MSE800

Professional Software Engineering

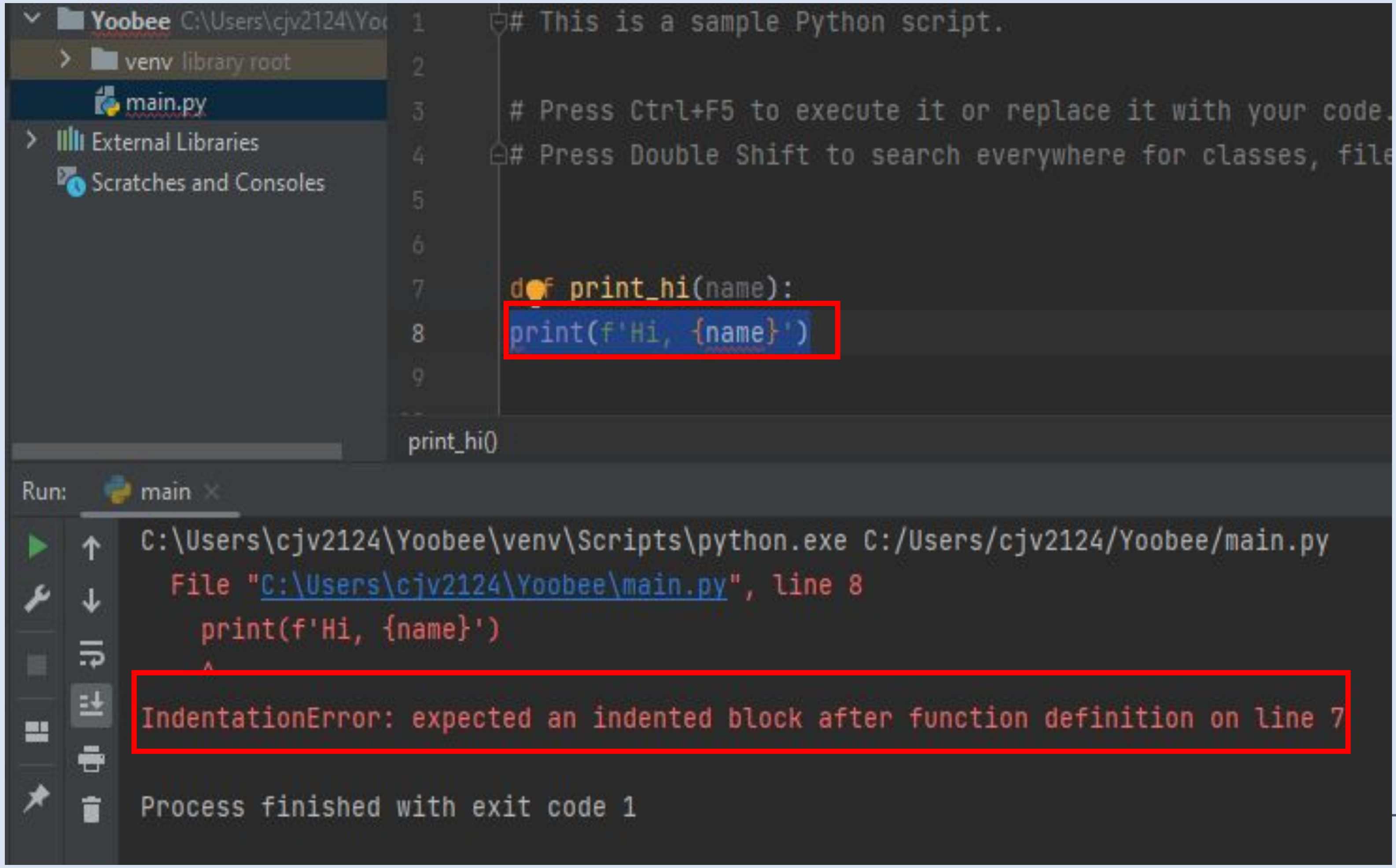
Programming Basic

Course Guideline:

- Programming Basics
  - Indentation
  - Variables
  - Data Input and Output
  - Comments
  - Data Type
  - Math Operators
- Exercises

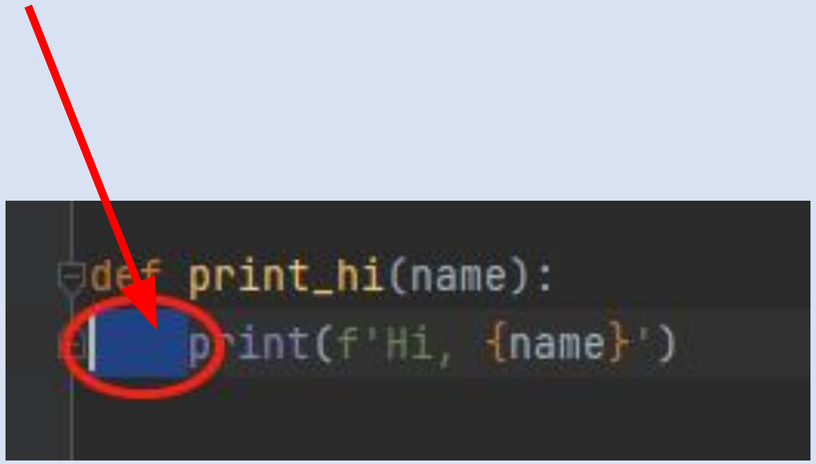
INDENTATION

Python uses indentation to indicate the block of code.



```
# This is a sample Python script.

# Press Ctrl+F5 to execute it or replace it with your code.
# Press Double Shift to search everywhere for classes, file

def print_hi(name):
print(f'Hi, {name}')
```

```
C:\Users\cjv2124\Yoobee\venv\Scripts\python.exe C:/Users/cjv2124/Yoobee/main.py
  File "C:\Users\cjv2124\Yoobee\main.py", line 8
    print(f'Hi, {name}')
    ^
IndentationError: expected an indented block after function definition on line 7

Process finished with exit code 1
```



```
def print_hi(name):
    print(f'Hi, {name}')
```

