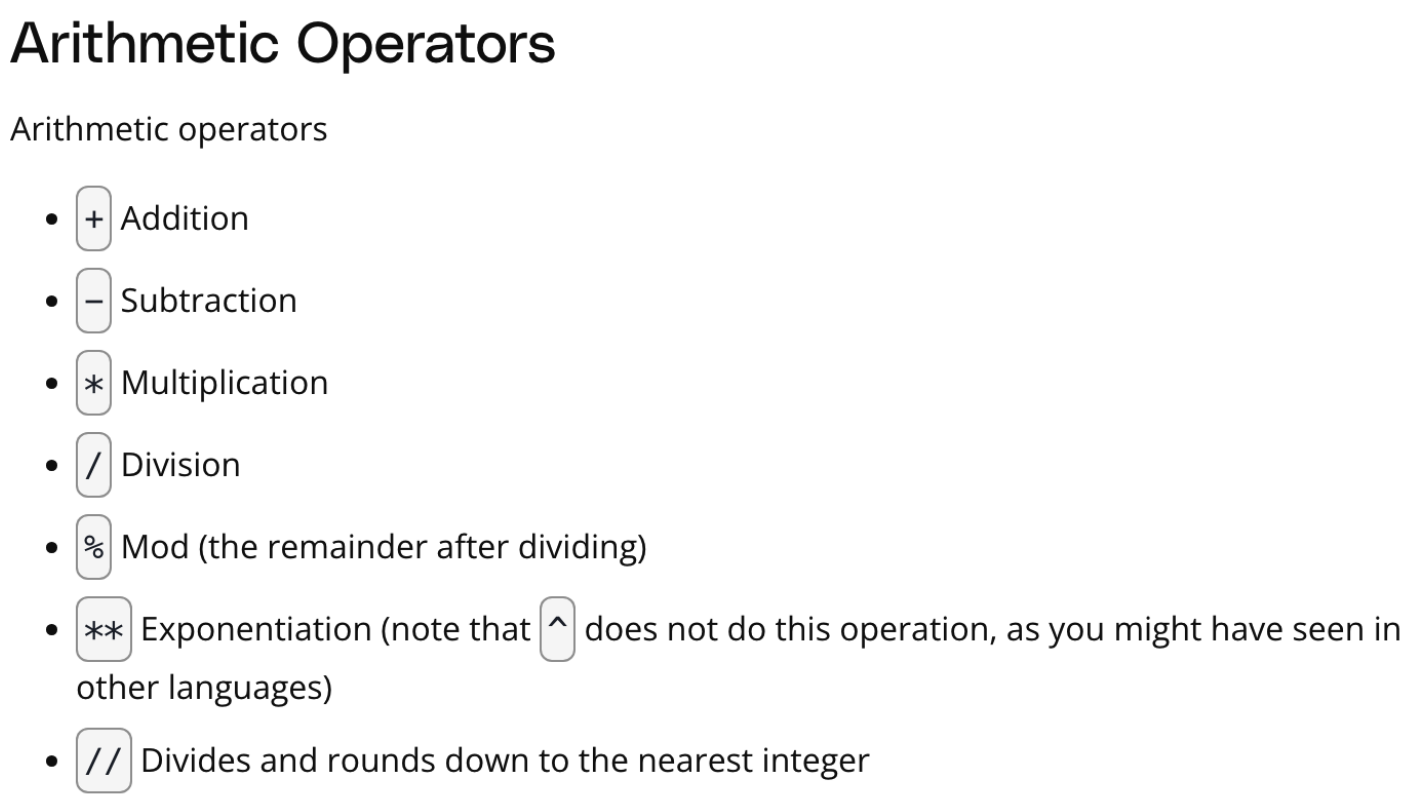
Programming AI in Python

Python Data Types and Operators

- Python is case sensitive
- Spacing matters



Arithmetic Operators

Arithmetic operators

- `+` Addition
- `-` Subtraction
- `*` Multiplication
- `/` Division
- `%` Mod (the remainder after dividing)
- `**` Exponentiation (note that `^` does not do this operation, as you might have seen in other languages)
- `//` Divides and rounds down to the nearest integer

The usual order of mathematical operations holds in Python, often referred to as **PEMDAS**: ***Parentheses***, ***Exponents***, ***Multiplication/Division***, ***Addition/Subtraction***.

If needed, you can review the order of mathematical operations in this Wikipedia article **Order of Operations**.

Bitwise operators are special operators in Python that you can learn more about **here** if you'd like.

Variable & Assignment Operators

Pythonic naming convention - use all lowercase letters and underscores to separate words
Ex. "views_per_day"

Variable assignment

```
x = 3
y = 4
z = 5
```

and

```
x, y, z = 3, 4, 5
```

Here are the assignment operators from the video.

