1- Strings

Strings in python are surrounded by either single quotation marks, or double quotation marks.

1.1- Creating a string

In [1]:
```
Uzair_string = "Welcome to Pharmacy!"
Uzair_string2 = 'The bright red fox jumped the fence.'
a_long_string = '''This is a
multi-line string.
It covers more than one line'''
##Above strings are Single quote Double quotes Triple quotes
```

In [3]:
```
Alpha_String = 'The word "Pharmacy" usually refers to a medicine'
triple_String = """Here's another way to embed "quotes" in a string"""
```

In [5]:
```
# String Casting means convert integer into string
my_number = 123
my_string = str(my_number)
my_string
```

Out[5]:
```
'123'
```

1.2- String Concatenation

In [7]:
```
# Concatenation means combining two things together.
string_1="Uzair is a good"
string_2=" data analyst"
string_3=string_1 + string_2
print (string_3)
```

```
Uzair is a good data analyst
```

1.3- String Methods

In [11]:
```
Meri_string = "This is a good string"
Meri_string
```

Out[11]:
```
'This is a good string'
```

In [12]:
```
# Strings in upper and lower case
Meri_string.upper()
```

Out[12]:
```
'THIS IS A GOOD STRING'
```

In [13]:
```
Meri_string.lower()
```

Out[13]:
```
'this is a good string'
```

In [16]:
```
# asking help in python
help(Meri_string.upper)
```

```
Help on built-in function upper:

upper() method of builtins.str instance
    Return a copy of the string converted to uppercase.
```

Introspection is an ability to determine the type of an object at runtime

In [17]:
```
type(Meri_string)
```

Out[17]:
```
str
```

1.4- String Slicing

String slicing means taking out a part of a string. The characters in the extracted part may be continuous or they may be present at regular intervals in the original string

In [19]:
```
Pak_string = "My Country Pakistan"
Pak_string[0:0]
```

Out[19]:
```
''
```

In [20]:
```
Pak_string = "My Country Pakistan"
Pak_string[0:-3]
```

Out[20]:
```
'My Country Pakis'
```

In [21]:
```
# Slicing down to a single character
print(Pak_string[0])
```

```
M
```

String Formatting AKA substitution

String formatting uses a process of string interpolation (variable substitution) to evaluate a string literal containing one or more placeholders, yielding a result in which the placeholders are replaced with their corresponding values. However you will also find yourself inserting integers and floats into strings quite often as well

In [22]:
```
my_string = "I like %s" % "rana"
my_string
```

Out[22]:
```
'I like rana'
```

In [23]:
```
# Examples with integers
int_string = "%i + %i = %i" % (3,3,6)
int_string
```

Out[23]:
```
'3 + 3 = 6'
```

In [24]:
```
# Examples with float
float_string = "%f" % (1.23)
float_string
```

Out[24]:
```
'1.230000'
```

In [25]:
```
# another float example
float_string2 = "%.2f" % (1.23)
float_string2
```

Out[25]:
```
'1.23'
```

In [26]:
```
# example
print("%(x)i + %(y)i = %(z)i" % {"x":1, "y":2, "z":3})
```

```
1 + 2 = 3
```

In [27]:
```
"Python is as simple as {0}, {1}, {2}".format("u", "v", "w")
```

Out[27]:
```
'Python is as simple as u, v, w'
```

In [29]:
```
xy = {"x":0, "y":10}
print("Graph a point at where x={x} and y={y}".format(**xy))
```

```
Graph a point at where x=0 and y=10
```

In [ ]:
```

```