Strings

A string is a data type in Python, which is a collection of characters. Characters include any letter, numbers and special characters also. For example: “Welcome”, ‘Lockdown 2.0’.

Syntax :

Variable_name = “string”

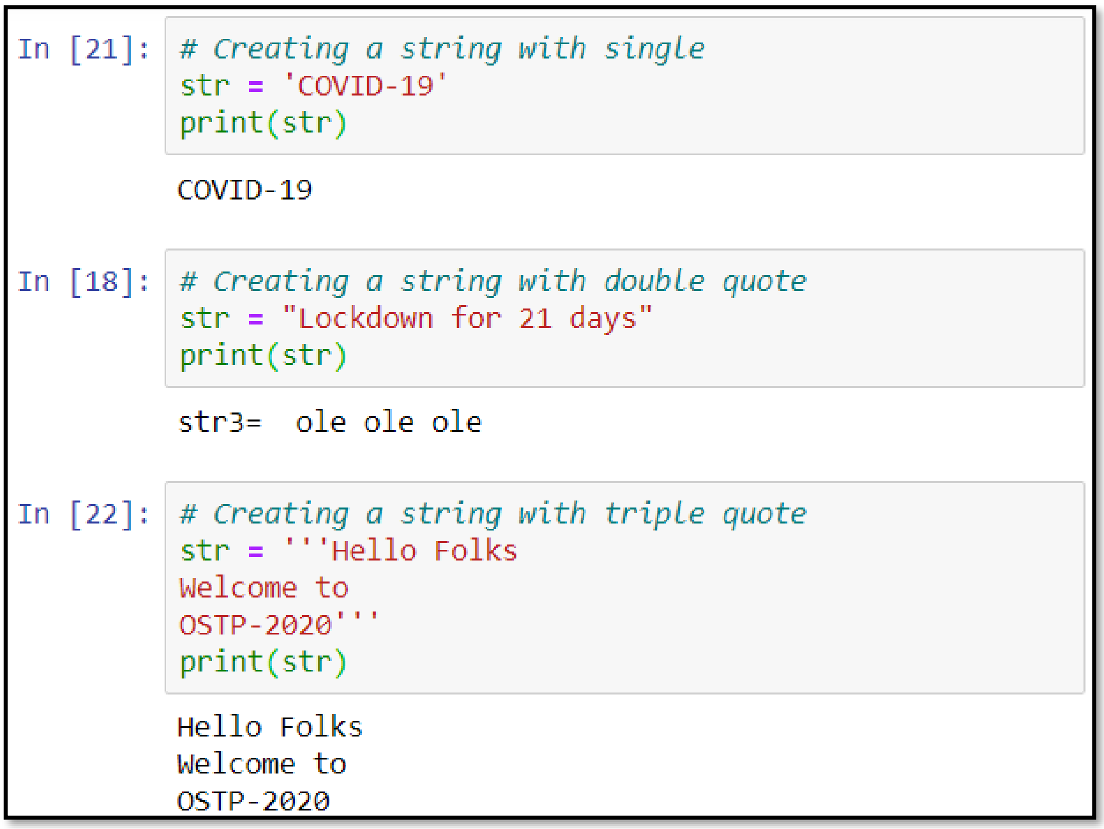
OR

Variable_name = ‘string’

1. How to create a string and assign it to a variable?

String can be created by enclosing the characters inside a single quote or double quote. Triple quote is also used in Python but generally used to represent multiline string.

Example:



```
In [21]: # Creating a string with single
         str = 'COVID-19'
         print(str)

         COVID-19

In [18]: # Creating a string with double quote
         str = "Lockdown for 21 days"
         print(str)

         str3=  ole ole ole

In [22]: # Creating a string with triple quote
         str = '''Hello Folks
         Welcome to
         OSTP-2020'''
         print(str)

         Hello Folks
         Welcome to
         OSTP-2020
```

2. How to access characters in string?

- We can access an individual characters using indexing and a range of characters using slicing.
- Index starts from 0. The index must be an integer.
- Trying to access a character out of the range will give an IndexError.
- Other than integer like float or other types it will give TypeError.

2.1 Negative indexing

- The index with -1 refers the last element in string.
- Similarly, -2 refers second last element in a string and so on.
- We can access a range of elements or extracting some part of string by using Slicing operator (colon).

