Python String

Introduction.
Strings are amongst the most popular types in Python. We can create them simply by enclosing characters in quotes. Python treats single quotes the same as double quotes. Creating strings is as simple as assigning a value to a variable

Objectives
Objectives by the end of this topic you should be able to:

- Create strings
- String slicing
- String methods

Learning activities
Learning Activity 7.1: Reading
Write a Python script that takes input from the user and displays that input back in upper and lower cases

Learning Activity 7.2: Journal
Write a Python program to remove the characters which have odd index values of a given string.

Learning Activity 7.3: Discussion
Write a Python program to remove the nth index character from a nonempty string

Assessment

Topic resourcess

1. The Python Tutorial¶. (n.d.). Retrieved from https://docs.python.org/3/tutorial/index.html
2. Mueller, J. P. (n.d.). *Beginning Programming with Python For Dummies*. S.l.: For Dummies.
3. (n.d.). Python 3.7.4 documentation. Retrieved from https://docs.python.org/3
4. (n.d.). Git Handbook. Retrieved from https://guides.github.com/introduction/git-handbook/
5. Shaw, Z. (2017). *Learn Python 3 the hard way: a very simple introduction to the terrifyingly beautiful world of computers and code*. Boston: Addison-Wesley.
6. Bader, D. (2018). *Python tricks: the book*. Vancouver, BC: Dan Bader.
7. Downey, A. B. (2015). *Think Python*. Sebastopol: OReilly.
8. Ramalho, L. (2016). *Fluent Python:*Beijing: OReilly.

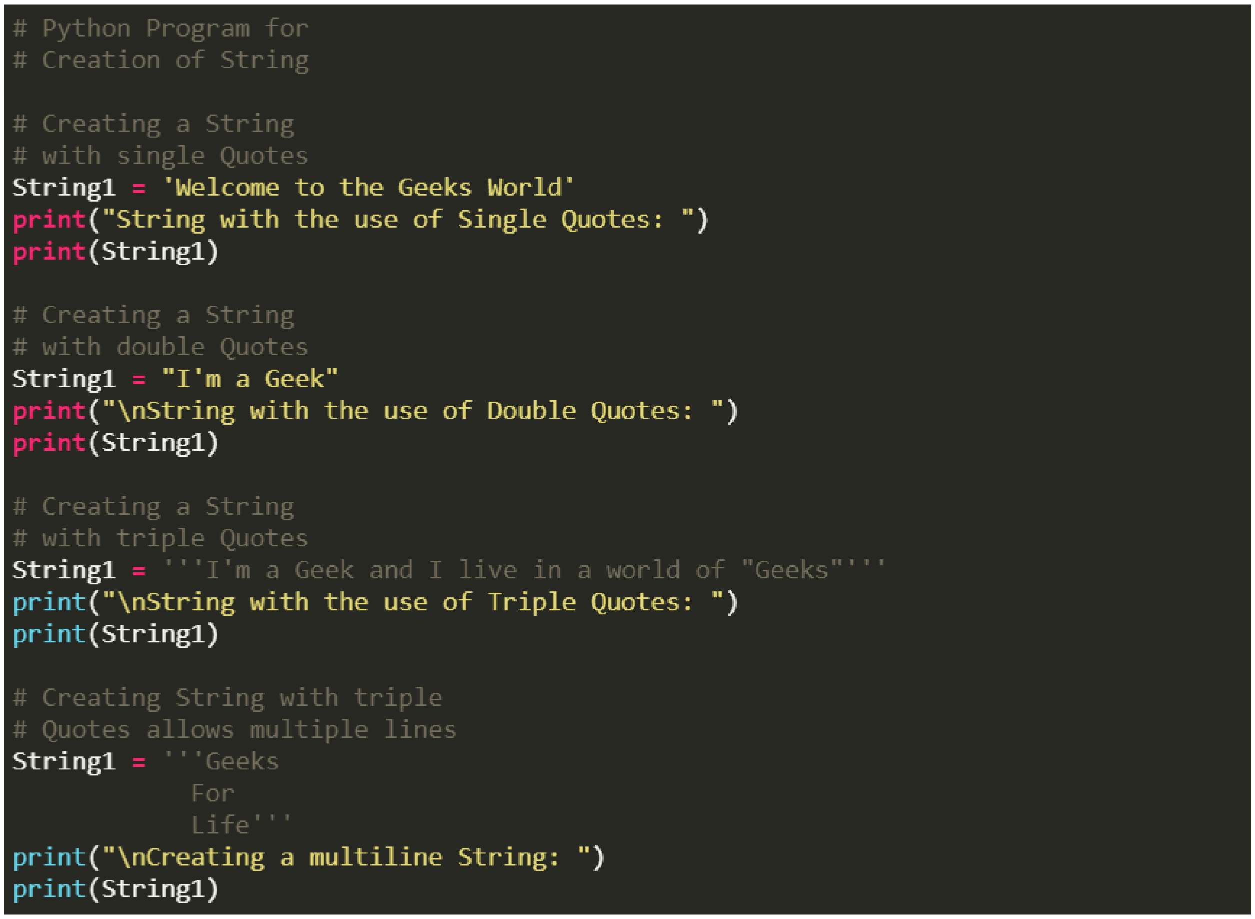
URL Links
https://www.tutorialspoint.com/python/python_strings.htm
https://www.w3schools.in/python-tutorial/strings/
https://data-flair.training/blogs/python-string/

Python String - NOTES

In Python, **Strings** are arrays of bytes representing Unicode characters. However, Python does not have a character data type, a single character is simply a string with a length of 1. Square brackets can be used to access elements of the string.

