Introduction to Strings
Part I

Course Title: Programming Language II
Course Code: CSE 111
Semester: Summer 2020
Lecture - 7

BRAC UNIVERSITY
Inspiring Excellence

Last Lecture

- Introduction
- Different types of functions
  - Built in functions
  - User Defined functions
  - Lambda functions
  - Recursion function
- Why function?

BRAC UNIVERSITY
Inspiring Excellence

Today's Lecture

- Introduction
- Indexing
- Mutability of String
- Basic String operations
  - Concatenation
  - Deletion
  - Repetition
  - Slicing

BRAC UNIVERSITY
Inspiring Excellence

String

- Array of bytes representing Unicode Characters
- Represented using single quotes(ex. 'Hello') or double quotes(ex. "Hello")
- Multiline Strings are represented using triple quotes. For example:

  '''Welcome to Python.
  Today we are going to learn Strings.
  Are you excited?'''
- A single character is also a String. For example:

  'P' or "P" is not a character, it is a String.
- Case sensitive. For example:

  "A" and "a" are two different Strings.
- Space is also a String. For example:

  " " or ' ' is a String
- Empty String ("" or '')

Correct representation:

"Hello there. I am Baymax. "
'Hello there. I am Baymax. '
'Hello'
"H"
'H'

Wrongs representation:

'Hello there. I am Baymax. "
"Hello there. I am Baymax. '
'Hello"
"H'

Cannot use mixture of quotes.

BRAC UNIVERSITY
Inspiring Excellence

Indexing

- All characters in a String are indexed. For example, in string "I am Baymax" the characters are indexed like this:

Characters	I		a	m		B	a	y	m	a	x
Index	0	1	2	3	4	5	6	7	8	9	10
Negative Index	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- Indexing starts at 0
- Characters in a String can be accessed using index
- Index must be an integer
- Space is also a character in the String
- Negative index can also be used
- Accessing index out of range will cause error

Example:

```
s = "I am Baymax"
print(s[0])
print(s[5])
print(s[1])
print(s[-1])
print(s[-9])
print(s[11])
print(s[-12])
print(s[20])
print(s[-40])
```

Output:

```
I
B
 
x
a
IndexError
IndexError
IndexError
IndexError
```

Space

BRAC UNIVERSITY
Inspiring Excellence

Indexing

- All characters in a String are indexed. For example, in string “I am Baymax” the characters are indexed like this:

Characters	I		a	m		B	a	y	m	a	x
Index	0	1	2	3	4	5	6	7	8	9	10
Negative Index	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- How can you print the last character of your String if you don’t the index of it?

Solution 1:

```
s = "I am Baymax"
print(s[-1])
```

← last character will always be at -1

Solution 2:

```
s = "I am Baymax"
print(s[len(s)-1])
```

`len(String)` is a function that returns the length of the string given as argument
Or
we can say **`len(String)`** is a function that returns the number of characters in the String given as argument. For example:

```
s = "I am Baymax"
print(len(s))
```

The output will be 11 as there are 11 characters in String S.

BRAC UNIVERSITY
Inspiring Excellence

Mutability of String

- Strings are **immutable.**
- Once a String is created the characters in it cannot be changed/deleted.
  For example,

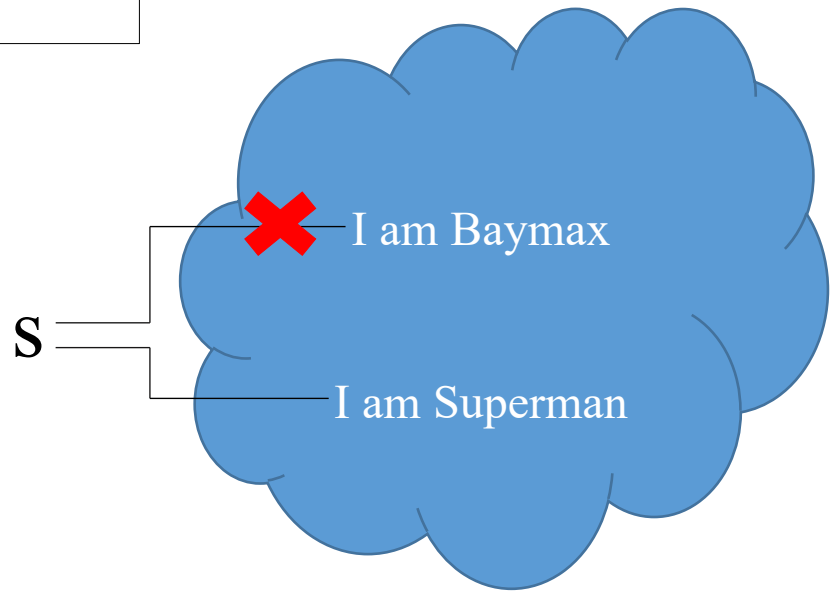
```
s = "I am Baymax"
s[0] = "5"
```

This line will give you an error like this:
TypeError: 'str' object does not support item assignment