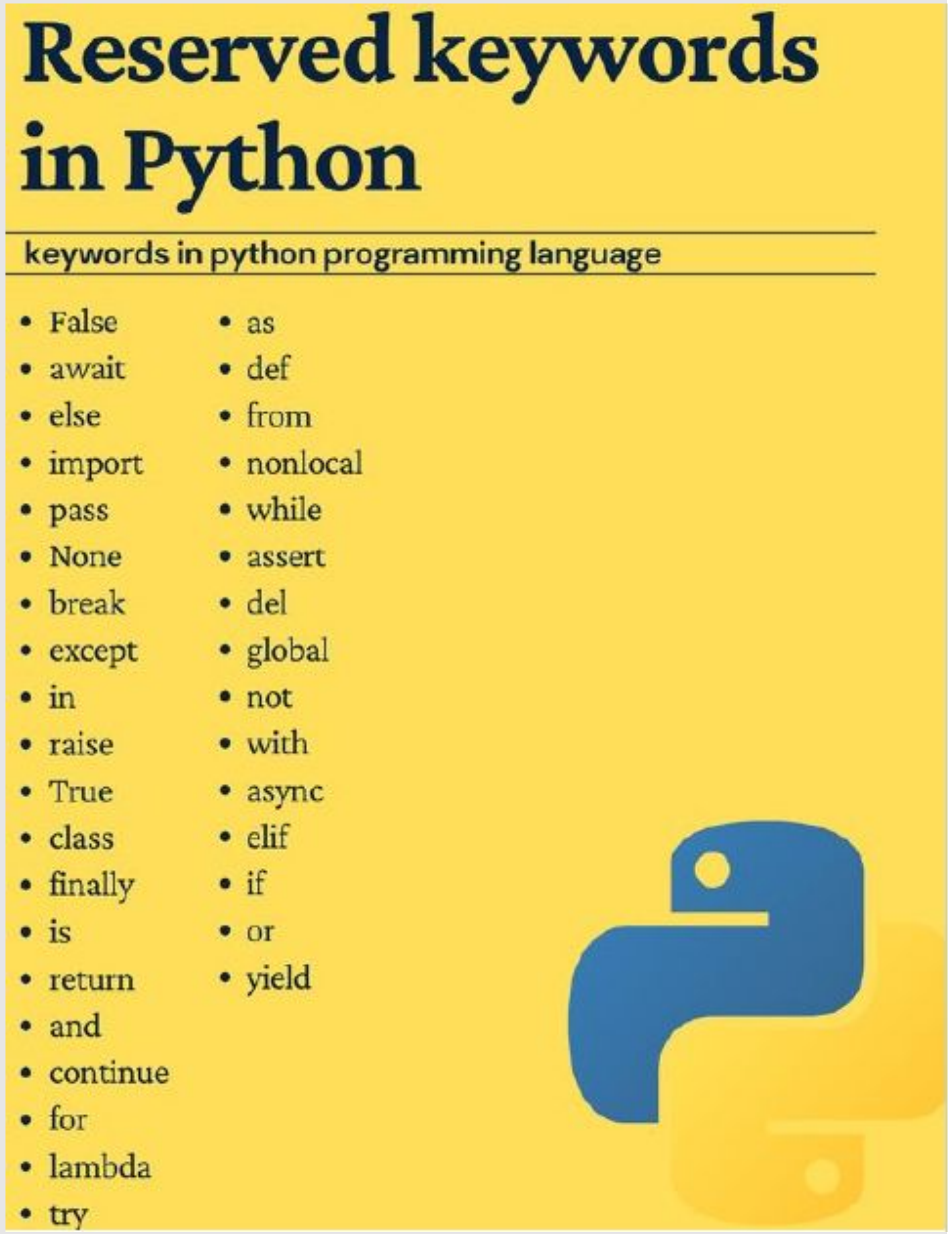
Variables

- **Variable: name that represents a value stored in the computer memory**
  - Used to access and manipulate data stored in memory
  - A variable references the value it represents
- **Assignment statement: used to create a variable and make it reference data**
  - General format is **variable = expression**
  - Example: age = 29
  - Assignment operator: the equal sign (=)
- **You can only use a variable if a value is assigned to it**

Variable Naming Rules

Rules for naming variables in Python:

- Variable name cannot be a Python keyword
- Variable name cannot contain spaces
- First character must be a letter or an underscore
- After first character, you may use (letters, digits, or underscores)
- Variable names should reflect their use
- Variable names are case-sensitive
  - Name, NAMA, and nAme are three different variables



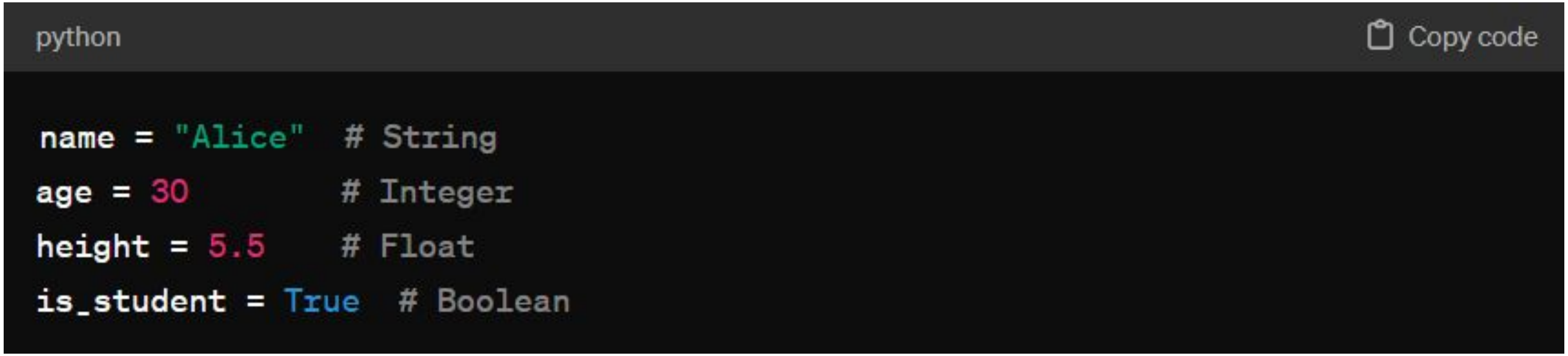
Reserved keywords in Python

keywords in python programming language

- False
- await
- else
- import
- pass
- None
- break
- except
- in
- raise
- True
- class
- finally
- is
- return
- and
- continue
- for
- lambda
- try
- as
- def
- from
- nonlocal
- while
- assert
- del
- global
- not
- with
- async
- elif
- if
- or
- yield

A variable in Python can refer to items of any type

- Python is a dynamic type language .
- The value of the variables is decided on runtime.



```python
name = "Alice"  # String
age = 30         # Integer
height = 5.5     # Float
is_student = True  # Boolean
```

Data Types

1. Built-in Data Types:

- **NoneType**: Represents the absence of a value, expressed as **None**.
- **Numeric Types**:
  - **int**: Integer type, e.g., **123**, **-456**, **0**.
  - **float**: Floating-point number, which has a decimal point, e.g., **123.45**, **-456.78**.
  - **complex**: Complex number, e.g., **1 + 2j**, **3 - 4j**.
- **Boolean Type**:
  - **bool**: Boolean value, either **True** or **False**.
- **Sequence Types**:
  - **str**: String type, e.g., **"Hello, World!"**.
  - **list**: List, a mutable and ordered sequence of items of different types, e.g., **[1, "a", True]**.
  - **tuple**: Tuple, an immutable sequence type, e.g., **(1, "a", True)**.
- **Set Types:**
  - **set**: Set, an unordered collection of unique items, e.g., **{1, 2, 3}**.
- **6. Mapping Type:**
  - **dict**: Dictionary, an unordered collection of key-value pairs, e.g., **{"key": "value", "number": 1}**.

2. Special Built-in Types:

- **range**: Represents an immutable sequence of numbers, typically used for looping.
- **bytes**: Byte type, e.g., **b'Hello'**.
- **bytearray**: A mutable array of bytes.

