Python
Bootcamp
& Masterclass

string methods 4

gknxt

.isprintable()

The `.isprintable()` method returns `True` if all characters in the string are printable (including space, but not newline) or the string is empty, `False` otherwise.

```
s = ''
s.isprintable()
t = ' '
t.isprintable()
u = 'Welcome Niño'
u.isprintable()
```

```
True

True

True
```

```
v = 'a\tb'
v.isprintable()
w = 'a\rb'
w.isprintable()
x = 'a\nb'
x.isprintable()
```

```
False

False

False
```

.isspace()

The `.isspace()` method returns `True` if there are only whitespace characters in the string and there is at least one character, `False` otherwise.

```
s = ' \t \n '
s.isspace()
t = '   '
t.isspace()
u = '\u0020'
u.isspace()
```

```
True

True

True
```

```
v = ''
v.isspace()
w = '  s  '
w.isspace()
```

```
False

False
```

ASCII(American Standard Code for Information Interchange)

ASCII defined 128 characters by using 7 bits of a byte (2^7 = 128) (8th bit → parity bit /error check)

The latest standard, Unicode Standard, supports more than 120 of the world's most spoken languages.

Presently, UTF-8 is the most common character encoding in the industry as UTF-8 is backward-compatible with ASCII and can represent any standard Unicode character (The first 128 UTF-8 characters precisely match the first 128 ASCII characters)

(UTF stands for Unicode Transformation Format)

Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value	Hex	Value
00	NUL	10	DLE	20	SP	30	0	40	@	50	P	60	`	70	p
01	SOH	11	DC1	21	!	31	1	41	A	51	Q	61	a	71	q
02	STX	12	DC2	22	"	32	2	42	B	52	R	62	b	72	r
03	ETX	13	DC3	23	#	33	3	43	C	53	S	63	c	73	s
04	EOT	14	DC4	24	$	34	4	44	D	54	T	64	d	74	t
05	ENQ	15	NAK	25	%	35	5	45	E	55	U	65	e	75	u
06	ACK	16	SYN	26	&	36	6	46	F	56	V	66	f	76	v
07	BEL	17	ETB	27	'	37	7	47	G	57	W	67	g	77	w
08	BS	18	CAN	28	(	38	8	48	H	58	X	68	h	78	x
09	HT	19	EM	29	)	39	9	49	I	59	Y	69	i	79	y
0A	LF	1A	SUB	2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
0B	VT	1B	ESC	2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
0C	FF	1C	FS	2C	,	3C	<	4C	L	5C	\	6C	l	7C	|
0D	CR	1D	GS	2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
0E	SO	1E	RS	2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
0F	SI	1F	US	2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL

gknxt

.isascii()

The `.isascii()` method returns `True` if the string is empty or all characters in the string are ASCII, `False` otherwise

```
s = ''
s.isascii()
t = ' '
t.isascii()
u = 'ñ'
u.isascii()
v = '/'
v.isascii()
```

```
True

True

False

True
```

.isalpha()

The `.isalpha()` method returns `True` if all characters in the string are alphabetic and there is at least one character, `False` otherwise (Alphabetic characters are those characters defined in the Unicode character database as “Letter”)

```
s = ''        # empty string is not an alphabet
s.isalpha()
t = ' '       # space is not an alphabet
t.isalpha()
u = 'hello '  # space is not an alphabet
u.isalpha()
u = '123'     # digit is not an alphabet
u.isalpha()
```

```
False

False

False

False
```

```
v = 'hello'
v.isalpha()
w = 'Niño'
w.isalpha()
```

```
True

True
```

`.isdecimal` ⊆ `.isdigit` ⊆ `.isnumeric`

The `.isdecimal()` method returns `True` if all characters in the string are decimal characters, and there is at least one character, `False` otherwise.

The `.isdigit()` method returns `True` if all characters in the string are digits, and there is at least one character, `False` otherwise.

The `.isnumeric()` method returns `True` if all characters in the string are numeric characters, and there is at least one character, `False` otherwise.