0	1	2	3	4	5	6	7	8	9	10
H	e	l	l	o		W	o	r	l	d
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Example:

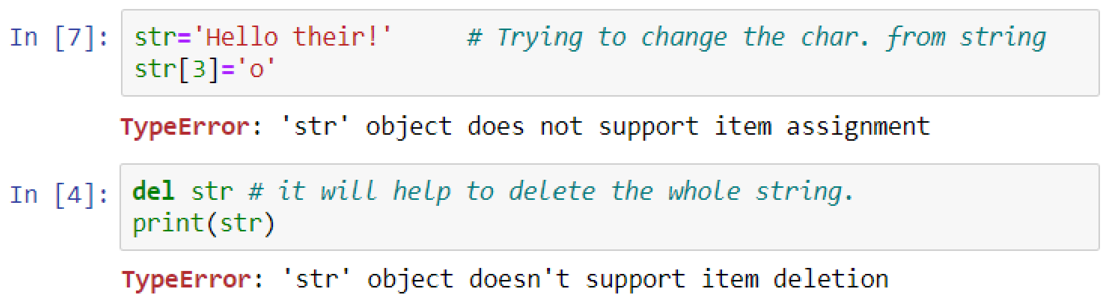
```
In [2]: str = "Basics of Python"
        print('str= ', str)

        str=  Basics of Python

In [3]: # Accessing first char.[Indexing]
        print('str[0]= ' ,str[0])

        str[0]=  B

In [4]: # Accessing Last char.[Negative Indexing]
        print('str[-1]= ' ,str[-1])

        str[-1]=  n

In [5]: # Slicing 2nd to 5th char.
        print('str[1:5]= ' ,str[1:5])

        str[1:5]=  asic

In [6]: # Slicing 6th to 2nd from last char.
        print('str[5:-2]= ' , str[5:-2])

        str[5:-2]=  s of Pyth
```

If we try to access index out of range or use any decimal values it will give error.

```
# Index must be in range
>>> str[20]
IndexError: string index out of range
# Index must be an integer
>>> str[1.5]
TypeError: string indices must be integers
```

3. How to change or delete a string?

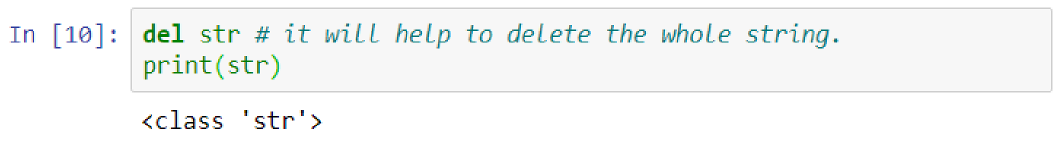
Strings are immutable. Immutable means elements of string cannot be changed once it has been assigned. We can simply reassign a string with same name or variable.



```
In [7]: str='Hello their!'      # Trying to change the char. from string
        str[3]='o'

TypeError: 'str' object does not support item assignment

In [4]: del str # it will help to delete the whole string.
        print(str)

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

We cannot delete or remove character from string. But we can delete complete string using keyword del.

Example:



```
In [10]: del str # it will help to delete the whole string.
         print(str)

         <class 'str'>
```

4. Python string operations

There are many operation in Python that can be performed in string which makes it one of the most used datatypes in Python.

4.1 Concatenation of Two or more strings

Joining of two or more string in one is called concatenation.

The + operator in Python is used to join two different string store in different variables.

Example:

```
In [19]: str1= 'Corona '
         str2='Virus '

         # Using + operator
         print('str= ', str1 + str2 )

         str=  Corona Virus
```

The * operator in Python is used to repeat the string for a given number of times.

Example:

```
In [18]: # Using * operator
         str3='ole '
         print('str3= ', str3 * 3)

         str3=  ole ole ole
```

4.2 Iterating through string

Using for loop we can iterate the string. Below is the example to print string element one after another.

Example:

```
In [8]: # Iterating string using FOR Loop.
        count =0
        for i in 'Python':
            print(i)

