Strings

June 29, 2022

1 Strings

1.0.1 Creating a String with single Quotes

[1]:
```
String1 = 'Welcome to the Programmers Hub'
print(String1)
```

```
Welcome to the Programmers Hub
```

1.0.2 Creating a String with double Quotes

[2]:
```
String2 = "I'm a Programmer"
print(String2)
```

```
I'm a Programmer
```

1.0.3 Creating a String with triple Quotes

[3]:
```
String3 = '''I'm a Programmer and I live in a world of "Python"'''
print(String3)
```

```
I'm a Programmer and I live in a world of "Python"
```

1.0.4 Creating String with triple Quotes allows multiple lines

[4]:
```
String4 = '''Programmers
    For
        Life'''
print(String4)
```

```
Programmers
    For
        Life
```

1.0.5 Use Quotes inside Python String

[5]:
```
print("Dogs are "love"")
```

```
  File "<ipython-input-5-c670e984dab2>", line 1
    print("Dogs are "love"")
```

```
SyntaxError: invalid syntax
```

[10]:
```
print('Dogs are "love" ')
```

```
Dogs are "love"
```

[11]:
```
print("Dogs are 'love'")
```

```
Dogs are 'love'
```

1.0.6 String Conversions

[12]:
```
print(chr(102)) #ASIIC
```

```
f
```

[13]:
```
print(ord('b'))
```

```
98
```

[14]:
```
a=str(20)
type(a)
```

[14]:
```
str
```

[15]:
```
a=20
a=str(a)

c="20"
c=int(c)


print(a)
print(c)
```

```
20
20
```

[16]:
```
#Cannot Covert Str words to integer
b="HSAI"
b=int(b)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-16-20af6ee13e9d> in <module>
      1 #Cannot Covert Str words to integer
      2 b="HSAI"
```

```
----> 3 b=int(b)

ValueError: invalid literal for int() with base 10: 'HSAI'
```

1.0.7 Accessing string characters in Python using Indexing

[17]:
```
str = 'Hello'
print('str = ', str)
print('len = ', len(str))
```

```
str =  Hello
len =  5
```

[18]:
```
#first character
print('first character = ', str[0])
#last character
print('last character = ', str[len(str)-1])
```

```
first character =  H
last character =  o
```

[19]:
```
#index must be in range
print(str[15])
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-19-dc6a75e25144> in <module>
      1 #index must be in range
----> 2 print(str[15])

IndexError: string index out of range
```

[20]:
```
# index must be an integer
print(str[1.0])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-20-6fc957eb75db> in <module>
      1 # index must be an integer
----> 2 print(str[1.0])

TypeError: string indices must be integers
```

1.0.8 String Slicing

Slicing Using Postive Indexing

[21]:
```
s = 'Python'

print(s[2:5]) #Positive indexing
```

```
tho
```

Slicing Using Negetive Indexing

[22]:
```
print(s[-4:-1]) #Negetive indexing
```

```
tho
```

[23]:
```
# You can omit the first or last index:

print(s[:2]) #str[start:End:Step]

print(s[:5])

print(s[2:])

print(s[2:len(s)])
```

```
Py
Pytho
thon
thon
```

[24]:
```
#To Print Complete String
s = "Python"
print(s[:2] + s[2:]) #[:n] + s[n:] == s

print(s[:])
```

```
Python
Python
```

[25]:
```
#Retruns Empty String If Start=Stop
print(s[2:2])
```

[26]:
```
#Start Greater Than Stop-Retruns Empty String
#End Index has to be greater than starting index
print(s[4:2])
```

[27]:
```
s = "Hello"
print(s[-1:])
```

```
o
```

Slicing With Step

[28]:
```
#slice with a step:

s = 'python'

print(s[0:5:2])

print(s[-1:-4:-1])
```

```
pto
noh
```

[29]:
```
#Reversing of String
print(s[::-1])
```

```
nohtyp
```

[30]:
```
s = '123451234512345123451234512345'

print(s[::5])

print(s[1::5])

print(s[2::5])

print(s[3::5])

print(s[::-5])
```

```
11111
22222
33333
44444
55555
```