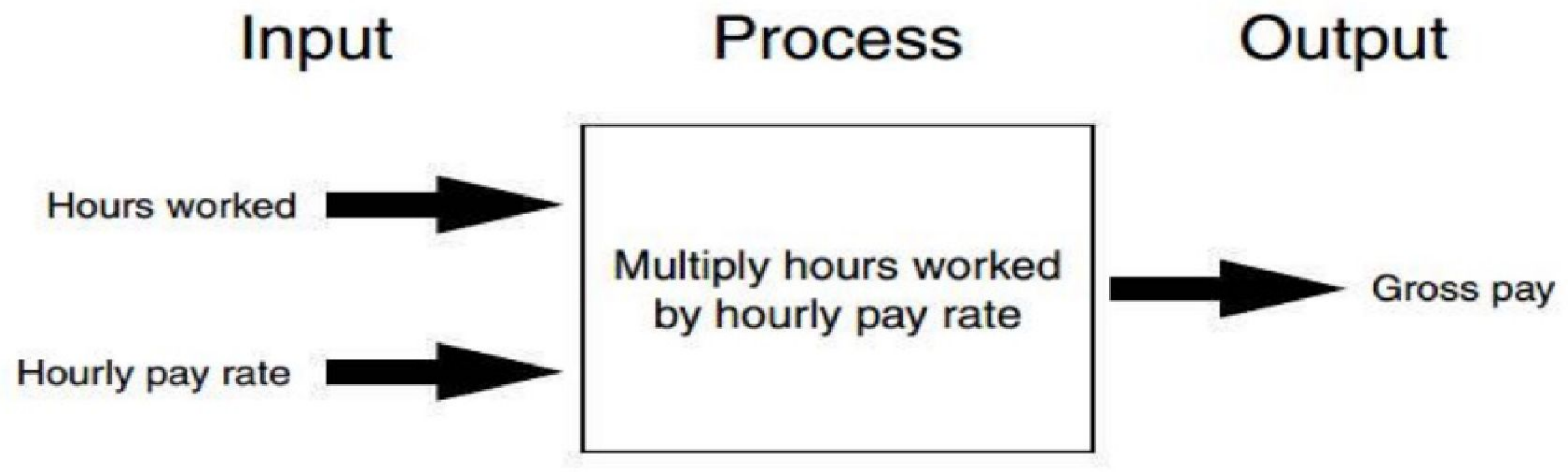
3. File Type:

- **file**: Used for file operations, created with the built-in **open** function.

Python's Standard Library also offers other data types, such as **decimal.Decimal, datetime.datetime**, and so on.

In addition, you can define your own data types (classes) and create objects from them; these are your custom data types.

Input, Processing, and Output



Typically, the computer performs a three-step process

- Receive input
    - Input: any data that the program receives while it is running
- Perform some process on the input
    - Example: mathematical calculation
- Produce output

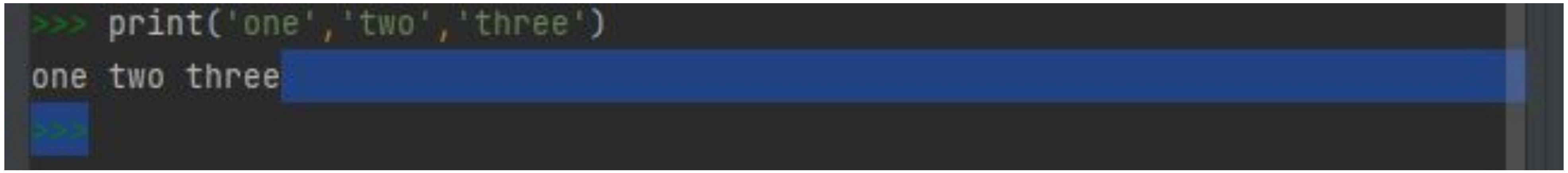
More About Data Output

print **function displays the line of output**

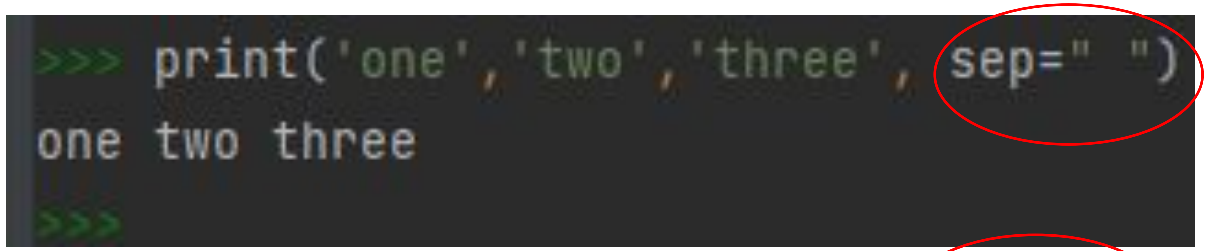
- ◦• Newline character (\n) at end of printed data
- ◦• Special argument end='*delimiter*' causes print to place *delimiter* at end of data instead of newline character

print **function uses space as an item separator**

- ◦• Special argument sep='*delimiter*' causes print to use *delimiter* as item separator



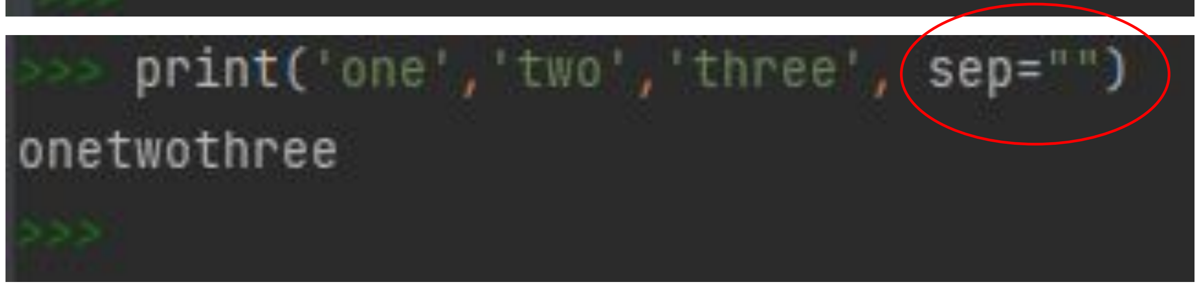
```
>>> print('one','two','three')
one two three
>>>
```



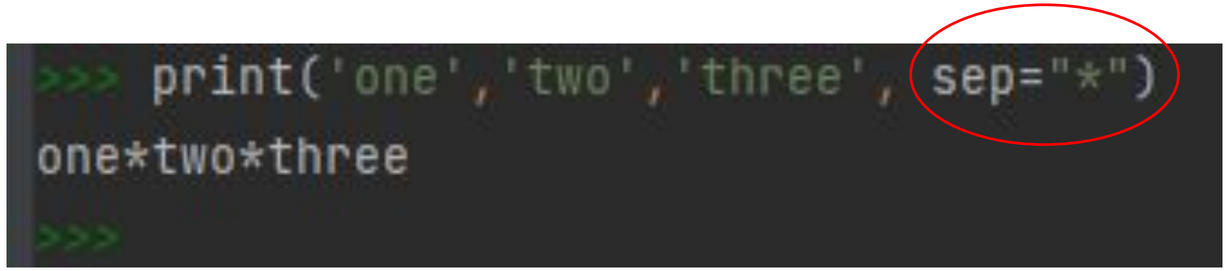
```
>>> print('one','two','three', sep=" ")
one two three
>>>
```

