Code Academy

# Chapter Two

PROGRAMMING WITH NUMBERS AND STRINGS

Sections 2.5
Input and Output

Code Academy

## Lecture Goals

- To create programs that read, and process inputs, and display the results

Code Academy

## 2.5 Input and Output

Code Academy

# User Input

- You can read a String from the console with the input() function:

```
name = input("Please enter your name")
```

  - The input function displays the string argument in the console window and places the cursor on the same line immediately following the string.
  - The program waits until the user types a name and when the **Enter** key is pressed
  - The sequence of characters are returned as string from the function input and stored in the variable name

- Converting a String variable to a number can be used if numeric (rather than string input) is needed

```
  age = int(input("Please enter age: "))

Or

   aString = input("Please enter age: ") # String input
   age = int(aString)                  # Converted to int
```

Code Academy

## Output

- You can display or print a line of text on the console with the print() function:

```
print("Hello World”)
```

- You can also prints numerical values. For example, the statement

```
print(3+4)
```

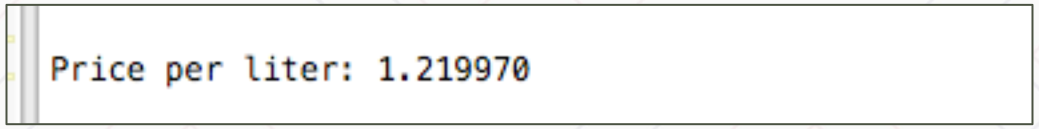
- You can pass multiple values to the function. For example,

```
print("The answer is", 6 * 7)     #The answer is 42
```

# Formatted output

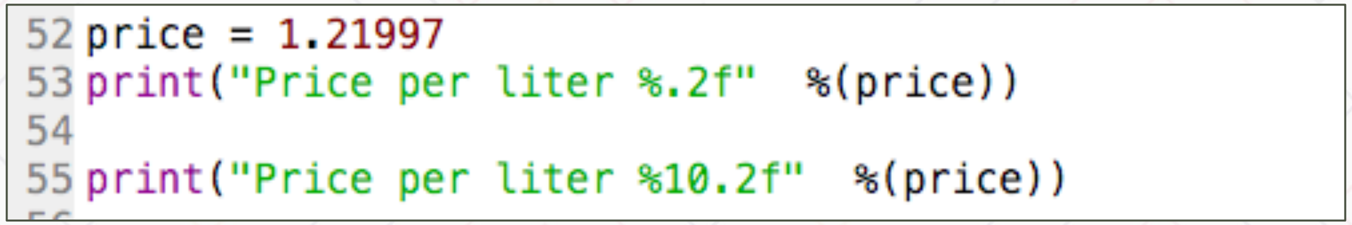
Code Academy

- Outputting floating point values can look strange:



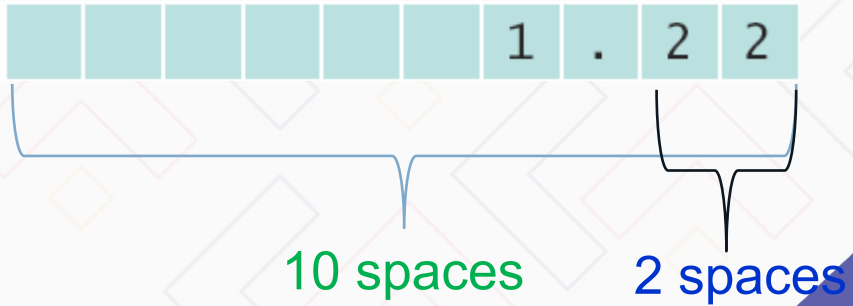
```
Price per liter: 1.219970
```

- To control the output appearance of numeric variables, use formatted output tools such as:

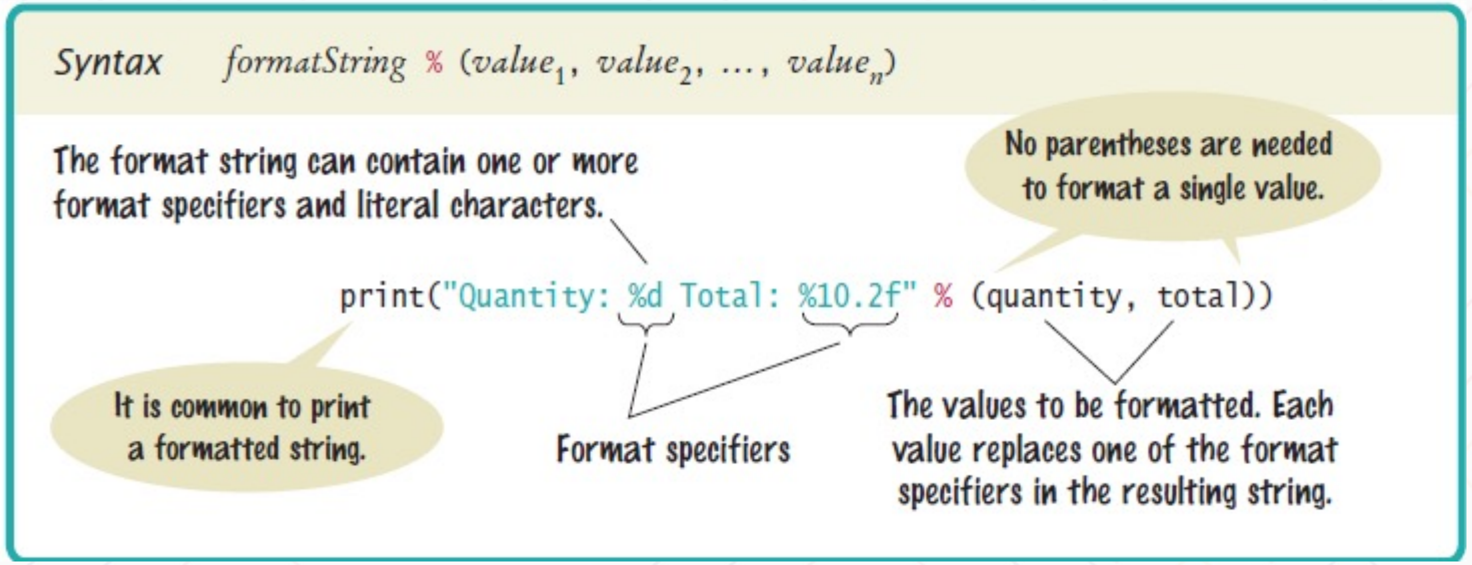


```
price = 1.21997
print("Price per liter %.2f"  %(price))

print("Price per liter %10.2f"  %(price))
```

The `%10.2f` is called a format specifier



# Syntax: formatting strings



*Syntax* *formatString* % (*value*$_1$, *value*$_2$, …, *value*$_n$)

```
print("Quantity: %d Total: %10.2f" % (quantity, total))
```

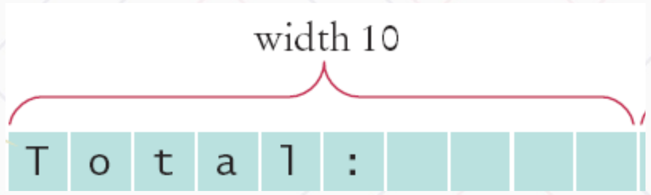
The format string can contain one or more format specifiers and literal characters.
No parentheses are needed to format a single value.
It is common to print a formatted string.
Format specifiers
The values to be formatted. Each value replaces one of the format specifiers in the resulting string.