1.0.9 String Manipulations

[31]:
```
# Python Program to Update character of a String - Strings are immutable

String1 = "Hello, I'm a Programmer"
print("Initial String: ",String1)
# Updating a character of the String

String1[2] = 'p'
print("Updating character at 2nd Index: ",String1)
```

```
Initial String:  Hello, I'm a Programmer
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-31-72b23989064e> in <module>
      5 # Updating a character of the String
      6
----> 7 String1[2] = 'p'
      8 print("Updating character at 2nd Index: ",String1)

TypeError: 'str' object does not support item assignment
```

[32]:
```
#  Updating entire String
String1 = "Welcome to the Programmers Hub"
print("Updated String: ",String1)
```

```
Updated String:  Welcome to the Programmers Hub
```

[33]:
```
# Python Program to Delete characters from a String
String1 = "Hello, I'm a Programmer"
print("Initial String: ",String1)

# Deleting a character  of the String
del String1[2]
print("Deleting character at 2nd Index: ",String1)
```

```
Initial String:  Hello, I'm a Programmer
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-33-30484c8b723b> in <module>
      4
      5 # Deleting a character  of the String
----> 6 del String1[2]
      7 print("Deleting character at 2nd Index: ",String1)

TypeError: 'str' object doesn't support item deletion
```

[34]:
```
# Deleting entire String

del String1
print("String1 Deleted ")
print(String1)
```

```
String1 Deleted
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
```

```
<ipython-input-34-b8cdd5d6c428> in <module>
      3 del String1
      4 print("String1 Deleted ")
----> 5 print(String1)

NameError: name 'String1' is not defined
```

1.0.10 String Formatting

[60]:
```
name = input()
age = int(input())

print('%s is %d years old' % (name, age))

print('{} is {} years old'.format(name, age))

print(f'{name} is {age} years old')  # f strings
```

```
Nitheesh
26
Nitheesh is 26 years old
Nitheesh is 26 years old
Nitheesh is 26 years old
```

Python program to demonstrate the use of formatting using %

[36]:
```
# Initialize variable as a string
variable = '15'
string = "Variable as string = %s" %(variable)
print (string)
```

```
Variable as string = 15
```

[37]:
```
# Printing as raw data
print ("Variable as raw data = %r" %(variable) )
```

```
Variable as raw data = '15'
```

[38]:
```
# Convert the variable to integer And perform check other formatting options
variable = int(variable) # Without this the below statement will give error.
string = "Variable as integer = %d" %(variable)
print (string )
print ("Variable as float = %f" %(variable) )
print ("Variable as float = %.1f" %(variable) )
```

```
Variable as integer = 15
Variable as float = 15.000000
Variable as float = 15.0
```

[39]:
```
# printing as any string or char after a mark
print ("Variable as hexadecimal = %x" %(variable) )
print ("Variable as octal = %o" %(variable) )
```

```
Variable as hexadecimal = f
Variable as octal = 17
```

Python string format( ) method

[40]:
```
# default order
default_order = "{}, {} and {}".format('John','Bill','Sean')
print('\n--- Default Order ---')
print(default_order)
```

```
--- Default Order ---
John, Bill and Sean
```

[41]:
```
# order using positional argument
positional_order = "{1}, {0} and {2}".format('John','Bill','Sean')
print('\n--- Positional Order ---')
print(positional_order)
```

```
--- Positional Order ---
Bill, John and Sean
```

[42]:
```
# order using keyword argument
keyword_order = "{s}, {b} and {j}".format(j='John',b='Bill',s='Sean')
print('\n--- Keyword Order ---')
print(keyword_order)
```

```
--- Keyword Order ---
Sean, Bill and John
```

[43]:
```
# order using Mixed arguments
mixed = "{0}, {1} and {j}".format('John','Bill',j='Sean')
print('\n--- Mixed Order ---')
print(mixed)
```

```
--- Mixed Order ---
John, Bill and Sean
```

[44]:
```
# string padding with right alignment
print("{:>10}".format("Python"))

# string padding with center alignment
print("{:^10}".format("Python"))
```

```
# string padding with center alignment and '*' padding character
print("{:*^20}".format("Python"))

# To demonstrate aligning of spaces - string alignment
print("{0:^20} was founded in {1:<10}!".format("Programmers Hub", 2020))

print("|{:<10}|{:^10}|{:>10}|".format('butter','bread','jam'))
```

```
    Python
  Python
*******Python*******
  Programmers Hub    was founded in 2020      !
|butter    |  bread   |       jam|
```