default

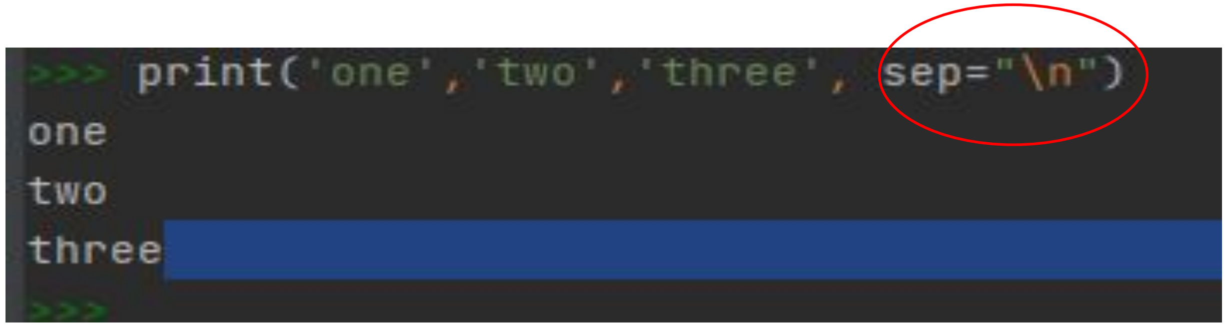
Same outputs



```
>>> print('one','two','three', sep="")
onetwothree
>>>
```



```
>>> print('one','two','three', sep="*")
one*two*three
>>>
```



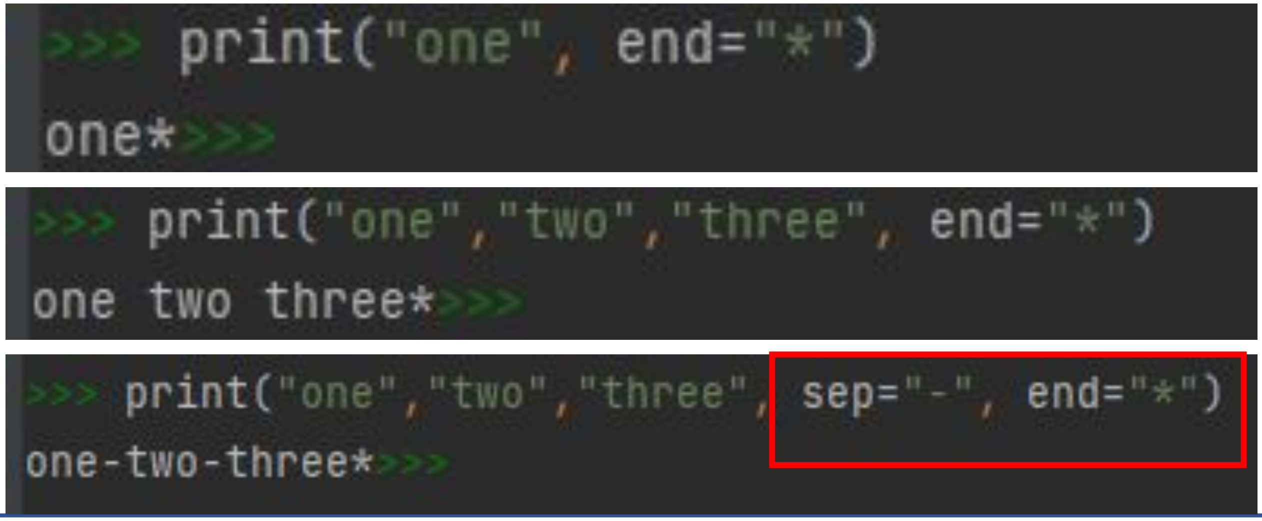
```
>>> print('one','two','three', sep="\n")
one
two
three
>>>
```

More About Data Output

print function displays the line of output

- Newline character at end of printed data
- Special argument end='*delimiter*' causes print to place *delimiter* at end of data instead of newline character

Examples:



```
>>> print("one", end="*")
one*>>>
```

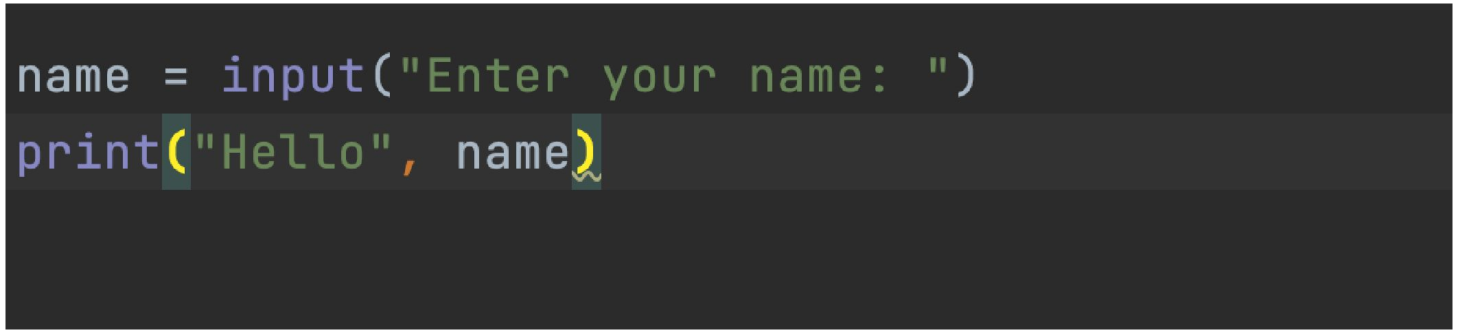
```
>>> print("one","two","three", end="*")
one two three*>>>
```

```
>>> print("one","two","three", sep="-", end="*")
one-two-three*>>>
```

Reading Input from the Keyboard

- **Most programs need to read input from the user**

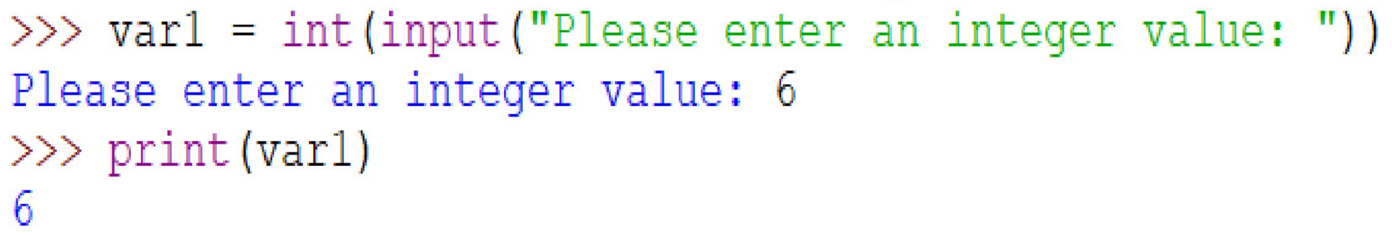
- **Built-in input function reads input from the keyboard**
  - Returns the data as a string
  - Format: ***variable*** **= input(*prompt*)**
    - ✔ Prompt is typically a string instructing the user to enter a value
  - Does not automatically display a space after the prompt, so make sure to add one:



```
name = input("Enter your name: ")
print("Hello", name)
```