Creating a String

Strings in Python can be created using single quotes or double quotes or even triple quotes.



```python
# Python Program for
# Creation of String

# Creating a String
# with single Quotes
String1 = 'Welcome to the Geeks World'
print("String with the use of Single Quotes: ")
print(String1)

# Creating a String
# with double Quotes
String1 = "I'm a Geek"
print("\nString with the use of Double Quotes: ")
print(String1)

# Creating a String
# with triple Quotes
String1 = '''I'm a Geek and I live in a world of "Geeks"'''
print("\nString with the use of Triple Quotes: ")
print(String1)

# Creating String with triple
# Quotes allows multiple lines
String1 = '''Geeks
            For
            Life'''
print("\nCreating a multiline String: ")
print(String1)
```

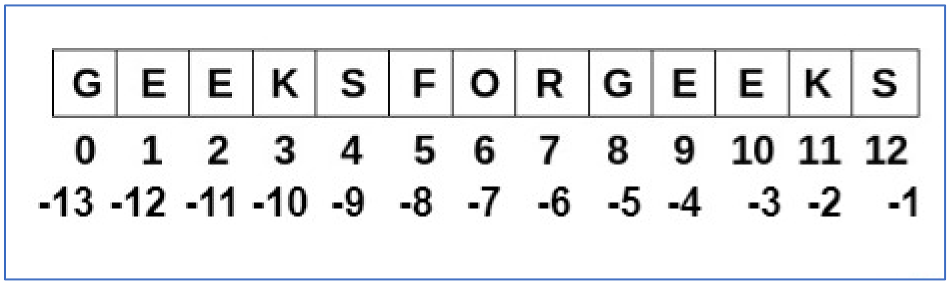
Output:

```
String with the use of Single Quotes:

Welcome to the Geeks World

String with the use of Double Quotes:

I'm a Geek
```

```
String with the use of Triple Quotes:

I'm a Geek and I live in a world of "Geeks"



Creating a multiline String:

Geeks

            For

            Life
```

Accessing characters in Python

In Python, individual characters of a String can be accessed by using the method of Indexing. Indexing allows negative address references to access characters from the back of the String, e.g. -1 refers to the last character, -2 refers to the second last character and so on.
While accessing an index out of the range will cause an **IndexError**. Only Integers are allowed to be passed as an index, float or other types will cause a **TypeError**.



G	E	E	K	S	F	O	R	G	E	E	K	S
0	1	2	3	4	5	6	7	8	9	10	11	12
-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```python
# Python Program to Access
# characters of String

String1 = "GeeksForGeeks"
print("Initial String: ")
print(String1)

# Printing First character
print("\nFirst character of String is: ")
print(String1[0])

# Printing Last character
print("\nLast character of String is: ")
print(String1[-1])
```

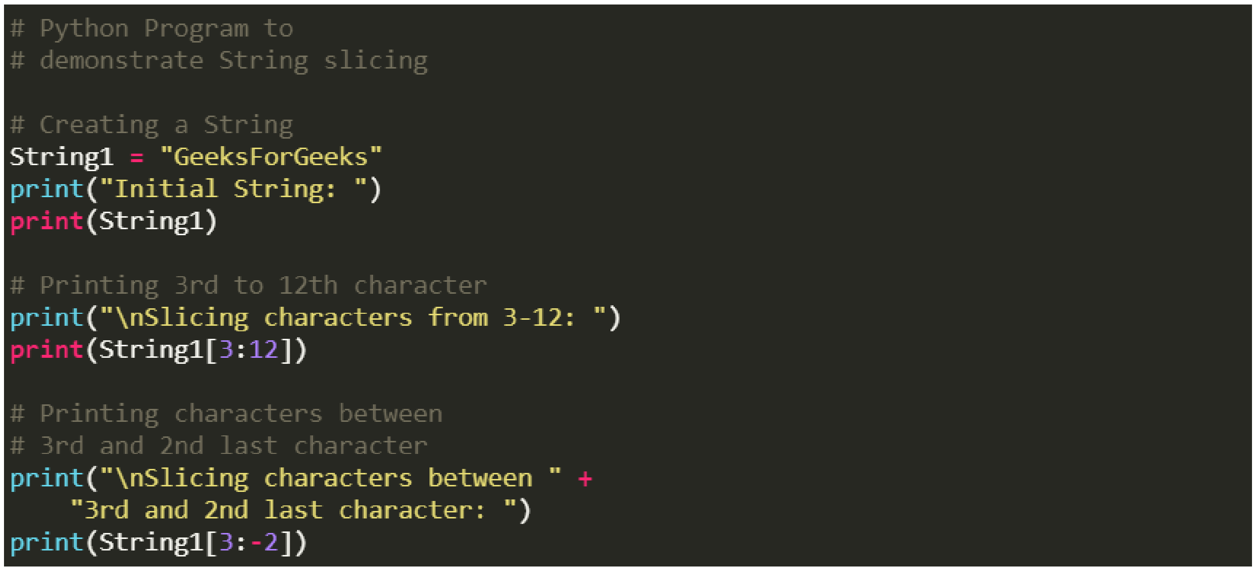
Output:

```
Initial String:

GeeksForGeeks
```

```
First character of String is:

G

Last character of String is:

s
```

String Slicing

To access a range of characters in the String, method of slicing is used. Slicing in a String is done by using a Slicing operator (colon).



```python
# Python Program to
# demonstrate String slicing

# Creating a String
String1 = "GeeksForGeeks"
print("Initial String: ")
print(String1)

# Printing 3rd to 12th character
print("\nSlicing characters from 3-12: ")
print(String1[3:12])

# Printing characters between
# 3rd and 2nd last character
print("\nSlicing characters between " +
    "3rd and 2nd last character: ")
print(String1[3:-2])
```

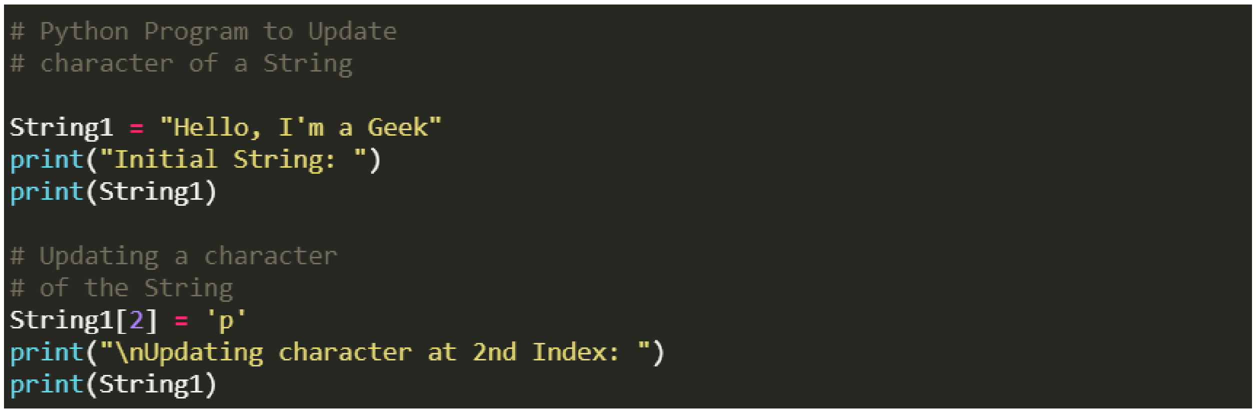
Output:

```
Initial String:

GeeksForGeeks



Slicing characters from 3-12:

ksForGeek



Slicing characters between 3rd and 2nd last character:

ksForGee
```

Deleting/Updating from a String

In Python, Updating or deleting of characters from a String is not allowed. This will cause an error because item assignment or item deletion from a String is not supported. Although deletion of entire String is possible with the use of a built-in del keyword. This is because Strings are immutable, hence elements of a String cannot be changed once it has been assigned. Only new strings can be reassigned to the same name.

Updating a character:

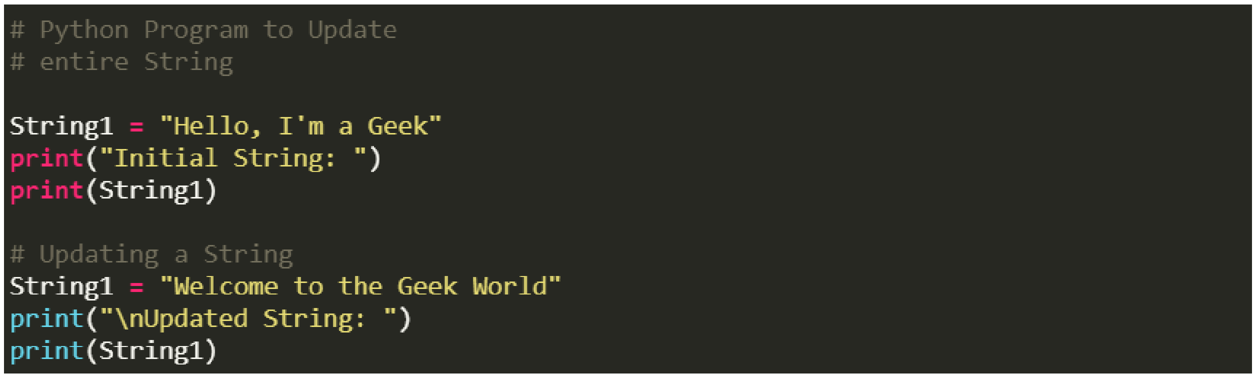


