Strings

In Python, string is a sequence of Unicode character. Unicode was introduced to include every character in all languages and bring uniformity in encoding.

Create a string
Strings can be created by enclosing characters inside a single quote or double quotes.

Example
```
my_string = 'Hello'
print(my_string)
my_string1 = raw_input("Enter a string")
print(my_string1)
my_string2 = """Hello, welcome to
        the world of Python"""
print(my_string2)
```

Access characters in a string

We can access individual characters using indexing and a range of characters using slicing. Index starts from 0. Trying to access a character out of index range will raise an IndexError. The index must be an integer. We can't use float or other types, this will result into TypeError.Python allows negative indexing for its sequences.

Example
```
str1 = 'program'
print('str = ', str1)
#first character
print('str[0] = ', str1[0])
#last character
print('str[-1] = ', str1[-1])
#slicing 2nd to 5th character
print('str[1:5] = ', str1[1:5])
#Reverse the string
print('str[::-1]=',str1[::-1])
```

Change or delete a string

Strings are immutable. This means that elements of a string cannot be changed once it has been assigned. We can simply reassign different strings to the same name.

We cannot delete or remove characters from a string. But deleting the string entirely is possible using the keyword del.

Example
```
my_string = 'perl'
my_string = 'Python'
print my_string
del my_string
print my_string
```

Concatenation of Two or More Strings

Joining of two or more strings into a single one is called concatenation.The + operator does this in Python.

The ***** operator can be used to repeat the string for a given number of times.

Example
```
str1 = 'Hello'
```

```
str2 ='World!'
# using +
print('str1 + str2 = ', str1 + str2)
# using *
print('str1 * 3 =', str1 * 3)
```

Replace,insert and remove one character from the given string using + operaot and : operator.

Example

```
str1 = 'program'
str1=str1[:3]+str1[4:]#Remove one character Output: proram
print str1
str1=str1[:3]+raw_input("Enter any character")+str1[3:]#insert one character Output: prosram
print str1
str1=str1[:3]+'g'+str1[4:]#replace one character  Output:program
print str1
```

String Formatting Operator

One of Python's coolest features is the string format operator %. This operator is unique to strings and makes up for the pack of having functions from C's printf() family.

%s string conversion via str() prior to formatting

%d signed decimal integer

%f floating point real number

Example

```
print "My name is %s and weight is %d kg!" % ('student', 45)
```

Output

My name is student and weight is 45 kg!

Escape Sequence

If we want to print a text like -He said, "What's there?"- we can neither use single quote or double quotes. This will result into SyntaxError as the text itself contains both single and double quotes.

One way to get around this problem is to use triple quotes. Alternatively, we can use escape sequences.

An escape sequence starts with a backslash and is interpreted differently. If we use single quote to represent a string, all the single quotes inside the string must be escaped. Similar is the case with double quotes.

Example

```
# using triple quotes
print('''He said, "What's there?"''')


# escaping single quotes
print('He said, "What\'s there?"')
```

escaping double quotes

print("He said, \"What's there?\"")

String methods

capitalize: The capitalize() method returns a string with first letter capitalized. It will modify the remaining characters into lower case.

Syntax: string.capitalize()

title: The title() method returns a title cased version of the string. Meaning, the first character of the each word is capitalized (if the first character is a letter).

Syntax:str.title()

lower: The lower() method returns the lowercased string from the given string. It converts all uppercase characters to lowercase.

Syntax:string.lower()

upper: The upper() method returns the uppercased string from the given string. It converts all lowecase characters to uppercase.

Syntax:string.upper()

endswith(): The endswith() method returns *True* if strings ends with the specified suffix. It returns *False* if string doesn't end with the specified suffix.

Syntax:str.endswith(suffix[, start[, end]])

startswith: The startswith() method returns *True* if the string starts with the specified prefix. It returns *False* if the string doesn't start with the specified prefix.

Syntax:str.startswith(prefix[, start[, end]])

isupper(): The isupper() method returns **True** if all characters in a string are uppercase characters.It returns **False** if any characters in a string are lowercase characters

Syntax: string.isupper()

islower(): The islower() method returns *True* if all alphabets that exist in the string are lowercase alphabets. It returns *False* if the string contains at least one uppercase alphabet.

Syntax: string.islower()

isdigit(): The isdigit() returns **True** if all characters in the string are digits. It returns **False** if at least one character is not a digit.

Syntax:string.isdigit()

isalpha(): The isalpha() returns **True** if all characters in the string are alphabets (can be both lowercase and uppercase). It returns **False** if at least one character is not alphabet.