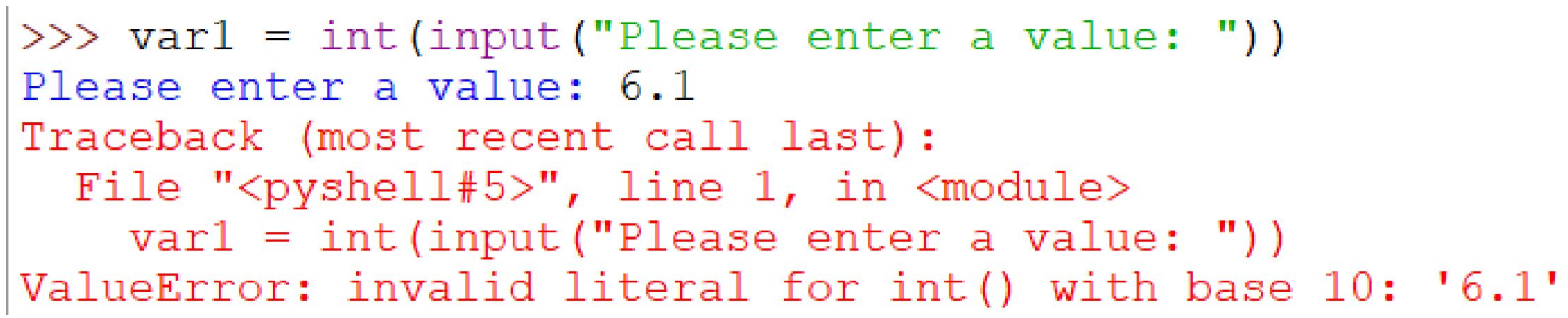
Reading Numbers with the *input* Function

- ***input*** **function always returns a string**
- **Built-in functions convert between data types**
  - int(*item*) converts *item* to an int
  - float(*item*) converts *item* to a float



```
>>> var1 = int(input("Please enter an integer value: "))
Please enter an integer value: 6
>>> print(var1)
6
```

If you enter a value that is not integer, an exception will be thrown:

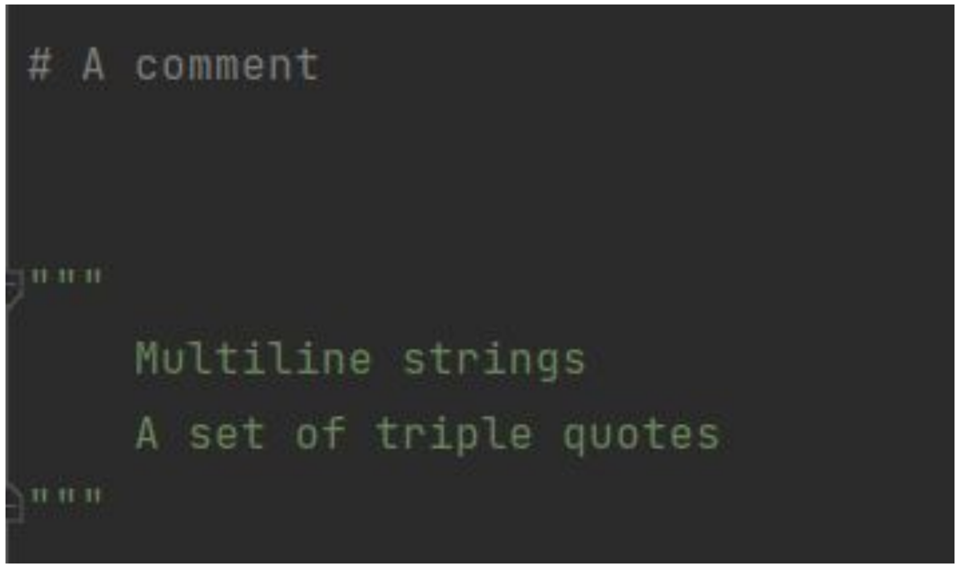


```
>>> var1 = int(input("Please enter a value: "))
Please enter a value: 6.1
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    var1 = int(input("Please enter a value: "))
ValueError: invalid literal for int() with base 10: '6.1'
```

Comments

Comments: notes of explanation within a program

- Ignored by Python interpreter
- Can not be run and is human-readable.
- Make your code easier to understand.
- Begin with a # character or triple quotes

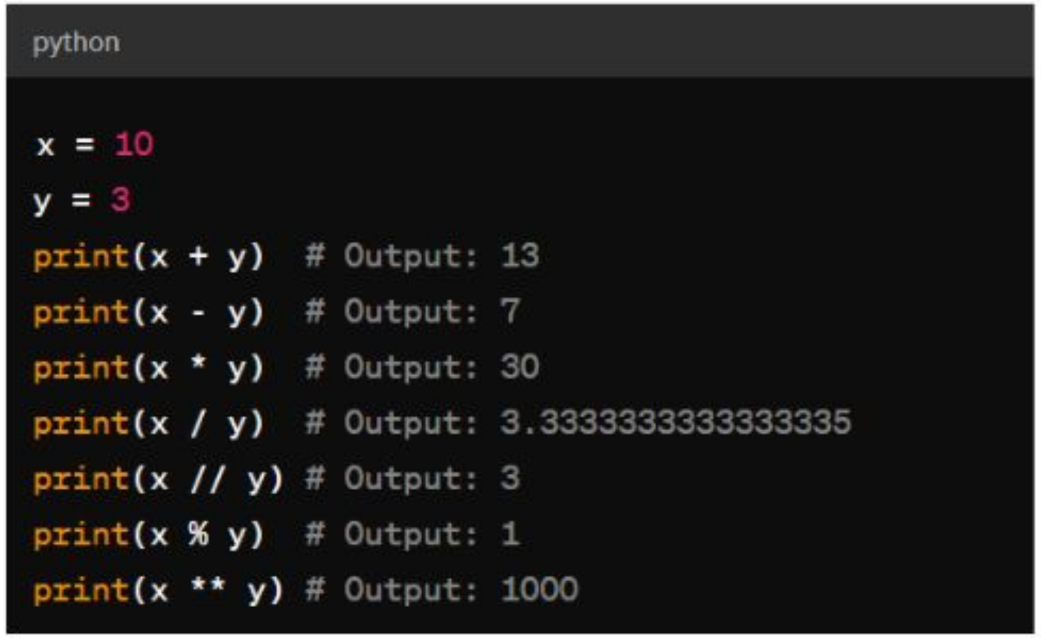


```
# A comment


"""
    Multiline strings
    A set of triple quotes
"""
```

