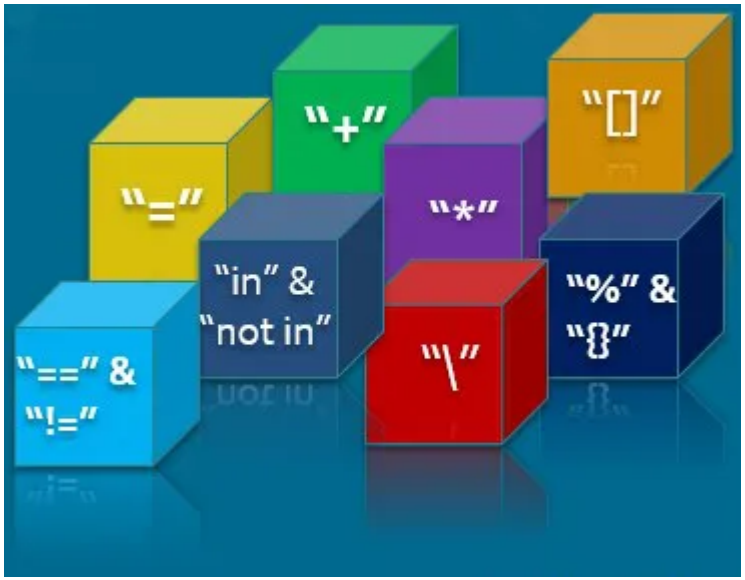
String Operations



Assignment Operator =

We can bind or assign a string to another variable:

In [1]:
```
# Assign string to variable
Name = "Big Data Analytics"
Name
```

Out[1]:
```
'Big Data Analytics'
```

Concatenate Operator +

In [2]:
```
Name1 = "Big Data"
Name2 = " Analytics"
print(Name1)
print(Name2)
```

```
Name = Name1+Name2
print(Name)
```

```
Big Data
 Analytics
Big Data Analytics
```

In [3]:
```
print(Name1 * 4)
```

```
Big DataBig DataBig DataBig Data
```

In [4]:
```
print(Name1)
print(Name2)
```

```
Big Data
 Analytics
```

In [5]:
```
print(Name1, end="")
print(Name2)
```

```
Big Data Analytics
```

In [6]:
```
print(Name1, end="==")
print(Name2)
```

```
Big Data== Analytics
```

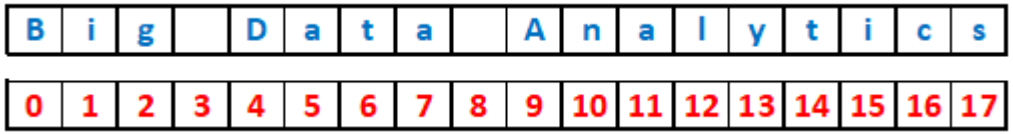
Repetition Operator *

In [7]:
```
print(Name1*4, end="")
print("\b\b")
```

```
Big DataBig DataBig DataBig Data
```

Indexing

It is helpful to think of a string as an ordered sequence. Each element in the sequence can be accessed using an index represented by the array of numbers:



In [8]:
```
Name = "Big Data Analytics"
```

In [9]:
```
print(Name)
```

```
Big Data Analytics
```

In [10]:
```
print(Name[6])
```

```
t
```

The first index can be accessed as follows:

[Tip]: Because indexing starts at 0, it means the first index is on the index 0.

Slicing operator [ ]

In [11]:
```
# Print the first element in the string
print(Name[0])
```

```
B
```

We can access index 6:

In [12]:
```
# Print the element on index 6 in the string
print(Name[6])
```

```
t
```

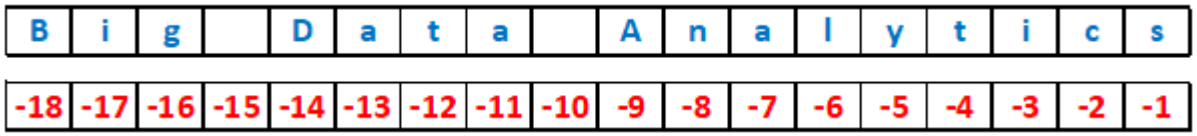
Moreover, we can access the 13th index:

In [13]:
```
# Print the element on the 13th index in the string
print(Name[13])
```

```
y
```

Negative Indexing

We can also use negative indexing with strings:



Negative index can help us to count the element from the end of the string.

