In [1]:

```
var1="hi"
var2="hello"
var3="Bye"
```

1

In [2]:

```
str.capitalize(var1)
```

Out[2]:

```
'Hi'
```

In [3]:

```
help(str.capitalize)
#Converts first character to Capital Letter
```

```
Help on method_descriptor:

capitalize(self, /)
    Return a capitalized version of the string.

    More specifically, make the first character have upper case and the rest
lower
    case.
```

2

In [4]:

```
str.casefold(var2)
```

Out[4]:

```
'hello'
```

In [5]:

```
help(str.casefold)
#converts to case folded strings
```

```
Help on method_descriptor:

casefold(self, /)
    Return a version of the string suitable for caseless comparisons.
```

3

In [6]:

```
str.upper(var1)
```

Out[6]:

```
'HI'
```

4

In [7]:

```
str.lower(var3)
```

Out[7]:

```
'bye'
```

5

In [8]:

```
str.count(var2,"l")
```

Out[8]:

```
2
```

6

In [9]:

```
str.find(var2,"o")
#shows indexing
```

Out[9]:

```
4
```

7

In [10]:

```
str.swapcase(var3)
#interchanges letter caps
```

Out[10]:

```
'bYE'
```

8

In [11]:

```
str.isalnum(var3)
```

Out[11]:

```
True
```

In [12]:

```
help(str.isalnum)
# if value is alphabet or number its true else false
```

```
Help on method_descriptor:

isalnum(self, /)
    Return True if the string is an alpha-numeric string, False otherwise.

    A string is alpha-numeric if all characters in the string are alpha-nume
ric and
    there is at least one character in the string.
```

9

In [13]:

```
str.center(var2,15)
#gives center allignment i.e making inverted comma's further
```

Out[13]:

```
'     hello     '
```

10

In [14]:

```
str.encode(var1)
#returns encoded string of given string
```

Out[14]:

```
b'hi'
```

In [15]:

```
# unicode string
string = 'pythön!'

# print string
print('The string is:', string)

# default encoding to utf-8
string_utf = string.encode()

# print result
print('The encoded version is:', string_utf)
```

```
The string is: pythön!
The encoded version is: b'pyth\xc3\xb6n!'
```

11

In [16]:

```
help(str.endswith)
```

```
Help on method_descriptor:

endswith(...)
    S.endswith(suffix[, start[, end]]) -> bool

    Return True if S ends with the specified suffix, False otherwise.
    With optional start, test S beginning at that position.
    With optional end, stop comparing S at that position.
    suffix can also be a tuple of strings to try.
```

In [17]:

```
text = "Python is easy to learn."

result = text.endswith('to learn')
# returns False
print(result)

result = text.endswith('to learn.')
# returns True
print(result)

result = text.endswith('Python is easy to learn.')
# returns True
print(result)

# Basically we can check the ending
```

```
False
True
True
```

12

In [18]:

```
str = 'xyz\t12345\tabc'

# no argument is passed
# default tabsize is 8
result = str.expandtabs()

print(result)

# to replace /t with spaces
```

```
xyz     12345   abc
```

13

In [19]:

```
# default arguments
print("Hello {}, your balance is {}.".format("Adam", 230.2346))

# positional arguments
print("Hello {0}, your balance is {1}.".format("Adam", 230.2346))

# keyword arguments
print("Hello {name}, your balance is {blc}.".format(name="Adam", blc=230.2346))

# mixed arguments
print("Hello {0}, your balance is {blc}.".format("Adam", blc=230.2346))

#just fits the words in format insode of {}
```

```
Hello Adam, your balance is 230.2346.
Hello Adam, your balance is 230.2346.
Hello Adam, your balance is 230.2346.
Hello Adam, your balance is 230.2346.
```

14

In [20]:

```
var2.index("hello")
#shows index number
```

Out[20]:

```
0
```

15

In [21]:

```
name = "M234onica"
print(name.isalnum())

# contains whitespace
name = "M3onica Gell22er "
print(name.isalnum())

name = "Mo3nicaGell22er"
print(name.isalnum())

name = "133"
print(name.isalnum())
#space made it false so only alphabets and numbers are true
```

```
True
False
True
True
```

16

In [22]:

```
name = "Monica"
print(name.isalpha())

# contains whitespace
name = "Monica Geller"
print(name.isalpha())

# contains number
name = "Mo3nicaGell22er"
print(name.isalpha())

#only alphabest are true
```

```
True
False
False
```

17- not sure what this does looks same like isdigit

In [23]:

```
s = "28212"
print(s.isdecimal())

# contains alphabets
s = "32ladk3"
print(s.isdecimal())

# contains alphabets and spaces
s = "Mo3 nicaG el l22er"
print(s.isdecimal())
#If all strings are decimal characters
```

```
True
False
False
```

18

In [24]:

```
s = "28212"
print(s.isdigit())

# contains alphabets and spaces
s = "Mo3 nicaG el l22er"
print(s.isdigit())
```

```
True
False
```

19-not sure about this either

In [25]:

```
str = 'Python'
print(str.isidentifier())

str = 'Py thon'
print(str.isidentifier())

str = '22Python'
print(str.isidentifier())

str = ''
print(str.isidentifier())

#True if the string is a valid identifier
#False if the string is not a invalid identifier
```

```
True
False
False
False
```

20

In [26]:

```
s = 'Space is a printable'
print(s)
print(s.isprintable())

s = '\nNew Line is printable'
print(s)
print(s.isprintable())

s = ''
print('\nEmpty string printable?', s.isprintable())

#prints all pritable characters and space, \n is not a printable char
```

```
Space is a printable
True

New Line is printable
False

Empty string printable? True
```

In [ ]:

```

```