PIAIC

PIAIC61773

(python) string buit-in functions:

capitalize() Method

The capitalize() method returns a string where the first character is upper case.

syntax: string.capitalize()

In [1]:
```
txt = "hello world welcome"
x = txt.capitalize()
print(x)
```

```
Hello world welcome
```

casefold() Method

The casefold() method returns a string where all characters are lower case.

syntax: string.casefold()

In [2]:
```
txt = "HELLO WORLD WELCOME"
x = txt.casefold()
print(x)
```

```
hello world welcome
```

count() Method

The count() method returns the number of times a specified value repeated in a string.

syntax: string.count(value, start, end)

where value= Required. A String. The string to value to search for start= Optional. An Integer. The position to start the search. Default is 0 end= Optional. An Integer. The position to end the search. Default is the end of the string

In [3]:
```
txt = "I love Python because it is modern programming language. Python is easy
to learn beacause of simple syntax."
x = txt.count("Python")
print(x)
```

```
2
```

let's look at another example having start and end value

In [4]:
```
txt = "I love Python because it is modern programming language. Python is easy
to learn beacause of simple syntax."
x = txt.count("Python",14,70)
print(x)
```

```
1
```

let's look at another example having only start value

In [5]:
```
txt = "I love Python because it is modern programming language. Python is easy
to learn beacause of simple syntax."
x = txt.count("Python",5)
print(x)
```

```
2
```

endswith() Method

The endswith() method returns True if the string ends with the specified value, otherwise False.

syntax: string.endswith(value, start, end)

where value= Required. The value to check if the string ends with start= Optional. An Integer specifying at which position to start the search end= Optional. An Integer specifying at which position to end the search

In [6]:
```
txt = "Hello, welcome to my world."
x = txt.endswith(".")
print(x)
```

```
True
```

Check if the string ends with the phrase "my world.":

In [7]:
```
txt = "Hello, welcome to my world."
x = txt.endswith("my world.")
print(x)
```

```
True
```

Check if position 5 to 11 ends with the phrase "my world.":

In [8]:
```
txt = "Hello, welcome to my world."
x = txt.endswith("my world.", 5, 11)
print(x)
```

```
False
```

center() Method

The center() method will center align the string, using a specified character (space is default) as the fill character.

syntax: string.center(length, character)

where length= Required. The length of the returned string character= Optional. The character to fill the missing space on each side. Default is " " (space)

In [9]:
```
txt = "PAKISTAN ZINDABAD"
x = txt.center(50, "*")
print(x)
```

```
****************PAKISTAN ZINDABAD****************
```

let's look at another example having no character by default it consider as space

In [10]:
```
txt = "PAKISTAN ZINDABAD"
x = txt.center(50)
print(x)
```

```
                  PAKISTAN ZINDABAD
```

encode() Method

The encode() method encodes the string, using the specified encoding. If no encoding is specified, UTF-8 will be used.

syntax: string.encode(encoding=encoding, errors=errors) where encoding= Optional. A String specifying the encoding to use. Default is UTF-8 errors= Optional. A String specifying the error method. Legal values are:

```
    'backslashreplace'  - uses a backslash instead of the character that could not
  be encoded
    'ignore'    - ignores the characters that cannot be encoded
    'namereplace'   - replaces the character with a text explaining the character
    'strict'    - Default, raises an error on failure
    'replace'   - replaces the character with a questionmark
    'xmlcharrefreplace' - replaces the character with an xml character
```

In [11]:
```
txt = "My name is Arslan"
x = txt.encode()
print(x)
```

```
b'My name is Arslan'
```

In [12]:
```
#UTF-8 encode the string:
txt = "My name is årslan"
x = txt.encode()
print(x)
```

```
b'My name is \xc3\xa5rslan'
```

expandtabs() Method

The expandtabs() method sets the tab size to the specified number of whitespaces.

syntax: string.expandtabs(tabsize) where: tabsize= Optional. A number specifying the tabsize. Default tabsize is 8

In [13]:
```
txt = "P\tI\tA\tI\tC"
x =  txt.expandtabs(2)
print(x)
```

```
P I A I C
```

In [14]:
```
txt = "P\tI\tA\tI\tC"
x =  txt.expandtabs(5)
print(x)
```

```
P    I    A    I    C
```

In [15]:
```
txt = "P\tI\tA\tI\tC"
x =  txt.expandtabs() #by default tabsize is 8
print(x)
```

```
P       I       A       I       C
```

find() Method

The find() method finds the first occurrence of the specified value.

The find() method returns -1 if the value is not found.

The find() method is almost the same as the index() method, the only difference is that the index() method raises an exception if the value is not found.

syntax: string.find(value, start, end) where value= Required. The value to search for start= Optional. Where to start the search. Default is 0 end= Optional. Where to end the search. Default is to the end of the string

In [16]:
```
txt = "hello world"
x = txt.find("world")
print(x)
```

```
6
```

In [17]:
```
txt = "hello world"
x = txt.find("welcome") #if no value found then it return -1
print(x)
```

```
-1
```

In [18]:
```
txt = "hello, welcome."
x = txt.find("e")
print(x)
```

```
1
```

In [19]:
```
txt = "Hello, welcome."
x = txt.find("e", 6, 12)
print(x)
```

```
8
```

index() Method

The index() method finds the first occurrence of the specified value.