# Format **flag** examples

Code Academy

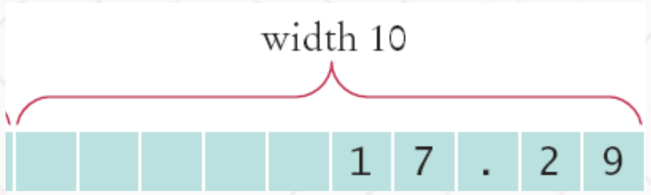
- Left Justify a String:

```
print("%-10s" %("Total:"))
```



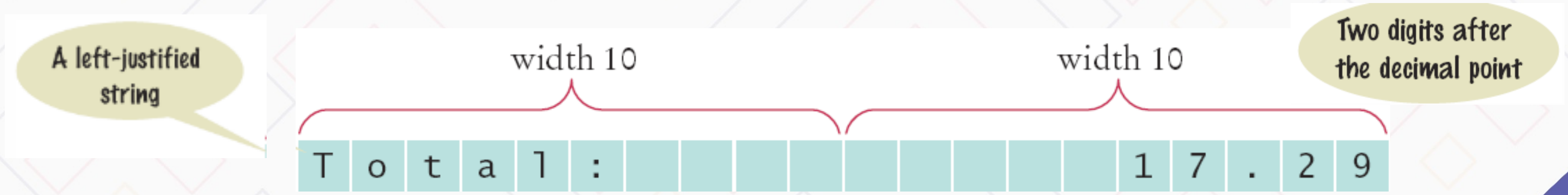
- Right justify a number with two decimal places

```
print("%10.2f" %(price))
```



