be a digit.

///////////////////////////////////////

**Example:**

txt = "50800"

x = txt.isdigit()

a = "\u0030" #unicode for 0

print("u0030")

print("\u0030")

b = "\u00B2" #unicode for ²

print(a.isdigit())

print(b.isdigit())

**Result:**

u0030

0

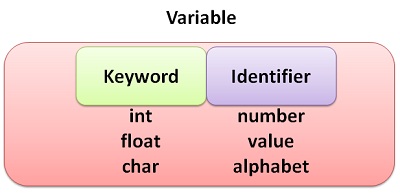
True

True

////////////////////////////////////////////////////

Difference between *keyword* and *identifier*.

Every language has keywords and identifiers, which are only understood by its compiler. Keywords are predefined reserved words, which possess special meaning. An identifier is a unique name given to a particular variable, function or label of class in the program. The identifier is only used to identify an entity uniquely in a program at the time of execution whereas, a variable is a name given to a memory location, that is used to hold a value. Variable is only a kind of identifier, other kinds of identifiers are function names, class names, structure names, etc.



**str.isidentifier()**

Return True if the string is a valid identifier according to the language definition, section [Identifiers and keywords](https://docs.python.org/3/reference/lexical_analysis.html#identifiers).

Call ‘[keyword.iskeyword()](https://docs.python.org/3/library/keyword.html#keyword.iskeyword)’ to test whether string s is a reserved identifier, such as [def](https://docs.python.org/3/reference/compound_stmts.html#def) and [class](https://docs.python.org/3/reference/compound_stmts.html#class).

///////////////////////////////////////////////////////////

**Example:**

from keyword import iskeyword

print('hello'.isidentifier(), iskeyword('hello'))

print('def'.isidentifier(), iskeyword('def'))

**Result:**

True False

True True

///////////////////////////////////

The *issnumeric()* methods returns “True” if all characters in the string are numeric characters, Otherwise, It returns “False”.

This function is used to check if the argument contains all numeric characters such as : integers, fractions, subscript, superscript, roman numerals etc.(all written in unicode).

**Syntax :**

*string*.isnumeric()

**Parameters:**

isnumeric() does not take any parameters

**Returns :**

1.True- If all characters in the string are numeric characters.

2.False- If the string contains 1 or more non-numeric characters.

////////////////////////////////////

**Example:**

a = '123ayu456'

print(a.isnumeric())

b = '123456'

print( b.isnumeric())

**Result:**

False

True

////////////////////////////////////////////

The *isprintable()* methods returns “True” if all characters in the string are printable or the string is empty, Otherwise, It returns “False”.  
This function is used to check if the argument contains any printable characters such as :

* Digits ( 0123456789 )
* Uppercase letters ( ABCDEFGHIJKLMNOPQRSTUVWXYZ )
* Lowercase letters ( abcdefghijklmnopqrstuvwxyz )
* Punctuation characters ( !”#$%&'()\*+, -./:;?@[\]^\_`{ | }~ )
* Space ( )

**Syntax :**

**string.isprintable()**

**Parameters:**

isprintable() does not take any parameters

**Returns :**

1.True- If all characters in the string are printable or the string is empty.

2.False- If the string contains 1 or more non printable characters.

///////////////////////////////////////////////////////////

**Example:**

a = 'My name is Ayush'

print(a.isprintable())

a = 'My name is \n Ayush'

print(a.isprintable())

a = ''

print(a.isprintable())

**Result:**

True

False

True

**Example:**

a ='hello\ngood\nmorning\nall dear friends'

b = ''

count = 0

for x in a:

if (x.isprintable()) == False:

count += 1

b += ' '

else:

b += x

print(count)

print(b)

**Result:**

3

hello good morning all dear friends

////////////////////////////////////////

The *isspace()* method returns ‘True’ if there are only whitespace characters in the string. If not, it return ‘False’.

Characters that are used for spacing are called whitespace characters. For example: tabs, spaces, newline etc.

The syntax of *isspace()* is:

string.isspace()

The *isspace()* method doesn't take any parameters.