The index() method raises an exception if the value is not found.

The index() method is almost the same as the find() method, the only difference is that the find() method returns -1 if the value is not found

syntax:string.index(value, start, end) where value Required. The value to search for start Optional. Where to start the search. Default is 0 end Optional. Where to end the search. Default is to the end of the string

In [20]:
```
txt = "I love Python because it is modern programming language. Python is easy
to learn beacause of simple syntax."
x = txt.index("Python")
print(x)
```

```
7
```

In [21]:
```
txt = "I love Python because it is modern programming language. Python is easy
to learn beacause of simple syntax."
x = txt.index("l")
print(x)
```

```
2
```

In [22]:
```
txt = "I love Python because it is modern programming language. Python is easy
to learn beacause of simple syntax."
x = txt.index("e", 5, 50)
print(x)
```

```
5
```

If the value is not found, the find() method returns -1, but the index() method will raise an exception:

In [23]:
```
txt = "I love Python because it is modern programming language. Python is easy
to learn beacause of simple syntax."
x = txt.index("z")
print(x)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-23-4dcd1483e2f9> in <module>
      1 txt = "I love Python because it is modern programming language. Pytho
n is easy to learn beacause of simple syntax."
----> 2 x = txt.index("z")
      3 print(x)

ValueError: substring not found
```

isalnum() Method

The isalnum() method returns True if all the characters are alphanumeric, meaning alphabet letter (a-z) and numbers (0-9). Example of characters that are not alphanumeric: (space)!#%&? etc.

syntax: string.isalnum()

In [ ]:
```
txt = "python3"
x = txt.isalnum()
print(x)
```

In [ ]:
```
txt = "python 3" # space
x = txt.isalnum()
print(x)
```

In [ ]:
```
txt = "python3.7" #dot of floating value
x = txt.isalnum()
print(x)
```

isdecimal() Method

The isdecimal() method returns True if all the characters are decimals (0-9).

This method is used on unicode objects.

syntax:string.isdecimal()

In [ ]:
```
txt = "\u0033" #unicode for 3
x = txt.isdecimal()
print(x)
```

In [ ]:
```
txt1 = "\u0030" #unicode for 0
txt2 = "\u0047" #unicode for G
x = txt1.isdecimal()
y = txt2.isdecimal()
print(x)
print(y)
```

zfill() Method

The zfill() method adds zeros (0) at the beginning of the string, until it reaches the specified length.

If the value of the len parameter is less than the length of the string, no filling is done.

syntax:string.zfill(len)

In [ ]:
```
txt = "PIAIC"
x = txt.zfill(10)
print(x)
```

In [ ]:
```
txt1 = "hello"
txt2 = "welcome to the jungle"
txt3 = "10.000"

print(txt1.zfill(10))
print(txt2.zfill(10))
print(txt3.zfill(10))
```

upper() Method

The upper() method returns a string where all characters are in upper case.

Symbols and Numbers are ignored.

syntax:string.upper()

In [ ]:
```
txt = "Hello world it's me arslan"
x = txt.upper()
print(x)
```

In [ ]:
```
txt = "Hello world it's me arslan 66." #with special character and number
x = txt.upper()
print(x)
```

lower() Method

The lower() method returns a string where all characters are lower case.

Symbols and Numbers are ignored.

synatx: string.lower()

In [ ]:
```
txt = "hELlo WoRld it's mE ArSLan"
x = txt.lower()
print(x)
```

In [ ]:
```
txt = "hELlo WoRld it's mE ArSLan 66."#with special character and number
x = txt.lower()
print(x)
```

title() Method

The title() method returns a string where the first character in every word is upper case. Like a header, or a title.

If the word contains a number or a symbol, the first letter after that will be converted to upper case.

syntax: string.title()

In [ ]:
```
txt = "Welcome to pIAIC"
x = txt.title()
print(x)
```

Note that the first letter after a non-alphabet letter is converted into a upper case letter:

In [ ]:
```
txt = "hello a1b2c2 and 2a3b4c"
x = txt.title()
print(x)
```

swapcase() Method

The swapcase() method returns a string where all the upper case letters are lower case and vice versa.

syntax: string.swapcase()

In [ ]:
```
txt = "Hello My Name Is aRSLAN"
x = txt.swapcase()
print(x)
```

strip() Method

The strip() method removes any leading (spaces at the beginning) and trailing (spaces at the end) characters (space is the default leading character to remove)

syntax: string.strip(characters) where characters Optional. A set of characters to remove as leading/trailing characters

In [24]:
```
txt = "     python     "
x = txt.strip()
print("of all languages", x, "is my favorite")
```

```
of all languages python is my favorite
```

