IBM

[**Python for Data Science, AI & Development**](https://www.coursera.org/learn/python-for-applied-data-science-ai/home/welcome)

**Module 1: Python Basics**

Table of Contents

[Introduction 3](#_Toc186924182)

[What is Python used for...? 3](#_Toc186924183)

[Python Basics 3](#_Toc186924184)

[1. Syntax 3](#_Toc186924185)

[Example of a simple program: 4](#_Toc186924186)

[2. Variables and Data Types 4](#_Toc186924187)

[3. Control Structures 4](#_Toc186924188)

[Conditional Statements (if-else / if-elif-else) 5](#_Toc186924189)

[Loops 5](#_Toc186924190)

[4. Functions 5](#_Toc186924191)

[5. Libraries and Modules 5](#_Toc186924192)

[Python in Data Science 6](#_Toc186924193)

[Python in AI 6](#_Toc186924194)

[Python in Development 6](#_Toc186924195)

[Why Python for Data Science, AI & Development? 7](#_Toc186924196)

[Python History and Development 7](#_Toc186924197)

[Key Differences Between Python 2 and Python 3 7](#_Toc186924198)

[Modules and pip in Python! 8](#_Toc186924199)

[Real-life Example: 8](#_Toc186924200)

[The pip command 9](#_Toc186924201)

[Summary: 9](#_Toc186924202)

[Our First Program with print function 10](#_Toc186924203)

[Quick Quiz 12](#_Toc186924204)

[Other Parameters of Print Statement 12](#_Toc186924205)

[Introduction to Jupyter 14](#_Toc186924206)

[Why Choose Jupyter? 14](#_Toc186924207)

# Introduction

Python is a **high-level programming language** known for its **simplicity, versatility, and readability**, which makes it an ideal choice for beginners and professionals alike. It is widely used in **data science, artificial intelligence (AI), and software development**, due to its extensive libraries, frameworks, and supportive community.

## What is Python used for...?

* Python is used in Data Visualization to create plots and graphical representations.
* Python helps in Data Analytics to analyze and understand raw data for insights and trends.
* It is used in AI and Machine Learning to simulate human behavior and to learn from past data without hard coding.
* It is used to create web applications.
* It can be used to handle databases.
* It is used in business and accounting to perform complex mathematical operations along with quantitative and qualitative analysis.

# Python Basics

## Syntax

Python's syntax is **clean, readable**, and uses indentation to define blocks of code rather than curly braces {} like many other languages. This makes Python very beginner-friendly.

### Example of a simple program:



* No need for semicolons (;) at the end of lines.
* Indentation (spaces or tabs) defines code blocks

## 2. Variables and Data Types

Variables are used to store data in Python. You don't need to declare the type of variable; Python infers it automatically.

A screenshot of a computer

Description automatically generated

## 3. Control Structures

Python uses control structures to manage the flow of a program. It uses indentation (not braces {}) to define codes.

### Conditional Statements (if-else / if-elif-else)

A screenshot of a computer

Description automatically generated

### Loops

* **For Loop**: Iterate over a sequence.

A close-up of a white background

Description automatically generated

* **While Loop**: Repeat until a condition is false.

A white background with black text

Description automatically generated

## 4. Functions

Functions help you organize code into reusable blocks. You can create a function on your own and there are may built-in functions like len(), type(), and range() in python.

A close-up of a computer code

Description automatically generated

## 5. Libraries and Modules

Python has an extensive standard library, and you can install additional libraries using **pip** (Python Package Index).

**Importing Libraries**

**A number and a number

Description automatically generated with medium confidence**

# Python in Data Science

Python is extremely popular in data science because it provides powerful tools for working with data. Let’s explore:

* 1. **Data Manipulation**: The **Pandas** library is essential for data manipulation and analysis.
  2. **Data Visualization**: Python offers powerful tools like **Matplotlib** and **Seaborn** for creating data visualizations.
  3. **Machine Learning**: **Scikit-learn** simplifies tasks such as regression, classification, and clustering.

# Python in AI

Python is a leading language for artificial intelligence because of its flexibility and strong libraries.

1. **Deep Learning**: Libraries like **TensorFlow** and **PyTorch** simplify building and training neural networks.
2. **Natural Language Processing (NLP)**: Tools like **spaCy** and **NLTK** assist in analyzing and processing text data.
3. **Computer Vision**: Python’s **OpenCV** library is ideal for handling image and video processing tasks.

# Python in Development

Python is used extensively for creating applications:

1. **Web Development**: Frameworks like **Django** and **Flask** are widely used for creating dynamic web applications.
2. **Game Development**: **Pygame** is perfect for building simple 2D games.
3. **Automation**: Python excels at automating repetitive tasks, including file handling, data scraping, and testing.