Syntax:string.isalpha()

isaplnum(): The isalnum() returns **True** if all characters in the string are alphanumeric.It returns **False** if at least one character is not alphanumeric

Syntax:string.isalnum()

swapcase(): The swapcase() method returns the string where all uppercase characters are converted to lowercase, and lowercase characters are converted to uppercase.

Syntax:string.swapcase()

len(): The len() function returns the number of items of an object.

Syntax:len(s)

count():count() method returns the number of occurrences of the substring in the given string. count() method only requires a single parameter for execution. However, it also has two optional parameters:

substring - string whose count is to be found.
start (Optional) - starting index within the string where search starts.
end (Optional) - ending index within the string where search ends.

Syntax:string.count(substring, start=..., end=...)

index: If substring exists inside the string, it returns the lowest index in the string where substring is found. If substring doesn't exist inside the string, it raises a ValueError exception. The index() method takes three parameters:

sub- substring to be searched in the string *str*.
start and **end**(optional) - substring is searched within **str[start:end]**

Syntax:str.index(sub[, start[, end]])

find(): The find() method returns an integer value.If substring exists inside the string, it returns the lowest index where substring is found. If substring doesn't exist inside the string, it returns -1. The find() method takes maximum of three parameters:

sub- It's the substring to be searched in the *str* string.

start and **end**(optional) - substring is searched within str[start:end]

Syntax:str.find(sub[, start[, end]])

rfind(): The rfind() method returns an integer value.If substring exists inside the string, it returns the highest index where substring is found. If substring doesn't exist inside the string, it returns -1. The rfind() method takes maximum of three parameters:

sub- It's the substring to be searched in the *str* string.

start and **end**(optional)- substring is searched within str[start:end]

Syntax:str.rfind(sub[, start[, end]])

replace(): The replace() method returns a copy of the string where *old* substring is replaced with the *new* substring. The original string is unchanged.If the *old* substring is not found, it returns the copy of the original string.Replace method have optional parameter called count.If *count* is not specified, replace() method replaces all occurrences of the *old* substring with the *new* substring.

Syntax:str.replace(old, new [, count])

split(): The split() breaks the string at the *separator* and returns a list of strings.The split() method takes maximum of 2 parameters

> **separator** (optional)- The is a delimiter. The string splits at the specified *separator*.
> If the *separator* is not specified, any whitespace (space, newline etc.) string is a separator.
>
> **maxsplit** (optional) - The *maxsplit* defines the maximum number of splits.
> The default value of *maxsplit* is -1, meaning, no limit on the number of splits.

Syntax: str.split([separator [, maxsplit]])

join(): It concatenates each element of an iterable (such as list, string and tuple) to the string and returns a string concatenated with the elements of an iterable.If the iterable contains any non-string values, it raises a **TypeError** exception.

Syntax: string.join(iterable)

Example

```
s="hello students good morning hello"

print s.capitalize()#Hello students good morning hello

print s.title()#Hello Students Good Morning Hello

print s.upper()##HELLO STUDENTS GOOD MORNING HELLO

print s.lower()#hello students good morning hello

print s.startswith("he")#True

print s.endswith("Lo")#False

print s.isupper()#Flase

print s.islower()#True

print s.isdigit()#False

print s.isalpha()#False(Because of spaces)

print s.isalnum()#False

print s.swapcase()#HELLO STUDENTS GOOD MORNING HELLO

print len(s)#33

print s.count('o')#5

print s.count('o',3,7)#1

print s.index('o')#4
```

```
print s.find('o')#4

print s.rfind('o')#32

print s.replace('he', 'He')#Hello students good morning Hello

print s.replace('he', 'He',1)#Hello students good morning hello

list1= s.split()

print list1#['hello', 'students', 'good', 'morning', 'hello']

list2=s.split(" ",2)

print list2#['hello', 'students', 'good morning hello']

sub=" "

string=sub.join(list1)

print string#hello students good morning hello
```

Excercise

1. Draw game boards that look like this

```
--- --- ---
|  |  |  |
 --- --- ---
|  |  |  |
 --- --- ---
|  |  |  |
 --- --- ---
```

2. Program to delete all punctuation from the string provided by the user

3. Consider 2 strings string1 and string2 and display the merged_string as the output. The merged_string should be capital letters from both the strings in the order they appear.
Note: Each character should be checked if it is a capital letter and then it should be merged.
Sample Input:
string1: I Like C
string2: Mary Likes Python
merged_string: ILCMLP

4. Given a string containing both upper and lower case letters. Write a Python program to count the number of repeated characters and display the maximum count of a character along with the character.
Sample Input: ABaBCbGc
Output:
2A
3B (three times B is repeated)

2C
1G

5. C Program to Find the Number of Vowels, Consonants, Digits and White space in a String

6. Write a Python Program to Check Whether a String is Palindrome or Not