Python Math Operators

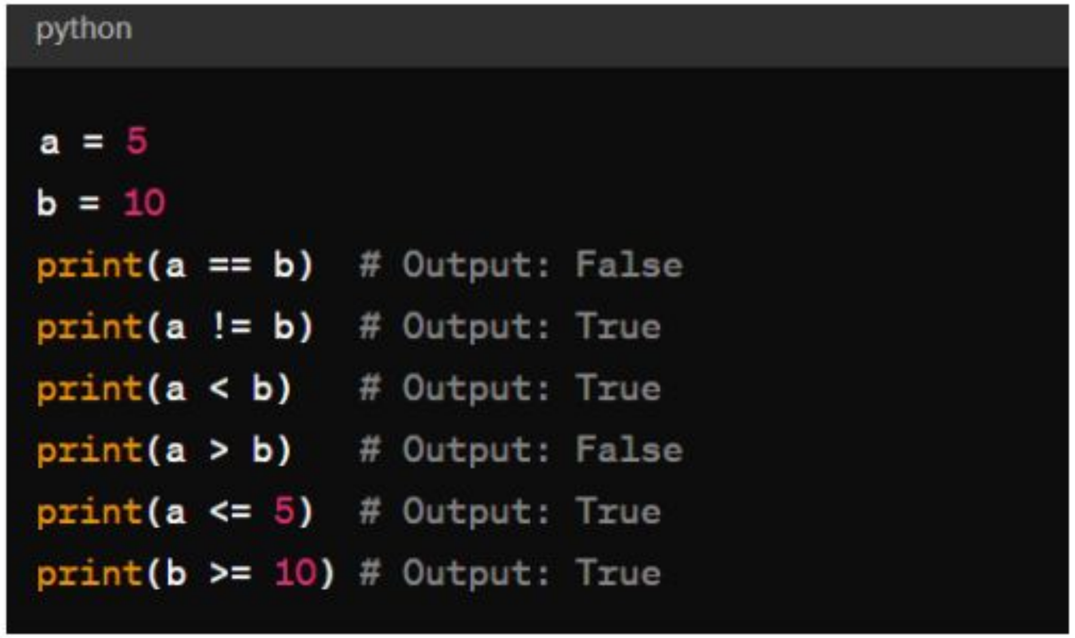
Symbol	Operation	Description
+	Addition	Adds two numbers
–	Subtraction	Subtracts one number from another
*	Multiplication	Multiplies one number by another
/	Division	Divides one number by another and gives the result as a floating-point number
//	Integer division	Divides one number by another and gives the result as a whole number
%	Remainder	Divides one number by another and gives the remainder
**	Exponent	Raises a number to a power

Mathematical Operators



```python
x = 10
y = 3
print(x + y)  # Output: 13
print(x - y)  # Output: 7
print(x * y)  # Output: 30
print(x / y)  # Output: 3.3333333333333335
print(x // y) # Output: 3
print(x % y)  # Output: 1
print(x ** y) # Output: 1000
```

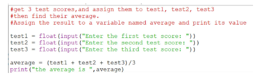
Comparison Operators



```python
a = 5
b = 10
print(a == b)  # Output: False
print(a != b)  # Output: True
print(a < b)   # Output: True
print(a > b)   # Output: False
print(a <= 5)  # Output: True
print(b >= 10) # Output: True
```

Exercise 1:

1. Ask the user to input 3 test scores & assign them to test1, test2, test3. These 3 variables should accept float values.
2. Find their average.
3. Assign the result to a variable named average and print its value



```
#get 3 test scores,and assign them to test1, test2, test3
#then find their average.
#Assign the result to a variable named average and print its value

test1 = float(input("Enter the first test score: "))
test2 = float(input("Enter the second test score: "))
test3 = float(input("Enter the third test score: "))

average = (test1 + test2 + test3)/3
print("the average is ",average)
```

Breaking Long Statements into Multiple Lines

- Long statements cannot be viewed on screen without scrolling and cannot be printed without cutting off
- Multiline continuation character (\): Allows to break a statement into multiple lines

```
result = var1 * 2 + var2 * 3 + \
         var3 * 4 + var4 * 5
```

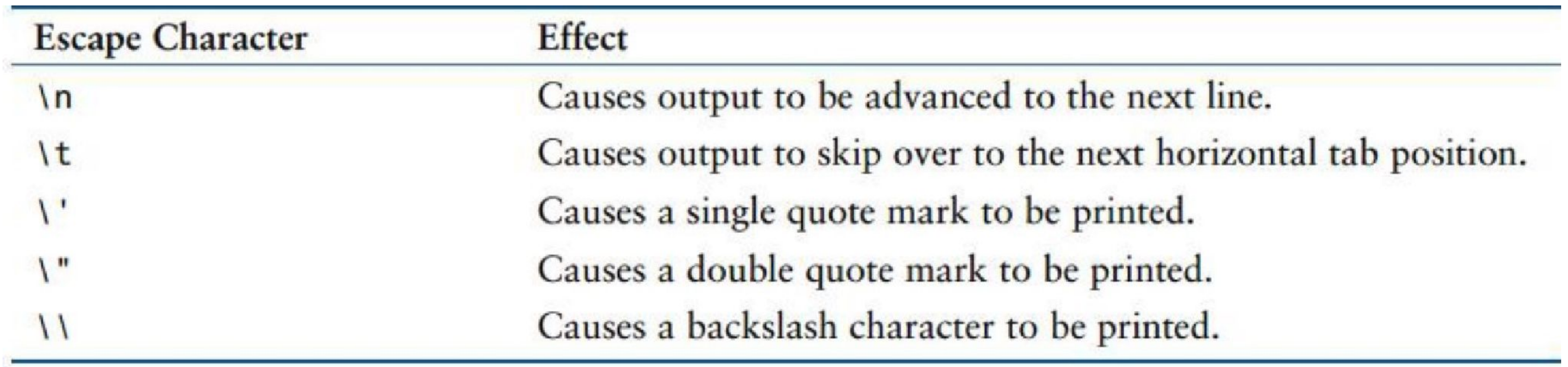
- Any part of a statement that is enclosed in parentheses can be broken without the line continuation character.

```
print("Monday's sales are", monday,
      "and Tuesday's sales are", tuesday,
      "and Wednesday's sales are", Wednesday)
```

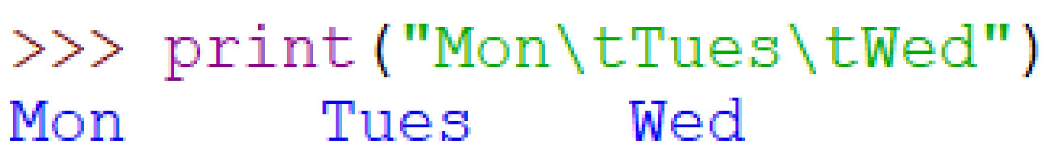
```
(value1 + value2 +
 value3 + value4 +
 value5 + value6)
```

More About Data Output

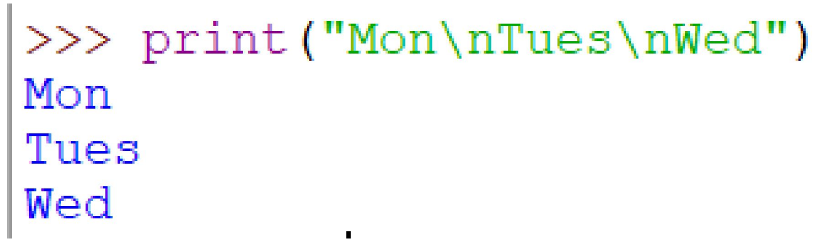
- Preceded by backslash (\)
    - Examples: newline (\n), horizontal tab (\t)



