IOTA ACADEMY

Become Job Ready...

Strings

- A string is a sequence of characters.
- Strings are immutable like tuples
- Computers do not deal with characters, they deal with numbers (binary). Even though you may see characters on your screen, internally it is stored and manipulated as a combination of 0's and 1's.
- This conversion of character to a number is called encoding, and the reverse process is decoding. ASCII and Unicode are some of the popular encoding used.
- In Python, string is a sequence of Unicode character. Unicode is modern format.

For more details about unicode

https://docs.python.org/3.3/howto/unicode.html

How to create a string?

Strings can be created by enclosing characters inside a single quote or double quotes.

Even triple quotes can be used in Python but generally used to represent multiline strings and docstrings.

In [13]:
```
myString = 'He doesn\'t \"know\" this'

print(myString)


myString = "Hello"
print(myString)


myString = '''Hello
I am Hemant'''
print(myString)
```

```
He doesn't "know" this
Hello
Hello
I am Hemant
```

How to access characters in a string?

We can access individual characters using indexing and a range of characters using slicing.

Index starts from 0.

Trying to access a character out of index range will raise an IndexError.

The index must be an integer. We can't use float or other types, this will result into TypeError.

Python allows negative indexing for its sequences.

In [1]:
```
myString = "Hello"

# print first Character
print(myString[0])

# print last character using negative indexing
print(myString[-1])

# slicing 2nd to 5th character
print(myString[2:5])
```

```
H
o
llo
```

If we try to access index out of the range or use decimal number, we will get errors.

In [2]:
```
print(myString[15])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[2], line 1
----> 1 print(myString[15])

IndexError: string index out of range
```

In [3]:
```
print(myString[1.5])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[3], line 1
----> 1 print(myString[1.5])

TypeError: string indices must be integers
```

How to change or delete a string ?

Strings are immutable. This means that elements of a string cannot be changed once it has been assigned.

We can simply reassign different strings to the same name.

In [7]:
```
myString = "Hello"
myString[4] = 's' # strings are immutable
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[7], line 2
      1 myString = "Hello"
----> 2 myString[4] = 's'

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

We cannot delete or remove characters from a string. But deleting the string entirely is possible using the keyword del.

In [8]:
```
del myString[0]   # it will give an error
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[8], line 1
----> 1 del myString[0]

TypeError: 'str' object doesn't support item deletion
```

In [9]:
```
del myString # delete complete string
```

In [10]:
```
print(myString)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[10], line 1
----> 1 print(myString)

NameError: name 'myString' is not defined
```

String Operations

Concatenation

Joining of two or more strings into a single one is called concatenation.

The + operator does this in Python. Simply writing two string literals together also concatenates them.

The * operator can be used to repeat the string for a given number of times.

In [11]:
```
s1 = "Hello"
s2 = "Satish"

# concatenation of 2 strings
print(s1 + s2)

print(s1+" "+s2)
#repeat string n times
print(s1 * 3)
```

```
HelloSatish
Hello Satish
HelloHelloHello
```

Iterating Through String

In [12]:
```
count = 0
seq = "Hello WorldL"
for l in seq.lower():
    if l == 'l':
        count = count + 1
print(count, 'letters found')
```

```
4 letters found
```

String Membership Test

In [13]:
```
print('l' in 'Hello World') # in operator to test membership
```

```
True
```

In [14]:
```
print('or' in 'Hello World')
```

```
True
```

String Methods

```
Some of the commonly used methods are lower(), upper(), join(), split(), find(), replace() etc
```

In [35]:
```
print(dir(str))
```

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

lower

Return a copy of the string converted to lowercase.

In [15]:
```
a = "IOTA"
a = a.lower()
print(a)
```

```
iota
```

upper

Return a copy of the string converted to uppercase.

In [16]:
```
b = "iota"
print(b.upper())
```

```
IOTA
```

split

Return a list of the words in the string, using sep as the delimiter string.

In [17]:
```
str1 = "This will split all words in a list"
str1.split()
```

Out[17]:
```
['This', 'will', 'split', 'all', 'words', 'in', 'a', 'list']
```

In [18]:
```
# example

str1 = "This will split all words in a list"

str1.split("i")
```

Out[18]:
```
['Th', 's w', 'll spl', 't all words ', 'n a l', 'st']
```

In [19]:
```
# example
str1 = "hello-how-are*you"

str1.split("-")
```

Out[19]:
```
['hello', 'how', 'are*you']
```

join

The string join() method returns a string by joining all the elements of an iterable (list, string, tuple), separated by the given separator.

In [20]:
```
'-'.join(['This', 'will', 'join', 'all', 'words', 'in', 'a', 'string'])
```

Out[20]:
```
'This-will-join-all-words-in-a-string'
```

In [21]:
```
# example
"*".join(["hello","how","are","you"])
```

Out[21]:
```
'hello*how*are*you'
```

find

The find() method returns the index of first occurrence of the substring (if found). If not found, it returns -1.

In [25]:
```
a = "Good Morning"  # returns the starting index when matched
a.find("Mo")
```

Out[25]:
```
5
```

replace

The replace() method replaces each matching occurrence of a substring with another string.

In [8]:
```
s1 = "Bad morning"

s2 = s1.replace("Bad", "Good")  # strings are immutable so creates a complete new string

print(s1)
print(s2)
```

```
Bad morning
Good morning
```

In [26]:
```
# example

s2 = "iotaacademy"
s2.replace("a","--")
```

Out[26]:
```
'iot----c--demy'
```

In [28]:
```
# example

num = "113,45,67"
num = num.replace(",","") # replace comma with empty string
print(num)
print(int(num))
```

```
1134567
1134567
```

isaplha

The isalpha() method returns True if all characters in the string are alphabets. If not, it returns False.

In [29]:
```
a = "iota"
a.isalpha()
```

Out[29]:
```
True
```

In [30]:
```
a = "iota234"
a.isalpha()
```

Out[30]:
```
False
```

isdigit

The isdigit() method returns True if all characters in a string are digits. If not, it returns False

In [31]:
```
a = "123"
a.isdigit()
```

Out[31]:
```
True
```

In [33]:
```
a = "iota"
a.isdigit()
```

Out[33]:
```
False
```

In [32]:
```
a = "123ddd"
a.isdigit()
```

Out[32]:
```
False
```

That's Great!