```
# Python Program to Update
# character of a String

String1 = "Hello, I'm a Geek"
print("Initial String: ")
print(String1)

# Updating a character
# of the String
String1[2] = 'p'
print("\nUpdating character at 2nd Index: ")
print(String1)
```

Error:

Traceback (most recent call last):
File "/home/360bb1830c83a918fc78aa8979195653.py", line 10, in
String1[2] = 'p'
TypeError: 'str' object does not support item assignment

Updating Entire String:

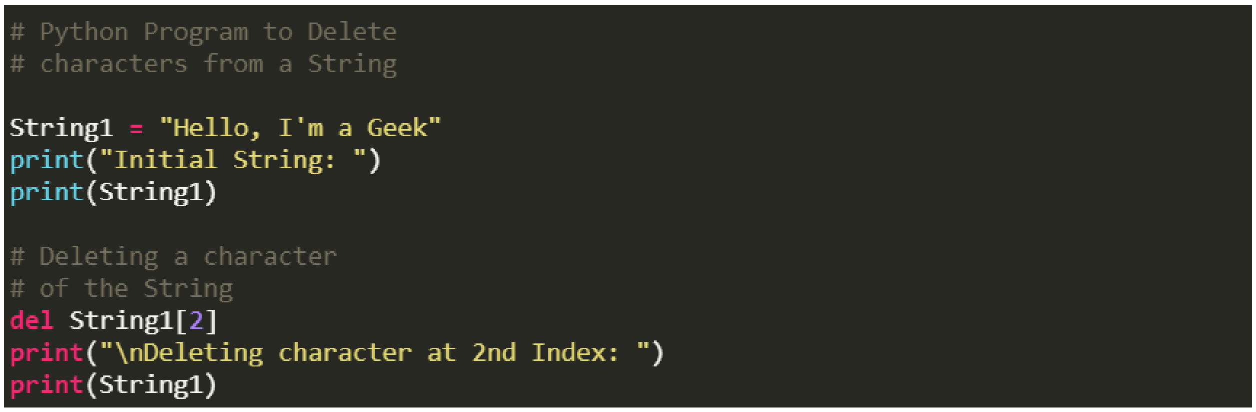


```
# Python Program to Update
# entire String

String1 = "Hello, I'm a Geek"
print("Initial String: ")
print(String1)

# Updating a String
String1 = "Welcome to the Geek World"
print("\nUpdated String: ")
print(String1)
```

Output:

```
Initial String:

Hello, I'm a Geek

Updated String:

Welcome to the Geek World
```

Deleting a character:



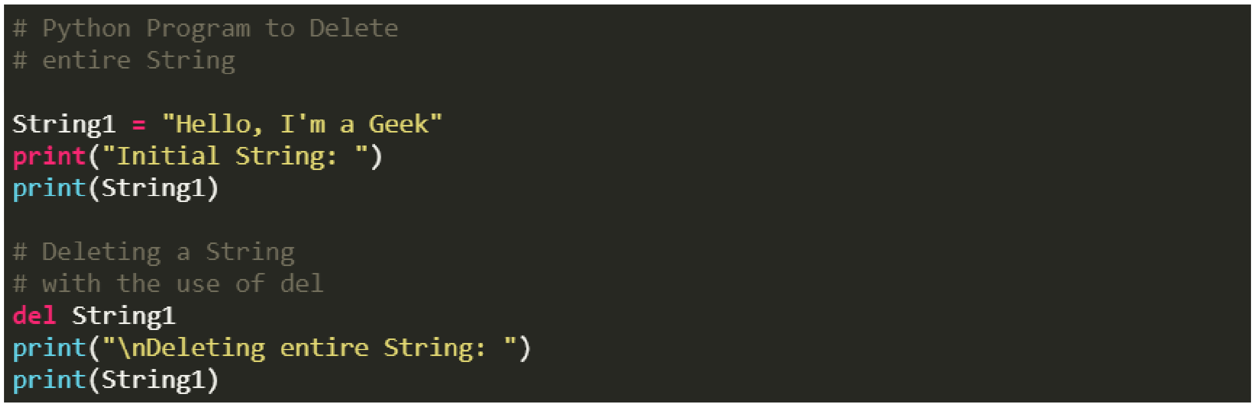
```python
# Python Program to Delete
# characters from a String

String1 = "Hello, I'm a Geek"
print("Initial String: ")
print(String1)

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

Error:

```
Traceback (most recent call last):
File “/home/499e96a61e19944e7e45b7a6e1276742.py”, line 10, in
del String1[2]
TypeError: ‘str’ object doesn’t support item deletion
```

