Python fundamentals day 17

Today's Agenda

- Strings Comparison
- Programs on strings
- String Operations

Strings Comparison

String comparison in python can be performed in three different ways:

1) Comparing Values of Strings using == operator.
2) Comparing References of Strings using id() function or is operator
3) Comparing String values by ignoring their cases using lower() and upper().

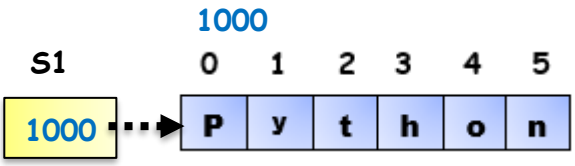
Let us Understand these different ways of comparing strings in detail with the help of codes as shown below:

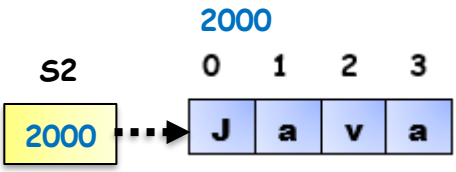
Case I) Comparing Values

Code:

```
s1 = "Python"
s2 = "Java"

if s1 == s2:
    print("String Values are equal")
else:
    print("String Values are Unequal")
```





OUTPUT:

String Values are Unequal

Explanation: In the above code we are comparing two strings using == operator which compares the strings based on their values. The two Strings, Python and Java are not equal and hence we get the output as String values are unequal.

Case II) Comparing References

Code:

```
s1 = "Python"
s2 = "Java"

if id(s1) == id(s2):
    print("String references are equal")
else:
    print("String references are Unequal")
```

OUTPUT:

String references are Unequal

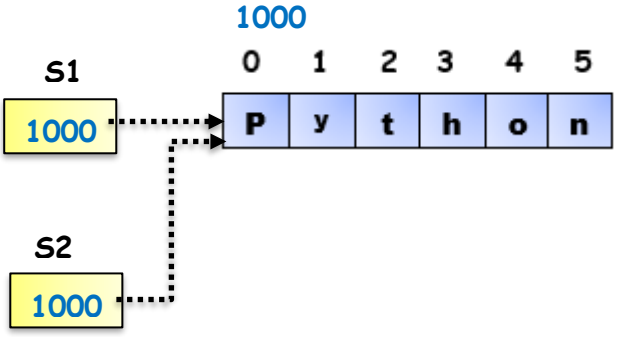
Explanation: In the above code we are comparing two strings References using **id()** function which compares the address of s1 and s2. S1 has address 1000 and s2 has address 2000 and hence we get the output as String references are unequal.

Case III) Comparing two similar string objects.

Code:

```
s1 = "Python"
s2 = "Python"

if s1 == s2:
    print("String Values are equal")
else:
    print("String Values  are Unequal")
```



Output:

```
String Values are equal
```

Explanation: In the above code we are trying to compare two similar string objects. In the first line, when you assign Python to s1, a new string object gets created and address is assigned to it which is then stored in S1. In the second line, you are now trying to create one more string object with the same value as Python.

Will python create one more object with the same value?

In order to answer this, one must remember **Strings in python are immutable,** hence two similar string objects are never created in the memory rather they are shared in the memory.

Thus, the same string object is now shared and its address is copied in s2 which now starts pointing to the same python object. When you now compare the values using == operator, we get the output as string values are equal as there is only one string with two references pointing to it.

Case IV) Comparing two similar string objects using is operator.