In [14]:
```
len(Name)
```

Out[14]:
```
18
```

The last element is given by the index -1:

In [15]:
```
# Print the last element in the string
print(Name[-1])
```

```
s
```

The first element can be obtained by index -17:

In [16]:
```
# Print the first element in the string
print(Name[-17])
```

```
i
```

We can find the number of characters in a string by using `len`, short for length:

In [17]:
```
# Find the length of string
len(Name)
```

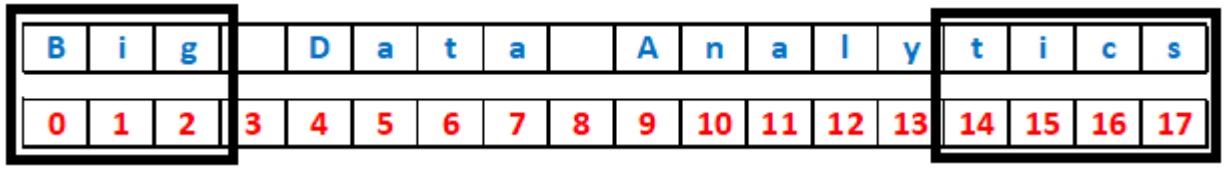
Out[17]: 18

Slicing

$$S[start:stop:step]$$

Start position *End position* *The increment*

We can obtain multiple characters from a string using slicing, we can obtain the 0 to 2nd and 14th to the 17th element:



[Tip]: When taking the slice, the first number means the index (start at 0), and the second number means the length from the index to the last element you want (start at 1)

In [18]:
```
Name[4:8]
```

Out[18]: 'Data'

Default Values --> start = 0 : stop = len(S) : step = 1

In [19]:
```
Name = 'Big Data Analytics'
```

In [20]:
```
Name[0:15:2]
```

Out[20]:
```
'BgDt nlt'
```

In [21]:
```
Name[::]     # Name[0:18:1]
```

Out[21]:
```
'Big Data Analytics'
```

In [22]:
```
Name[0:3]
```

Out[22]:
```
'Big'
```

In [23]:
```
Name[:3]
```

Out[23]:
```
'Big'
```

In [24]:
```
Name[4:]   # 4:18:1
```

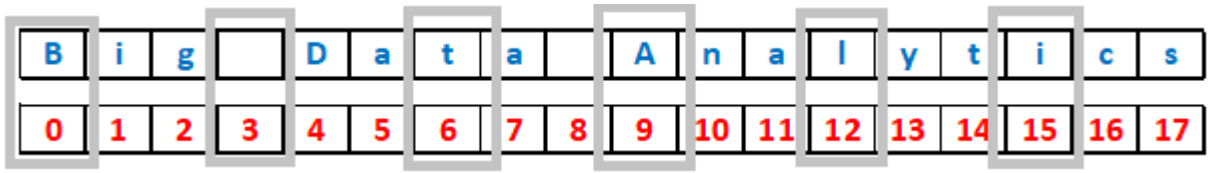
Out[24]:
```
'Data Analytics'
```

In [25]:
```
Name[4::1] # 4:18:1
```

Out[25]:
```
'Data Analytics'
```

Stride

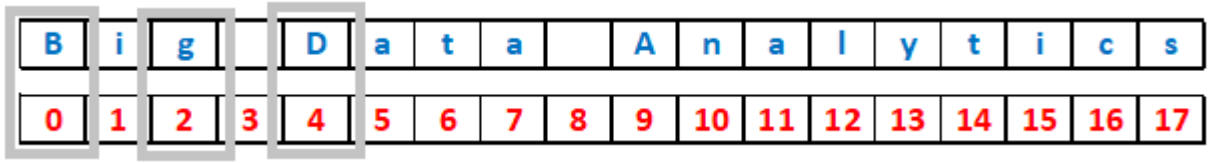
We can also input a stride value as follows, with the '2' indicating that we are selecting every second variable:



In [26]:
```
# Get every second element. The elments on index 1, 3, 5 ...
Name[::2]   # [0:18:2]
```

