Often you will want to "inject" a variabe into your string for printing. For example: my_name="Jose" print ("Hello" + my_name)

There are multiple ways to format strings for printing variables in them This is known as string interpolation (sticking a variable into a string)

Lets explore two methods for this: .format() method f-strings (formatted string literals)

Formatting with the .format() method A good way to format objects into your string for print statements is with the string .format() method. The syntax is:

'String here {} then also {}'.format('something1','something2')

In [1]:
```
print('This is a string {}'.format('INSERTED'))
```
```
This is a string INSERTED
```

So what we see above is the 'INSERTED' string was inputted into 'This is a string'

Strings can also be inputted into a specific format position

In [2]:
```
print('The {} {} {}' .format('fox', 'brown', 'quick'))
```
```
The fox brown quick
```

It will input the strings in the same order you supplied them so {fox} {brown} {quick} But we want this to be gramatically correct so we can output based on position like so:

In [3]:
```
print('The {2} {1} {0}' .format('fox', 'brown', 'quick'))
```
```
The quick brown fox
```

In [4]:
```
print('The {0} {0} {0}' .format('fox', 'brown', 'quick'))
```
```
The fox fox fox
```

We can do index position as pictured above, however we can also position based on key words

In [6]:
```
print('The {q} {b} {f}' .format(f='fox', b='brown', q='quick'))
```
```
The quick brown fox
```

In [7]:
```
print('The {f} {f} {f}' .format(f='fox', b='brown', q='quick'))
```
```
The fox fox fox
```

Float formatting follows "{value:width.precision f}" This allows us to adjust the width and precision of the floating point number

In [8]:
```
result = 100/777
```

In [9]:
```
result
```
Out[9]:
```
0.1287001287001287
```

In [11]:
```
print("The result was {}" .format(result))
```
```
The result was 0.1287001287001287
```

In [13]:
```
print("The result was {r}" .format(r=result))
```
```
The result was 0.1287001287001287
```

In [14]:
```
print("The result was {r:1.3f}" .format(r=result))
```
```
The result was 0.129
```

By doing {r:1.3f} above, we set: r = result width = 1 precision=3 (3 places past the decimal point)

In [15]:
```
print("The result was {r:10.3f}" .format(r=result))
```
```
The result was      0.129
```

In [16]:
```
print("The result was {r:1.5f}" .format(r=result))
```
```
The result was 0.12870
```

Lets do another example where result is equal to 104.12345

In [17]:
```
result =104.12345
```

In [18]:
```
print("The result was {r:1.5f}" .format(r=result))
```
```
The result was 104.12345
```

In [ ]:
```

```

f-strings (formatted string literals) examples:

In [19]:
```
name = "Jose"
```

In [20]:
```
print (f"Hello, his name is {name}")
```
```
Hello, his name is Jose
```

In [33]:
```
name = "Sam"
age = 3
```

In [38]:
```
print (f'{name} is {age} years old')
```
```
Sam is 3 years old
```