4 String manipulation

Strings are used to record the text information such as name. In Python, Strings act as “Sequence” which means Python tracks every element in the String as a sequence. This is one of the important features of the Python language.

For example, Python understands the string "hello' to be a sequence of letters in a specific order which means the indexing technique to grab particular letters (like first letter or the last letter).

As we now know string contains set of characters, let's check how can we manipulate and take subset of string, how can we access charcters out of string and do some manipulation on strings.

Subset of string can be access by using slice operator [] or [:](String index starts with 0 and -1 at the end). Lets look at the example.

Python 3 Alert!

Note that, In Python 3, print is a function and not a statement. So you would print statements like this: print('Hello World')

If you want to use this functionality in Python2, you can import form the **future** module.

Caution: After importing this; you won't be able to choose the print statement method anymore. So pick the right one whichever you prefer depending on your Python installation and continue on with it.

4.2 String operations

Accessing element from string

We know strings are a sequence, which means Python can use indexes to call all the sequence parts. Let's learn how String Indexing works.

• We use brackets `[]` after an object to call its index.

• We should also note that indexing `starts at 0` for Python.

Now, Let's create a new object called s and the walk through a few examples of indexing.

In [2]:

```
string ="this is string manupilation"
```

In [3]:

```
# Fetch first character of a string
print(string[0])
```

```
t
```

In [10]:

```
# Fetch last character of a string
print(string[-1])

#or
l=len(string)

print(string[l-1]) #because indexing stars from zero and length is calculated from 1
```

```
n
n
```

In [12]:

```
# This slice operation will help us to fetch substring from a string.
# [ Index given before colon will be starting index and index given after colon will be e
# it is not considered to print]
#  And if no index is given it will consider till end of string index
print(string[1:])
```

```
his is string manupilation
```

In [13]:

```
# This will give character starting from index 1 and ending index 2 (Last will not consia
print(string[1:3])
```

```
hi
```

Note the above slicing. Here we're telling Python to grab everything from 1 up to 3. It doesn't include the 3rd index. You'll notice this a lot in Python, where statements and are usually in the context of "up to, but not including".

In [14]:

```
# It will give all the characters but not the last three characeters
print(string[:-3])
```

```
this is string manupilat
```

In [15]:

```
#It will give the last two char
print(string[-3:])
```

```
ion
```

In [17]:

```
print(string[2:20])
```

```
is is string manup
```

Index and slice notation is used to grab elements of a sequenec by a specified step size (where in 1 is the default size). For instance we can use two colons in a row and then a number specifying the frequency to grab elements. For example:

In [18]:

```
string[:]
```

Out[18]:

```
'this is string manupilation'
```

In [20]:

```
string[::]
```

Out[20]:

```
'this is string manupilation'
```

In [21]:

```
string[::1]
```

Out[21]:

```
'this is string manupilation'
```

In [23]:

```
string[::2] #here 2 represents the step size
```

Out[23]:

```
'ti ssrn auiain'
```

In [25]:

```
string[::-1] #Here -1  represent steps in reverse order
```

Out[25]:

```
'noitalipunam gnirts si siht'
```

In [26]:

```
string[::-2]
```

Out[26]:

```
'niaiua nrss it'
```

In [32]:

```
string[11:4:-1] #in this characters from 4th to 11th index will be displayed in reverse c
```

Out[32]:

```
'irts si'
```

In [34]:

```
string[11:4:1] #the above is incorrect if -1 is replaced by 1
```

Out[34]:

```
''
```

String Concationation

In [36]:

```
print(string+ " Course")
```

```
this is string manupilation Course
```

Inbuilt String Functions

In [37]:

```
dir(str)
```

Out[37]:

```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getnewargs__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mod__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rmod__',
 '__rmul__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'capitalize',
 'casefold',
 'center',
 'count',
 'encode',
 'endswith',
 'expandtabs',
 'find',
 'format',
 'format_map',
 'index',
 'isalnum',
 'isalpha',
 'isascii',
 'isdecimal',
 'isdigit',
 'isidentifier',
 'islower',
 'isnumeric',
 'isprintable',
 'isspace',
 'istitle',
 'isupper',
 'join',
 'ljust',
 'lower',
 'lstrip',
 'maketrans',
```

In [38]:

```
#Length of a string
len(string)
```

Out[38]:

```
27
```

In [ ]:

In [45]:

```
### Change strings to upper & lower case

# Changes to upper case
print(string.upper()+"\n")

# Changes to lower case
print(string.lower()+"\n")

# Swap case from lower to upper & upper to lower
print(string.swapcase()+"\n")

print(string.title()+"\n")

print(string.capitalize()+"\n")
```

```
 'partition',
 'removeprefix',
 'removesuffix',
 'replace',
 'rfind',
 'rindex',
 'rjust',
 'rpartition',
 'rsplit',
 'rstrip',
 'split',
 'splitlines',
 'startswith',
 'strip',
 'swapcase',
 'title',
 'translate',
 'upper',
 'zfill']
```

```
this is string manupilation

THIS IS STRING MANUPILATION

This Is String Manupilation

This is string manupilation
```

In [ ]:

```

```

In [47]:

```
###Find character index /word in a string

# To find a character in a string, use find and it will give index of that character
print(string.find("n"))

# If it is not able to find character , it will give index as a -1
print(string.find('x'))
```

```
12
-1
```

In [ ]:

```

```

In [49]:

```
### Count characters in a string

# To count no. of characters in a string, can use count method
print(string.count(' '))

print(string.count('a'))
```

```
3
2
```

In [ ]:

```
```

In [53]:

```
### String split operation

# To split string at certain space/character, will return list of strings after splitting
print(string.split(' '))

print(string.split('n'))
```

```
['this', 'is', 'string', 'manupilation']
['this is stri', 'g ma', 'upilatio', '']
```

note --spliting does not include seperation character to include seperation character use **.partition** function

In [52]:

```
print(string.partition('n'))
```

```
('this is stri', 'n', 'g manupilation')
```

note --.**partion** function only seperates from the fist charater it finds only

In [ ]:

```
```

```
### Reverse string
better use [::-1 ] method
```

In [2]:

```
name = "asit"
print(''.join(reversed(name)))
```

```
tisa
```

In [ ]:

```
```

In [61]:

```
### Removing characters from the end of the string
string_1=" this ia string manupilation"

# Strip will remove white space from both end of the strings
string.strip(" ")
```

Out[61]:

```
'this is string manupilation'
```

In [62]:

```
# removes leading character from a string
string_1.lstrip(" ")
```

Out[62]:

```
'this ia string manupilation'
```

In [63]:

```
# removes trailing character from a string
string_1.rstrip(" ")
```

Out[63]:

```
' this ia string manupilation'
```

note strip is only for removing whitespaces from ends

In [ ]:

```

```

In [67]:

```
### Join operation in string

" ".join("hello") # this joins spaces between every character the string

"a".join("Welcome to ineuron")
```

Out[67]:

```
'Waealacaoamaea ataoa aianaeauaraoan'
```

In [ ]:

```

```

In [68]:

```
### Replace string

string_n = "welcome to india"
string_n.replace("to","from")
```

Out[68]:

```
'welcome from india'
```

In [5]:

```
string_2= "thi is a strin"
string_2.replace("thi","this").replace("strin","string")
```

Out[5]:

```
'this is a string'
```

In [ ]:

```
```

In [71]:

```
# Formatting

##The `center()` method allows you to place your string 'centered' between a provided str
string_n.center(20,'z')
```

Out[71]:

```
'hello   hi'
```

In [73]:

```
'hello\thi'.expandtabs() #this expand tabs
```

Out[73]:

```
'hello   hi'
```

In [ ]:

```
```

In [74]:

```
### Checking string case
# Check if string are in upper case
print(string.isupper())

# Check if string are in lower case
print(string.islower())

# Check if string contains space
print(string.isspace())

# Check if string contains digit
print(string.isdigit())

# Check if string endswith character n
print(string.endswith('n'))

# Check if string startswith character n
print(string.startswith('i'))

#check if all char in string are alphanumeric
a = "abcd1234"
print(a.isalnum())

#test if string contains title words
a="Abcdef"
print(a.istitle())
```

```
False
True
False
False
True
False
True
True
```

In [ ]:

```

```

Create to check_palindrome() function

In [94]:

```
palin= "asdfdsa"
def checkpalindrome(palin):
    x=palin
    y=palin[::-1]
    l=len(palin)
    palin_count=0
    for i in range(0,l):
        if x[i]!=y[i]:
            print('Not a Palindrome')
            break
        else: palin_count+=1
    if palin_count==l:
        print("its a palindrome")
```

In [95]:

```
checkpalindrome("asdffdsa")
```

```
its a palindrome
```

In [24]:

```
# We can use index to iterate string reverse direction
string="this is string manupilation"
len(string)

for i in range(len(string)-1,-1,-1):
    print(string[i],end="")

#**Note** -in range the middle -1 os added cuz for loop to stop at 0 we have to put -1.
```

```
noitalipunam gnirts si siht
```

In [32]:

```
#best solution
for i in range(len(string)):
    print(string[len(string)-(i+1)],end="")
```

```
noitalipunam gnirts si siht
```

In [ ]:

```

```