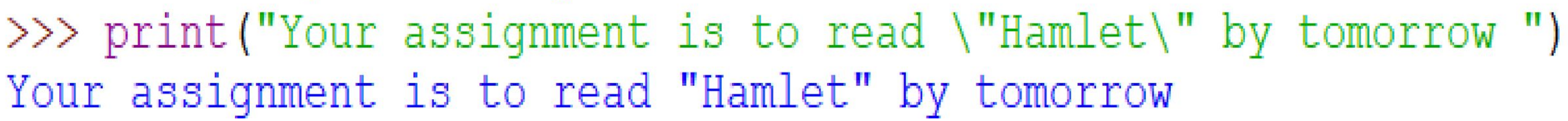
Escape Character	Effect
\n	Causes output to be advanced to the next line.
\t	Causes output to skip over to the next horizontal tab position.
\'	Causes a single quote mark to be printed.
\"	Causes a double quote mark to be printed.
\\	Causes a backslash character to be printed.



```
>>> print("Mon\tTues\tWed")
Mon     Tues    Wed
```



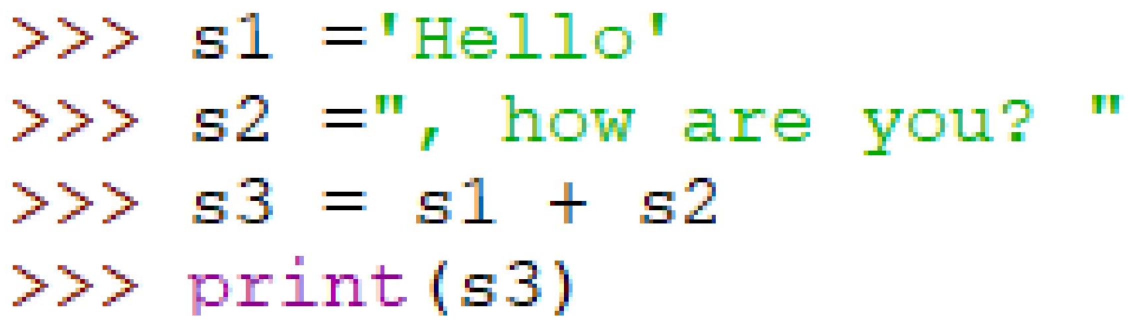
```
>>> print("Mon\nTues\nWed")
Mon
Tues
Wed
```



```
>>> print("Your assignment is to read \"Hamlet\" by tomorrow ")
Your assignment is to read "Hamlet" by tomorrow
```

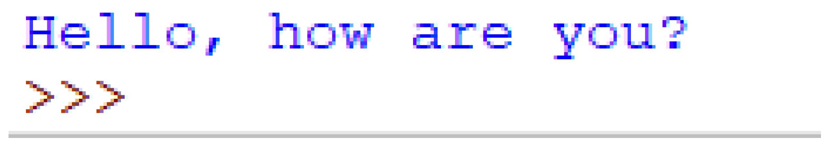
Concatenation using the + operator:

- **When + operator used on two strings in performs string concatenation**



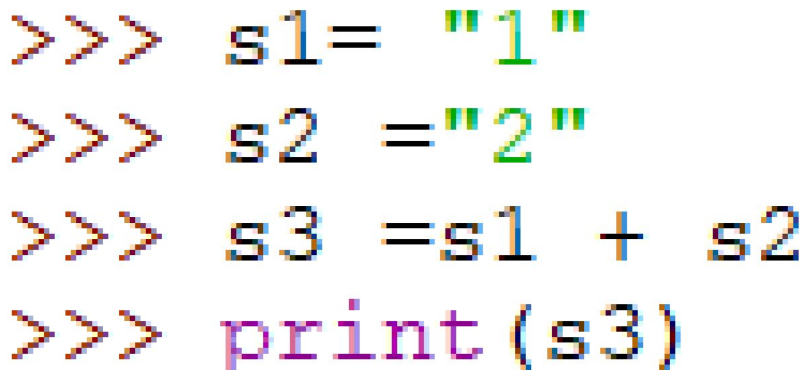
```
>>> s1 ='Hello'
>>> s2 =", how are you? "
>>> s3 = s1 + s2
>>> print(s3)
```

output



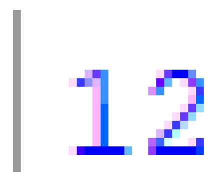
```
Hello, how are you?
>>>
```

s1, s2, s3 here are strings, not numbers:



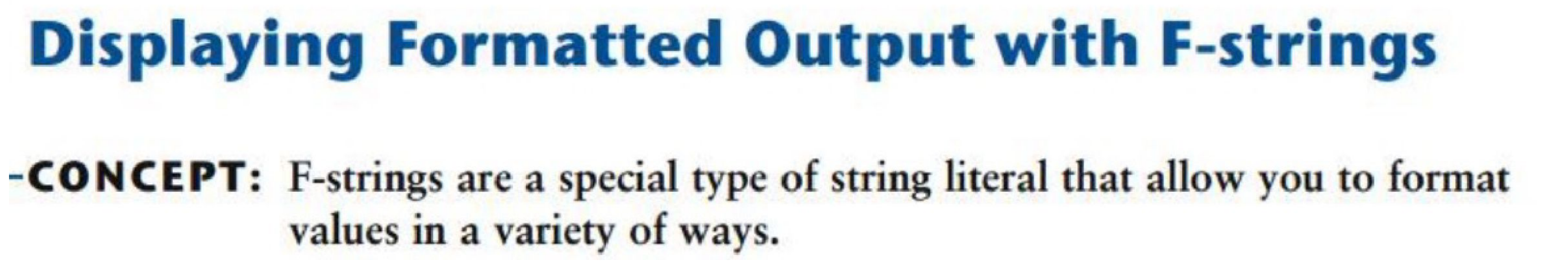
```
>>> s1= "1"
>>> s2 ="2"
>>> s3 =s1 + s2
>>> print(s3)
```