Deleting Entire String:

Deletion of entire string is possible with the use of del keyword. Further, if we try to print the string, this will produce an error because String is deleted and is unavailable to be printed.



```python
# Python Program to Delete
# entire String

String1 = "Hello, I'm a Geek"
print("Initial String: ")
print(String1)

# Deleting a String
# with the use of del
del String1
print("\nDeleting entire String: ")
print(String1)
```

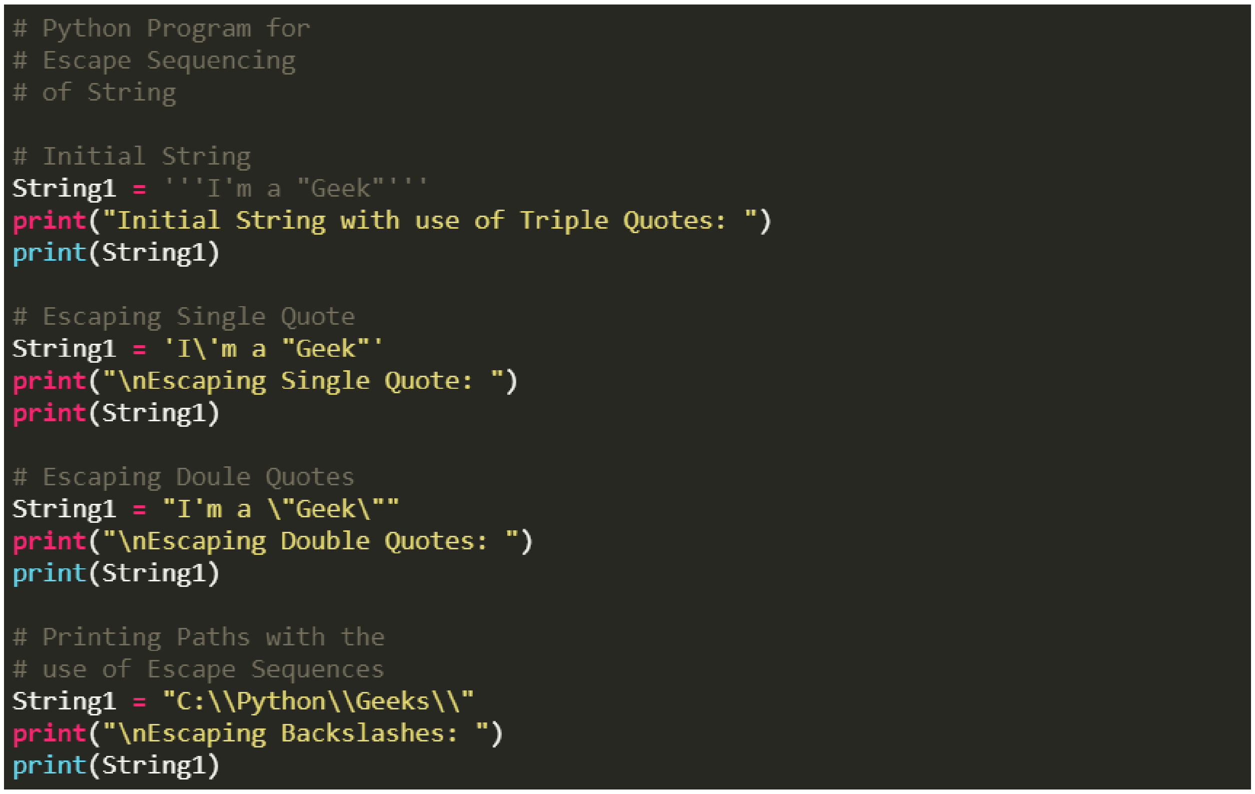
Error:

```
Traceback (most recent call last):
File “/home/e4b8f2170f140da99d2fe57d9d8c6a94.py”, line 12, in
print(String1)
NameError: name ‘String1’ is not defined
```

Escape Sequencing in Python

While printing Strings with single and double quotes in it causes **SyntaxError** because String already contain ns Single and Double Quotes and hence cannot be printed with the use of either

of these. Hence, to print such a String either Triple Quotes are used or Escape sequences are used to print such Strings.
Escape sequences start with a backslash and can be interpreted differently. If single quotes are used to represent a string, then all the single quotes present in the string must be escaped and same is done for Double Quotes.



```python
# Python Program for
# Escape Sequencing
# of String

# Initial String
String1 = '''I'm a "Geek"'''
print("Initial String with use of Triple Quotes: ")
print(String1)

# Escaping Single Quote
String1 = 'I\'m a "Geek"'
print("\nEscaping Single Quote: ")
print(String1)

# Escaping Doule Quotes
String1 = "I'm a \"Geek\""
print("\nEscaping Double Quotes: ")
print(String1)

# Printing Paths with the
# use of Escape Sequences
String1 = "C:\\Python\\Geeks\\"
print("\nEscaping Backslashes: ")
print(String1)
```