# Why Python for Data Science, AI & Development?

* **Ease of Use**: Python's simple syntax reduces the learning curve.
* **Community Support**: A vast community helps with troubleshooting and learning.
* **Rich Libraries**: Libraries like NumPy, Pandas, TensorFlow, and Flask cater to specific needs.
* **Cross-Platform**: Works on Windows, macOS, and Linux.
* **Versatility**: Works across web development, AI, and data science.

# Python History and Development

* **Python 2** and **Python 3** are two major versions of the Python programming language.

Python 3 was introduced to address limitations and inconsistencies in Python 2, leading to improved functionality and modernized syntax. **Python 2** was released in 2000. It's not updated anymore and reached the "end of life" in 2020, meaning no more official support or fixes.

* **Python 3** was released in 2008. It’s the current version and actively maintained with new features and updates.

## Key Differences Between Python 2 and Python 3

* 1. **Printing text is different:**
* **In Python 2**, you write print "Hello" without parentheses.
* **In Python 3**, you must use parentheses: print("Hello").
  1. **Dealing with text (Unicode vs. ASCII):**
* **Python 2** uses ASCII by default, which is good for English but struggles with other languages. So, Unicode handling requires a special prefix (u).
* **Python 3** uses Unicode by default, which handles all languages and special characters better. (Think emojis! 😊)
  1. **Division behaves differently:**
* **In Python 2**, dividing integers (e.g., 5/2) gives a whole number: 2.
  + You’d need to write 5.0 / 2.0 to get 2.5.
* **In Python 3**, dividing integers (5/2) gives a float result: 2.5 by default.
  1. **Libraries are mostly Python 3-friendly now:**
* Many modern libraries and tools only support **Python 3**.
* If you use Python 2, you might miss out on new technology.
  1. **Input function is clearer in Python 3:**
* **In Python 2**, you use raw\_input() to get input as a string, but input() evaluates it like code (can be risky).
* **In Python 3**, you just use input(), which always takes input as a string (simpler and safer).
  1. **Syntax Improvements**
* Python 3 introduces many new features such as type hints, f-strings for formatted strings, and async programming.

## Modules and pip in Python!

Module is like a code library which can be used to borrow code written by somebody else in our python program. There are two types of modules in python:

* Built in Modules - These modules are ready to import and use and ships with the python interpreter. there is no need to install such modules explicitly.
* External Modules - These modules are imported from a third-party file or can be installed using a package manager like pip or conda. Since this code is written by someone else, we can install different versions of a same module with time.

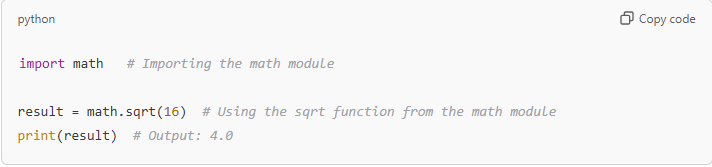
A module is like a container that holds related pieces of code, which can include functions, classes, and variables. You can think of a module as a toolbox. Each module has a specific purpose, like a toolbox containing tools for a particular task.

**Real-life Example:**

Imagine you have a kitchen. The kitchen has different drawers for different items: one drawer for spoons, another for knives, and another for forks. These drawers organize your tools in a way that makes it easier to find the one you need.

Similarly, in Python, a module is like a drawer that contains tools (functions, classes, etc.) you might need. You can import a module into your program to use its tools.

For example:



Here, math is the module, and sqrt is one of the tools (functions) it contains.

## The pip command

**pip** is a tool used to **install** and manage Python packages (which are like modules). It's like going to a **store** to buy new tools or equipment. When you need a tool that isn't in your current toolbox (module), you can use pip to go to the store and get it.

**Real-life Example:**

Imagine you need a **specialized tool** that you don’t have in your toolbox, like a **wrench** for a very specific type of bolt. You can go to the **store** (the internet, in this case) and buy the wrench (using pip), bringing it back to your toolbox to use whenever you need it.

In Python, **pip** helps you get packages (modules) from the **Python Package Index (PyPI)** so you can use them in your program.

It can be used as a package manager [pip](https://pip.pypa.io/en/stable/) to install a python module. Lets install a module called pandas using the following command

pip install pandas

**Using a module in Python (Usage)**

We use the import syntax to import a module in Python. Here is an example code:

A white background with black text

Description automatically generated

**Summary:**

* **Module**: Like a **toolbox** that contains different tools (functions, classes) to help you perform tasks.
* **pip**: Like a **store** where you can go to **buy** (install) new tools (packages/modules) that you need.

So, when you use pip, you're essentially going to the store to get new tools that you can later use in your Python programs by importing them as modules!

## Our First Program with print function

Today we will write our first ever python program from scratch. It will consist of a bunch of print statements. print can be used to print something on the console in python.