If a string is `decimal`, then it'll also be `digit` and `numeric`. (`numeric is a superset of digit which is a superset of decimal)` If a string is `digit`, then it'll also be `numeric`.

isdecimal()==True (so, isdigit()==True and isnumeric()==True)

Almost all **digits** from all supported languages are **decimals** (only few are shown below)

"0123456789"	ENGLISH DIGITS (0 – 9)	"౦౧౨౩౪౫౬౭౮౯"	KANNADA DIGITS (0 – 9)
"٠١٢٣٤٥٦٧٨٩"	ARABIC-INDIC DIGITS (0 – 9)	"൦൧൨൩൪൫൬൭൮൯"	MALAYALAM DIGITS (0 – 9)
"०१२३४५६७८९"	DEVANAGARI DIGITS (0 – 9)	"๐๑๒๓๔๕๖๗๘๙"	THAI DIGITS (0 – 9)
"০১২৩৪৫৬৭৮৯"	BENGALI DIGITS (0 – 9)	"๐๑๒๓๔๕๖๗๘๙"	LAO DIGITS (0 – 9)
"੦੧੨੩੪੫੬੭੮੯"	GURMUKHI DIGITS (0 – 9)	"०১২৩৪৫৬৭৮৯"	MYANMAR DIGITS (0 – 9)
"૦૧૨૩૪૫૬૭૮૯"	GUJARATI DIGITS (0 – 9)	"๐๑๒๓๔๕๖๗๘๙"	KHMER DIGITS (0 – 9)
"୦୧୨୩୪୫୬୭୮୯"	ORIYA DIGITS (0 – 9)	"０１２３４５６７８９"	FULLWIDTH DIGITS (0 – 9)
"௦௧௨௩௪௫௬௭௮௯"	TAMIL DIGITS (0 – 9)	"𝟎𝟏𝟐𝟑𝟒𝟓𝟔𝟕𝟖𝟗"	MATH BOLD DIGITS (0 – 9)
"౦౧౨౩౪౫౬౭౮౯"	TELUGU DIGITS (0 – 9)	"𝟘𝟙𝟚𝟛𝟜𝟝𝟞𝟟𝟠𝟡"	MATH DOUBLE-STRUCK DIGITS (0 – 9)

gk nxt

isdecimal()==False but, isdigit()==True (so, isnumeric()==True)

Some **special digits** from many languages are **digits** (only few are shown below)

"⁰¹²³⁴⁵⁶⁷⁸⁹"	SUPERSCRIPT DIGITS ZERO TO NINE
"₀₁₂₃₄₅₆₇₈₉"	SUBSCRIPT DIGITS ZERO TO NINE
"0.1.2.3.4.5.6.7.8.9."	DIGITS WITH PERIOD ZERO TO NINE
"0,1,2,3,4,5,6,7,8,9,"	DIGITS WITH COMMA ZERO TO NINE
"⓪①②③④⑤⑥⑦⑧⑨"	CIRCLED DIGITS ZERO TO NINE
"❿❶❷❸❹❺❻❼❽❾"	NEGATIVE CIRCLED DIGITS ZERO TO NINE
"(1)(2)(3)(4)(5)(6)(7)(8)(9)"	PARENTHESIZED DIGITS ONE TO NINE
"❶❷❸❹❺❻❼❽❾"	DINGBAT NEGATIVE CIRCLED SANS-SERIF DIGITS ONE TO NINE
"᎑᎒᎓᎔᎕᎖᎗᎘᎙"	ETHIOPIC DIGITS ONE TO NINE

gknxt

isdecimal()==False and isdigit()==False but, isnumeric()==True)

Some **chars** from many languages are **numeric** (only few are shown below)

" ½ ⅓ ¼ ⅕ ⅙ ⅔ ¾ ⅖ ⅗ ⅘ ⅚ ⅜ ⅝ ⅞ "	VULGAR FRACTIONS
" Ⅰ Ⅱ Ⅲ Ⅳ Ⅴ Ⅵ Ⅶ Ⅷ Ⅸ Ⅹ Ⅺ Ⅻ Ⅼ Ⅽ Ⅾ Ⅿ"	ROMAN NUMERALS
"⑩⑪⑫⑬⑭⑮⑯㊷㊸㊹㊺㊻㊼㊽㊾㊿"	CIRCLED NUMBER (10 – 50)
" 🔟 2⃣0⃣ 3⃣0⃣ 4⃣0⃣ 5⃣0⃣ 6⃣0⃣ 7⃣0⃣ 8⃣0⃣"	CIRCLED NUMBERS ON BLACK SQUARE (10 – 80)
" ⑽ ⑾ ⑿ ⒀ ⒁ ⒂ ⒃ ⒄ ⒅ ⒆ ⒇"	PARENTHESIZED NUMBERS (10 – 20)
" 10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20."	NUMBERS WITH FULL STOP (10 – 20)
" ⓫ ⓬ ⓭ ⓮ ⓯ ⓰ ⓱ ⓲ ⓳ ⓴"	NEGATIVE CIRCLED NUMBERS (11 – 20)
" 〡 〢 〣 〤 〥 〦 〧 〨 〩 〸 〹 〺"	HANGZHOU NUMERAL (1 – 10, 20, 30)
" ፲ ፳ ፴ ፵ ፶ ፷ ፸ ፹ ፺ ፻ ፼"	ETHIOPIC NUMBERS (10 – 90, 100, 10000)