Output:

```
Initial String with use of Triple Quotes:

I'm a "Geek"


Escaping Single Quote:

I'm a "Geek"


Escaping Double Quotes:

I'm a "Geek"


Escaping Backslashes:

C:\Python\Geeks\
```

To ignore the escape sequences in a String, **r** or **R** is used, this implies that the string is a raw string and escape sequences inside it are to be ignored.

```
# Printing Geeks in HEX
String1 = "This is \x47\x65\x65\x6b\x73 in \x48\x45\x58"
print("\nPrinting in HEX with the use of Escape Sequences: ")
print(String1)

# Using raw String to
# ignore Escape Sequences
String1 = r"This is \x47\x65\x65\x6b\x73 in \x48\x45\x58"
print("\nPrinting Raw String in HEX Format: ")
print(String1)
```

Output:

```
Printing in HEX with the use of Escape Sequences:

This is Geeks in HEX


Printing Raw String in HEX Format:

This is \x47\x65\x65\x6b\x73 in \x48\x45\x58
```

Formatting of Strings

Strings in Python can be formatted with the use of format() method which is very versatile and powerful tool for formatting of Strings. Format method in String contains curly braces {} as placeholders which can hold arguments according to position or keyword to specify the order.

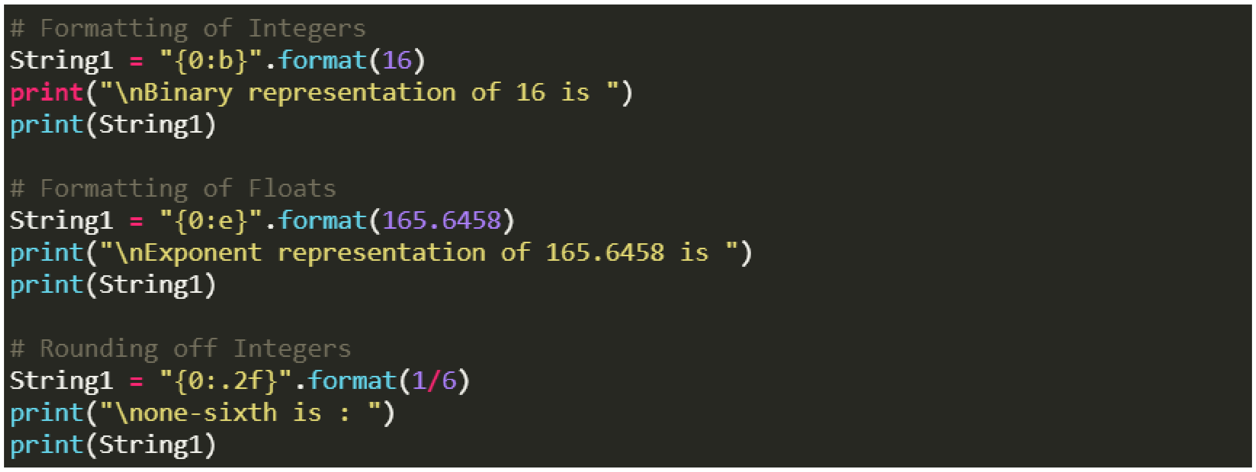
```
# Python Program for
# Formatting of Strings

# Default order
String1 = "{} {} {}".format('Geeks', 'For', 'Life')
print("Print String in default order: ")
print(String1)

# Positional Formatting
String1 = "{1} {0} {2}".format('Geeks', 'For', 'Life')
print("\nPrint String in Positional order: ")
print(String1)

# Keyword Formatting
String1 = "{l} {f} {g}".format(g = 'Geeks', f = 'For', l = 'Life')
print("\nPrint String in order of Keywords: ")
print(String1)
```

Output:

```
Print String in default order:
```

```
Geeks For Life


Print String in Positional order:

For Geeks Life


Print String in order of Keywords:

Life For Geeks
```

Integers such as Binary, hexadecimal, etc. and floats can be rounded or displayed in the exponent form with the use of format specifiers.

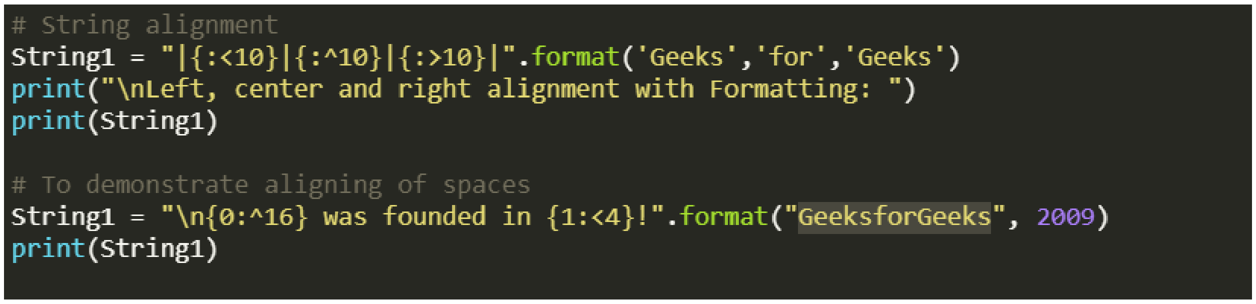


```python
# Formatting of Integers
String1 = "{0:b}".format(16)
print("\nBinary representation of 16 is ")
print(String1)

# Formatting of Floats
String1 = "{0:e}".format(165.6458)
print("\nExponent representation of 165.6458 is ")
print(String1)

# Rounding off Integers
String1 = "{0:.2f}".format(1/6)
print("\none-sixth is : ")
print(String1)
```