Code:

```
s1 = "Python"
s2 = "Python"

if s1 is s2:
    print("String Refrences are equal")
else:
    print("String Refrences  are Unequal")
```

Output:

```
String Refrences are equal
```

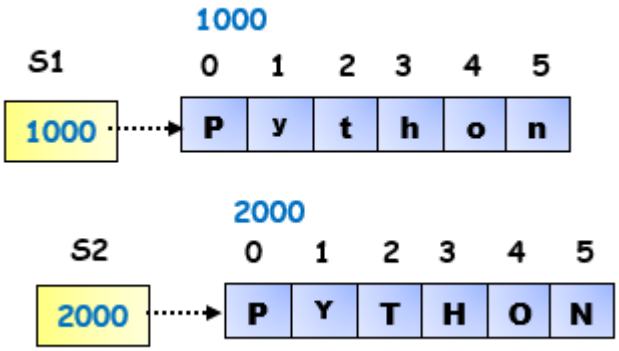
Explanation: In the above code we are trying to compare two string references using is operator. Since s1 and s2 are pointing to the same object, they both have the same address which is 1000 and hence we get the output as String references are equal.

Case V) Comparing two similar string objects with different cases.

Code:

```
s1 = "python"
s2 = "PYTHON"

if s1 == s2:
    print("String Values are equal")
else:
    print("String Values  are Unequal")
```



Output:

String Values are Unequal

Explanation: In the above code we are trying to compare two string Values using == operator which are dissimilar in values as s1 is python in lower case and s2 is PYTHON in upper case and hence we get the output as String values are unequal.

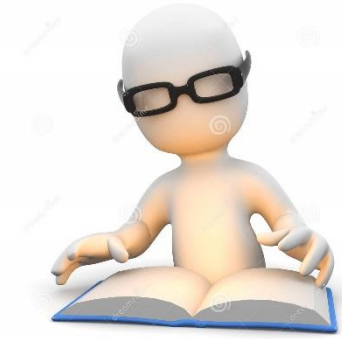
Programs on strings

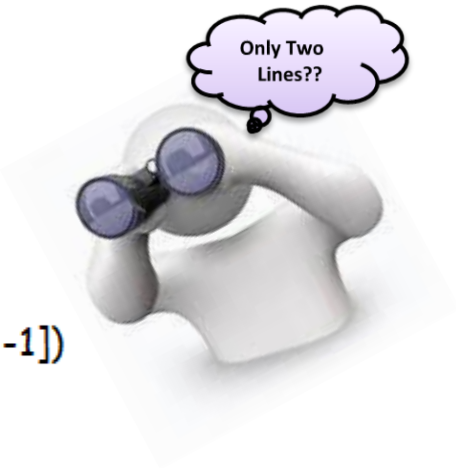
Let us now code some important programs on strings which are commonly asked in many interviews.

1) Write a program to reverse the second half of a string.

Soln: One of the efficient ways of coding the above question is by making use of Slicing Technique as shown below.

STEPS:





Code:

```
s = input("Enter a string\n")
print(s[ : len(s)//2 :] + s[len(s)-1 : (len(s)//2)-1 : -1])
```

Output:

```
Enter a string
PYTHON
PYTNOH
```

String Operations

String operations can be performed by making use of + and * operators.

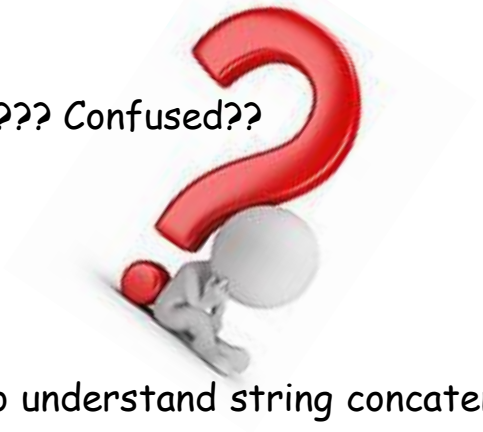
+ → Performs **Concatenation** when used between two strings.

* → Performs **Replication** when used between two strings.

STRING CONCATENATION

+ Operator, When applied between two strings does not perofrm addition instead performs concatenation which simply means it joins the strings and creates a new string.

Didn't get it??? Confused??

