Formatting Strings

Print to PDF

Let us understand how we can replace placeholders in Python using different variations of formatting strings.

- While we can concatenate strings, it will by default fail when we try to concatenate string with non string.
- Conventionally we tend to use `str` to convert the data type of the non-string values that are being concatenated to string.
- From Python 3, they have introduced placeholders using `{}` with in a string.
- We can replace the placeholders either by name or by position.

Using placeholders will improve the readability of the code and it is highly recommended.

```
print('Hello World')
```

```
Hello World
```

```
print('Hello' + ' ' + 'World')
```

```
Hello World
```

```
# This will fail as we are trying to concatenate 0 to a string
print('Hello' + 0 + 'World')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-3-bf74b0067f9c> in <module>
      1 # This will fail as we are trying to concatenate 0 to a string
----> 2 print('Hello' + 0 + 'World')

TypeError: must be str, not int
```

```
print('Hello' + str(0) + 'World')
```

```
Hello0World
```

```
i = 0
print('Hello' + str(i) + 'World')
```

```
Hello0World
```

```
# Replacing placeholders by name
i = 0
print(f'Hello{i}World')
```

```
Hello0World
```

```
# Replacing placeholders by name
print('Hello{i}World'.format(i=0))
```

```
Hello0World
```

```
# Replacing placeholders by position
print('Hello{}World'.format(0))
```

```
Hello0World
```

```
# Replacing placeholders by position
print('Hello{}World{}'.format(0, 1))
```

```
Hello0World1
```

```python
# These are the approaches which are commonly used
i = 0
s1 = f'str1: Hello{i}World'
print(s1)
s2 = 'str2: Hello{j}World'.format(j=i)
print(s2)
```

```
str1: Hello0World
str2: Hello0World
```