


PYTHON PROGRAMMING

STRING OBJECT

String object

- String is a **group of characters** enclosed in single (') or double (")

value = "Hello"

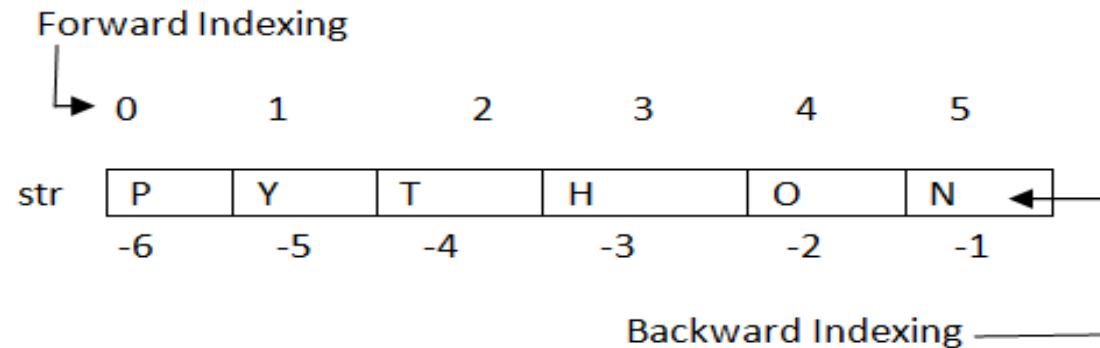


```
PyStringObject {  
    value = "Hello"  
}
```

- String is a **sequence**, i.e., is *an ordered collection of values*
- Strings are **immutable**, i.e. *We cannot change an existing **string***

Accessing Strings

- Strings are stored as individual characters in a **contiguous memory location**
- Strings can be **accessed** from both the directions in **forward and backward** , one character at a time using []. The expression in brackets, known as **Index**



Forward Index:
`str[0]='P', str[1]='Y'`

Backward Index: `str[-1]='N', str[-2]='O' ...`

String – Immutable property

- Once the string is created, you can't change an existing string. i.e. **immutable**,

- Example 1

```
str1="Python Programming"
```

```
str1[7] = 'J'
```

ERROR: 'str' object does not support item assignment

- Example 2

```
str1 = "welcome"
```

```
id(str1)      # 2381521555888
```

```
str2 = "Welcome"
```

```
id(str2)      # 2381521446368
```

```
str2 += " python"
```

```
id(str2)      # 2381521558768
```

String – Immutable property

deleting data

```
str3 = "Python"  
print(str3)
```

```
del str3  
print(str3)
```

Concatenation and Replication operations

- “+” operator : Combines values on either side of the operator
- “*” operator : Concatenates multiple copies of a string to create new strings
also known as **replication** Operator

Example:

```
str1 = "Python "  
str2 = "Programming"  
print(str1+str2)  
print(str1 *3)
```

Membership Operators on String

- **"in"** operator :
return **true** if substring is present in the specified string , else **false**.
- **"not in"** operator:
return **true** if substring does not exist in the specified string, else **false**.

Example:

```
str1="Python Programming"
```

```
"Program" in str1
```

```
"Program" not in str1
```

String Slicing

- Returns part of the string based on expression

Rules for Slicing notation:

1. `[n:m]` returns “nth” character to “mth” character,
2. `[:n]` slice starts at the beginning of the string.
3. `[n:]` slice goes to the end of the string
4. `[:]` returns total string
5. If the first index is \geq to the second index, result is an **empty string**,

String Slicing examples

```
str1="Python Programming"
```

```
str1[0:6]
```

```
str1[7:18]
```

```
str1[:6]
```

```
str1[7:]
```

```
str1[:]
```

```
str1[5:3]
```

String Functions – len()

- len() : returns the number of characters in a string

Ex:-

```
str1="Python Programming"  
print (len(str1))
```

#calculate the length of a string

```
def string_length(str1):  
    count = 0  
    for char in str1:  
        count += 1  
    return count  
print(string_length('python programming'))
```

String Traversal

- Processing string , one character at a time from starting character, select each character in turn, do something to it, and continue until the end, known as **traversal**.

