Python - Strings

Strings are sequence of characters. It can hold zero or more than zero characters in it. String is an object in Python. Python provides many functions to work on with the strings.

Creation

In [ ]:

```
"this is string" # create a string
```

In [ ]:

```
my_str = 'string in single quotes'  # single quotes can be used around characters
```

In [ ]:

```
my_str2 = "string in single quotes"  # double quotes can be used around characters
```

In [ ]:

```
my_str3 = """string is spread
acroos multiple lines"""  # triple quotes can be used to create multi line strings
```

In [ ]:

```
type(my_str)  #use type to check the type of string variable
```

In [ ]:

```
string_with_spaces = " x y 1 2 3 " #string can have spaces in it
```

In [ ]:

```
string_with_special_characters = "#$%@"
```

In [ ]:

```
empty_string = ''  #empty string has nothing in it
```

In [ ]:

```
empty_string
```

In [ ]:

```
len('this is string')
```

In [ ]:

```
len(empty_string)  # len can be use to determine the length of string
```

In [ ]:

```
len(my_str)
```

Concatenation and Repetition

In [ ]:

```
my_str + "  string are useful"
```

In [ ]:

```
my_str + "    " + my_str2 # string concatenation
```

In [ ]:

```
(my_str + " ") * 3 #string repetition
```

In [ ]:

```
'*'*20
```

Indices

String characters has index associated with it starting from zero to len(string) - 1. Individual characters can be accessed by indices. Negative indexing is also avaialble.

In [ ]:

```
my_string = 'this is my string'
```

In [ ]:

```
my_string[0]   # accessing first character of string
```

In [ ]:

```
my_string[1]  # accessing second character of string
```

In [ ]:

```
length = len(my_string)  # accessing last character of string, gives error as index has to
my_string[length]
```

In [ ]:

```
length = len(my_string)
my_string[length - 1]
```

In [ ]:

```
my_string[-1] # accessing last character of string using negative indexing
```

In [ ]:

```
my_string[-2] # accessing second last character of string using negative indexing
```

Slicing

Slice is portion of string accessed using the indices. [] operator allows us to extract part of string using the indices.

In [ ]:

```
my_string = 'this is my string'
```

In [ ]:

```
my_string[ : ]  # extracts complete string
```

In [ ]:

```
my_string[ 0 : 3 ]  # extracts first three characters of string
```

In [ ]:

```
my_string[ 5 : 10 ]  # extracts fifth to ninth character of string
```

In [ ]:

```
my_string[ 3 :  ]  # extract all characters of string from third character onwards
```

In [ ]:

```
my_string[  : 5 ]  # extracts first four characters of string
```

In [ ]:

```
my_string[ 0 : 10 : 2] # from first to nineth character, extract every second character
```

In [ ]:

```
my_string[-2 : ] # extract last two characters of string
```

Escape Sequences

Back slashes represent the beginning of escape sequences. Escape sequences represent strings that may be difficult to input.

In [ ]:

```
print(" Python is beautiful! \n I love Python!" )   # The output is given by a new line
```

In [ ]:

```
print(" Python is beautiful! \t I love Python!" ) # back slash "t" represents a tab
```

In [ ]:

```
print(" Python is beautiful! \\ I love Python!" )  # to place a back slash in your string,
```

The in operator

The in operator is helpful to find out if a string contains some other character or string

In [ ]:

```
my_string = "this is string"
```

In [ ]:

```
'i' in my_string  # check whether i is present in string
```

In [ ]:

```
'i' not in my_string # check whether is is not present in string
```

In [ ]:

```
'this' in my_string
```

Immutability

Strings are immutable i.e. they can not be changed. You can change the individual characters of string.

In [ ]:

```
my_string = "this is my string"
```

In [ ]:

```
my_string[0] = 'p'   # not allowed, as strings are immutable i.e. can not be modified
```

In [ ]:

```
my_string = 'this is new string'
# You can assign some different value to the string variable but its currently assigned val
```

In [ ]:

```
my_string = "this is my string"

first_char = 'p'
my_new_string = first_char + my_string[1:] # concat the part to be changed to the remaining

print("Original string ----> ", my_string)
print("Modofied string ----> ", my_new_string)
```

Looping

One can iterate over the characters present in the string using the for loop.

In [ ]:

```
my_string = "python is beautiful!"
```

In [ ]:

```
for string in my_string:
    print(string)
```

In [ ]:

```
for i in range(len(my_string)):
    print( "(", i, ") character -----> ", my_string[i])
```

Exercise

Q1 Accept string input from user and then print the following with the user entered string - length of string - repeat the string 5 times - first and last characters of the string - first and last three characters of string - string backwards - string with last character removed

In [2]:

```
#Try it here
```

In [ ]:

```

```