gknxt

```
a = ''              # empty string
a.isdecimal()
a.isdigit()
a.isnumeric()
```

```
False

False

False
```

```
b = ' '             # space
b.isdecimal()
b.isdigit()
b.isnumeric()
```

```
False

False

False
```

```
c = '0'             # NUMBER ZERO
c.isdecimal()
c.isdigit()
c.isnumeric()
```

```
True

True

True
```

```
d = '〇'             # IDEOGRAPHIC NUMBER ZERO
d.isdecimal()
d.isdigit()
d.isnumeric()
```

```
False

False

True
```

```
e = '98.4'            # float
e.isdecimal()
e.isdigit()
e.isnumeric()
```

```
False

False

False
```

```
f = '-40'            # negative number
f.isdecimal()
f.isdigit()
f.isnumeric()
```

```
False

False

False
```

```
g = '0 3 8'            # white space between digits
g.isdecimal()
g.isdigit()
g.isnumeric()
```

```
True

True

True
```

```
i = 'Ⅹ'                # Roman numeral 10
i.isdecimal()
i.isdigit()
i.isnumeric()
```

```
False

False

True
```

```
h = '⑴②❸'             # bracketed, circled & negative circled digits
h.isdecimal()
h.isdigit()
h.isnumeric()
```

```
False

True

True
```

```
j = 'X'                # letter X
j.isdecimal()
j.isdigit()
j.isnumeric()
```

```
False

False

False
```

.isalnum( )

The **.isalnum( )** method returns **True** if all characters in the string are alphanumeric and there is at least one character, **False** otherwise.

A character c is alphanumeric if any one of the following returns **True**:

.isalpha( )

.isdecimal( )

.isdigit( )

.isnumeric( )

```
k = '-40'
k.isalnum()
```

```
False
```

```
l = ' '          # space
l.isalnum()
```

```
False
```

```
m = 'Niño'
m.isalnum()
```

```
True
```

len( )

The built-in function `len(s)` returns the length (the number of items/characters) of `s`. The argument, `s`, can be a sequence (string, bytes, tuple, list, or range) or a collection (dictionary, set, or frozen set)

```
len('')
len('\t')
len('\n')
len('3+4j')
len('Hello')
len('-23')
len('Niño')
len('🙏')
```

```
0
1
1
4
5
3
4
1
```

gk nxt

str( )

The built-in function `str(obj)` returns a string representation of the `obj`

```
str()
str('')
str(' ')
str(12)
str(24.7)
```

```
''

''

' '

'12'

'24.7'
```

```
str(3+4j)
str(True)
str('Hello')
str(12 + 6/2 * 3)
str(-23)
str('🙏')
```

```
'(3+4j)'

'True'

'Hello'

'21.0'

'-23'

'🙏'
```

ord( ) and chr( )

The built-in function `ord(c)` returns an integer representing the Unicode code point of `c`. This is the inverse of `chr()`

The built-in function `chr(i)` returns the string representing a character whose Unicode code point is the integer `i`. This is the inverse of `ord()`

```
ord('A')
ord('a')
ord('$')
ord('€')
ord('\N{euro sign}')
ord('\N{dollar sign}')
ord('\N{indian rupee sign}')
ord('\N{bitcoin sign}')
ord('\N{PERSON WITH FOLDED HANDS}')
```

```
65
97
36
8364
8364
36
8377
8383
128591
```

```
chr(65)
chr(97)
chr(36)
chr(8364)
chr(8377)
chr(8383)
chr(9786)
chr(128591)
```

```
'A'
'a'
'$'
'€'
'₹'
'₿'
'⏀'
'🙏'
```

Online Resources

For best python resources, please visit:

gknxt.com/python/

gknxt


Python
Bootcamp
& Masterclass
Thank You
for your Rating & Review
gknxt