Output:

```
Binary representation of 16 is

10000


Exponent representation of 165.6458 is

1.656458e+02


one-sixth is :

0.17
```

A string can be left() or center(^) justified with the use of format specifiers, separated by colon(:).

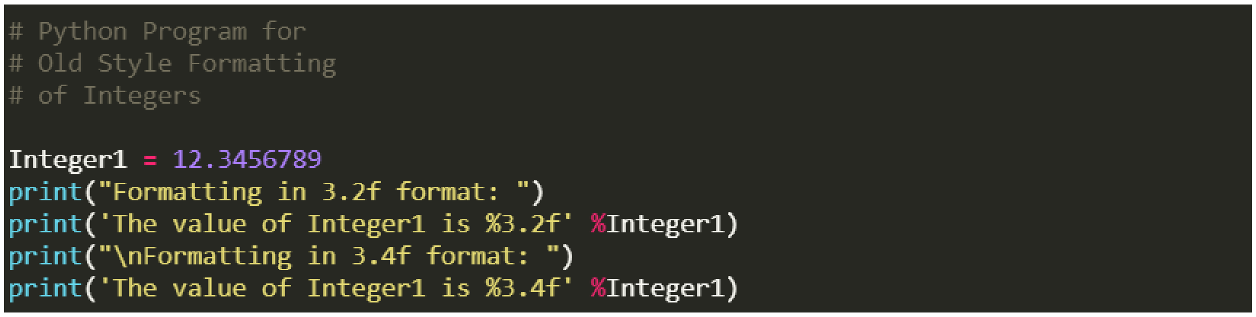


```python
# String alignment
String1 = "|{:<10}|{:^10}|{:>10}|".format('Geeks','for','Geeks')
print("\nLeft, center and right alignment with Formatting: ")
print(String1)

# To demonstrate aligning of spaces
String1 = "\n{0:^16} was founded in {1:<4}!".format("GeeksforGeeks", 2009)
print(String1)
```

Output:

```
Left, center and right alignment with Formatting:

|Geeks     |   for    |     Geeks|



 GeeksforGeeks   was founded in 2009 !
```

Old style formatting was done without the use of format method by using **%** operator



```python
# Python Program for
# Old Style Formatting
# of Integers

Integer1 = 12.3456789
print("Formatting in 3.2f format: ")
print('The value of Integer1 is %3.2f' %Integer1)
print("\nFormatting in 3.4f format: ")
print('The value of Integer1 is %3.4f' %Integer1)
```

Output:

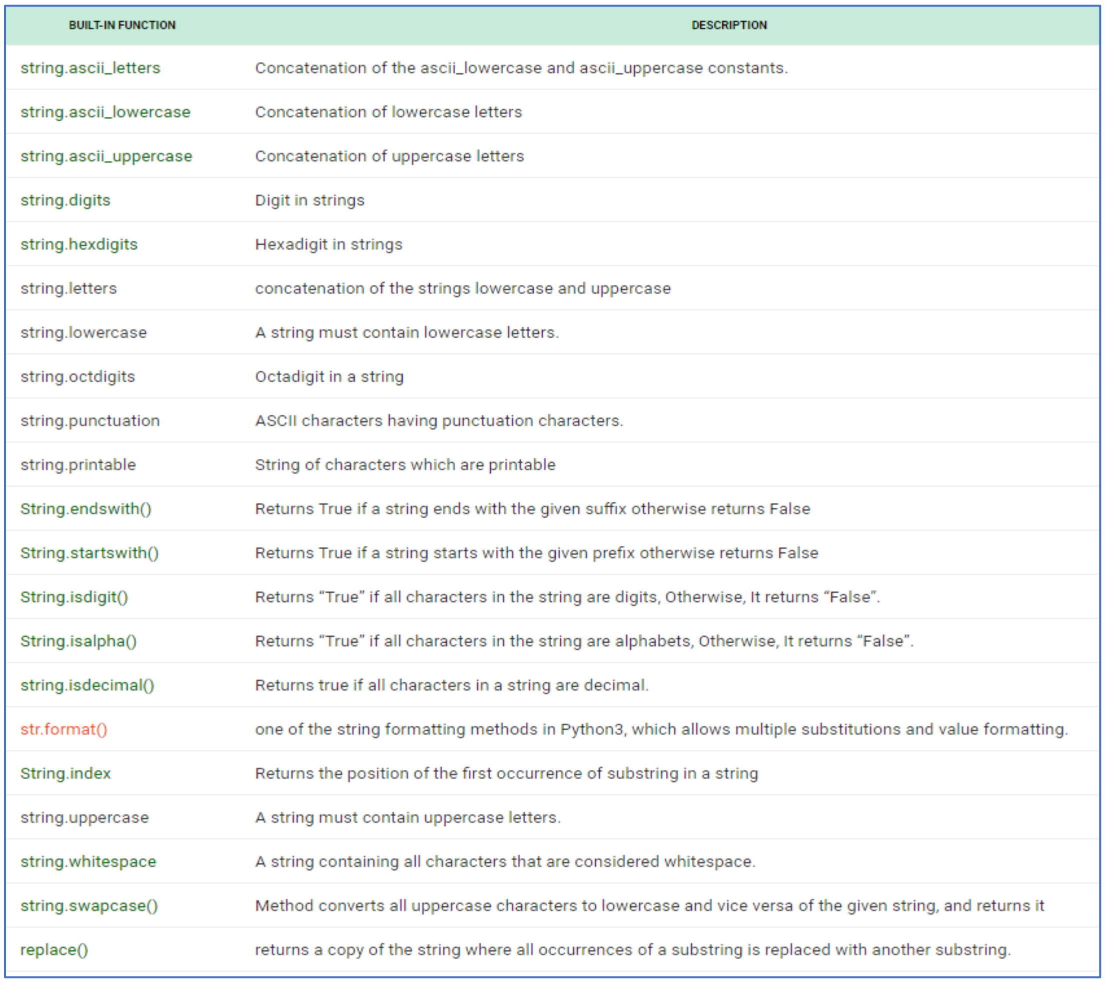
```
Formatting in 3.2f format:

The value of Integer1 is 12.35



Formatting in 3.4f format:

The value of Integer1 is 12.3457
```

Useful String Operations

- Logical Operators on String
- String Formatting using %
- String Template Class
- Split a string
- Python Docstrings
- String slicing
- Find all duplicate characters in string

- Reverse string in Python (5 different ways)
- Python program to check if a string is palindrome or not

String constants



BUILT-IN FUNCTION	DESCRIPTION
string.ascii_letters	Concatenation of the ascii_lowercase and ascii_uppercase constants.
string.ascii_lowercase	Concatenation of lowercase letters
string.ascii_uppercase	Concatenation of uppercase letters
string.digits	Digit in strings
string.hexdigits	Hexadigit in strings
string.letters	concatenation of the strings lowercase and uppercase
string.lowercase	A string must contain lowercase letters.
string.octdigits	Octadigit in a string
string.punctuation	ASCII characters having punctuation characters.
string.printable	String of characters which are printable
String.endswith()	Returns True if a string ends with the given suffix otherwise returns False
String.startswith()	Returns True if a string starts with the given prefix otherwise returns False
String.isdigit()	Returns "True" if all characters in the string are digits, Otherwise, It returns "False".
String.isalpha()	Returns "True" if all characters in the string are alphabets, Otherwise, It returns "False".
string.isdecimal()	Returns true if all characters in a string are decimal.
str.format()	one of the string formatting methods in Python3, which allows multiple substitutions and value formatting.
String.index	Returns the position of the first occurrence of substring in a string
string.uppercase	A string must contain uppercase letters.
string.whitespace	A string containing all characters that are considered whitespace.
string.swapcase()	Method converts all uppercase characters to lowercase and vice versa of the given string, and returns it
replace()	returns a copy of the string where all occurrences of a substring is replaced with another substring.

Revision questions

1. Write a Python program to calculate the length of a string. Go to the editor
Click me to see the sample solution

2. Write a Python program to count the number of characters (character frequency) in a string. Go to the editor
Sample String : google.com'
Expected Result : {'o': 3, 'g': 2, '.': 1, 'e': 1, 'l': 1, 'm': 1, 'c': 1}
Click me to see the sample solution

3. Write a Python program to get a string made of the first 2 and the last 2 chars from a given a string. If the string length is less than 2, return instead of the empty string. Go to the editor
Sample String : 'w3resource'
Expected Result : 'w3ce'
Sample String : 'w3'

Expected Result : 'w3w3'
Sample String : ' w'
Expected Result : Empty String
Click me to see the sample solution

4. Write a Python program to get a string from a given string where all occurrences of its first char have been changed to '$', except the first char itself. Go to the editor
Sample String : 'restart'
Expected Result : 'resta$t'
Click me to see the sample solution

5. Write a Python program to get a single string from two given strings, separated by a space and swap the first two characters of each string. Go to the editor
Sample String : 'abc', 'xyz'
Expected Result : 'xyc abz'
Click me to see the sample solution