- Print multiple values:

```
print("%-10s%10.2f" %("Total: ", price))
```



Code Academy

# Format Specifier Examples

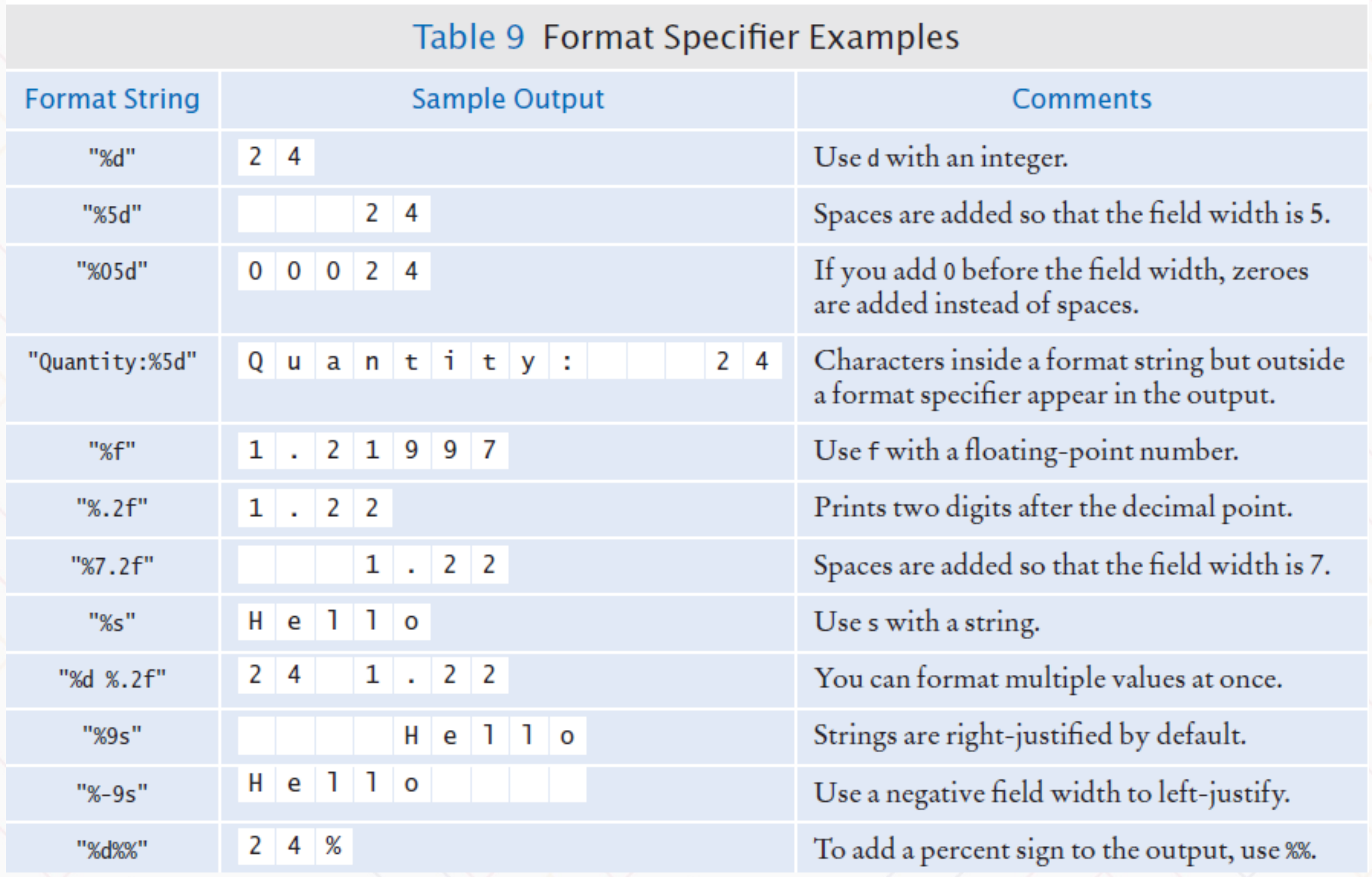
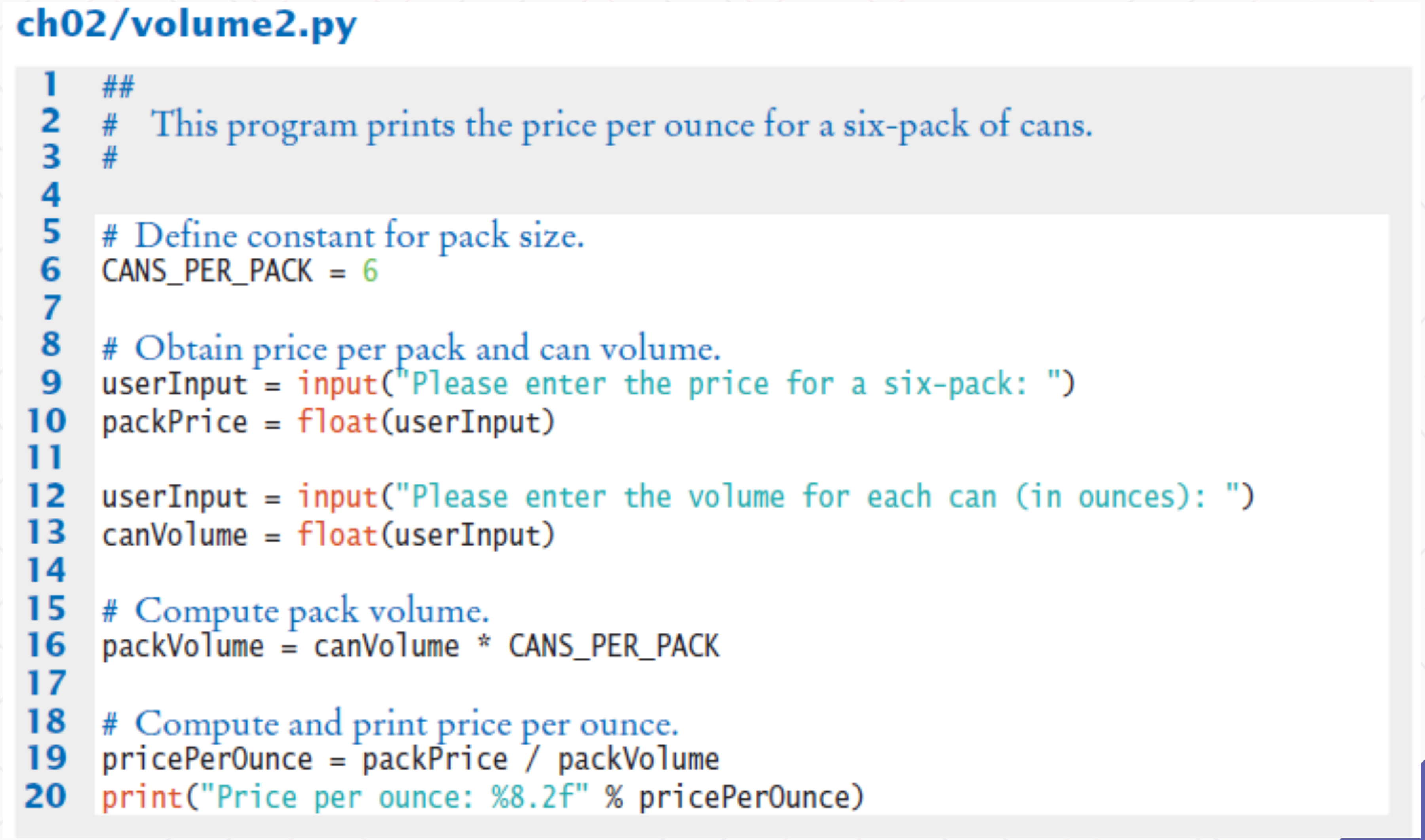