In [25]:
```
#Remove the leading and trailing characters:
txt = "..,,,,,,llttgg.....PIAIC....rll"
x = txt.strip(",.grlt")
print(x)
```

```
PIAIC
```

startswith() Method

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

syntax: string.startswith(value, start, end) where value= Required. The value to check if the string starts with start= Optional. An Integer specifying at which position to start the search end= Optional. An Integer specifying at which position to end the search

Check if the string starts with "Hello":

In [26]:
```
txt = "Hello world."
x = txt.startswith("Hello")
print(x)
```

```
True
```

Check if position 7 to 20 starts with the characters "wel":

In [27]:
```
txt = "Hello, welcome to my world."
x = txt.startswith("wel", 7, 20)
print(x)
```

```
True
```

splitlines() Method

The splitlines() method splits a string into a list. The splitting is done at line breaks

syntax: string.splitlines(keeplinebreaks) where keeplinebreaks= Optional. Specifies if the line breaks should be included (True), or not (False). Default value is not (False)

In [28]:
```
txt = "hello world\nWelcome to piaic"
x = txt.splitlines()
print(x)
```

```
['hello world', 'Welcome to piaic']
```

In [29]:
```
#Split the string, but keep the line breaks:

txt = "hello world \nWelcome to piaic"
x = txt.splitlines(True)
print(x)
```

```
['hello world \n', 'Welcome to piaic']
```

split() Method

The split() method splits a string into a list.

You can specify the separator, default separator is any whitespace.

Note: When maxsplit is specified, the list will contain the specified number of elements plus one.

syntax: string.split(separator, maxsplit) where separator= Optional. Specifies the separator to use when splitting the string. By default any whitespace is a separator maxsplit= Optional. Specifies how many splits to do. Default value is -1, which is "all occurrences"

In [30]:
```
txt = "welcome to PIAIC"
x = txt.split()
print(x)
```

```
['welcome', 'to', 'PIAIC']
```

In [31]:
```
#Split the string, using comma, followed by a space, as a separator:

txt = "hello, my name is Arslan, I am 23 years old"
x = txt.split(", ")
print(x)
```

```
['hello', 'my name is Arslan', 'I am 23 years old']
```

In [32]:
```
#Use a hash character as a separator:

txt = "apple#banana#cherry#orange"
x = txt.split("#")
print(x)
```

```
['apple', 'banana', 'cherry', 'orange']
```

In [33]:
```
#Split the string into a list with max 2 items:

txt = "apple#banana#cherry#orange"

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

```
['apple', 'banana#cherry#orange']
```

isupper() Method

The isupper() method returns True if all the characters are in upper case, otherwise False.

Numbers, symbols and spaces are not checked, only alphabet characters.

syntax: string.isupper()

In [34]:
```
#Check if all the characters in the text are in upper case:

txt = "HELLO PIAIC WORLD"
x = txt.isupper()
print(x)
```

```
True
```

In [35]:
```
#Check if all the characters in the texts are in upper case:

a = "Hello World"
b = "hello 123"
c = "MY NAME IS ARSLAN"

print(a.isupper())
print(b.isupper())
print(c.isupper())
```

```
False
False
True
```

islower() Method

The islower() method returns True if all the characters are in lower case, otherwise False.

Numbers, symbols and spaces are not checked, only alphabet characters.

syntax: string.islower()

In [36]:
```
txt = "hello world!"
x = txt.islower()
print(x)
```

```
True
```

In [37]:
```
#Check if all the characters in the texts are in lower case:

a = "Hello world!"
b = "hello 123"
c = "mynameisarslan"

print(a.islower())
print(b.islower())
print(c.islower())
```

```
False
True
True
```

isspace() Method

The isspace() method returns True if all the characters in a string are whitespaces, otherwise False.

syntax: string.isspace()

In [38]:
```
#Check if all the characters in the text are whitespaces:

txt = "   "
x = txt.isspace()
print(x)
```

```
True
```

In [39]:
```
#Check if all the characters in the text are whitespaces:

txt = "   s   "
x = txt.isspace()
print(x)
```

```
False
```

isdigit() Method

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

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

syntax: string.isdigit()

In [40]:
```
#Check if all the characters in the text are digits:

txt = "69096"
x = txt.isdigit()
print(x)
```

```
True
```

In [41]:
```
#Check if all the characters in the text is alphabetic:

a = "\u0030" #unicode for 0
b = "\u00B2" #unicode for ²

print(a.isdigit())
print(b.isdigit())
```

```
True
True
```

replace() Method

The replace() method replaces a specified phrase with another specified phrase.

Note: All occurrences of the specified phrase will be replaced, if nothing else is specified.

syntax: string.replace(oldvalue, newvalue, count)

where oldvalue= Required. The string to search for newvalue= Required. The string to replace the old value with count= Optional. A number specifying how many occurrences of the old value you want to replace. Default is all occurrences

In [42]:
```
#Replace the word "C++":

txt = "I like C++"
x = txt.replace("C++", "Python")
print(x)
```

```
I like Python
```

In [ ]:
```

```