PYTHON REVISION TOUR - II

CHAPTER - 2

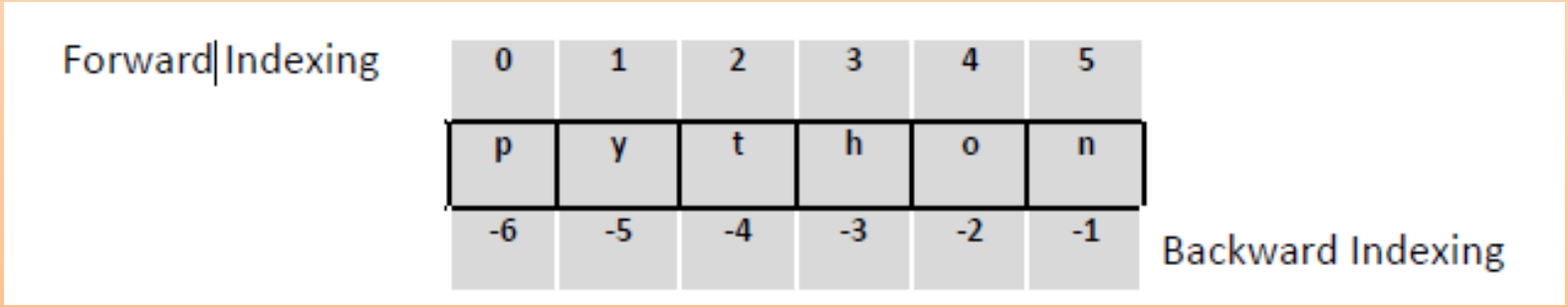
STRINGS

- Any number of valid characters within quotation marks.
- It can hold any type of known characters like alphabets, numbers, special characters etc.

Example: “Ram”, ‘Computer’, “2456”,”%$#@”

- **String as a sequence of Characters**

–String is stored as individual characters in contiguous location with two way indexing



STRING FUNCTIONS

- Length of a string – len()

Example 1:

Name = “Computer”

L = len(Name)

print(“The length of the string is “, l)

Output: The length of the string is 8

Example 2:

Name = "Computer"

Name[4]= 'b'

TypeError: 'str' object does not support item assignment

Strings are immutable

Traversing a string

- Iterating through a element of String , one character at a time.
- Each character are accessible through unique index.

```
str='python'
for ch in str:
    print(ch)
```

```
p
y
t
h
o
n
```

```
for ch in str:
    print(ch,end=' @ ')
```

```
p @ y @ t @ h @ o @ n @
```

```
for i in range(len(str)):
    print(str[i])
```

```
p
y
t
h
o
n
```

String Operators

Basic Operators

+ operator is Concatenation Operator

Example :

```
print("Computer " + "Science")
print('Revision' + ' Tour' + ' 2')
print('3'+ '5')
```

Output

Computer Science
Revision Tour 2
35

* operator is Replication Operator

Example 2:

```
print("Computer " * "Science")
```

TypeError: can't multiply sequence by non-int of type 'str‘

```
print('Revision' * ' 2')
```

TypeError: can't multiply sequence by non-int of type 'str‘

```
print('Revision' *  2)
```

Output: RevisionRevision

Membership Operators

Operator

* in

* not in

- **Working**

Returns true if character /substring occurs in a given string, otherwise false.

Returns false if character /substring occurs in a given string, otherwise true.

```
ch='a'
st1='pan'
st2='panel'
st3='japan'
t1=ch in st1
t2=st1 in st2
t3=st1 not in st3
print(t1,t2,t3, sep='    ')
```

```
True    True    False
```

Comparison Operators

- All relational Operators are comparison operator (<, <=, >, >=, ==,!=).
- •In case of characters Uppercase letters are considered smaller than the lowercase letters.
- •Python compares two strings through relational operators using character by character comparison of their Unicode Values.

Ordinal/ unicode values

```
'comp'<'computer'
```

True

```
'comp'<'Computer'
```

False

```
'COMPUTER'>'computer'
```

False

```
'computer'>'compUter'
```

True

Characters	Ordinal Value
'0' to '9'	48 to 57
'A' to 'Z'	65 to 90
'a' to 'z'	97 to 122

```
ord('4')
```

52

```
ord('B')
```

66

```
ord('d')
```

100

```
chr(52)
```

'4'

```
chr(66)
```

'B'

```
chr(100)
```

'd'

String Slice

- String slice is the part of a String containing some contiguous character from the string.
- For Example ‘or’ , ‘corpor’, ‘tion’, ‘n’ are slice of String ‘corporation’.

