LECTURE

FOUR

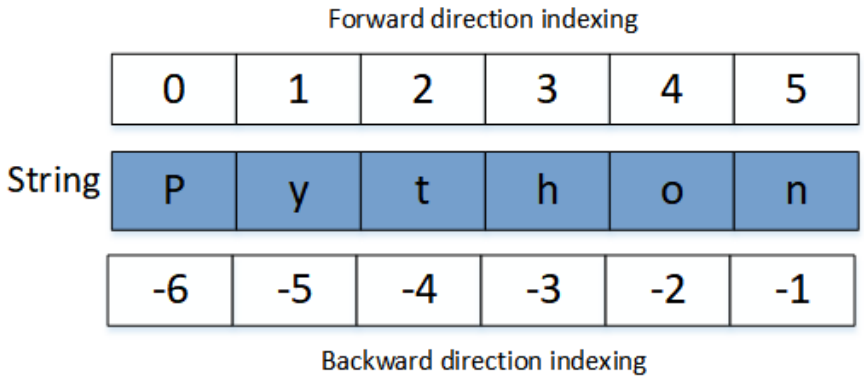
String

- A string is a sequence of characters.
- A character is simply a symbol.
- In Python a string is a predefined object which contains characters.
- Python recognizes both **single** quotes (') and **double** quotes (") as valid ways to determine the boundaries of something. Even **triple** quotes can be used in Python but **generally** used to represent **multiline** strings.
- We can access individual characters using **indexing** and a **range** of characters using **slicing** with the **bracket** operator. **Index** starts from **0**. Trying to access a character out of index range will raise an **IndexError**. The index must be an **integer**. Python allows **negative** indexing for its sequences.
- 'str' object does not support item assignment

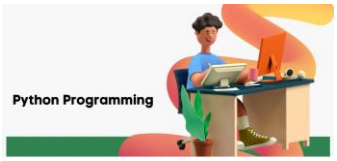
Example: String Quotes

```
x="Iraq"          # double quote
y='Baghdad'       #single quotes
z='" Baghdad is the capital of Iraq and the second-largest city in the Arab world after Cairo"'      # triple quote
```

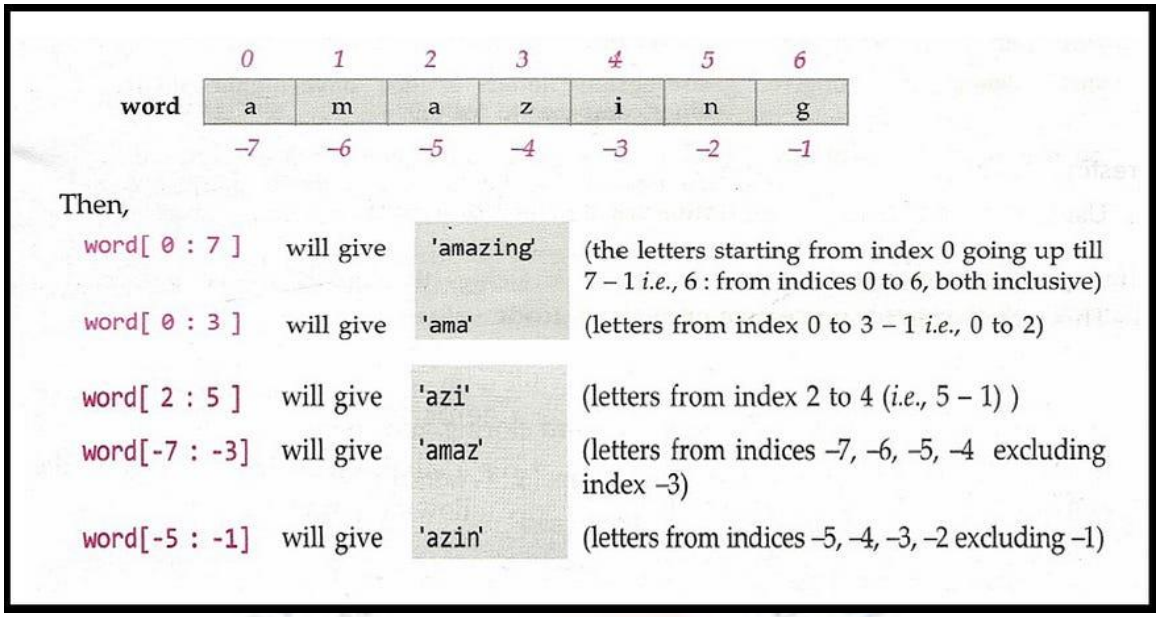
Example: String Indexing




```
m='python'
print(m[0])
print(m[4])
print(m[-2])
```



