Operators, Boolean Expressions, and Comments

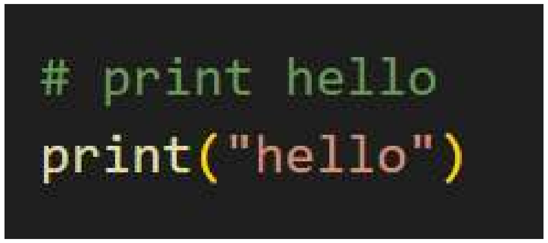
can code communities

Comments

- A comment is a line of code that **doesn’t get run**
- You use comments to document your code - describe what your code is doing in words. Comments are a type of **documentation**.
- Comments are important if you’re doing individual work to remind you what your own code does
- They’re especially important if you’re doing team work, to explain to your team what the code does.
    - In most development jobs, you’ll be working on a team, so you need to document your code with good comments.

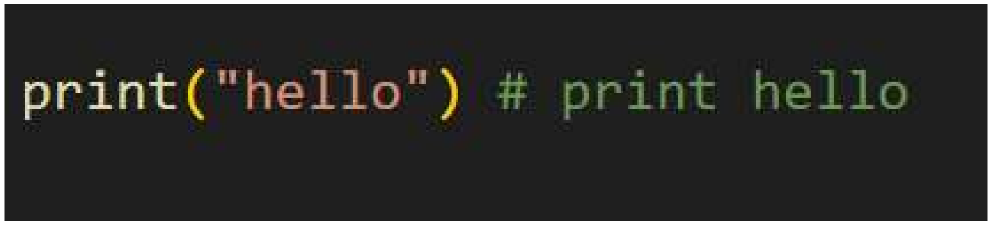
Single-Line Comments

- In Python, you start a single line comment with #
- Everything after the # symbol is ignored when you run your code, so you can put whatever you want after it
- You can start a line with #, or put the comment on the same line as a line of code



```python
# print hello
print("hello")
```

Standard single-line comment

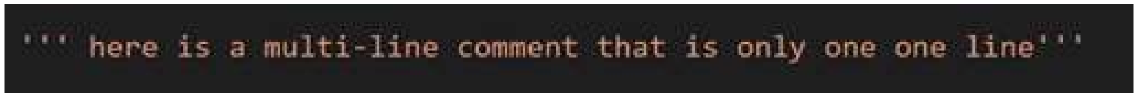


```python
print("hello") # print hello
```

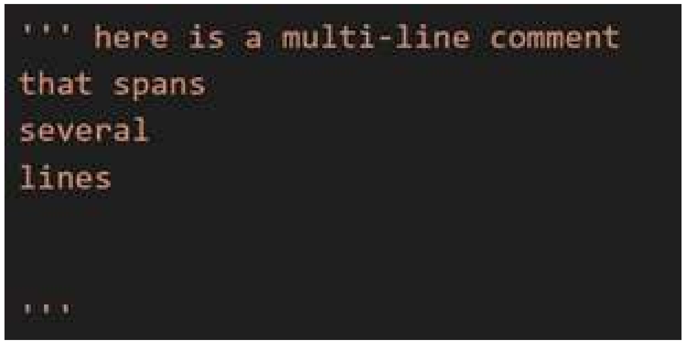
In-line comment

Multi-Line Comments

- You can also write a comment that spans multiple lines.
- In Python, you open the comment with three quotes, and close it with three quotes. Everything between the quotes is a comment.
  - Just like for strings, they can be single or double quotes.
- Multi-line comments can be as many lines as you want, or even just one line.



```
''' here is a multi-line comment that is only one one line'''
```



```
''' here is a multi-line comment
that spans
several
lines


'''
```

PEP 8 Comments

https://www.python.org/dev/peps/pep-0008/#comments

Example

Add comments to the perimeter of a rectangle example from last class.

Exercise

Add comments to the Fahrenheit to Celsius conversion exercise from last class.

Console

The console is like an active python coding environment.

In here, we can type out code and run it line by line.

There are some things that can’t be done in the console.

We also can not save our code when working in the console.

We will not really be using the console in this class.

Arithmetic Operators Review

+ Addition

- Subtraction

* Multiplication

/ Division

** Exponents

// Integer Division

% Modulo/mod/remainder

Keep in mind

Order of operations: PEMDAS

- Parentheses
- Exponents
- Multiplication/Division and Mod, for Python
- Addition/Subtraction

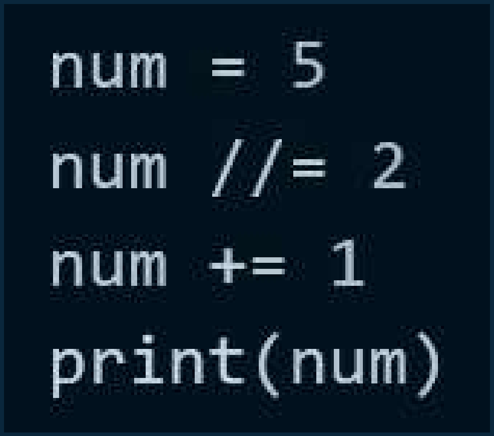
Use parentheses for clarification if need

Shortcut Operators

- For each arithmetic operator, you can use shortcut operators if you’re adding to an existing value.
- For example, if you have a variable named num and you want to add num to it, that would be `num = num + 5`
- Using shortcut operators, we can shorten this to `num += 5`
- These lines do the exact same thing: They add 5 to the existing value of num, and then store it back in num.

Exercise

What is the output of this code?



```
num = 5
num //= 2
num += 1
print(num)
```

Exercise

Change your code that converts Fahrenheit to Celsius to **only** use shortcut operators.

Boolean Review

A boolean is a data type in Python like ints, floats, or strings. In Python, booleans are called bool.

A boolean must be either `True` or `False`.

Booleans have a lot of uses that we will see later on in this course. You usually use booleans if there are two possible conditions, like a yes/no question.

Boolean Operators

`<`	Less Than
`<=`	Less Than or Equal To
`>`	Greater Than
`>=`	Greater Than or Equal To
`==`	Equals
`!=`	Not Equals
`and`	Both True
`or`	One True
`not`	Opposite
`is`	Are they the same Object?
`in`	Inside of

Using any of these operators on two values results in a boolean - the result is either True or False.

Example

Go back to the perimeter of a rectangle example and print whether the length is larger than the width.

Exercise

Go back to the Fahrenheit to Celsius conversion exercise, and print whether the Fahrenheit is greater than the Celsius.

Converting Data to Booleans

- You can’t directly cast data to a boolean like with other data types
    - The bool() function evaluates whether the object exists or not. So it will always return True, unless you pass in an empty object, False, 0, or None
- If you want to cast a string to a boolean, you have two options:
    - You can manually check if the string == "True" or == "False", in which case set it equal to True or False. (More on this when we learn about conditionals)
    - You can cast it to a boolean using eval(), the evaluate function

`eval("True")` returns True

`eval("False")` returns False

More on eval(): https://realpython.com/python-eval-function/