100 Days Python Challenge

DAY4

Strings:- Indexing, Concatenate String, Escape Sequence, String Operation

Strings

In strings we deal with what are strings, Indexing, Slicing, Concatination, Escape Sequences, String operations

What are Strings?

Strings are any charecter, number or letter which are enclosed within double quotes(" ") are Strings.

In [1]:

```
##the strings can be implemented as follows.

a='ABC'
##using type keyword we can see what is the type of assigned value.

print(type(a))

##below output prints as <Class 'str'> it belongs to str class
### str is short for String.
```

```
<class 'str'>
```

In [10]:

```
a="abc" # notice any change? here we can also use double and single quotes
        #to denote a string.
print(type(a))
```

```
<class 'str'>
```

In [11]:

```
a='1 2 3 4 5 6 7' # the strings can also be integer numbers and spaces

print(type(a))
```

```
<class 'str'>
```

In [6]:

```
a= '@#2_#]&*^%$' # The srings can also be special Character
print(type(a))
```

```
<class 'str'>
```

In [7]:

```
a=" 1 a 2 b " #this is one way or printing the string
print(a)
```

```
 1 a 2 b
```

In [8]:

```
print("1 a 2 b") #this is another way of printing the string
#Note:- We can use both single  and double quotes to represent the string.
```

```
1 a 2 b
```

In [9]:

```
a=" 1 a 2 b " #this is one way or printing the string
a  #we can also print a function without using the print statement
#we just need to enter the assigned variable.
```

Out[9]:

```
' 1 a 2 b '
```

INDEXING

#### “Indexing” means referring to an element of an iterable by its position within the iterable. (or) we can assume it as String is a ordered sequence , each element in the sequence is accessed using an index represented as number

```
Still Not understood? u will know it as we start implementing
```

In [13]:

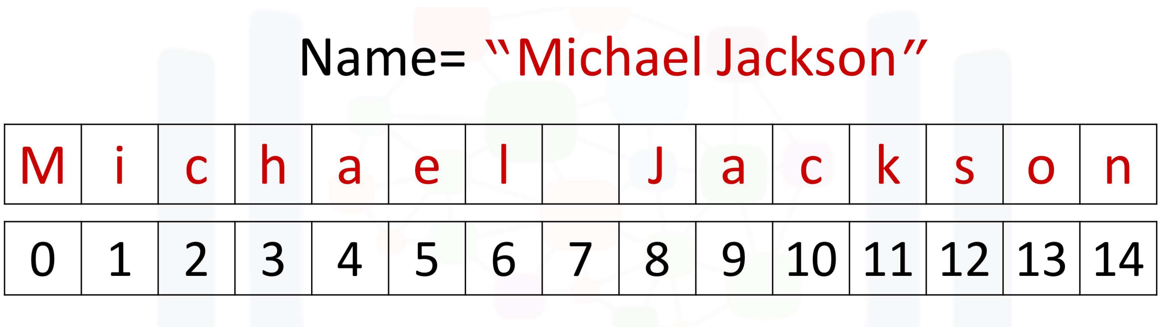
```
##inorder for us to understand the Indexing of string we need
##to create the string.

string="I am superman"
print(string)
```

```
I am superman
```

In [22]:

```
#if we want to print a single element from the above cell string,
#we need to see which position where it is
#in this case in string 'I am superman' i need the letter a,
# the string starts from I so, we need to count from o as it is counted
# as n.....(n-1), other words the count starts from 0 till to the last.
#NOTE:- while indexing it will also consider the spaces
#still not understood?
```



Credits... SKILLS NETWORK

```
in the above cell, the String variable is Name and the assigned string is 'Michel
jackson" so each letter is assigned with a number starting from 0 which is nothing but
the index
```

In [16]:

```
#we willprint the first element in the string which we have implement above
print(string[0])
```

```
I
```

In [18]:

```
#we will print the 7th element in the string which we implement above
print(string[7]) #here from 0th positon till the 7th positon it considered
                 #spaces also as it is the part of the string
```

```
p
```

In [21]:

```
# Print the element on the 13th index in the string

print(string[12])
```

```
n
```

```
Note:- here the each letter in the string we implement above has a position which starts
from 0 for first character and till nth character.
```

Negative Indexing

we can use Negetive indexing with Strings

Negative index can help us to count the element from the end of the string.

using the same "I am superman" string which we implement above we can utilize the same to implement the negetive indexing also

In [24]:

```
# Print the last element in the string

print(string[-1])
```

```
n
```

the first element can be obtained by index-13.