Example: String Slicing



	0	1	2	3	4	5	6
word	a	m	a	z	i	n	g
	–7	–6	–5	–4	–3	–2	–1

Then,

`word[ 0 : 7 ]`	will give	`'amazing'`	(the letters starting from index 0 going up till 7 – 1 *i.e.*, 6 : from indices 0 to 6, both inclusive)
`word[ 0 : 3 ]`	will give	`'ama'`	(letters from index 0 to 3 – 1 *i.e.*, 0 to 2)
`word[ 2 : 5 ]`	will give	`'azi'`	(letters from index 2 to 4 (*i.e.*, 5 – 1))
`word[-7 : -3]`	will give	`'amaz'`	(letters from indices –7, –6, –5, –4 excluding index –3)
`word[-5 : -1]`	will give	`'azin'`	(letters from indices –5, –4, –3, –2 excluding –1)

Operations and Functions on Strings

Strings have their own set of permissible operations. In general, strings can be:

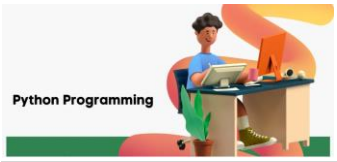
- Concatenated (joined): The **+ operator** does this in Python. Simply writing two string together also concatenates them.
- Replicated: The *** operator** can be used to repeat the string for a given number of times.

Example: Concatenated String and Replicated

```
s1 = 'Hello '
s2 ='World!'
s= s1 + s2
m= s1 * 3
print(s)
print(m)
```

Output

```
Hello World!
Hello Hello Hello
```



❖ ASCII/UNICODE code: If you want to know a specific character's ASCII/UNICODE code point value, you can use a function named **ord()**.

Example: ASCII/UNICODE
c1 = 'a' c2 = 'A' c3= '0' print(ord(c1)) print(ord(c2)) print(ord(c3))

Output

97
65
48

❖ Character code: If you know the code point (number) and want to get the corresponding character, you can use a function named **chr().**

Example: Character code
c1 = 97 c2 = 65 c3= 48 print(chr(c1)) print(chr(c2)) print(chr(c3))

Output

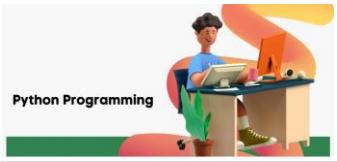
a
A
0

❖ python **len** function: it returns the number of items (length) in an object. use the len() to get the length of the given string.

Example: len
m="python" print(len(m))

Output

6



Common Python String Methods

There are numerous methods available with the string object, some of the commonly used methods are:

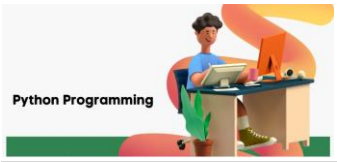
Method	Description
capitalize()	Converts the first character to upper case
upper()	Converts a string into upper case
lower()	Converts a string into lower case
count()	Returns the number of times a specified value occurs in a string
find()	Searches the string for a specified value and returns the position of where it was found
replace()	Returns a string where a specified value is replaced with a specified value
split()	Splits the string at the specified separator, and returns a list

Example: Common Python String Methods

```
x="google Company"
print(x. capitalize())
print(x. upper ())
print(x. lower ())
print(x. count ("o"))
print(x. find ("o"))
print(x. replace ('o','m'))
print(x. split ())
```

Output

```
Google company
GOOGLE COMPANY
google company
3
1
gmmgle cmmpany
['google', 'company']
```



Example (1): Write Python Program to read your name and print each character of your name in line	

name=input(" enter your name= ")
for i in name:
  print(i)
	name=input(" enter your name= ")
	for i in range(len(name)):
	  print(name[i])

Example (2)): Write Python Program to read your name and print each character in reverse order using function called “reverse_name”

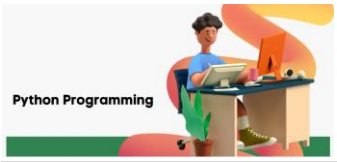
```
def reverse_name(name):
    i = -1
    while i >= -len(name):
        print(name[i])
        i = i - 1
name = input("Enter your name: ")
reverse_name(name)
```

Example (3): Write Python Program to read word and find the count of each character in string using function called “count_character”

```
def count_character(word):
    for i in word:
        print(word.count(i))
s = input("Enter a word: ")
count_character(s)
```

Example (4): Write Python Program to read statement and replace all character (r) with character (m)

```
s=input("enter statement= ")
print(s.replace('r','m'))
```



Example (5): Write Python Program to read statement and print index of vowel characters in statement

```
s=input("enter statement= ")
v="aeiou"
for i in s:
        if i in v:
                print(i,"=",s. find (i))
```

Example (6): Write Python Program to read statement and print count of vowel characters in statement

```
s=input("enter statement= ")
v="aeiou"
c=0
for i in s:
        if i in v:
                c=c+1
print(c)
```

Example (7): Write Python Program to read sentence and print the count of words which end with "tion" using function called “count_tion”

```
def count_tion(sentence):
    words = sentence.split()
    count = 0
    for word in words:
        if word[-4:0]("tion"):
            count += 1
    return count

sentence = input("Enter a sentence: ")
result = count_tion(sentence)
print("Number of words ending with 'tion':", result)
```