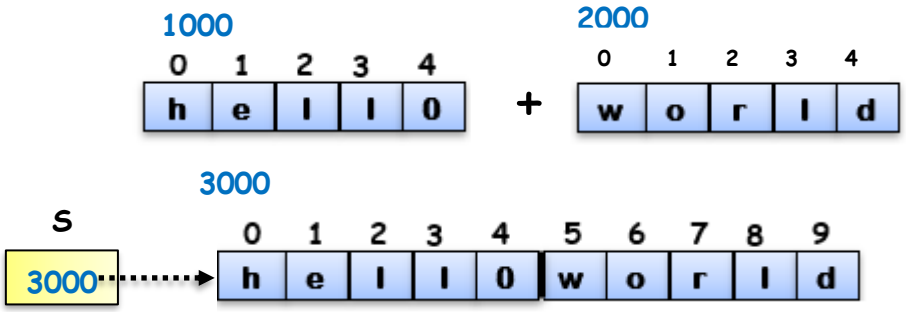


Let us try to understand string concatenatiion with the help of coding examples.

Example1)

Code:

```
s = "hello" + "world"
print(s)
```



Output:

helloworld

Explanation: In the above example, hello is one string object and world is another string object, when + operator is used between these two strings, the string values does not change as strings are immutable, rather a new string object gets created which consists of the concatenated values of hello and world. It is to this string reference s is assigned. Since hello string object and world string object do not have any references pointing to them (Reference count is zero), they are now treated as garbage objects and are collected by garbage collector

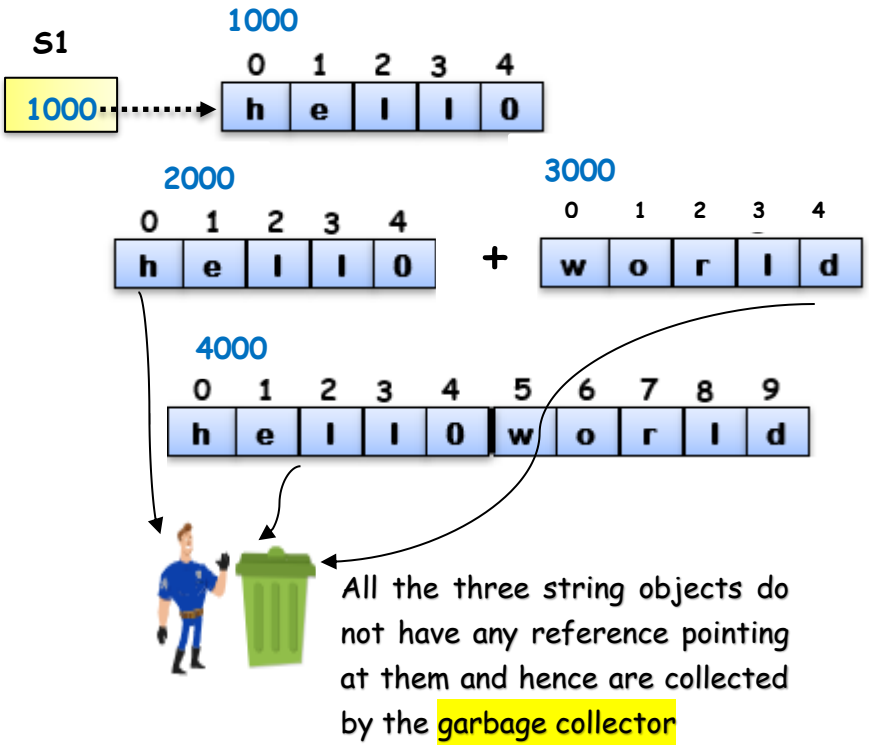
Example2)

Code:

```
s1 = "hello"
print(s1)
s1 + "world"
print(s1)
```

Output:

```
hello
hello
```



Explanation: In the above example, hello is one string object to which s1 is pointing to. In the third line, we are attempting to change the value of an immutable string by concatenating it with another string called "world" and hence a copy of string object "hello" gets created. To this string we are now trying to modify by concatenationg it with world string object. As already disccussed, strings cannot be modified, due to which a new string object helloworld gets created to which there is no refernce assigned and hence it becomes a garbage object. All the string objects with zero refrence count are now collcted by the garbage collector and hence the only object in the memory is hello with s1 as refrence poitinting to it, thus the ouput we got is hello two times.

