Video Script: Intro to Python & Colab (Voiceover)

[Intro Music Fades In and Out]

(VOICEOVER)
Hello and welcome to the first video in our Python and Machine Learning Foundations series. This module is designed to provide you with the Python skills necessary to start programming in Python, particularly as you engage with data analysis and machine learning.
So, if you're new to programming, our goal is to demonstrate that Python is not only powerful but also highly accessible. Let's begin!

[ON SCREEN: Slide 2 - "Introduction to Python"]

(VOICEOVER)
First, we'll introduce Python. At its core, it's defined as a high-level, versatile programming language. What do I mean by 'high-level'?.

[ON SCREEN: Slide 3 - "What is Python?"]

(VOICEOVER)
"High-level" means it's designed with an intuitive human readable syntax that abstracts away complex computer operations, making it easier for developers to read and write. As for its history, Python was created in the late 1980s by a Dutch programmer, Guido van Rossum.
Fun fact: the name is not a reference to the reptile. Apparently, Guido was a fan of the British comedy series *Monty Python's Flying Circus*, which influenced his choice for a name which he also felt was "unique, memorable and mysterious"

[ON SCREEN: Slide 4 - "Why Python?"]

(VOICEOVER)
So, why has this language become so integral to modern computing? Its impact is substantial. Python is a dominant force in both industry and academia.
Most importantly for this course, it is the de facto standard for Artificial Intelligence and Machine Learning. Its clean syntax allows researchers and developers to focus on complex algorithms rather than cumbersome code. Furthermore, its extensive ecosystem of open-source libraries—such as TensorFlow, PyTorch, and Scikit-learn—provides the essential tools for building and deploying sophisticated models.

[ON SCREEN: Slide 5 - "IDE – Google Colab"]

(VOICEOVER)
Now that we understand what Python is, we need to discuss where we will write and execute it. For that, we'll be using an IDE, and our specific tool of choice is Google Colab.

[ON SCREEN: Slide 6 - "What is an IDE?"]

(VOICEOVER)
IDE stands for Integrated Development Environment. You can think of it as a programmer's workshop or as I like to call it, a programmer's playground. It's an application that consolidates the essential tools for software development, such as a code editor, a compiler or interpreter, and debugging tools, into a single, streamlined interface.

[ACTION: Screen recording switches from the slide deck to a live view of a terminal session where Python interpreter will be demoed]

Here are some examples of how you can write and execute Python code. Since Python is an interpreted language i.e., it is executed line-by-line rather than an entire script file at once, we can quickly test code with the Python interpreter in the terminal. This wouldn't qualify as an IDE because it lacks debugging tools and other functions like code completions that make an IDE a developer's most important tool.

[ACTION: Screen recording switches from the slide deck to a live view of a VS Code instance and a new file having a python script which is then executed.]

A very popular IDE for developing Python applications is VS Code and as you can see on your screen, you can write full Python scripts and execute the whole file. But, for ease of getting started and an out-of-box experience, we are going to use Google Colab as stated earlier.

[ON SCREEN: Slide 7 - "Google Colab(oratory)"]

(VOICEOVER)
Google Colab offers several distinct advantages for our purposes. It requires no local installation, operating entirely within a web browser with a Google account. This ensures a consistent environment for all users. It also facilitates easy collaboration and provides access to powerful cloud-based hardware, like GPUs which will become relevant in later machine learning modules.

[ACTION: Screen recording switches from the slide deck to a live view of a web browser opening Google Colab.]

(VOICEOVER)
Alright, let's see it in action. To get started, navigate to colab.research.google.com. You'll see a pop-up window. We can close this for now and create a new notebook by selecting File and then New notebook.
This is the Colab notebook environment. It is composed of blocks called **cells**. There are two primary types. **Code cells**, like this one, are for writing and executing Python

code. And **Text/Markdown cells**, which can be added with the + Text button, are for documentation and notes.

To execute a code cell, you can either click the play icon or use the keyboard shortcut Shift + Enter to run the current cell and create a new one below it, which is the method I will be using. You can also do Ctrl + Enter to run the current cell and avoid creating a new cell below.

