

String Formatting and Formatted Output

ENGR 102

Purpose

- Start taking control of printed output
- Printed output is a form of communication
 - A nice format can make that communication more effective
- By controlling the number of decimal places displayed, we can communicate the proper number of significant digits for a particular calculation

String Formatting

- There are many (thousands) of ways to control the format of strings in Python
- Included here is a discussion of how to use f-strings to control string format
- Feel free to investigate other methods for formatting strings
 - See zyBooks sections 3.9, 3.12, 7.16, and 8.15
- Learn at least one method to control the number of decimal places of printed data

Strategy for Printing Formatted Numbers

- First format an f-string, then print the string

```
mynum = 123.456789  
mystr = f'The value of mynum is {mynum:6.2f}'  
print(mystr)
```

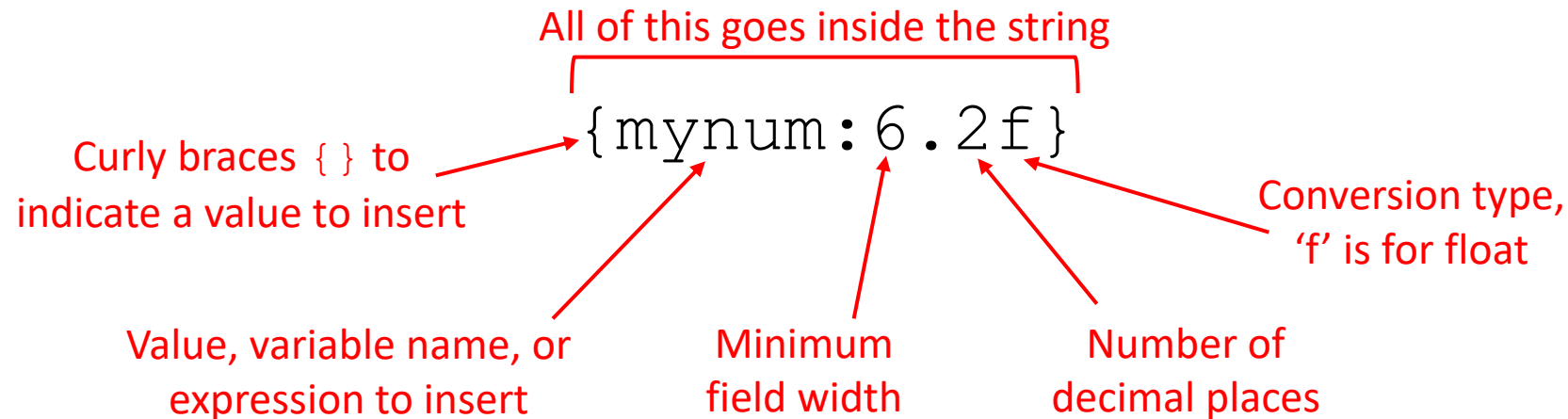
Output:

```
The value of mynum is 123.46
```

Conversion Specifier

```
mystr = (f)'The value of mynum is {mynum:6.2f}'
```

Note the f goes
before the string



Conversion Specifier

```
mynum = 123.456789  
mystr = f'The value of mynum is {mynum:6.2f}'  
print(mystr)
```

Output:

The value of mynum is 123.46

In this case, Python formatted the value of mynum to a field width of 6 and with 2 decimal places

Note the value is rounded up rather than truncated

Using Conversion Specifiers in `print()`

- Instead of creating a formatted string, then printing the string, we can put all of that in the `print()` function call

```
mynum = 123.456789  
print(f'The value of mynum is {mynum:6.2f}')
```

Output:

```
The value of mynum is 123.46
```

Printing more than one value at once

- We can print multiple values with multiple conversions

```
num1 = 123.456789
```

```
num2 = 867.5309
```

```
num3 = -0.0235
```

```
print(f'num1 = {num1:6.2f} num2 = {num2:7.1f} num3 = {num3:8.3f}')
```

Output:

```
num1 = 123.46 num2 = 867.5 num3 = -0.024
```


Other possible conversion specifiers

Value	Presentation Type	Example
b	Binary integer	11101
d	Decimal integer	29
e	Exponential	1.02e+02
f	Floating point	102.0
s	String	'12'
x	Hexadecimal integer	1d

Mixed example


```
A = 123.45678
```

```
B = 0.00008675309
```

```
C = 2029
```

```
D = 'Howdy!'
```

```
print(f'Here are all four values: {A:.2f} {B:.3e} {C:d} {D:s}')
```



Output:

```
Here are all four values: 123.46 8.675e-05 2029 Howdy!
```

Formatting a Table of Data

```
from math import sin, cos, pi

Theta = "theta"
Func1 = "cos()"
Func2 = "sin()"

theta1 = 0 # radians
theta2 = pi/6 # radians
theta3 = pi/4 # radians
theta4 = pi/3 # radians
theta5 = pi/2 # radians

# a good first guess is to match the width of the labels with the width of the data fields
print(f"{Theta:6s} {Func1:6s} {Func2:6s}")
print(f"'(rad)':6s {'----':6s {'----':6s}")

print(f"{theta1:<6.2f} {cos(theta1):<6.2f} {sin(theta1):<6.2f}") # use < for left justify
print(f"{theta2:<6.2f} {cos(theta2):<6.2f} {sin(theta2):<6.2f}") # use > for right justify
print(f"{theta3:<6.2f} {cos(theta3):<6.2f} {sin(theta3):<6.2f}") # use ^ for center
print(f"{theta4:<6.2f} {cos(theta4):<6.2f} {sin(theta4):<6.2f}") # strings are default left justified
print(f"{theta5:<6.2f} {cos(theta5):<6.2f} {sin(theta5):<6.2f}") # numbers are default right justified11
```

Printed Results

- The code from the previous slide produces this output
- The decimal points are aligned and the numbers are printed to two decimal place accuracy
- This is a lot easier using loops – be patient – we'll get there

theta	cos()	sin()
(rad)	----	----
0.00	1.00	0.00
0.52	0.87	0.50
0.79	0.71	0.71
1.05	0.50	0.87
1.57	0.00	1.00

Formatting a Table of Data (another example)

```
# try a phone list
First1 = "Emma"
Last1 = "Smith"
Phone1 = "(555)234-5678"
First2 = "Noah"
Last2 = "Johnson"
Phone2 = "(555)234-2233"
First3 = "Olivia"
Last3 = "Williams"
Phone3 = "(555)234-9826"

print(f"{First1:>8s} {Last1:>10s} {Phone1:>14s}")
print(f"{First2:>8s} {Last2:>10s} {Phone2:>14s}")
print(f"{First3:>8s} {Last3:>10s} {Phone3:>14s}")
```

Printed Results

- The code from the previous slide produces this output
- Here, the names are justified
- We'll learn how to take more control over this behavior in the coming weeks

Emma	Smith	(555) 234-5678
Noah	Johnson	(555) 234-2233
Olivia	Williams	(555) 234-9826

More things to try

```
from math import pi
```

```
precision = 30
```

```
print(f"pi = {pi:.{precision}f}")
```

```
print(f'{2**30:,d} bytes in arrays saved')
```

```
print(f"{3**3+2=} in binary is {3**3+2:b}")
```

Moving Forward

- Future assignments will require you to print the output in a specified format
- Please get in the habit of creating nicely formatted output for all of your programs