 # Strings

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

```
student1 = "santosh"

student2 = 'akshay'

print(student1)

print(student2)
```

 # Quotes Inside Quotes

 # You can use quotes inside a string, as long as they don't match the quotes surrounding the string:

```
print("It's alright")

print("He is called 'Johnny'")

print('He is called "Johnny"')
```

 # Multiline Strings

You can assign a multiline string to a variable by using three quotes:

```
a = """Lorem ipsum dolor sit amet,

consectetur adipiscing elit,

sed do eiusmod tempor incididunt

ut labore et dolore magna aliqua."""

print(a)
```

 # Or three single quotes:

```
a = '''Lorem ipsum dolor sit amet,consectetur adipiscing elit,

sed do eiusmod tempor incididunt

ut labore et dolore magna aliqua.'''

print(a)
```

Note: in the result, the line breaks are inserted at the same position as in the code.

 # Strings are Arrays

 # In Python, a string is a group (or sequence) of characters. Each character in the string has a position called an index, and it starts from 0.

 # Python does not have a separate character data type. If you want a single character, it's just a string with length 1

 # Get the character at position 1 (remember that the first character has the position 0):

```
a = "Hello, World!"
print(a[1])
```

 # Looping Through a String

 # Since strings are arrays, we can loop through the characters in a string, with a for loop.

```
for x in "banana":
  print(x)
```

 # String Length

 # To get the length of a string, use the len() function.

```
a = "Hello,  World!"
print(len(a))
```

 # Check String

 # To check if a certain phrase or character is present in a string, we can use the keyword in.

```
txt = "The best things in life are  free!"

print("free" in txt)
```

 # Use it in an if statement:

txt = "The best things in life are free!"

```
if "free" in txt:
  print("Yes, 'free' is present.")
```

 # Check if NOT

 # To check if a certain phrase or character is NOT present in a string, we can use the keyword not in.

```
txt = "The best things in life are free!"

print("expensive" not in txt)
```

 # Use it in an if statement:

```
txt = "The best things in life are free!"

if "expensive" not in txt:
  print("No, 'expensive' is NOT present.")
```

 # Slicing

 # You can return a range of characters by using the slice syntax.

```
b = "Hello, World!"

print(b[2:5])
```

 # Slice From the Start

 # By leaving out the start index, the range will start at the first character:

```
b = "Hello, World!"
print(b[:5])
```

 # Slice To the End

 # By leaving out the end index, the range will go to the end:

```
b = "Hello, World!"
print(b[2:])
```

Negative Indexing

Use negative indexes to start the slice from the end of the string:

```
b = "Hello, World!"
print(b[-5:-2])
```

 # Python has a set of built-in methods that you can use on strings.

```
a = "Hello, World!"
print(a.upper())
```

 # Lower Case

#The lower() method returns the string in lower case:

```
a = "Hello, World!"
print(a.lower())
```

R**emove Whitespace**

Whitespace is the space before and/or after the actual text, and very often you want to remove this space.

```
a = " Hello, World! "
print(a.strip())
```

 # Now the spaces at the start and end are gone. Only the middle space between "Hello," and "World!" remains (because strip() does not remove spaces between words).

 # Replace String

The replace() method replaces a string with another string:

```
a = "Hello, World!"

print(a.replace("H", "J"))
```

Split String

The split() method returns a list where the text between the specified separator becomes the list items.

```
a = "Hello, World!"

print(a.split(","))
```

 # String Concatenation

To concatenate, or combine, two strings you can use the + operator.

```
a = "Hello"

b = "World"

c = a + b

print(c)
```

 # To add a space between them, add a " ":

```
a = "Hello"

b = "World"

c = a + "   " + b

print(c)
```

 # Python - Format - Strings

 # F-Strings (formatted string literals) are a way to embed variables or expressions directly inside a string using curly braces {}.

 # They were introduced in Python 3.6 and are now the preferred and fastest way to format strings.

```
name = "Akshay"

age = 19

print(f"My name is {name} and I am {age} years old.")
```

A placeholder can contain Python code, like math operations:

```
txt = f"The price is {20 * 5} dollars"

print(txt)
```

 # Escape Character

 # To insert characters that are illegal in a string, use an escape character.

 # An escape character is a backslash \ followed by the character you want to insert.

```
txt = "We are the so-called \"Vikings\" from the north."

print(txt)
```