- You can change entire String.
  For example,

```
s = "I am Baymax"
print(s)
s = "I am Superman"
print(s)
```

Output:
I am Baymax
I am Superman



BRAC UNIVERSITY
Inspiring Excellence

Iterating over String

- Use loops to iterate over a String character by character
- Using '**while**' loop:
  - Use the loop control variable as index

  For example,

```
index=0
s = "I am Baymax"
while index < len(s):
    print(s[index])
    index+=1
```

Output:

```
I

a
m

B
a
y
m
a
x
```

- Using '**for**' loop
  - No need to maintain indexing

```
s = "I am Baymax"
for char in s:
    print(char)
```

Tip:
Use while loop to iterate over Strings if you need index along with characters in the String; Otherwise use for loop.

BRAC UNIVERSITY
Inspiring Excellence

String Operations (Concatenation)

- '+' operator to concat/merge two Strings.

  For example,

```
s = "Hello there."
s = s + "I am Baymax"
print(s)
```

Output:
Hello there.I am Baymax

No space

- '+' operator does not add space between two Strings automatically.
- We can concat as many Strings as you like with '+' operator.

  For example,

```
s = "Hello"
s = s + " " + "there." + " " + "I am Baymax"
print(s)
```

Output:
Hello there. I am Baymax

BRAC UNIVERSITY
Inspiring Excellence

String Operations (Deletion)

- **'del'** keyword to unbind reference to a String.

```
Example:
s = "I am Baymax"
print(s)
del s
print(s)
```

```
Output:
I am Baymax
NameError: name 's' is not defined
```

- Cannot delete a character from a String (Immutability)

```
Example:
s = "I am Baymax"
del s[0]
```

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

BRAC UNIVERSITY
Inspiring Excellence

String Operations (Repetition)

- '*' operator to repeat a String multiple times.

  For example,

```
s = "I am Baymax"
print(s * 4)
```

Output:
I am BaymaxI am BaymaxI am BaymaxI am Baymax

- Can be used instead of loops.

  For example,

```
for x in range(0,4):
    print("Hi")
```

```
print("Hi\n"*4,end="")
```

Output:
Hi
Hi
Hi
Hi

Both will produce the same result.

BRAC UNIVERSITY
Inspiring Excellence

String Operations (Slicing)

Characters	I		a	m		B	a	y	m	a	x
Index	0	1	2	3	4	5	6	7	8	9	10
Negative Index	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- Syntax:

  String[start: stop: step]

  start: From which index to start, inclusive. If not set, default value is 0

  stop: At which index to stop slicing, exclusive. If not set, default value is last index of the String

  step: Optional. How many step to take. If not set, default value is 1.

For example:

```
s = "I am Baymax"
print(s[2:4])
print(s[:4])
print(s[5:])
print(s[:])
print(s[2:9:2])
print(s[9:5:-1])
```

Output:

am
I am
Baymax
I am Baymax
a am
amya

BRAC UNIVERSITY
Inspiring Excellence

Checking String Membership

- Use '**in**' keyword to find whether a String is present inside another String or not.
- Syntax: String1 **in** String2
  - if String1 is present inside String2, the result will be **True** ; Otherwise the result will be **False**.

For example,

```
s1 = "I am Baymax"
s2 = "am"
print(s2 in s1)
s3 = "Hello"
print(s3 in s2)
```

Output:
True
False

- Used in '**if**' statement as condition.

For example,

```
s1 = "I am Baymax"
s2 = "am"
if s2 in s1:
    print("Found")
else:
    print("Not found ")
```

Output:
Found

BRAC UNIVERSITY
Inspiring Excellence

Summary

- String is a sequence of Unicode characters.
- Characters inside Strings are indexed and can be accessed using index number.
- String is immutable.
- There are several basic operations of String.
  - Concatenation : Merges to Strings
  - Deletion: Unbinds the reference of a String
  - Repetition: Repeats a String several times.
  - Slicing: Creates substring from a String

BRAC UNIVERSITY
Inspiring Excellence

Next Lecture

- Escape sequence
- String formatting
- Functions of String class

BRAC UNIVERSITY
Inspiring Excellence

BRAC UNIVERSITY
Inspiring Excellence