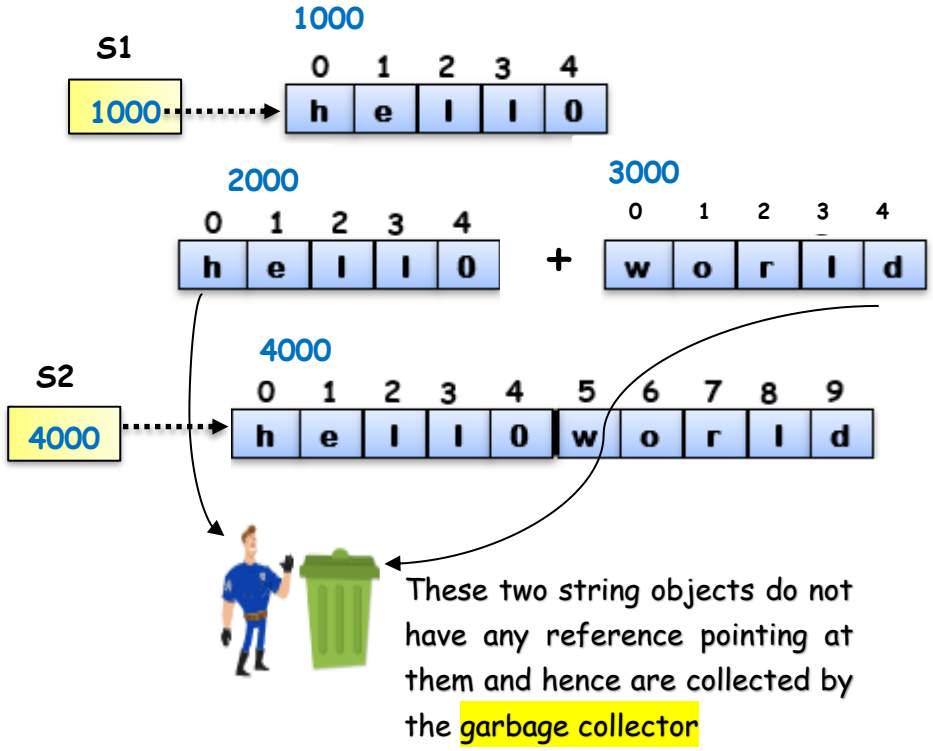
Example3)

Code:

```
s1 = "hello"
print(s1)
s2 = s1 + "world"
print(s1)
print(s2)
```

Output

hello
hello
helloworld



These two string objects do not have any reference pointing at them and hence are collected by the garbage collector

Example4)

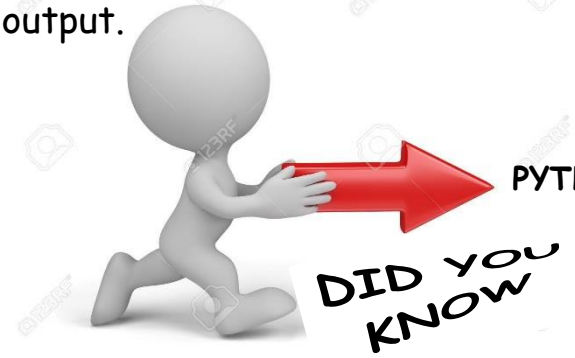
Code:

```
s1 = "hello"
print(s1)
s1 = s1 + "world"
print(s1)
```

Output

hello
helloworld

Try to draw the memory map for the above code and understand the output.



PYTHON IS AN OPEN SOURCE LANGUAGE.