        P
        y
        t
        h
        o
        n
```

4.3 String Membership test

To check if sub string is present in large string or not, using keyword in and not in.

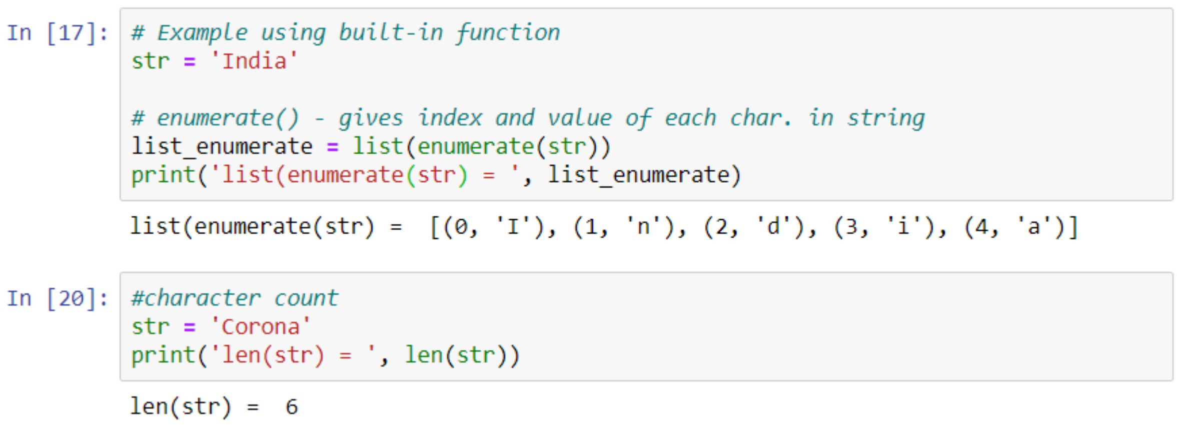
Example:

```
In [9]: # Membership Test
        'o' in 'Python'

Out[9]: True

In [10]: 'o'  not  in 'Python'

Out[10]: False
```

5. Built – in functions to work with python

- Various Built-in function that works with strings.
- Some of the commonly used are enumerate( ) and len( ).
- The **enumerate( )** function returns the enumerate object which contains the index and value of each element of string. This can be usful in iteration.
- Similarly, **len( )** function is used to returns the length(number of characters) of the string.

Example:

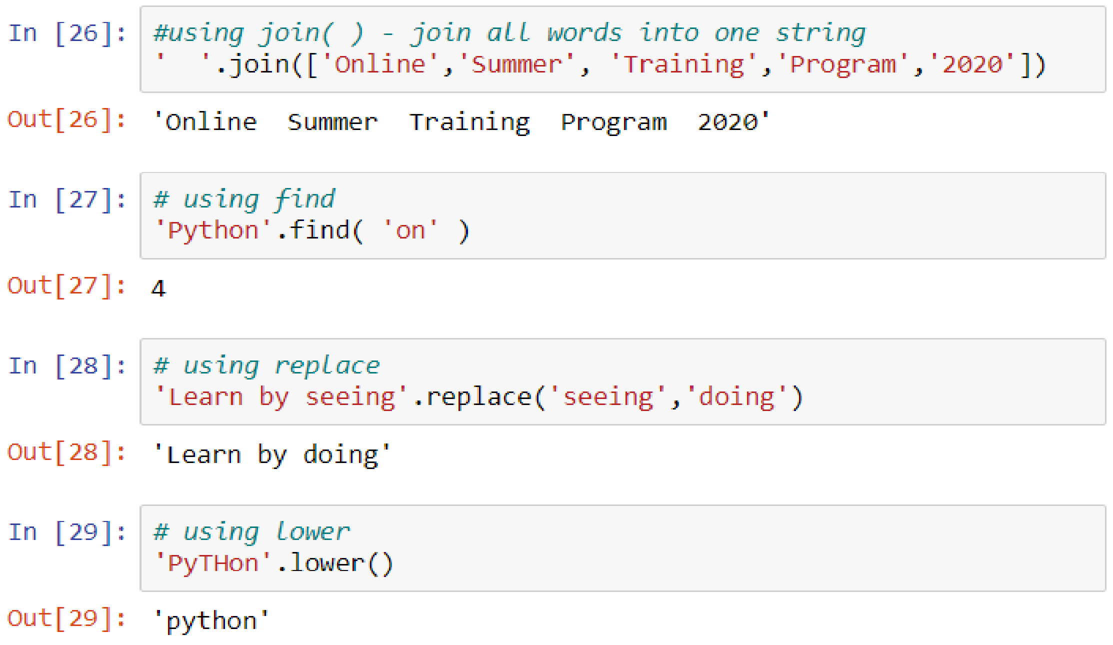


```
In [17]: # Example using built-in function
         str = 'India'

         # enumerate() - gives index and value of each char. in string
         list_enumerate = list(enumerate(str))
         print('list(enumerate(str) = ', list_enumerate)

         list(enumerate(str) =  [(0, 'I'), (1, 'n'), (2, 'd'), (3, 'i'), (4, 'a')]

