Strings in Python

Single quote ' ' or double quote " " use for single line string

In [3]:
```
print('Happy')  # single quote ' '
print("Coding") # double quote " "
```

```
Happy
Coding
```

triple quote use for multiple line strings

In [1]:
```
print('''Stay
    positive
        and keep
            smiling''')
```

```
Stay
    positive
        and keep
            smiling
```

Indexing of string

In [8]:
```
a="Every day is a new day"
print(a[2])
print(a[-3])
```

```
e
d
```

Slicing of string

It extract a portion (substring) of a string by specifying its starting index and ending index.

[start:Stop:Step]

```
Start - starting Index (inclusive)
Stop - ending Index (exclusive)
Step (optional) - skips the step to next index. interval between chara
cters to be included in the slice.
```

In [69]:
```
r="Good programmer with great habits"

print(r[5:16]) # Slicing from index 5 to 16 (excluding 16)
print(r[21:])  # Slicing from index 21 to the end of the string
print(r[:15])  # Slicing from the beginning to index 15 (excluding 15)
print(r[:])    # Slicing the entire string
```

```
programmer
great habits
Good programmer
Good programmer with great habits
```

In [32]:
```
x = 'ABCDEF'
print(x[::1])
print(x[::-1]) # Slicing in reverse order. from the right end of the string
print(x[::-2]) # reverse order and skips 1 step.
print(x[::2]) # Get every other character from 0 to the end. step of 2.
```

```
ABCDEF
FEDCBA
FDB
ACE
```

length function

In [10]:
```
x = 'I am mastering the Python Language'
len(x)
```

Out[10]: 34

String Methods

upper method

In [18]:
```
a = "I am A data Analyst"
a.upper()
```

Out[18]: 'I AM A DATA ANALYST'

lower method

In [32]:
```
a.lower()
```

Out[32]: 'i am a data analyst'

replace method

In [33]:
```
# String is immutable but word or letter can be replaced using this function.
a.replace("Analyst", "Scientist")
```

Out[33]:
```
'I am A data Scientist'
```

find method

In [42]:
```
# it returns index position
a.find("a")
```

Out[42]:
```
2
```

split method

In [27]:
```
# Remove spaces at the beginning and at the end of the string
b = "   apple   "
b.strip()
```

Out[27]:
```
'apple'
```

Count Method

In [31]:
```
list1 = [1, 4, 2, 9, 7, 8, 9, 3, 9]
list1.count(9)
```

Out[31]:
```
3
```

String Concatenation

In [7]:
```
x = "abc"
y = 'ABC'
full_name = x + y
print(full_name)
```

```
abcABC
```

Formatting

Two ways:

1. format() method
2. using % operator

format() method

In [13]:
```
# Add placeholders curly brackets { }
price = 51
"The price is {} dollars".format(price)
```

Out[13]: 'The price is 51 dollars'

In [14]:
```
"The price is {} dollars".format(20)
```

Out[14]: 'The price is 20 dollars'

Decimal value in format()

In [15]:
```
# use paramenters in placeholder {}

"The price is {:.2f} dollars".format(51.23)
```

Out[15]: 'The price is 51.23 dollars'

multiple input value

In [17]:
```
"Numbers are {}, {}, {}".format(20,30,40)
```

Out[17]: 'Numbers are 20, 30, 40'

Index numbers in placeholder

In [18]:
```
"Numbers are {0}, {1}, {2}, {1}".format(15,25,35) # here 15 is 0 index, 25 is i
```

Out[18]: 'Numbers are 15, 25, 35, 25'

Named Indexes in format()

In [61]:
```
"Numbers are {num1}, {num2}, {num3}".format(num1 = 20,num2 = 30,num3 = 40)
```

Out[61]: 'Numbers are 20, 30, 40'

using f-strings

f prefix, and expressions inside curly braces { } are evaluated and replaced with their values

In [19]:
```
num1 = 20
num2 = 30
num3 = 40

f"Numbers are {num1}, {num2}, {num3}"
```

Out[19]:
```
'Numbers are 20, 30, 40'
```

Using % operator

1. %s use for string

2. %d use for numbers

In [26]:
```
name = "Alok"
age = 30
"Hello, my name is %s and I am %d years old." %(name, age)
```

Out[26]:
```
'Hello, my name is Alok and I am 30 years old.'
```

%d round off the decimal and rational numbers

In [54]:
```
frac_num = 8/3
"%d is equal to 8/3. Actual value is 2.66" % frac_num
```

Out[54]:
```
'2 is equal to 8/3. Actual value is 2.66'
```

In [55]:
```
dec_num = 10.9785
"%d is equal to 10.9785 using this operator" %dec_num
```

Out[55]:
```
'10 is equal to 10.9785 using this operator'
```

The %s automatically converts a numeric value to a string without throwing an error.

In [27]:
```
"Hello, my name is %s and I am %s years old." %(name, age)
```

Out[27]:
```
'Hello, my name is Alok and I am 30 years old.'
```

note : The %d, however, can only be used for numeric values. Otherwise, an error is returned.

String Alignment With Formating

In [45]:
```
# Left <, center^ and right > alignment

String1 = "|{:<15}|{:^30}|{:>10}|".format('Code','Looks','Good')
String1
```

Out[45]:
```
'|Code           |            Looks             |      Good|'
```