String Interpolation in Python

String Interpolation refers to the technique of inserting values of variables in place of placeholders in a string. Using string interpolation, one can dynamically insert values inside a string. In Python, there are four ways using which we can perform string interpolation, namely, modulo (%), `format()` method, formatted strings, and template class. In this article, we will learn in detail about these methods.

String Interpolation using the Modulo (%) in Python

In Python, we can use the Modulo (%) for formatting strings. This approach is similar to the `printf()` in the `C` programming language. Following are the placeholders we can use inside strings for formatting.

```
%d = Integer
%f = Float
%s = String
%x = Hexadecimal
%o = Octal
```

Note that it is necessary to place a modulo character before characters inside strings to be considered as placeholders. Following is the syntax for using Modulo (%).

```
"<string with placeholders>" % (<comma separated values or variables>)
```

Now that we are done with some brief introduction let us understand how to use this concept practically with the help of some examples. Refer to the following Python code for the same.

```
a = 2000
b = "Hi"
c = "Python"
d = True
e = 3.14
print("%d + 1000 = 3000" % (a))
print("%s is a web framework written in %s" % ("Django", c))
print("[%d, %s, %s, %s, %f]" % (a, b, c, d, e))
print("%s! My favourite programming language is %s" % (b, c))
print("The value of PI or π: %f" % (e))
```

Output:

```
2000 + 1000 = 3000
Django is a web framework written in Python
[2000, Hi, Python, True, 3.140000]
```

```
Hi! My favourite programming language is Python
The value of PI or π: 3.140000
```

Note how placeholders are used inside strings for formatting and what placeholder is used for what kind of value.

String Interpolation using the `format()` method in Python

In Python, the `format()` method can br used for formatting strings. This method is similar to the previous one but here, `{}` acts as a placeholder for every type of value. Following is the syntax for the `format()` method.

```
"<string with placeholdes>".format(<comma separated values and variables>)
```

Refer to the following Python code to understand this concept better with the help of some relevant examples.

```
a = 2000
b = "Hi"
c = "Python"
d = True
e = 3.14
print("{} + 1000 = 3000".format(a))
print("{} is a web framework written in {}".format("Django", c))
print("[{}, {}, {}, {}, {}]".format(a, b, c, d, e))
print("{}! My favourite programming language is {}".format(b, c))
print("The value of PI or π: {}".format(e))
```

Output:

```
2000 + 1000 = 3000
Django is a web framework written in Python
[2000, Hi, Python, True, 3.14]
Hi! My favourite programming language is Python
The value of PI or π: 3.14
```

String Interpolation using the formatted strings in Python

Formatted strings are unique strings in Python. When prefixed with an `f` character, regular strings are known as formatted strings. Formatted strings are also known as f-strings. These strings are used to insert string representations of variables and objects inside strings dynamically. We can add `{}` inside strings, and inside these blocks, we can add variables or logic that return some value.

Following is the syntax for formatted strings.

```
```

```
f"{<variable>} {<variable>} {<variable>}"
```

Now let us understand this concept with the help of some relevant examples. Refer to the following Python code for the same.

```
def hello():
    return "Hello"

a = 2000
b = "Hi"
c = "Python"
d = True
e = 3.14
print(f"{a} + 1000 = 3000")
print(f"Django is a web framework written in {c}")
print(f"[{a}, {b}, {c}, {d}, {e}]")
print(f"{hello()}! My favourite programming language is {c}")
print(f"The value of PI or π: {e}")
print(f"1000 + 2000 = {1000 + 2000}")
print(f"10 * 20 - 25 = {10 * 20 - 25}")
```

Output:

```
2000 + 1000 = 3000
Django is a web framework written in Python
[2000, Hi, Python, True, 3.14]
Hello! My favourite programming language is Python
The value of PI or π: 3.14
1000 + 2000 = 3000
10 * 20 - 25 = 175
```

String Interpolation using the template class in Python

Python has an in-built module, `string`, which has a class `Template` that allows us to create dynamic strings. Using this method, we can perform `$`- based substitutions. For example, for this string `"Hello $name"`, `name` is a variable, and we can provide a value for this variable. We can use the `substitute()` of the `Template` class to substitute values. This method accepts a dictionary, where variables of the template string are keys, and they point to values. Instead of using a dictionary, we can also provide keyword arguments.

Note that if the key is not found, the Python interpreter will throw a `KeyError`. To avoid that, one can either make sure that the requried keys exist in the dictionary or we can use the `safe_substitute()` method, that will leave the string unchanged for that placeholder.

Now that we are done with some theory let us understand this concept with the help of some relevant examples. Refer to the following Python code for the same.

```
```

```
from string import Template

def pi():
    return 3.14

t1 = Template("$a + 1000 = 3000")
t2 = Template("$package is a web framework written in $language")
t3 = Template("$greeting! My favourite programming language is $language")
t4 = Template("[$a, $b, $c, $d, $e]")
t5 = Template("The value of PI or π: $pi")
print(t1.substitute({ "a": 2000 }))
print(t2.substitute({ "package": "Flask", "language": "Python" }))
print(t3.substitute({ "greeting": "Hey", "language": "Python" }))
print(t4.substitute(a = 2000, b = "Hi", c = "Python", d = True, e = 999.909))
print(t5.substitute(pi = pi()))
```

Output:

```
2000 + 1000 = 3000
Flask is a web framework written in Python
Hey! My favourite programming language is Python
[2000, Hi, Python, True, 999.909]
The value of PI or π: 3.14
```

Refer to the following Python code for the use of `safe_substitute()` method.

```
from string import Template

def pi():
    return 3.14

t1 = Template("$a + 1000 = 3000")
t2 = Template("$package is a web framework written in $language")
t3 = Template("$greeting! My favourite programming language is $language")
t4 = Template("[$a, $b, $c, $d, $e]")
t5 = Template("The value of PI or π: $pi")
print(t1.safe_substitute({ }))
print(t2.safe_substitute({ "package": "Flask" }))
print(t3.safe_substitute({ "language": "Python" }))
print(t4.safe_substitute(c = "Python", d = True, e = 999.909))
print(t5.safe_substitute())
```

Output:

```
$a + 1000 = 3000
Flask is a web framework written in $language
```

```
$greeting! My favourite programming language is Python
[$a, $b, Python, True, 999.909]
The value of PI or π: $pi
```