The *isspace()* method returns:

* True -  if all characters in the string are whitespace characters
* False -  if the string is empty or contains at least one non-printable() character

**Example:**

s = '\t'

print(s.isspace())

s = ' a '

print(s.isspace())

s = ''

print(s.isspace())

s = ' '

print(s.isspace())

s = '\n'

print(s.isspace())

**Result:**

True

False

False

True

True

**Example:**

a = "Good Morning Friends"

x = a.istitle()

print(x)

**Result:**

True

/////////////////////////////////////////////////

The partition() method splits the string at the first occurrence of the argument string and returns a tuple containing the part before separator, argument string and the part after the separator.

The syntax of partition() is:

string.partition(separator)

The partition() method takes a string parameter separator that separates the string at the first occurrence of it.

The partition method returns a 3-tuple containing:

* the part before the separator, separator parameter, and the part after the separator if the separator parameter is found in the string
* string itself and two empty strings if the separator parameter is not found

////////////////////////////////

**Example:**

a = "hello good morning"

print(a.partition('good'))

print(a.partition('not'))

a = "hello good morning, have a good day!"

print(a.partition('good'))

**Result:**

('hello ', 'good', ' morning')

('hello good morning', '', '')

('hello ', 'good', ' morning, have a good day!')

**Example:**

a = "hello good morning, have a good day!"

print(a.rpartition('good'))

**Result:**

('hello good morning, have a ', 'good', ' day!')

/////////////////////////////////////////////////

The *rsplit()* method splits a string into a list, starting from the right.

Syntax

*string*.rsplit(*separator, maxsplit*)

*Separator* is optional. Itspecifies the separator to use when splitting the string. By default any whitespace is a separator.

*Maxsplit* is optional. It specifies how many splits to do. Default value is -1, which is "all occurrences".

If no "maxsplit" is specified, this method will return the same as the [*split()*](https://www.w3schools.com/python/ref_string_split.asp) method.

////////////////////////////////////////////////

**Example:**

a = "Bus, Truck, Car"

x = a.rsplit(", ")

print(x)

x = a.rsplit(", ", 1)

print(x)

x = a.rsplit(", ", 2)

print(x)

x = a.rsplit(", ", 5)

print(x)

**Result:**

['Bus', 'Truck', 'Car']

['Bus, Truck', 'Car']

['Bus', 'Truck', 'Car']

['Bus', 'Truck', 'Car']

/////////////////////////////////////////

The *startswith()* method returns *True* if the string starts with the specified value, otherwise *False*.

## Syntax

*string*.startswith(*value, start, end*)

*Value* is required. To check if the string starts with *value*.

*Start* is optional. An Integer specifying at which position to start the search.

*End* is optional. An Integer specifying at which position to end the search.

////////////////////////////////////////////////

**Example:**

#Check if position 6 to 15 starts with the characters "goo":

a = "hello good morning"

x = a.startswith("go", 6, 15)

print(x)

x = a.startswith("g", 6, 15)

print(x)

x = a.startswith("m", 6, 15)

print(x)

**Result:**

True

True

False

**Example:**

print ('I am % d years old' %20) # 'd' represent digit and '%' represent starting position

print ('I am % s years old' %'some') # 's' represent string and '%' represent starting position

**Result:**

I am 20 years old

I am some years old

**Example:**

a = '%d' %100

print(a)

print(type(a))

**Result:**

100

<class 'str'>

**Example:**

b = '%s' %'hai'

print(b)

print(type(b))

**Result:**

hai

<class 'str'>

**Example:**

print('%i' % 0b1111) # Converting binary value to integer

print(int(0b1111)) # Converting binary value to integer

print(bin(15)) # Converting integer to binary value

print(hex(15)) # Converting integer to hexadecimal string

**Result:**

15

15

0b1111

0xf

**Example:**

d = 'z'

print(ord(d)) #'ord()' function is used to convert a character to integer.

**Result:**

122

**Example:**

d = 122

print(chr(d)) #'chr()' function is used to convert an integer to character.

**Result:**

z

**Example:**

s = "100100"

c = int(s,2)

print ("After converting to integer base 2 : ",c)

# printing string converting to int base 2

print(bin(c))

**Result:**

After converting to integer base 2 : 36

0b100100

**Example:**

# printing integer converting to octal string

c = oct(56)

print ("After converting 56 to octal string : ",c)

# printing integer converting from octal string

print(int(c,8))

**Result:**

After converting 56 to octal string : 0o70

56