Symbol	Example	Equivalent
`=`	`x = 2`	`x = 2`
`+=`	`x += 2`	`x = x + 2`
`-=`	`x -= 2`	`x = x - 2`

You can also use `*=` in a similar way, but this is less common than the operations shown below. You can find some practice with much of what we have already covered **here**.

Python Best Practices

For all the best practices, see the PEP8 Guidelines.

You should limit each line of code to **80** characters, though **99** is okay for certain use cases. **You can thank IBM for this ruling**.
Why are these conventions important? Although how you format the code doesn't affect how it runs, following standard style guidelines makes code easier to read and consistent among different developers on a team.

Python Errors

In general, there are two types of errors to look out for

- Exceptions
- Syntax

An Exception is a problem that occurs when the code is running, but a 'Syntax Error' is a problem detected when Python checks the code before it runs it. For more information, see the Python tutorial page on Errors and Exceptions.

Booleans, Comparison Operators, and Logical Operators

The bool data type holds one of the values `True` or `False`, which are often encoded as `1` or `0`, respectively.

There are 6 comparison operators that are common to see in order to obtain a `bool` value:

Comparison Operators

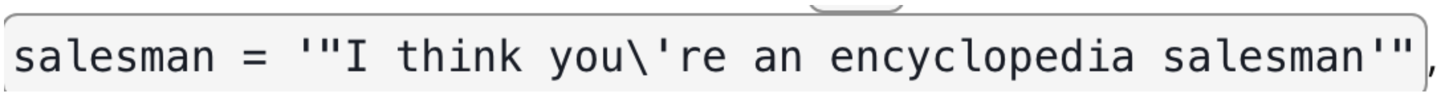
Symbol Use Case	Bool	Operation
5 < 3	False	Less Than
5 > 3	True	Greater Than
3 <= 3	True	Less Than or Equal To
3 >= 5	False	Greater Than or Equal To
3 == 5	False	Equal To
3 != 5	True	Not Equal To

And there are three logical operators you need to be familiar with:

Logical Use	Bool	Operation
5 < 3 `and` 5 == 5	False	`and` - Evaluates if all provided statements are True
5 < 3 `or` 5 == 5	True	`or` - Evaluates if at least one of many statements is True
`not` 5 < 3	True	`not` - Flips the Bool Value

<u>Strings (Immutable datatype)</u>

- You can use single or double quotes for strings ‘ ‘ or “ “
- Quotation marks should be preceded by a backslash ex.
- Strings cannot be modified



```
salesman = '"I think you\'re an encyclopedia salesman'",
```

The `len()` function

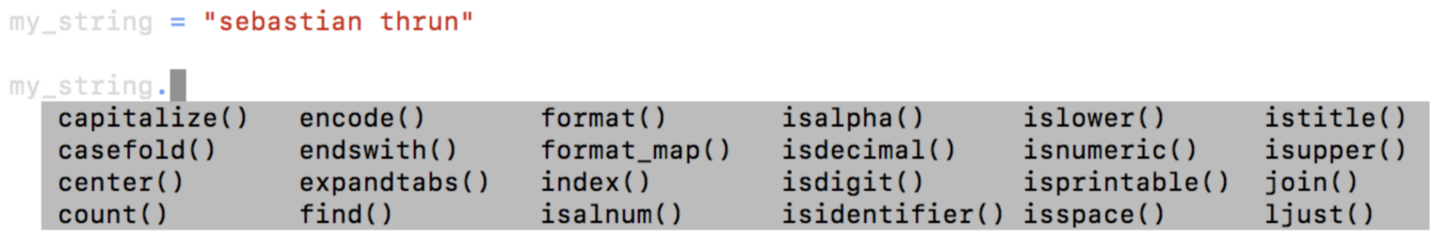
`len()` is a built-in Python function that returns the length of an object, like a string. The length of a string is the number of characters in the string. This will always be an integer.

There is an example above, but here's another one:

```
print(len("ababa") / len("ab"))
2.5
```

You know what the data types are for len("ababa") and len("ab"). Notice the data type of their resulting quotient here.

Other string methods



```
my_string = "sebastian thrun"

my_string.
    capitalize()   encode()       format()       isalpha()       islower()      istitle()
    casefold()     endswith()     format_map()   isdecimal()     isnumeric()    isupper()
    center()       expandtabs()   index()        isdigit()       isprintable()  join()
    count()        find()         isalnum()      isidentifier()  isspace()      ljust()
```

Another important string method: `split()`

A helpful string method when working with strings is the .split method. This function or method returns a data container called a **list** that contains the words from the input string. We will be introducing you to the concept of lists in the next video.
The split method has two additional arguments (*sep* and *maxsplit*). The *sep* argument stands for "separator". It can be used to identify how the string should be split up (e.g., whitespace characters like space, tab, return, newline; specific punctuation (e.g., comma, dashes)). If the *sep* argument is not provided, the default separator is whitespace.
True to its name, the *maxsplit* argument provides the maximum number of splits. The argument gives maxsplit + 1 number of elements in the new list, with the remaining string being returned as the last element in the list. You can read more about these methods in the Python documentation too.