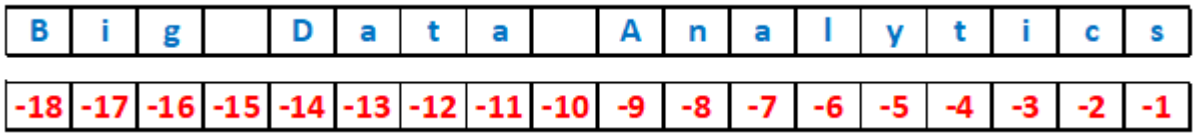
Out[26]:
```
'BgDt nltc'
```

We can also incorporate slicing with the stride. In this case, we select the first five elements and then use the stride:



In [27]:
```
Name[0:5:2]
```

Out[27]:
```
'BgD'
```



Default Values for -ve indexing --> start = -len(S) : stop = -1 : step = 1

In [28]:
```
Name[:-1]
```

Out[28]:
```
'Big Data Analytic'
```

In [29]:
```
Name[-18:-1:1]
```

Out[29]:
```
'Big Data Analytic'
```

In [30]:
```
Name[-18:0:1]  # wrong usage
```

Out[30]:
```
''
```

In [31]:
```
Name[-18::1]
```

Out[31]:
```
'Big Data Analytics'
```

In [32]:
```
Name[::-1]
```

Out[32]:
```
'scitylanA ataD giB'
```

In [33]:
```
Name[::-2]
```

Out[33]:
```
'siyaAaa i'
```

In [34]:
```
Name[0:18:-1]
```

Out[34]:
```
''
```

In [35]:
```
Name[17:0:-1]
```

Out[35]:
```
'scitylanA ataD gi'
```

In [36]:
```
Name[17::-1]
```

Out[36]:
```
'scitylanA ataD giB'
```

In [37]:
```
print(Name[17::-1])
```

```
scitylanA ataD giB
```

String Operations

There are many string operation methods in Python that can be used to manipulate the data. We are going to use some basic string operations

```
String a = new String("now is");
String b = new String("the time");
String c = new String(" the");
```

instance method call	*return type*	*return value*
a.length()	int	6
a.charAt(4)	char	'i'
a.substring(2, 5)	String	"w i"
b.startsWith("the")	boolean	true
a.indexOf("is")	int	4
a.concat(c)	String	"now is the"
b.replace("t","T")	String	"The Tim"
a.split(" ")	String[]	{ "now", "is" }
b.equals(c)	boolean	false

on the data.

Let's try with the method `upper`; this method converts lower case characters to upper case characters:

In [38]:
```
# Convert all the characters in string to upper case
A = "Big Data Analytics"
print("before upper:", A)
```

```
before upper: Big Data Analytics
```

In [39]:
```
A.upper()
```

Out[39]:
```
'BIG DATA ANALYTICS'
```

In [40]:
```
A.lower()
```

Out[40]:
```
'big data analytics'
```

The method `replace` replaces a segment of the string, i.e. a substring with a new string. We input the part of the string we would like to change. The second argument is what we would like to exchange the segment with, and the result is a new string with the segment changed:

In [41]:
```
# Replace the old substring with the new target substring is the segment has been found in the string
B = A.replace('Analytics', 'Tools')
B
```

Out[41]:
```
'Big Data Tools'
```

The method `find` finds a sub-string. The argument is the substring you would like to find, and the output is the first index of the sequence.

Name = "Big Data Analytics"

In [42]:
```
# Find the substring in the string. Only the index of the first elment of substring in string will be the output
Name.find('a')
```

Out[42]:
```
5
```

In [43]:
```
# Find the substring in the string.
Name.find('Data')
```

Out[43]:
```
4
```

If the sub-string is not in the string then the output is a negative one.

In [44]:
```
# If cannot find the substring in the string

Name.find('Tools')
```

Out[44]:
```
-1
```

In [45]:
```
Nm = Name.split(' ')
Nm
```

Out[45]:
```
['Big', 'Data', 'Analytics']
```

In [46]:
```
Nm[0]
```

Out[46]:
```
'Big'
```

In [47]:
```
lines = "Mary had a little lamb Little lamb, little lamb Mary had a little lamb \
Its fleece was white as snow And everywhere that Mary went Mary went, Mary went \
Everywhere that Mary went The lamb was sure to go"
```

In [48]:
```
fnd = input("Enter text to search: ")
idx = lines.find(fnd)
if  idx > -1:
    Res = f"found at index: {idx}"
else:
    Res = "Not found"

print(f"Given text '{fnd}' {Res}")
```

```
Enter text to search: lamb
Given text 'lamb' found at index: 18
```

In [ ]:
```

```