Chapter 3

Strings

Strings

- In Python (as in most object-oriented languages) a string is an object of a string class, but it is set up to act as a sequence
- It holds a collection of individual items, in this case letters
- Placed sequentially in memory
- Items are numbered starting with 0

Creating strings

- Already familiar with the following:

```
name = input(“Enter your name:  ”)
```

- You may also store literals

```
salutation = “To whom it may concern”
```

Strings in memory

- Each item is stored in an individual memory space sequentially
- May be accessed individually using the index
- The first index (first character in the string) is 0
- All other indices are the number of steps away from the first one

Memory Example

className = “CSE 1284”

Values stored	Index
‘C’	**0**
‘S’	1
‘E’	2
‘ ’	3
‘1’	4
‘2’	5
‘8’	67
‘4’	

Accessing a single character

- Use the variable name (really an object name, but we’ll cover that when we get to classes) and the index.
- Example:

```
className = “CSE 1284”
firstCharacter = className[0]
```

Number of characters in a string

- len() function
- Example:

```
numberCharacters = len(className)
```

Formatting output (f-strings)

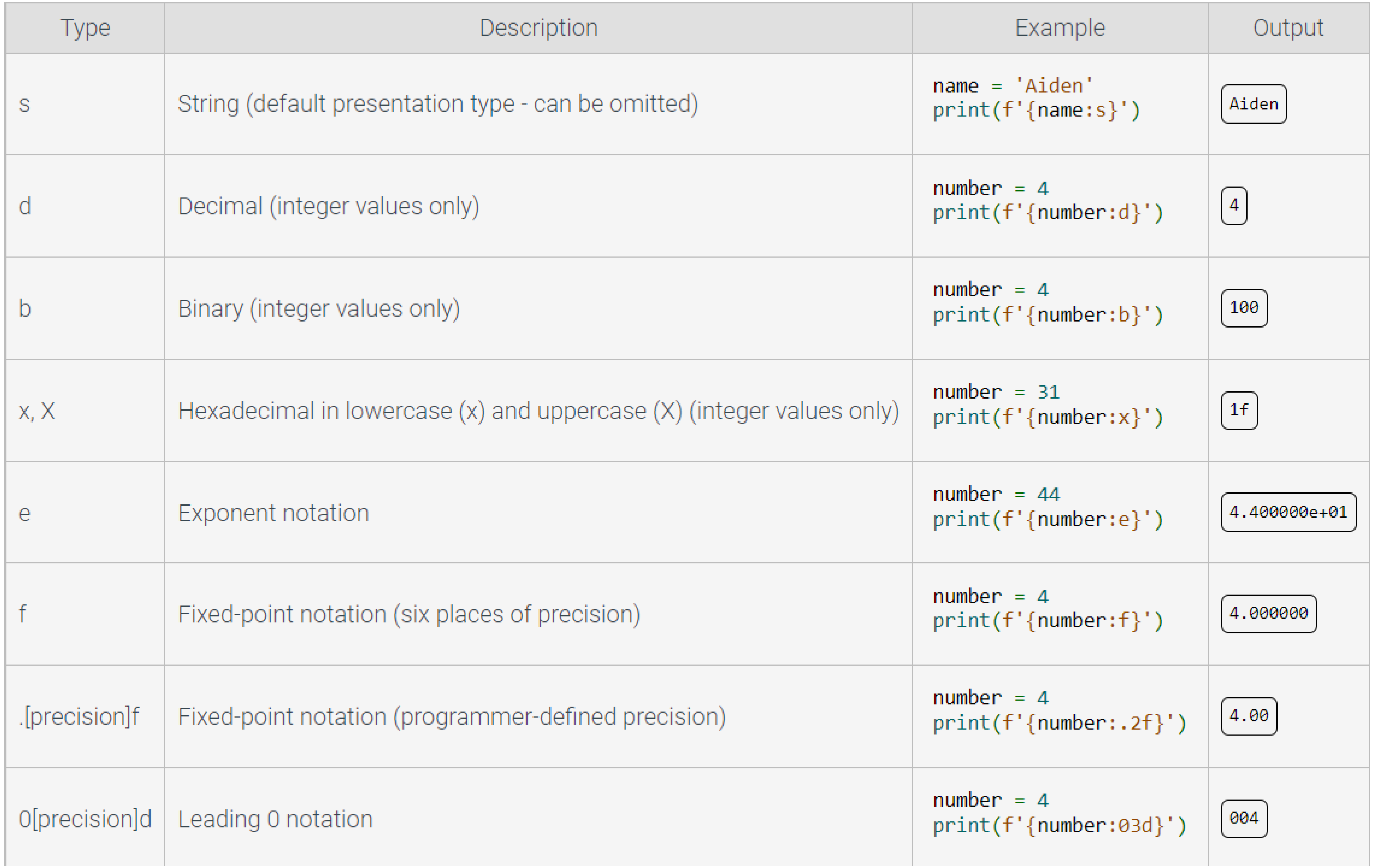
```
print(f'{variable} text to be printed')
```

Expression or value to
be printed found in {}

Example:

```
x = 5
y = 6
print(f'The point has coordinates ({x}, {y}).’)
```

Prints: The point has coordinates (5, 6).



Type	Description	Example	Output

s	String (default presentation type - can be omitted)	`name = 'Aiden'`
		`print(f'{name:s}')`
			`Aiden`

d	Decimal (integer values only)	`number = 4`
		`print(f'{number:d}')`
			`4`

b	Binary (integer values only)	`number = 4`
		`print(f'{number:b}')`
			`100`

x, X	Hexadecimal in lowercase (x) and uppercase (X) (integer values only)	`number = 31`
		`print(f'{number:x}')`
			`1f`

e	Exponent notation	`number = 44`
		`print(f'{number:e}')`
			`4.400000e+01`

f	Fixed-point notation (six places of precision)	`number = 4`
		`print(f'{number:f}')`
			`4.000000`

.[precision]f	Fixed-point notation (programmer-defined precision)	`number = 4`
		`print(f'{number:.2f}')`
			`4.00`

0[precision]d	Leading 0 notation	`number = 4`
		`print(f'{number:03d}')`
			`004`