Formatted Strings and Type Conversions in Python 3

This document contains examples of formatted strings, including integers, floats, hexadecimal, octal, binary representation, and type conversions in Python 3 and above. We will cover different formatting types using format(), f-strings, and type conversions.

1. Basic String Formatting

```
name = "Reece"
age = 22

# Using format()
"My name is {} and I am {} years old.".format(name, age)

# Using f-strings (Python 3.6+)
f"My name is {name} and I am {age} years old."
```

2. Integer Formatting

```
num = 42

# Default
"The number is {}".format(num)

# Right aligned with width of 5
"{:5}".format(num)

# Left aligned with width of 5
"{:<5}".format(num)
```

Center aligned with width of 5

"{:^5}".format(num)

3. Float and Decimal Formatting

pi = 3.141592653589793

Default

"Pi is {:.2f}".format(pi)

f-string with precision

f"Pi to 4 decimal places: {pi:.4f}"

Exponential notation

f"Pi in scientific notation: {pi:.2e}"

4. Hexadecimal, Octal, and Binary

num = 255

Hexadecimal

f"Hexadecimal (lowercase): {num:x}"

f"Hexadecimal (uppercase): {num:X}"

Octal

f"Octal: {num:o}"

Binary

f"Binary: {num:b}"

Including prefix

f"Hexadecimal with prefix: {num:#x}"

f"Octal with prefix: {num:#o}"

f"Binary with prefix: {num:#b}"

5. Percentage Formatting

value = 0.25

Default percentage

f"Percentage: {value:.0%}"

Percentage with two decimal places

f"Percentage (2 decimal): {value:.2%}"

6. Padding with Zeros

num = 42

Zero padded to width 5

f"Zero-padded: {num:05}"

7. Thousands Separator

large_number = 1234567890

Adding commas for thousands

f"With commas: {large_number:,}"

```
# With decimal

num = 1234.5678

f"Formatted with commas and 2 decimals: {num:,.2f}"
```

8. Aligning Text

```
text = "Hello"


# Left align

f"Left aligned: '{text:<10}'"


# Right align

f"Right aligned: '{text:>10}'"


# Center aligned

f"Center aligned: '{text:^10}'"
```

9. Custom Fill Characters

```
num = 42


# Right align with dots

f"Right aligned with dots: '{num:.>10}'"


# Left align with stars

f"Left aligned with stars: '{num:*<10}'"
```

10. Floating-Point with Specific Width and Precision

```
value = 1234.56789

# Floating-point with width 10, precision 2
f"Width 10, precision 2: {value:10.2f}"

# Center align with specific width and precision
f"Center align: {value:^10.2f}"
```

11. Handling Negative and Positive Signs

```
num_pos = 42
num_neg = -42

# Show + for positive numbers
f"Positive sign: {num_pos:+d}, Negative sign: {num_neg:+d}"

# Default behavior (no + for positive)
f"Default sign: {num_pos:d}, {num_neg:d}"
```

12. Type Conversions (Binary, Hex, Octal, Decimal)

```
# Binary to decimal
bin_num = '1010'
dec_from_bin = int(bin_num, 2)

# Hexadecimal to decimal
hex_num = 'a'
dec_from_hex = int(hex_num, 16)
```

```
# Octal to decimal
oct_num = '12'
dec_from_oct = int(oct_num, 8)

# Decimal to binary
decimal = 10
binary = bin(decimal)

# Decimal to hexadecimal
hexadecimal = hex(decimal)

# Decimal to octal
octal = oct(decimal)

# Example conversions
print(f"Binary 1010 to decimal: {dec_from_bin}")  # 10
print(f"Hexadecimal 'a' to decimal: {dec_from_hex}")  # 10
print(f"Octal '12' to decimal: {dec_from_oct}")  # 10
print(f"Decimal 10 to binary: {binary}")  # '0b1010'
print(f"Decimal 10 to hexadecimal: {hexadecimal}")  # '0xa'
print(f"Decimal 10 to octal: {octal}")  # '0o12'
```