**Slide 2**:  
  
In this module, you'll learn about some basic concepts of the Python programming language, using NASA as context to apply them. Actual engineers working at NASA use these ideas everyday when working on projects.

**Slide 5**:

Over the course of this module, we'll be using comments in our code. Learning how to use comments efficiently will greatly improve the readability of your code and help you learn programming topics faster.

**Slide 6**:

## What are comments?

In programming, comments are words that the computer will not read when running the code. Their sole purpose is to help developers who are reading and writing code. Comments are great for explaining complex code to others, helping you remember what a certain part of your code does, and reminding you of what needs to be added to your code.

**Slide 7**:

## How to use comments

In Python, comments are as simple as including a # before the comment that you would like to add.

As noted previously, comments are not run by the computer. If you look at the code below, 2 + 2 is commented out so the computer will only recognize and compute 2 + 3. Try to copy this code into a cell and click the play button.

Another great use of comments is to add TODOs to let you know what still needs to be done. In this example, the comment is reminding a developer to finish a piece of code at a later time.

**Slide 8**:

Block comments are many lines of comments strung together. They can be useful to help explain code to a user who might be unfamiliar with the concepts you're using or to describe what an entire program does.

**Slide 9:**

Inline comments are written on the same line as written code. Use them sparingly because they can add clutter to your code. They're good at explaining a tricky line of code.

**Slide 10:**

Math is used frequently in computer science. The concepts you'll learn in this module will be great tools to add to your coding arsenal. Python has features that allow you to do many different mathematical computations.

**Slide 11:**

As with a calculator, you can use the addition + , subtraction - , multiplication \* , and division / operators in Python to add, subtract, multiply, and divide numbers.

For example, write the following comment and code in a cell, then select the run button to see the output.

**Slide 12:**

There are also more advanced calculations that Python can compute.

**Slide 16:**

# Create variables in Python Notebooks for future data analysis

Completed100 XP

* 6 minutes

In this unit, you'll learn about variables in Python. Variables are containers that you can store data in and use at a different time.

There are four main types of data variables that you'll encounter throughout this content:

* Integers (int): These are whole numbers like 1, 4, 10, -5.
* Floats: These are decimal numbers like 0.3, 1.6, 17.4, -3.5.
* String: These are chains of characters that are surrounded by single or double quotes like "hi", "NASA", "Space Rocks", "54321".
* Boolean: Represents either True or False.

**Slide 17:**

A unique aspect of Python is that you don't need to tell the program what type of variable you're making. For example, in some languages, if you want to make an integer, you must first let the computer know that you're about to create an integer. In Java, for example, you might write: int intVar = 0;.

Later on in our program, we'll be using variables to store the number of a certain type of rock we find. One of the main types of moon rock is called basalt. We can make a variable named basaltRockCount and give it a value of 0 by using the following code:

**Slide 21:**

Strings are one of the most unusual of all variables, because they can take on almost any key from your keyboard. Python also has much pre-made code (called functions) that you can call to manipulate your strings in various ways.

**Slide 25:**

Lists are ways to store a lot of data in Python and are similar to variables. They are a collection of values stored in the form of a list. Lists are useful in Python because often you'll need more than one piece of data. Lists play a huge role in data science, because the goal of data science is to take large amounts of related data and analyze it. That's why the popular data science Python packages such as Pandas and NumPy have additional data structures such as dataframes and arrays, which are essentially lists with additional features.

**Slide 26:**

Defining lists in Python is similar to creating variables. Start by making a name for your list and then set it equal to a value. However, lists can hold multiple values.

**Slide 27:**

As you can see, you must include square brackets and then include the values you want separated by commas.

Lists can have values that come in all forms (int, float, string) and can even mix and match different types.

**Slide 28:**

There are many pre-made functions included in Python, which we can use to modify a list.

For example, we can add items to a list by calling the .append() function followed by the data that we want to add in parentheses. Doing this will add an item to the end of a list. Let's add a rock type to our original list.

**Slide 29:**

We can also delete items from the end of a list by calling the .pop() function. We'll now delete soil from the rock types list.

**Slide 30:**

Finally, we can look at and change the value from anywhere within the list. To see what value is at a certain point in the list, use square brackets after the list name to look at that specific value. Everything in Python is zero-indexed, meaning that counting begins at 0, not 1. So if we look at the value in the first position in a list, we would use: listName[0].

**Slide 31:**

We can also change a specific value in the list at a specific point by coding:

listName[position in list to change value] = newValue.

In our rock type example, if we wanted to change the value in the third position of the list to be "soil", we could do so as follows:

**Slide 32:**

Now that we've created some variables and given them values, we can learn how to view the values beyond running a notebook cell. We'll use a function called print(). Notebooks are useful because each cell will output the value of the final command without the need to do use the print() function. However, it might be useful to print values or messages between larger code blocks, for debugging, logging, or informational purposes. Without the print() function, the cell will print only the last value:

**Slide 34:**

The print() function, like the functions we called earlier when using strings, is code we can use that has been written by others. The print() function will make the computer write the values within the values you pass in as a parameter.

**Slide 36:**

The print function is very helpful, because the people using our programs won't always be able see our code or know how to read it. The print function allows us to show them only the values we want them to see.

**Slide 38:**

In this module, you learned about some basic computer science principles and applied them to NASA themed examples in the Python programming language. You used variables to count space rocks, output functions to print data about space launches, and string functions to help print data about an astronaut.