In [20]: #character count
         str = 'Corona'
         print('len(str) = ', len(str))

         len(str) =  6
```

There are number of methods available with string object.

Some commonly used methods are join(), find(), replace(), lower(), upper(), split(), capitalize(), starswith(),endswith(),etc.

Example:



```
In [26]: #using join( ) - join all words into one string
         '  '.join(['Online','Summer', 'Training','Program','2020'])

Out[26]: 'Online  Summer  Training  Program  2020'

In [27]: # using find
         'Python'.find( 'on' )

Out[27]: 4

In [28]: # using replace
         'Learn by seeing'.replace('seeing','doing')

Out[28]: 'Learn by doing'

In [29]: # using lower
         'PyTHon'.lower()

Out[29]: 'python'
```

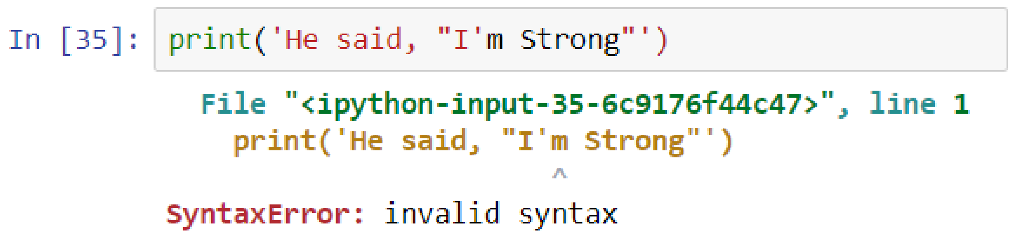
```
In [29]: # using lower
         'PyTHon'.lower()

Out[29]: 'python'

In [30]: # using upper
         'ostp-20'.upper()

Out[30]: 'OSTP-20'

In [31]: # using split( ) - split all words into list
         'Hello Python Trainees'.split()

Out[31]: ['Hello', 'Python', 'Trainees']

In [32]: # using capatilize
         'learn by doing'.capitalize()

Out[32]: 'Learn by doing'

In [34]: # using startswith & endswith
         # it gives the value either True or False
         str='Hello World!'

         print(str.startswith('Hello'))
         print(str.endswith('Folks'))
         print(str.endswith('World'))

         True
         False
         False
```

6. Python string Formatting

6.1 Escape sequence

- If we want to print a text like - I’m Rahul Sharma. Here we cannot use single quote.
- Let’s take another example - He said, ”I’m strong”. In this example we neither use single quote nor double quotes.
- This will results in SyntaxError as above both example itself contain both single or double quotes.

Example:



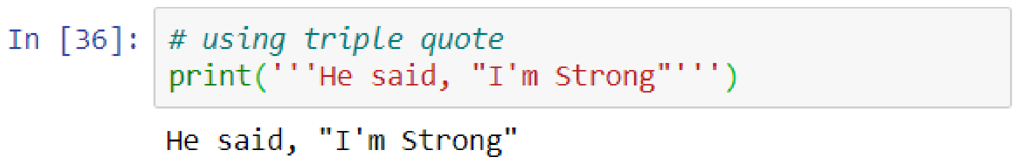
```
In [35]: print('He said, "I'm Strong"')

  File "<ipython-input-35-6c9176f44c47>", line 1
    print('He said, "I'm Strong"')
                      ^
SyntaxError: invalid syntax
```

One of the solution is by using triple quotes. Other solution is, we can use escape sequence.

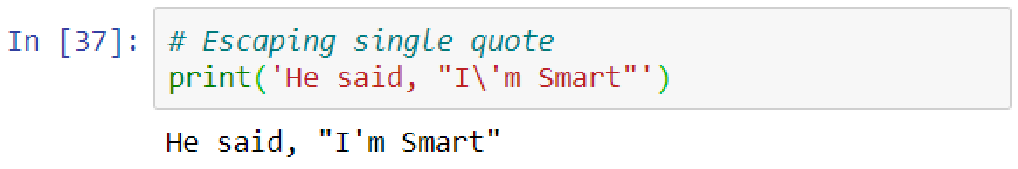
Escape sequences are starts with backslash(\ ) and is interpreted differently.

If we use single quote to represent string then all the single quotes in string must be escaped. Similar is the case with Double quote.

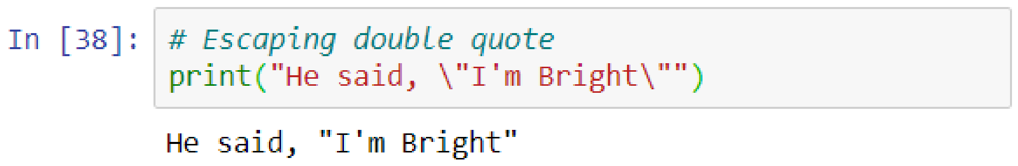
Example:



```
In [36]: # using triple quote
         print('''He said, "I'm Strong"''')

         He said, "I'm Strong"
```



```
In [37]: # Escaping single quote
         print('He said, "I\'m Smart"')

         He said, "I'm Smart"
```

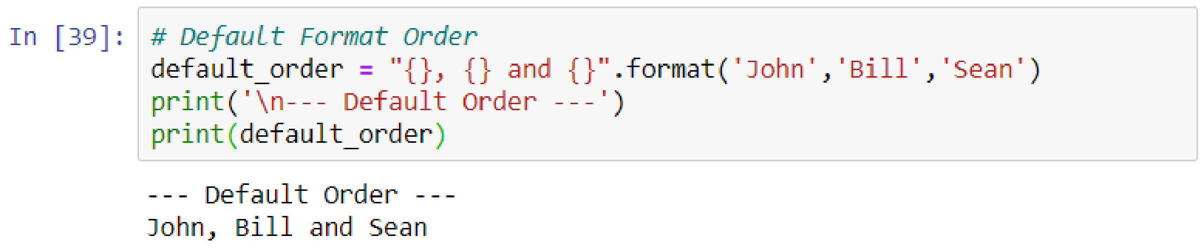


```
In [38]: # Escaping double quote
         print("He said, \"I'm Bright\"")

         He said, "I'm Bright"
```

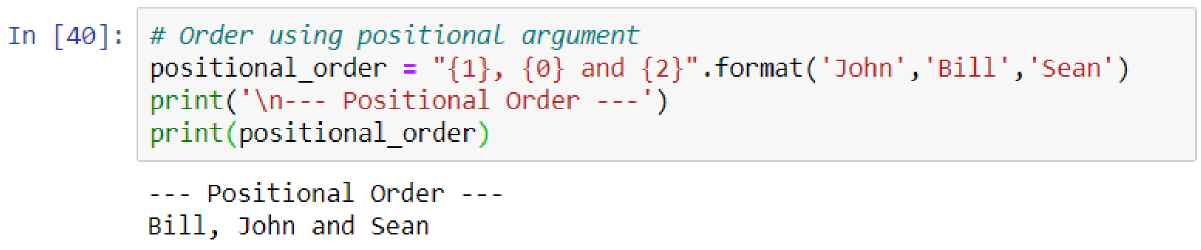
6.2 Formatting of string

- String in Python can be formatted by using format( ) method which is mostly used for formatting string.
- Format strings contain curly braces {} as a placeholder or replacement field which gets replaced.
- It can hold arguments according to position or keyword to specify the order.

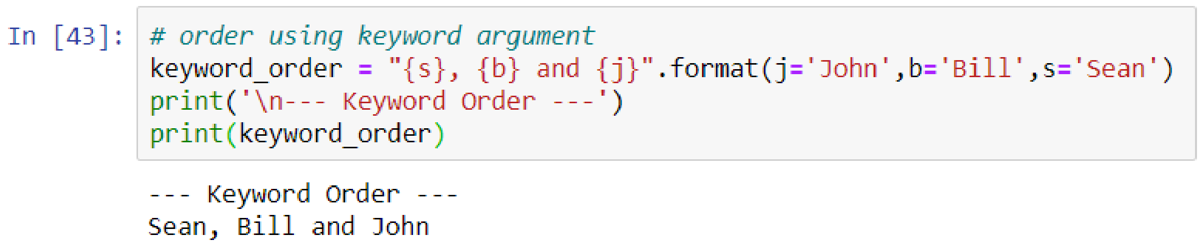
Example:



```
In [39]: # Default Format Order
         default_order = "{}, {} and {}".format('John','Bill','Sean')
         print('\n--- Default Order ---')
         print(default_order)