output



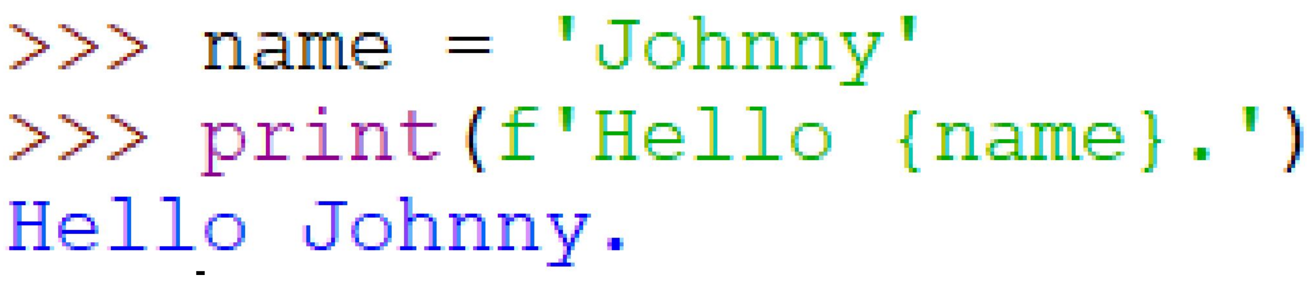
```
12
```

Formatting Numbers



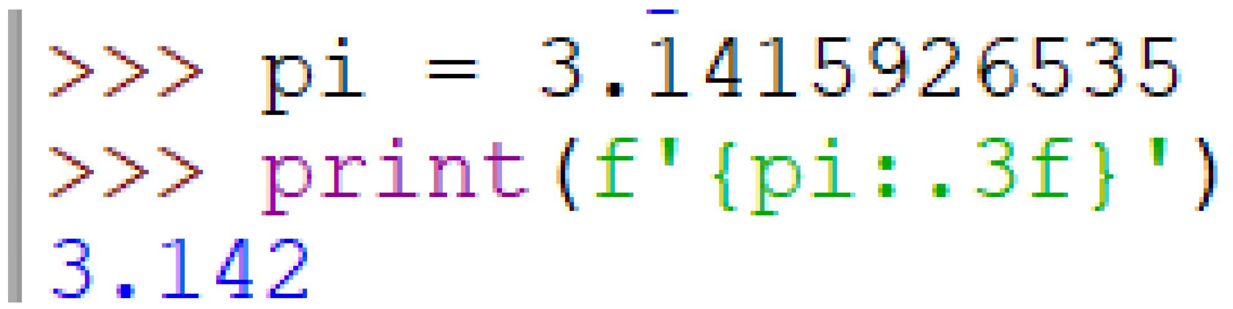
Displaying Formatted Output with F-strings

CONCEPT: F-strings are a special type of string literal that allow you to format values in a variety of ways.



```
>>> name = 'Johnny'
>>> print(f'Hello {name}.')
Hello Johnny.
```

Round a number to 3 decimal places:

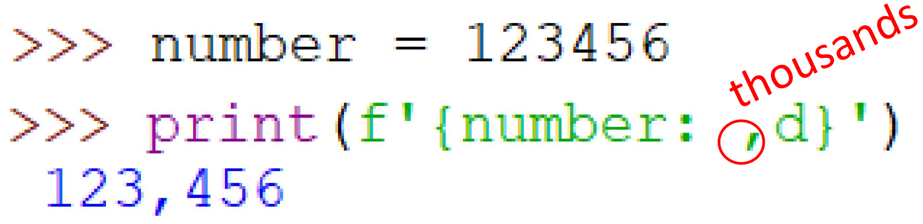


```
>>> pi = 3.1415926535
>>> print(f'{pi:.3f}')
3.142
```

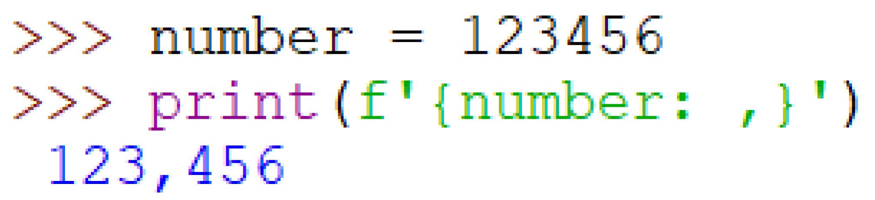
1. Make number 123456 to be printed with a comma separator

Use d as the type designator

thousands separator

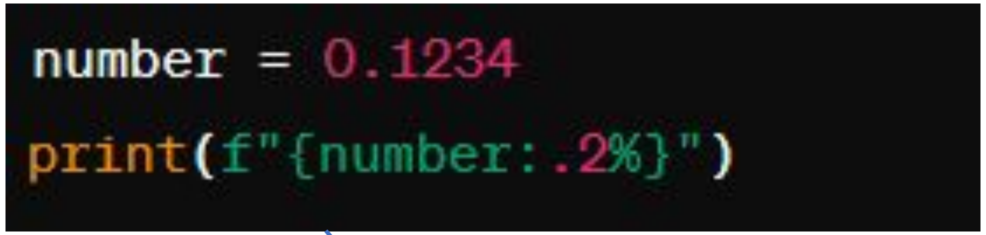


```
>>> number = 123456
>>> print(f'{number: ,d}')
 123,456
```



```
>>> number = 123456
>>> print(f'{number: ,}')
 123,456
```

3. Make 0.1234 to be multiplied by 100 using the % sign:

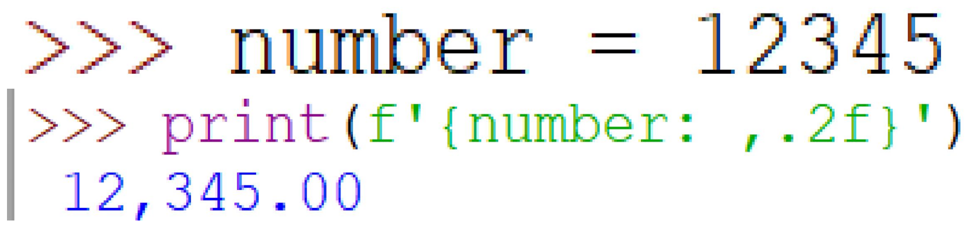


```
number = 0.1234
print(f"{number:.2%}")
```

output

12.34%

2. Floating point value rounded to 2 decimal places with a comma separator.



```
>>> number = 12345
>>> print(f'{number: ,.2f}')
 12,345.00
```

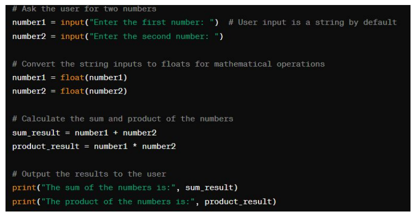
Questions

Exercises

Problem 1: Simple Math Input/Output

Objective: Create a Python script that asks the user for two numbers and then calculates the sum and product of these numbers. The script should:

- Use proper indentation to maintain readability.
- Use variables to store the user inputs and results.
- Perform data input and output operations.
- Include comments that explain what each part of the code is doing.
- Handle data type conversion because **`input`** returns string values.

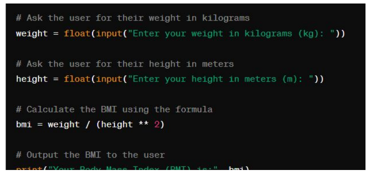


```python
# Ask the user for two numbers
number1 = input("Enter the first number: ")  # User input is a string by default
number2 = input("Enter the second number: ")

# Convert the string inputs to floats for mathematical operations
number1 = float(number1)
number2 = float(number2)

# Calculate the sum and product of the numbers
sum_result = number1 + number2
product_result = number1 * number2

# Output the results to the user
print("The sum of the numbers is:", sum_result)
print("The product of the numbers is:", product_result)
```

Problem 2: Build a BMI Calculator

Objective: Create a Python script that asks the user for the weight and height and then calculates the Body Mass Index (BMI) score. You can try to use F-strings for output.

Notes : BMI score = An individual's weight in kilograms by the square of the height in meters



```python
# Ask the user for their weight in kilograms
weight = float(input("Enter your weight in kilograms (kg): "))

# Ask the user for their height in meters
height = float(input("Enter your height in meters (m): "))

# Calculate the BMI using the formula
bmi = weight / (height ** 2)

# Output the BMI to the user
print("Your Body Mass Index (BMI) is:", bmi)
```

☺ Thank you