Accessing portions of a string

A string is a sequence of characters. Strings are an important data type and commonly used in multiple scenarios. In the set of notebooks that follow, we will discuss various aspects of string manipulation starting with the string concatenation operator.

We often wish to extract portions of a string. In this notebook, we will also study how individual characters in a string can be accessed by using the appropriate subscript.

The first character in a string has an index (subscript) of 0 while the last character has an index that is one less than the total number of characters in the string.

We also take a look at the len() function which returns the number of elements in its argument. When called with a string as its argument, the len() function returns the number of characters in the string.

The + symbol serves as the addition operator when its operands are *numeric* values.

The + symbol serves as the concatenation operator when its operands are *string* values.)

In [1]:

```
'''
In the following lines of code, we accept two integer values from the user, compute the sum and th
en display it.
Here the + symbol functions as an addition operator and adds the two integer values together.
'''

n1 = int(input('Enter integer 1: '))
n2 = int(input('Enter integer 2: '))
print('The sum of', n1, 'and', n2, 'is', n1+n2)
```

```
Enter integer 1: 15
Enter integer 2: 27
The sum of 15 and 27 is 42
```

In [2]:

```
'''
If we  apply the concatenation operator (the + symbol) to string values, the results are very diff
erent.
'''
n1 = input('Enter integer 1: ')
n2 = input('Enter integer 2: ')

print('The concatenation of', n1, 'and', n2, 'is', n1 + n2)
```

```
Enter integer 1: 15
Enter integer 2: 27
The concatenation of 15 and 27 is 1527
```

In [6]:

```
'''
In this code, we ask the user to provide a string as input.  We then use the len() function to fin
d the number of characters
in the string. The code prints out the nth character in the string where n is a number obtained fr
om the user.
'''
my_str = input('Enter a string: ')
str_len = len(my_str) #The len() function gives you the number of characters in the string
argument
print(str_len)
```

```
Enter a string: This is a string
16
```

In [8]:

```
'''
Before concatenating a string with an integer, the integer must be first converted to string.
'''
n1 = int(input('Enter an integer between 0 and ' + str(str_len - 1) + ': ' ))
print(my_str[n1])
```

```
Enter an integer between 0 and 15: 5
i
```

In [9]:

```
'''
As described above, string indices start at 0 and end at 1 less than the number of characters in t
he string.
However, the indices can also be negative values ranging from -1 to -n where n is the number of ch
aracters in the string.
That is, if a string has 10 characters, valid index values are 0 to 9 and -1 to -10.  The last cha
racter in the string
has an index of -1 and the first will have an index of -n

So what happens if we use a negative index?
If the negative index is a valid value as described above, the character associated with the index
will be returned.
Otherwise, we get an error.
'''
my_str = 'This string has 28 chracters'
print(my_str[-27])
print(my_str[-28])
print(my_str[27])
print(my_str[89]) #This will result in an error
#print(my_str[-89]) #This will also result in an error
```

```
h
T
s

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-9-5054028e71f7> in <module>()
     12 print(my_str[-28])
     13 print(my_str[27])
---> 14 print(my_str[89]) #This will result in an error
     15 #print(my_str[-89]) #This will also result in an error

IndexError: string index out of range
```