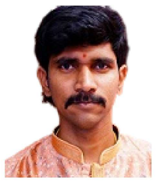


String Methods

str.upper()

- returns a copy of string with all characters in uppercase

In [1]:

```
s="hello world"
s.upper()
```

Out[1]:

```
'HELLO WORLD'
```

str.lower()

- returns a copy of string with all characters in lowercase

In [2]:

```
s= "HELLO WORLD"
s.lower()
```

Out[2]:

```
'hello world'
```

str.swapcase()

- returns a copy of string with all uppercase charaters converted to lower case & viceversa

In [3]:

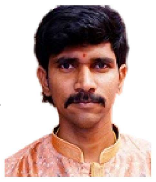
```
s="hEllO 2 WorLD"
s.swapcase()
```

Out[3]:

```
'HeLLo 2 wORld'
```

str.capitalize()

- returns a copy of string with first letter capitilized and remaning all to lowercase



In [4]:

```
s="HeLLo World"
s.capitalize()
```

Out[4]:

```
'Hello world'
```

str.title()

- returns a copy of string with each word capitalized

In [5]:

```
s="gooD moRNing"
s.title()
```

Out[5]:

```
'Good Morning'
```

str.strip()

- returns a copy of the string with leading & trailing whitespace removed

In [6]:

```
s = " Python is easy "
print(s.strip())
```

```
Python is easy
```

str.lstrip()

- returns a copy of the string with leading whitespace removed

In [7]:

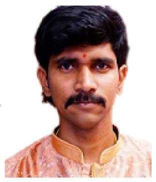
```
s = " Python "
s.lstrip()
```

Out[7]:

```
'Python '
```

str.rstrip()

- returns a copy of the string with trailing whitespace removed



In [8]:

```
s = " Python "
s.rstrip()
```

Out[8]:

```
' Python'
```

str.replace()

- returns a copy of the string with all the occurences of a specified substring replaced with another substring

In [9]:

```
s = "Good Morning..."
print(s)
```

```
Good Morning...
```

In [10]:

```
id(s)
```

Out[10]:

```
2361997608176
```

In [11]:

```
s = s.replace("Good", "Bad")
print(s)
```

```
Bad Morning...
```

In [12]:

```
id(s)
```

Out[12]:

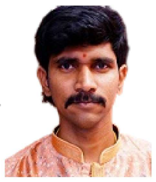
```
2361997597424
```

str.split()

- splits the string into a list of substrings based on the specified delimiter
- if no delimiter is specified, then whitespace is selected as default delimiter

In [13]:

```
st = 'my name ,is SRK'
st.split()
```



Out[13]:

```
['my', 'name', ',is', 'SRK']
```

str.join()

- join a list of strings into a single string, using a specified delimiter

In [14]:

```
s= ['my', 'name', 'is', 'SRK']
''.join(s)
```

Out[14]:

```
'mynameisSRK'
```

str.index()

- returns the index of the first occurence of the string searched for. (raises value error if not found)

In [15]:

```
s="python is very very very easy"
s.index("python")
```

Out[15]:

```
0
```

str.rindex()

- returns the index of the last occurence of the string searched for. (raises value error if not found)

In [16]:

```
s="python is very very very easy"
print(s.rindex("very"))
```

```
20
```

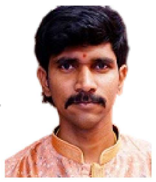
str.find()

- returns the index of the first occurence of the substring in a string
- returns -1 if the substring is not found

In [17]:

```
s="python is very easy"

print(s.find("e"))
print(s.find("b"))
```



```
11
-1
```

str.rfind()

- returns the index of the last occurence of the substring in a string
- returns -1 if the substring is not found

In [18]:

```
s="python is very easy"

print(s.rfind("e"))
print(s.rfind("b"))
```

```
15
-1
```

str.count()

- returns the number of no-overlapping occurences of a substring in the searched string

In [19]:

```
s="Good Morning"
s.count("o")
```

Out[19]:

```
3
```

str.startswith()

- returns a boolean stating whether a string starts with the sprecified prefix

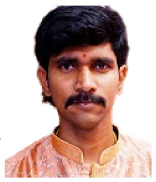
In [20]:

```
my_str='Siva123'
print(my_str.startswith('Sh'))
```

```
False
```

str.endswith()

- returns a boolean stating whether a string ends with the sprecified suffix



In [21]:

```
my_str='Siva123'
print(my_str.endswith("123"))
```

```
True
```

str.isalnum()

- returns a boolean stating whether a string contains only letters & digits

In [22]:

```
my_str='Siva@123'
my_str.isalnum()
```

Out[22]:

```
False
```

str.isalpha()

- returns a boolean stating whether a string contains only letters

In [23]:

```
my_str='Siva'
print(my_str.isalpha())
```

```
True
```

str.isdigit()

- returns a boolean stating whether a string contains only digits

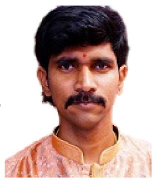
In [24]:

```
my_str='123'
print(my_str.isdigit())
```

```
True
```

str.islower()

- returns a boolean stating whether a string is in lower case



In [25]:

```
my_str='siva123'
print(my_str.islower())
```

```
True
```

str.isupper()

- returns a boolean stating whether a string is in upper case

In [26]:

```
my_str='SIVA 123'
print(my_str.isupper())
```

```
True
```

str.isspace()

- returns a boolean stating whether a string contains only whitespace characters

In [27]:

```
my_str='          '
print(my_str.isspace())
```

```
True
```

str.istitle()

- returns a boolean stating whether a sting in title case

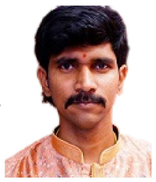
In [28]:

```
my_str='Siva123'
print(my_str.istitle())
```

```
True
```

str.removeprefix()

- returns a string with the given prefix string removed if present.



In [29]:

```
mystring ="Python"
mystring.removeprefix("Py")
```

Out[29]:

```
'thon'
```

str.removesuffix()

- returns a string with the given suffix string removed if present.

In [30]:

```
mystring ="Python"

mystring.removesuffix("on")
```

Out[30]:

```
'Pyth'
```

In [31]:

```
# Strings are immutable. This means that elements of a string cannot be changed once it 
# We can simply reassign different strings to the same name.

myString = "Hello"
myString[4]='s'                 # strings are immutable
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
Cell In[31], line 5
      1 # Strings are immutable. This means that elements of a string cann
ot be changed once it has been assigned.
      2 # We can simply reassign different strings to the same name.
      4 myString = "Hello"
----> 5 myString[4]='s'

TypeError: 'str' object does not support item assignment
```