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



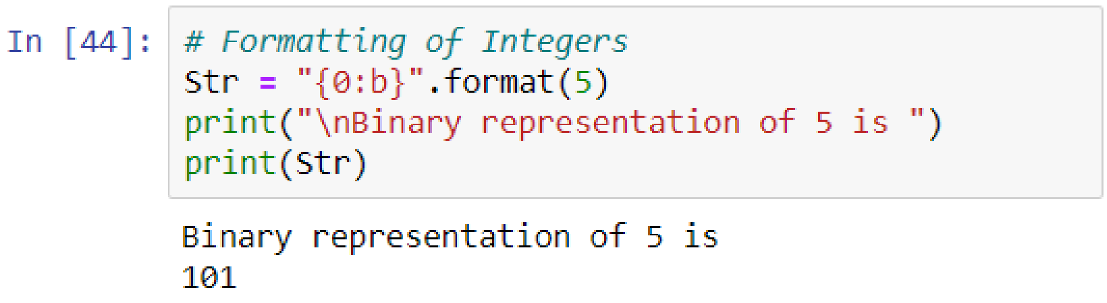
```
In [40]: # Order using positional argument
         positional_order = "{1}, {0} and {2}".format('John','Bill','Sean')
         print('\n--- Positional Order ---')
         print(positional_order)

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



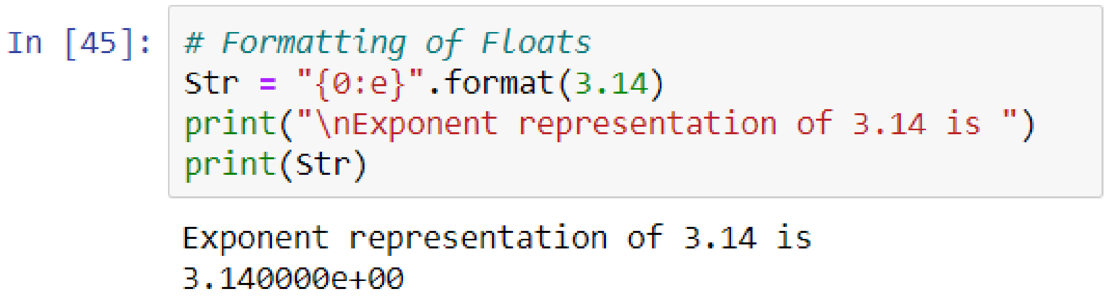
```
In [43]: # order using keyword argument
         keyword_order = "{s}, {b} and {j}".format(j='John',b='Bill',s='Sean')
         print('\n--- Keyword Order ---')
         print(keyword_order)

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

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

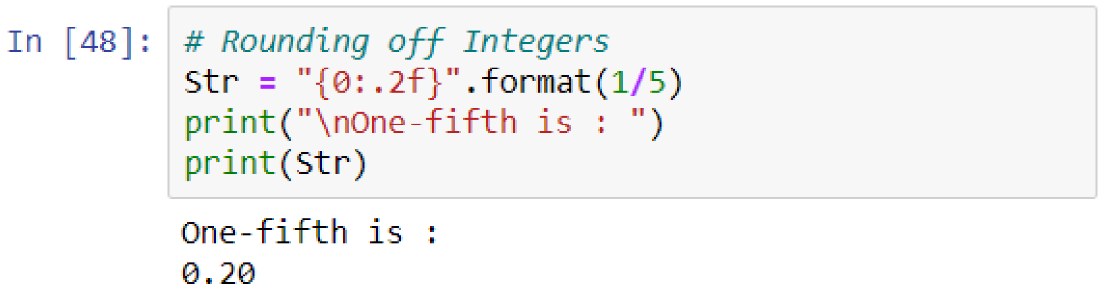
Example:



```
In [44]: # Formatting of Integers
         Str = "{0:b}".format(5)
         print("\nBinary representation of 5 is ")
         print(Str)

         Binary representation of 5 is
         101
```



```
In [45]: # Formatting of Floats
         Str = "{0:e}".format(3.14)
         print("\nExponent representation of 3.14 is ")
         print(Str)

         Exponent representation of 3.14 is
         3.140000e+00
```



```
In [48]: # Rounding off Integers
         Str = "{0:.2f}".format(1/5)
         print("\nOne-fifth is : ")
         print(Str)

         One-fifth is :
         0.20
```

Video link: https://youtu.be/Ctqi5Y4X-jA

For more detail(optional): https://youtu.be/QGLNQwfTO2w

References:-

1. https://www.tutorialspoint.com/python/python_strings.htm
2. https://www.hackerearth.com/practice/python/getting-started/string/tutorial/
3. https://www.geeksforgeeks.org/python-strings/
4. https://www.w3schools.com/python/python_strings.asp
5. https://realpython.com/python-strings/