[45]:
```
#Truncating strings with format()

#Truncating strings to 3 letters
print("{:.3}".format("caterpillar"))

# truncating strings to 3 letters and padding
print("{:5.3}".format("caterpillar"))

#truncating strings to 3 letters, padding and center alignment
print("{:^5.3}".format("caterpillar"))
```

```
cat
cat
 cat
```

[46]:
```
# Simple number formatting

# integer arguments
print("The number is:{:d}".format(123))

# Convert decimal integers floating point numeric constants
print ("This site is {0:f}% securely {1}!!".format(100, "encrypted"))

# To limit the precision
print ("My average of this {0} was {1:.2f}%".format("semester", 78.234876))

# For no decimal places
print ("My average of this {0} was {1:.0f}%".format("semester", 78.234876))

# octal, binary and hexadecimal format
print("bin: {0:b}, oct: {0:o}, hex: {0:x}".format(12))
```

```
The number is:123
This site is 100.000000% securely encrypted!!
```

```
My average of this semester was 78.23%
My average of this semester was 78%
bin: 1100, oct: 14, hex: c
```

[47]:
```
# Number formatting with padding for int and floats

# integer numbers with minimum width
print("{:5d}".format(12))

# width doesn't work for numbers longer than padding
print("{:2d}".format(1234))

# padding for float numbers
print("{:8.3f}".format(12.2346))

# integer numbers with minimum width filled with zeros
print("{:05d}".format(12))

# padding for float numbers filled with zeros
print("{:08.3f}".format(12.2346))
```

```
   12
1234
  12.235
00012
0012.235
```

Python Concatenation and Repetition

[48]:
```
str1 = 'Hello'
str2 ='World!'
```

[49]:
```
#Concatenation

print('Concatenation', str1 + str2)

print("With Space : ", str1 + " " + str2)
```

```
Concatenation HelloWorld!
With Space :  Hello World!
```

[50]:
```
#Replication
print("Replication : ", str1 * 10)
```

```
Replication :  HelloHelloHelloHelloHelloHelloHelloHelloHelloHello
```

[51]:
```
#Concatenation and Replication
print('-'* 10 + str1 + '-'* 10)
```

```
----------Hello----------
```

Escape Sequences -

[52]:
```
#printing single quote
str1 = 'Hi, I\'m a programmer'
print(str1)
```

```
Hi, I'm a programmer
```

[53]:
```
#printing double quotes
str2 = "\"Hello world\""
print(str2)
```

```
"Hello world"
```

[54]:
```
#printing Backslash
str3 = "D:\\work_folder\\python_works"
print(str3)
```

```
D:\work_folder\python_works
```

[55]:
```
#printing Backspace
str4 = "Hi, I\b am a programmer"
print(str4)
```

```
Hi,  am a programmer
```

[56]:
```
#printing Horizontal Tab
str5 = "Hi, I\t am a programmer"
print(str5)
```

```
Hi, I    am a programmer
```

[57]:
```
#printing NewLine
str6 = "Hi, I\n am a programmer"
print(str6)
```

```
Hi, I
 am a programmer
```

[58]:
```
#using hexadecimal values
str7 = "This is  \x49\x6E\x63\x6C\x75\x64\x65\x48\x65\x6C\x70"
print(str7)
```

```
This is  IncludeHelp
```

Ignoring escape sequences - RawStrings

[59]:
```
#ignoring single quote escape sequences
str1 = r"Hi, I\'m IncludeHelp"
#ignoring double quotes escape sequences
str2 = r"\"Hello world\""
#ignoring path escape sequences
```

```
str3 = R"D:\\work_folder\\python_works"
#ignoring hexadecimal values escape sequences
str4 = R"This is  \x49\x6E\x63\x6C\x75\x64\x65\x48\x65\x6C\x70"

print(str1)
print(str2)
print(str3)
print(str4)
```

```
Hi, I\'m IncludeHelp
\"Hello world\"
D:\\work_folder\\python_works
This is  \x49\x6E\x63\x6C\x75\x64\x65\x48\x65\x6C\x70
```