0	1	2	3	4	5	6	7	8	9	10
c	o	r	p	o	r	a	t	i	o	n
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

0	1	2	3	4	5	6	7	8	9	10
c	o	r	p	o	r	a	t	i	o	n
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
str='corporation'
str[0:4]
```

```
'corp'
```

```
str[3:6]
```

```
'por'
```

```
str[-5:-2]
```

```
'ati'
```

```
str[:-2]
```

```
'corporati'
```

```
str[2:]
```

```
'rporation'
```

```
str[2:8:2]
```

```
'roa'
```

```
str[::-2]
```

```
'niaorc'
```

```
str[::2]
```

```
'croain'
```

String Function and Method

- Python offers many built-in functions for string manipulation

Function	Functionality	Example

string.capitalize( )	Returns a copy of the string with its first character capitalized in a sentence.	`str1="I Love Icecream"`
		`print(str1.capitalize())`
		`I love icecream`

string.find
(sub, start, end)
	Returns the lowest index in the string where the substring sub is found within the slice range of start and end.
	Returns -1 if sub is not present.
		`str1.find('Ice',5,10)`
		`7`
		`str1.find('tea',5,10)`
		`-1`

Function	Functionality	Example

string.isalnum( )	Returns true if the character in the string are alphanumeric and there is at least one character. Otherwise false.	`st1="ram123"`
		`st2="ram"`
		`st3="123"`
		`st4=' '`
		`st1.isalnum()`
		True
		`st2.isalnum()`
		True
		`st3.isalnum()`
		True
		`st4.isalnum()`
		False

string.isalpha( )	Returns true if all the characters in the string are alphabet and there is at least one character. Otherwise false.	`st1.isalpha()`
		False
		`st2.isalpha()`
		True
		`st3.isalpha()`
		False
		`st4.isalpha()`
		False

string.isdigit( )	Returns true if all the characters in the string are digits and there is at least one character. Otherwise false.	`st1.isdigit()`
		False
		`st2.isdigit()`
		False
		`st3.isdigit()`
		True
		`st4.isdigit()`
		False

Function	Functionality	Example

string.islower( )	Returns true if all the cased characters in the string are lowercase and there is at least one character. Otherwise false.	`st5="COMPUTER"`
		`st6="Computer"`
		`st7="computer"`
		`st5.islower()`
		`False`
		
		`st8="computer123"`
		`st8.islower()`
		`True`
		
		`st6.islower()`
		`False`
		
		`st7.islower()`
		`True`

string.isupper( )	Returns true if all the cased characters in the string are uppercase and there is at least one character. Otherwise false.	`st9="Computer123"`
		`st9.isupper()`
		`False`
		
		`st10="COMPUTER123"`
		`st10.isupper()`
		`True`
		
		`st5.isupper()`
		`True`
		
		`st6.isupper()`
		`False`
		
		`st7.isupper()`
		`False`

Function	Functionality	Example

string.lower( )	Returns a copy of the string converted to lowercase.	`st5="COMPUTER"`
		`st6="Computer"`
		`st7="computer"`
		`st5.lower()`
		`'computer'`
		
		`st6.lower()`
		`'computer'`
		
		`st7.lower()`
		`'computer'`

string.upper( )	Returns a copy of the string converted to uppercase.	`st5.upper()`
		`'COMPUTER'`
		
		`st6.upper()`
		`'COMPUTER'`
		
		`st7.upper()`
		`'COMPUTER'`

string.isspace( )	Returns true if there are only whitespaces characters in the string and there is at least one character. Otherwise false.	`st3="123"`
		`st4=' '`
		`st4.isspace()`
		`True`
		
		`st3.isspace()`
		`False`
		
		`str1="i love icecream"`
		`print(str1.isspace())`
		`False`

Function	Functionality	Example

string.lstrip([chars])	Returns a copy of the string with leading characters removed.
	If used without any argument , it removes the leading whitespaces.
		`str1="the great india place"`
		`str1.lstrip("t")`
		`'he great india place'`
		`str1="the great india place"`
		`str1.lstrip("there")`
		`' great india place'`
		`str1="the great india place"`
		`str1.lstrip("the ground")`
		`'at india place'`

string.rstrip([chars])	Returns a copy of the string with trailing characters removed.
	If used without any argument , it removes the leading whitespaces.
		`str1="the great india place"`
		`str1.rstrip("the ground")`
		`'the great india plac'`
		`str1.rstrip("the placard")`
		`'the great indi'`
		`str1.rstrip("the card")`
		`'the great india pl'`