Working of String methods in Python

In [ ]:
```
#Create a string
st="python Programming"
```

In [ ]:
```
#display the string
print(st)
type(st)
```

In [ ]:
```
#Methods of <class 'str'>
dir(st)
```

Syntax: strobject.MethodName(args)

Every Method creates a new string object and returns it, without modifying the invoking string

In [ ]:
```
#use of capitalize()
st="python Programming"
s1=st.capitalize()     #makes the first charcter capital and rest Lower
print(s1)
print(st)
```

In [ ]:
```
#use of lower()
st="python Programming"
s1=st.lower()   #returns a new string in lowercase
print(s1)
```

In [ ]:
```
#use of upper()
st="python Programming"
s1=st.upper()   #returns a new string in uppercase
print(s1)
```

In [ ]:
```
#use of swapcase()
st="python Programming"
s1=st.swapcase()   #returns a new string in changing the case
print(s1)
```

In [ ]:
```
#use of title()
st="python programming"
s1=st.title()   #returns a new string in title case format
print(s1)
```

In [ ]:

```
#use of count()
st="python programming"
ct=st.count('n',0,6)   #returns an integer value representing the number
print(ct)
```

In [ ]:

```
#use of startswith()
st="python programming"
flag=st.startswith('program',7,20)   #returns True or False
print(flag)
```

In [ ]:

```
#use of endswith()
st="python programming"
flag=st.endswith('on')   #returns True or False
print(flag)
```

In [ ]:

```
#use of find()
st="python programming"
pos=st.find('p',7,13)   #returns the position of first occurrence of sub
print(pos)                #else -1 if not found
```

In [ ]:

```
#use of index()
st="python programming"
pos=st.index('prog')   #returns the position of first occurrence of subs
print(pos)               #else raise an exception "Value Error" if not fo
```

In [ ]:

```
#use of replace()
st="python programming"
s=st.replace('ming','s',1)   #returns a new string by replacing old with
print(s)   #python programs
print(st)
```

In [ ]:

```
#use of center()
st="python programming"
st.center(12,'*')   #returns a padded version of the string with spaces
```

In [ ]:

```
#use of ljust/rjust()
st="python programming"
st.rjust(12,'$')   #returns a padded version of the string with spaces i
```

Usage of isX String Methods

In [ ]:
```
#use of islower()
st="Python programming"
st.islower()   #returns True if all characters are in lowercase else Fal
```

In [ ]:
```
#use of isupper()
st="Python programming"
st.upper().isupper()   #returns True if all characters are in uppercase
```

In [ ]:
```
#use of isalpha()
st="Python@programming"
st.isalpha()   #returns True if all characters are alphabets else False
```

In [ ]:
```
#use of isdigit()
st="123abc"
st.isdigit()   #returns True if all characters are digits else False
```

In [ ]:
```
#use of isalnum()
st="123ABC@"
st.isalnum()   #returns True if all characters are either alphabets or d
```

In [ ]:
```
#use of isspace()
st="    \t\n"
st.isspace()   #returns True if all characters are spaces else False
```

In [ ]:
```
#use of istitle()
st="Python Program"
st.istitle()   #returns True if the string is in title case else False
```

In [ ]:

In [ ]:
```
 #usage of split()
s='python,programming'
ls=s.split(',') #return a list of words splitted based on deleimiter (sp
print(ls)
```

Working of Pyperclip module

```
In [35]: import pyperclip
         #pyperclip.copy('Hello world!')  #copy to clipboard "Hello world!"
         pyperclip.paste()   # Hello world!
```

```
Out[35]: "Python is an interpreted, high-level and general-purpose programming la
         Created by Guido van Rossum and first released in 1991, Python's design
         phy emphasizes code readability with its notable use of significant whit
         Its language constructs and object-oriented approach aim to help program
         ite clear, logical code for small and large-scale projects"
```

Working With Strings

```
In [18]: name='This is Alice's cat'
         print(name)
```

```
  File "<ipython-input-18-380228323f35>", line 1
    name='This is Alice's cat'
                        ^
SyntaxError: invalid syntax
```

Usage of Double Quotes

```
In [20]: name='This is Alice"s cat'
         print(name)
```

```
This is Alice"s cat
```

Usage of Escape Characters

```
In [21]: name='This is Alice\'s cat'   #backward slash
         print(name)
```

```
This is Alice's cat
```

```
In [23]: name='This is Alice\'s \t The cat is black in color \n Thank You'
         print(name)
```

```
This is Alice's          The cat is black in color
 Thank You
```

In [28]:

```python
#Create a string that contains 'That is Carol\'s cat.'
name="That is Carol\\\'s cat."  #raw string
print(name)
```

```
That is Carol\'s cat.
```

MultiLine Strings

In [29]:

```python
#Create a string that contains multiple lines using escape character
name="python \n vsem \n svit"
print(name)
```

```
python
 vsem
 svit
```

In [32]:

```python
#Create a string that contains multiple lines using multiline strings
name='''python's programming \t  cse ise
v sem
svit'''
print(name)
```

```
python's programming      cse ise
v sem
svit
```

MultiLine Comments

In [33]:

```python
'''This is a test Python program.
Written by Al Sweigart al@inventwithpython.com
This program was designed for Python 3, not Python 2.
'''
def spam():
    """This is a multiline comment to help
    explain what the spam() function does."""
    print('Hello!')

spam()
```

```
Hello!
```

In [ ]:

```python

```