Using While

```
str1="Python Programming"
```

```
len1 = len(str1)
```

```
index = 0
```

```
while (index < len1):
```

```
    letter = str1[index]
```

```
    print(letter)
```

```
    index = index + 1
```

Using for loop

```
str1="Python Programming"
```

```
for l in str1:
```

```
    print(l)
```

String count()

- **count()** returns the number of occurrences of substring sub in the range [start, end].
 - start and end are optional

```
str = "Python is an interpreted language"
```

```
str.count('i')
```

```
str.count('i', 7, 20)
```

Splitting strings

- **split()** : returns a list of all the words in the string, using str as the separator

```
str = "Python is an interpreted language"  
str.split()
```

Note: split(str) where str is separator
if not specified, splits on all whitespace

```
str = "Python,is,an,interpreted,language"  
str.split(',')
```

String join()

- join(iterable) : joins a list of strings using the object calling the string as the separator
 - **iterable** includes List, Tuple, String, Dictionary and Set

```
str1 = "Python is"
```

```
str2 = "a Programming language"
```

```
" ".join([str1,str2])
```

PYTHON PROGRAMMING

STRING OBJECT part 2

Searching for substring using find()

- **find()** method returns the **index of first occurrence** of the **substring**

else returns **-1**

syntax: integer find(sub[, start[, end]])

```
str = "Python is a programming language"
```

```
str.find('i')
```

```
str.rfind('p',10,20)
```

```
str.rfind('i')
```

- **rfind()** method returns highest index

Searching for substring using index()

- `index()` : returns index of a substring.
else raises an exception.

`index(sub[, start[, end]])`

```
str = "Python is a programming language"
```

```
str.index('i')
```

```
str.rindex('p',10,20)
```

```
str.index('z')
```

```
# raises exception
```

Check for char digit alphanumeric upper lower

- **isalpha()**: returns True if all characters in the string are alphabets, else returns False.
- **isdigit()** : returns True if all characters in a string are digits, else returns False.

function which accepts a sentence and finds the number of letters and digits in the sentence

```
s = input("Input a string")
d=l=0
for c in s:
    if c.isdigit():
        d=d+1
    elif c.isalpha():
        l=l+1
    else:
        pass
print("Letters", l)
print("Digits", d)
```

Similarly string has various functions

Islower()

Isupper()

Isalnum()

Replacing char/word

- `replace(oldstr, newstr)` :replaces a substring with an alternative string

```
str = "Python is an interpreted language"
```

```
str.replace('i','I')
```

```
str.replace("an",'a')
```

Removing spaces

- `strip()` : Remove spaces at the beginning and at the end of the string
- `strip(characters)` : Remove the leading and trailing characters

```
str="  welcome to python  "  
print (str.lstrip())
```

```
str1="  welcome to python  "  
print (str.lstrip(" "))
```

String processing using `startswith()` and `endswith()`

- **`startswith()`** : returns **True** if a string starts with the specified prefix(string), else **False**
syn: `startswith(prefix[, start[, end]])`
- **`endswith()`** :returns **True** if a string ends with the specified suffix, else returns **False**.

chr()

- The **chr()** method converts an integer to its unicode character and returns it.
- The **chr()** method returns:
 - a unicode character of the corresponding integer argument (in the range 0 to 1,114,111)
 - ValueError - for an out of range integer number
 - TypeError - for a non-integer argument

print(chr(97)) # a is the unicode character of 97

print(chr(65)) # A is the unicode character of 65

ord()

- **The ord() function returns an integer representing the Unicode character.**
 - The ord() function takes a single parameter:
 - ch - a Unicode character
 - The ord() function returns an integer representing the Unicode character.
- `print(ord('A'))` # 65