Table 9 Format Specifier Examples

| Format String | Sample Output | Comments |
|---|---|---|
| "%d" | `24` | Use d with an integer. |
| "%5d" | `   24` | Spaces are added so that the field width is 5. |
| "%05d" | `00024` | If you add 0 before the field width, zeroes are added instead of spaces. |
| "Quantity:%5d" | `Quantity:   24` | Characters inside a format string but outside a format specifier appear in the output. |
| "%f" | `1.21997` | Use f with a floating-point number. |
| "%.2f" | `1.22` | Prints two digits after the decimal point. |
| "%7.2f" | `   1.22` | Spaces are added so that the field width is 7. |
| "%s" | `Hello` | Use s with a string. |
| "%d %.2f" | `24 1.22` | You can format multiple values at once. |
| "%9s" | `    Hello` | Strings are right-justified by default. |
| "%-9s" | `Hello    ` | Use a negative field width to left-justify. |
| "%d%%" | `24%` | To add a percent sign to the output, use %%. |

# Input and Output Example



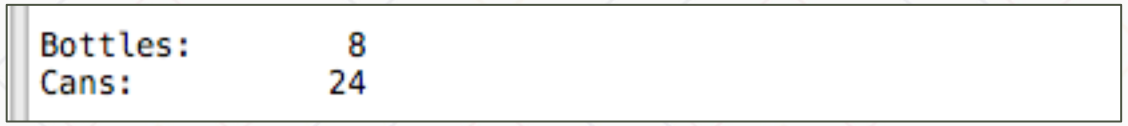
ch02/volume2.py

```
##
#  This program prints the price per ounce for a six-pack of cans.
#

# Define constant for pack size.
CANS_PER_PACK = 6

# Obtain price per pack and can volume.
userInput = input("Please enter the price for a six-pack: ")
packPrice = float(userInput)

userInput = input("Please enter the volume for each can (in ounces): ")
canVolume = float(userInput)

# Compute pack volume.
packVolume = canVolume * CANS_PER_PACK

# Compute and print price per ounce.
pricePerOunce = packPrice / packVolume
print("Price per ounce: %8.2f" % pricePerOunce)
```

## Example

Code Academy

- Using the string format operator, print the values of the variables `bottles` and `cans` so that the output looks like this:



```
Bottles:       8
Cans:         24
```

The numbers to the right should line up. (You may assume that the numbers are integers and have at most 8 digits.)

```
print("Bottles: %8d" %bottles)

print("Cans:    %8d" %cans)

Or

print("%-8s %8d" %("Bottles:", bottles))

print("%-8s %8d" %("Cans:", cans))
```

Code Academy

# Summary: python overview

- You can convert between integers, floats and strings using the respective functions: int(), float(), str()
- Python libraries are grouped into modules. Use the import statement to use methods from a module.
- Use the input() function to read keyboard input in a console window.
- Use the format specifiers to specify how values should be formatted.