String formatting in python

String formatting can be done in python using the format method.

```
txt = "For only {price:.2f} dollars!"
print(txt.format(price = 49))
```

f-strings in python

It is a new string formatting mechanism introduced by the PEP 498. It is also known as Literal String Interpolation or more commonly as F-strings (f character preceding the string literal). The primary focus of this mechanism is to make the interpolation easier.

When we prefix the string with the letter 'f', the string becomes the f-string itself. The f-string can be formatted in much same as the str.format() method. The f-string offers a convenient way to embed Python expression inside string literals for formatting.

Example

```
val = 'Geeks'
print(f"{val}for{val} is a portal for {val}.")
name = 'Tushar'
age = 23
print(f"Hello, My name is {name} and I'm {age} years old.")
```

Output:

```
Hello, My name is Tushar and I'm 23 years old.
```

In the above code, we have used the f-string to format the string. It evaluates at runtime; we can put all valid Python expressions in them.

We can use it in a single statement as well.

Example

```
print(f"{2 * 30}"
```

Output:

Docstrings in python

Python docstrings are the string literals that appear right after the definition of a function, method, class, or module.

Example

```
def square(n):
   '''Takes in a number n, returns the square of n'''
   print(n**2)
square(5)
```

Here,

'''Takes in a number n, returns the square of n''' is a docstring which will not appear in output

Output:

```
25
```

Here is another example:

```
def add(num1, num2):
   """
   Add up two integer numbers.

   This function simply wraps the ``+`` operator, and does not
   do anything interesting, except for illustrating what
   the docstring of a very simple function looks like.

   Parameters
   ----------
   num1 : int
      First number to add.
   num2 : int
      Second number to add.

   Returns
   -------
   int
      The sum of ``num1`` and ``num2``.

   See Also
   --------
   subtract : Subtract one integer from another.

   Examples
   --------
   >>> add(2, 2)
```

```
4
>>> add(25, 0)
25
>>> add(10, -10)
0
"""
return num1 + num2
```

Python Comments vs Docstrings

Python Comments

Comments are descriptions that help programmers better understand the intent and functionality of the program. They are completely ignored by the Python interpreter.

Python docstrings

As mentioned above, Python docstrings are strings used right after the definition of a function, method, class, or module (like in Example 1). They are used to document our code.

We can access these docstrings using the **doc** attribute.

Python doc attribute

Whenever string literals are present just after the definition of a function, module, class or method, they are associated with the object as their **doc** attribute. We can later use this attribute to retrieve this docstring.

Example

```
def square(n):
   '''Takes in a number n, returns the square of n'''
   return n**2

print(square.__doc__)
```

Output:

```
Takes in a number n, returns the square of n
```

PEP 8

PEP 8 is a document that provides guidelines and best practices on how to write Python code. It was written in 2001 by Guido van Rossum, Barry Warsaw, and Nick Coghlan. The primary focus of PEP 8 is to improve the readability and consistency of Python code.

PEP stands for Python Enhancement Proposal, and there are several of them. A PEP is a document that describes new features proposed for Python and documents aspects of Python, like design and style, for the community.

The Zen of Python

Long time Pythoneer Tim Peters succinctly channels the BDFL’s guiding principles for Python’s design into 20 aphorisms, only 19 of which have been written down.

Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

Easter egg

import this

Recursion in python

Recursion is the process of defining something in terms of itself.

Python Recursive Function

In Python, we know that a function can call other functions. It is even possible for the function to call itself. These types of construct are termed as recursive functions.

Example:

```
def factorial(num):
   if (num == 1 or num == 0):
      return 1
   else:
      return (num * factorial(num - 1))

# Driver Code
num = 7;
print("Number: ",num)
print("Factorial: ",factorial(num))
```

Output:

```
number:  7
Factorial:  5040
```