In [27]:

```
# Print the first element in the string

print(string[-13])
```

```
I
```

We can find the length of the strings using 'len' function

In [31]:

```
len(string) #here we use the variable name
```

Out[31]:

```
13
```

Note:- indexing starts indexing from 0 and the len reads from 1

In [32]:

```
len("I am superman") ##this is another way of representing the string.
```

Out[32]:

```
13
```

Slicing

We can obtain multiple characters from a string using slicing, we can obtain starting, middle or end values from the string.(or) “Slicing” means getting a subset of elements from an iterable based on their indices.

From the same string which we implemented above.

In [34]:

```
print(string)
```

```
I am superman
```

In [36]:

```
# Take the slice of variable name with only index 0 to index 3

string[0:4]
```

Out[36]:

```
'I am'
```

In [40]:

```
# Take the slice on variable name with only index 8 to index 11

string[8:14]
```

Out[40]:

```
'erman'
```

Note:- When Slicing the two index values u implement for instance [8:14] the 8th index should be the start of index and n+1 should be your next index

for instance:

In [42]:

```
string="I am superman" #this is the string we implement starts from 0 ends at
                        #ends at 12, so inorder to obtain the 12th element
                        #we need to use the next higher value of
                        # that is
string[0:14] #note:- there are only 13 elements but 14th is required to print
            # the 13th element. Its is something like [0:n+1th element].
```

Out[42]:

```
'I am superman'
```

Stride

We can also input a stride value as follows, with the '2' indicating that we are selecting every second variable:

In [45]:

```
# Get every second element. The elments on index 1, 3, 5 ...

string[::2]
```

Out[45]:

```
'Ia uemn'
```

In [48]:

```
# Get every second element in the range from index 0 to index 4

string[0:5:2]
```

Out[48]:

```
'Ia '
```

Concatenate Strings

combining strings by using the addition symbols, and the result is a new string that is a combination of both:

In [54]:

```
# Concatenate two strings
            #the string right below, we implement already above
new_string = string + ", I am best"
new_string #Remember the + operator while concatinating dosent provide space
```

Out[54]:

```
'I am superman, I am best'
```

In [58]:

```
# Print the string for 3 times

3 * "Hulk " #we need to add space to seperate the character
```

Out[58]:

```
'Hulk Hulk Hulk '
```

In [60]:

```
# Concatenate strings

string = "loki"
string1 = string + " is the best"
string1
```

Out[60]:

```
'loki is the best'
```

Escape Sequences

Back slashes represent the beginning of escape sequences. Escape sequences represent strings that may be difficult to input. For example, back slash "n" represents a new line. The output is given by a new line after the back slash "n" is encountered:

In [62]:

```python
# New line escape sequence

print(" I am \n the best")
```

```
 I am 
 the best
```

In [64]:

```python
# Tab escape sequence

print(" I am \t the best" )
```

```
 I am 	 the best
```

In [66]:

```python
# Include back slash in string

print(" I am \\ the best" )
```

```
 I am \ the best
```

In [69]:

```python
# r will tell python that string will be display as raw string

print(r" VIVEK \ is the best" )
```

```
 VIVEK \ is the best
```

String Operations

There are many string operation methods in Python that can be used to manipulate the data. We are going to use some basic string operations on the data.

In [71]:

```python
# Convert all the characters in string to upper case

a = "i am the best"
print("before upper:", a)
b = a.upper()
print("After upper:", b)
```

```
before upper: i am the best
After upper: I AM THE BEST
```

In [77]:

```
# Replace the old substring with the new target
#substring is the segment has been found in the string

a = "loki is the best" #the given string.
b = a.replace('loki', 'vivek') # this line will help replace loki to vivek.
b
```

Out[77]:

```
'vivek is the best'
```

In [80]:

```
# Find the substring in the string. Only the index of the first elment of substring in stri

name = "Thor Odinson"
name.find('son')
```

Out[80]:

```
9
```

In [82]:

```
name.find('Odin')
```

Out[82]:

```
5
```

If the sub-string is not in the string then the output is a negative one. For example, the string 'Jasdfasdasdf' is not a substring:

In [83]:

```
#Like this
name.find("i am vivek")
```

Out[83]:

```
-1
```