Operators on String

Concatenation :

```
s1 = ‘ hello ‘
s2 = ‘ world ‘
s3 = s1+ s2
s3 = ‘ helloworld ’
```

- It will concatenate the two string and gives the new string . Because string is immutable it will not modify it . It will create a new string

```
s4 = ‘ hello ‘   ‘ world ‘
```

Output : hello world

```
s5 = ‘ hello ‘ + 15        #error
```

- We have to do type casting with string

```
s5 = ‘ hello ‘ + str(15)
```

Output : hello15

Repetition

- It will repeat

```
s1 = ‘ hi ‘
s1 * 3
```

Output : hihihi

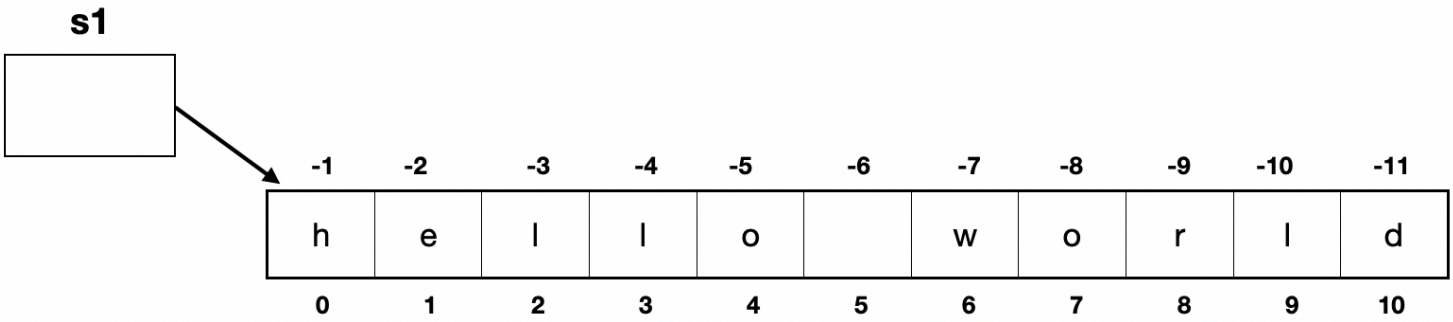
But the thing is it should be integer

- You can have same string multiple times by using multiplication

```
s1 * 2.5
```

Indexing :

```
s1 = ‘ hello world ‘
```



- You can access any character of the string using index also called as substring

```
s1[ 4 ] —— o
s1[ 6 ] —— w
s1[ -5 ] —— o
```

Slicing :

s1 [start : end : step]

- Slicing will work just like loop

s1= ‘hello world ‘
s1[0 : len (s1) : 1]

Output : hello w

```
>>> s1[0:len(s1):1]
'Hello World'
>>> s1[:len(s1):1]
'Hello World'
>>> s1[::1]
'Hello World'
>>> s1[3::]
'lo World'
>>> s1[6::]
'World'
>>> s1[6:8:]
'Wo'
>>> s1[::2]
'HloWrd'
>>> s1[::-1]
'dlroW olleH'
>>> s1[-1:-len(s1)-1:-1]
'dlroW olleH'
>>> s1[-1::-1]
'dlroW olleH'
>>> s1[-1::-2]
'drWolH'
```

in :

It will say if a character is present in the string or not . If it is present then it will return True or else False .

h in s1 —— True
world in s1 —— True
me in s1 —— False

not in :

It will say if it is not present . Then it will return True or else False .

me not in s1 —— True
world not in s1 —— False