Let's begin writing code. As is customary whenever learning a new programming language, let's start with the Hello World program. Some would say it is cliche, but we call this a classic over here folks.

[ACTION: Type hello world program and execute.]

One of the first things one must know is how to show output which we are doing with the "print" statement here. All this does is to print to the console which traditionally refers to a terminal screen but in the modern context refers to whichever channel of display output is available. In colab, it is the output attached at the end of each cell.

The next basic thing one needs to know is how to write comments in the code. Comments are text in a block of code which are there only to provide context to the human developer reading some code. It is ignored by the compiler. In Python, the syntax is to use a "#" character. Anything you type after this will be a comment. For a multi-line comment, you can use triple quotes and include anything within and that will be ignored by the compiler.

Now, in the context of Colab and this code notebook style in particular, let's quickly discuss what Markdown cells are. Markdown is a simple syntax to represent formatted text. For example, you can specify a 'H1' heading which is text represented as big title text with a single '#'. Note this is markdown now and not python anymore. No code will be run in these cells. This is the neat part of Jupyter (the original python package name) notebooks. Consider these markdown cells as fancy comments where you can take documentation of your code to the next level. You basically have a word processor combined with a code editor here which is really powerful for scientific and quick prototyping of ideas. This is also why it is popular to use Jupyter Notebooks or Colab to test ML models or explore a dataset with data visualisations because this mode allows you to see the code and visualisations/documented reasoning for the code in one go.

For the best experience from this course and videos, we encourage you to code along with the video and then practice with the practice question(s) or challenge that will be described at the end of each video. Now as a bonus, you can find keyboard shortcuts to help you become a power user with Colab and master the IDE. For your task, I encourage you to spin up a Colab instance if you haven't already and mess around with it. Thank you and happy coding!

—

The first concept to understand is the **variable**. A variable is essentially a named reference to a location in memory where a value is stored.

Let's define one. I'll type message and assign it the text value "Hello, Python!".

(Live typing in a code cell)

```
message = "Hello, Python!"
```

(VOICEOVER)
Here, message is the variable name, and "Hello, Python!" is the value it stores. To display the value, we can use the print() function.
(Live typing in the same cell)

```
message = "Hello, Python!"
print(message)
```

(VOICEOVER)
Now I'll press Shift + Enter. As you can see, the output is displayed directly below the cell.
Next, let's briefly discuss **data types**. The text we just used is a **string**. We can also work with numerical data. Whole numbers are **integers**, and numbers with a fractional component are **floats**.

(Live typing in a new code cell)

```
# An integer
transaction_count = 10
print(transaction_count)

# A float
pi_value = 3.14159
```

```
print(pi_value)
```

(VOICEOVER)
And of course, we can perform arithmetic operations. The operators are standard: plus, minus, an asterisk for multiplication, and a forward slash for division.
(Live typing in a new code cell)

```
a = 10
b = 5
print(a + b)
print(a / b)
```

(VOICEOVER)
As you can see, the syntax is quite straightforward. The final concept for this introduction is conditional logic, which allows a program to execute different code blocks based on whether a condition is met. We use an if-else statement for this.
Let's look at an example.

(Live typing in a new code cell)

```
error_threshold = 0.05
calculated_error = 0.03

if calculated_error < error_threshold:
    print("Model performance is acceptable.")
else:
    print("Model performance is below the required threshold.")
```

(VOICEOVER)
Since the value of calculated_error is less than the error_threshold, the first block is executed. If I were to change the calculated error to a higher value, such as 0.06, and re-run the cell...
(Live editing of the cell)

(VOICEOVER)
...the else block is executed instead. This is the fundamental way we control program flow.
[ACTION: Screen recording switches back to the slide deck.]

[ON SCREEN: Slide 8 - "Thank you"]

(VOICEOVER)
To summarise, we have discussed the origins and significance of Python, particularly its role in AI and machine learning. We have also introduced the Google Colab IDE and demonstrated the use of variables, data types, and conditional statements.
This provides a solid foundation for our next session, where we will explore loops and functions to create more complex and efficient programs.

Thank you for your time. I encourage you to experiment with these concepts in your own notebook. I will see you in the next video.

[Outro Music Fades In]