Here are some examples for the `.split()` method.

1. A basic split method:

~~~
new_str = "The cow jumped over the moon."
new_str.split()```
Output is:
```Python
['The', 'cow', 'jumped', 'over', 'the', 'moon.']```

2. Here  the separator is space, and the maxsplit argument is set to 3.
    ```Python
    new_str.split(' ', 3) ```
    Output is:
    ```Python
['The', 'cow', 'jumped', 'over the moon.']```

3. Using '.' or period as a separator.
```Python
new_str.split('.')```
Output is:
```Python
['The cow jumped over the moon', '']```

4. Using no separators but having a maxsplit argument of 3.
    ```Python
    new_str.split(None, 3)```
    Output is:
    ```Python
['The', 'cow', 'jumped', 'over the moon.']```
~~~

Additional Practice Resources

When many students are getting started, they often always want more practice. There are a number of great websites you can use for coding exercises and solutions. Two that you should definitely take advantage of are HackerRank and Codewars.
Note: You may find some of the exercises require knowledge on concepts you haven't learned yet. Feel free to google them, or wait until you've gone through all the lessons in this course.
I encourage you to create a profile on both and commit to improving your Python programming skills! As you get better, you can advance to harder problems and sites with even greater challenges. If you spend a lot of time on this, you'll really become a Python programming master.
Happy coding and see you in the next lesson!

Data Structures

Lists (Mutable datatype) - Lists are used to store multiple items in a single variable.
List items are ordered (means that the items have a defined order and that order will not change), changeable (meaning we can change, add, remove items after it has been created), and allow duplicate values(meaning lists can have items with the same value).

Slicing with lists

You saw that we can pull more than one value from a list at a time by using **slicing**. When using slicing, it is important to remember that the `lower` index is `inclusive` and the `upper` index is `exclusive`.

```
>>> list_of_random_things[:2]
[1, 3.4]
```

Are You `in` or `not in`?

You saw that we can also use `in` and `not in` to return a **bool** of whether an element exists within our list, or if one string is a substring of another.

```
>>> 'this' in 'this is a string'
True
>>> 'in' in 'this is a string'
True
>>> 'isa' in 'this is a string'
False
>>> 5 not in [1, 2, 3, 4, 6]
True
>>> 5 in [1, 2, 3, 4, 6]
False
```

Useful Functions for Lists I

1. `len()` returns how many elements are in a list.
2. `max()` returns the greatest element of the list. How the greatest element is determined depends on what type of objects are in the list. The maximum element in a list of numbers is the largest number. The maximum element in a list of strings is the element that would occur last if the list were sorted alphabetically. This works because the the `max()` function is defined in terms of the greater than comparison operator. The `max()` function is undefined for lists that contain elements from different, incomparable types.
3. `min()` returns the smallest element in a list. min is the opposite of max, which returns the largest element in a list.
4. `sorted()` returns a copy of a list in order from smallest to largest, leaving the list unchanged. Note again that for string objects, sorted smallest to largest means sorting in alphabetical order.

Tuples(Immutable)

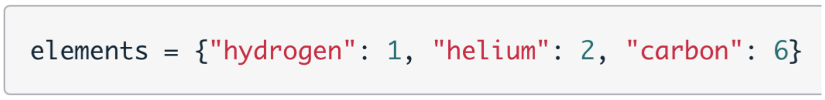
- Cannot be sorted
- tuples written with round brackets
- tuples are ordered (means that the items have a defined order and that order cannot be changed), unchangeable(cannot change, add or remove items after creation), allow duplicate(since tuples are indexed, they can have items with the same value)

Sets(Mutable) Used to collect unique elements

-Sets are mutable, unordered collections of unique elements
-add method adds item to the set
-pop method removes a random item from the set

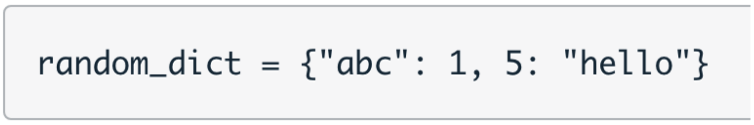
Dictionaries(Mutable)

-dictionary is mutable data type that stores mappings of unique keys to values
-create a dictionary by using squal brackets “ { }”
-dictionaries are unordered
-duplicates not allowed



```
elements = {"hydrogen": 1, "helium": 2, "carbon": 6}
```

-Dictionaries are mutable, but their keys need to be any immutable type, like strings, integers, or tuples. It's not even necessary for every key in a dictionary to have the same type! For example, the following dictionary is perfectly valid:



```
random_dict = {"abc": 1, 5: "hello"}
```