***Note: REPL: - Read Evaluate Print Loop***

In order to print in python, a valid object should be given. We use print function in python to pass a value (string, number, etc.) directly to it.



print()

We use ***single ‘ ‘ or double “ “*** quotation in python to ***print string*** however number and variable doesn’t need them. We can also print multiple values.



We can ***also format Output with f-strings*** in more readable ways using python. Note we need to ***use curly brackets {}*** for each variable.

A screenshot of a chat

Description automatically generated

**Controlling Line Breaks**

By default, print() adds a newline (a **line break**) after each call. This means that each print() statement will show its output on a new line.

A white rectangular object with a white line

Description automatically generated

If you don’t want a new line after the output, you can ***use the end argument*** to control what happens after the output. By default, ***end="\n***" adds a new line, but you can change this behavior.

A white background with text

Description automatically generated

## Other Parameters of Print Statement

1. object(s): Any object, and as many as you like. Will be converted to string before printed
2. sep = 'separator': Specify how to separate the objects, if there is more than one. Default is ' '
3. end='end': Specify what to print at the end. Default is '\n' (line feed)
4. file: An object with a write method. Default is sys.stdout

## Quick Quiz

* 1. **Write a program to print a poem in Python. Choose the poem of your choice and publish your repl.**

A screenshot of a computer

Description automatically generated

# Comments and Escape sequence

## Python Comments

A comment is a part of the coding file that the programmer does not want to execute, rather the programmer uses it to either explain a block of code or to avoid the execution of a specific part of code while testing.

* # is used for disabling single line in python
* """ or ''' is used for disabling block of codes and it needs to have same closing quotes
* Or we can select lines of codes and press ctrl + / to disables codes

A white background with black text

Description automatically generated

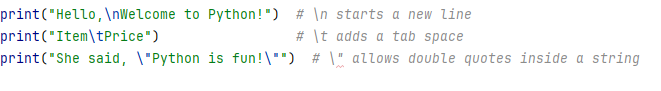
## Escape Sequences in Python:

Escape sequences are special characters in a string that perform actions like starting a new line, adding a tab space, or using quotation marks within a string.

Escape sequences start with a backslash (\).

Common Escape Sequences:

* \n - New Line
* \t - Tab Space
* \' - Single Quote
* \" - Double Quote



**Output**

Item Price

She said, "Python is fun!"

p is greater than 5.

## Errors in Python

Errors occur when the Python interpreter encounters something it cannot process, preventing the program from running or completing its task. Understanding errors helps debug code effectively and ensures smoother program execution.

**Types of Errors in Python**

1. **Syntax Errors**:
   * These occur when Python's syntax rules are broken. These are detected during parsing before execution.
   * A screenshot of a computer

     Description automatically generated
2. **Indentation Errors**:

* Python relies on indentation to define code blocks. Misaligned or missing indentation causes this error.

A screenshot of a computer

Description automatically generated

1. **Name Errors**:

* Raised when attempting to access a variable, function, or module that is not defined.

**A screenshot of a computer

Description automatically generated**

1. **Type Errors:**

* Occur when an operation is applied to an object of an inappropriate type.

A screenshot of a computer

Description automatically generated

1. **Value Errors**:

* Occur when a function receives the right type of argument but an invalid value.

A white card with black text

Description automatically generated

1. **Index Errors**:

* Raised when trying to access an index that doesn’t exist in a list, string, or tuple.

A white background with black text

Description automatically generated

1. **Key Errors**:

* Occur when trying to access a dictionary key that doesn’t exist.

A white background with black text

Description automatically generated

1. **ZeroDivisionError**:

* Raised when dividing by zero, which is mathematically undefined.

A screenshot of a phone

Description automatically generated

1. **Attribute Errors**:

* Raised when an invalid attribute is accessed for an object.

A screenshot of a phone

Description automatically generated

## How to Handle Errors in Python

Python provides the **try-except** mechanism to gracefully handle errors and prevent program crashes.

**Basic Error Handling Syntax**

**A white screen with black text

Description automatically generated**

* 1. **Handle Division by Zero:**

**A white rectangular object with green text

Description automatically generated**

* 1. **Handle Multiple Errors**:

**A white background with green text

Description automatically generated**

* 1. **Catch All Errors**: Exception Handling (Not recommended for debugging purposes):

**A white rectangular object with red and green text

Description automatically generated**

* 1. **Using finally** (Code that always runs):

**A white background with green and black text

Description automatically generated**

* 1. **else Block in Try-Except:**

**A white background with green text

Description automatically generated**

* 1. **Raising Exceptions**

**A white background with green text

Description automatically generated**

## Tips for Error Handling

* Be specific about the error type in except to avoid catching unrelated errors.
* Use finally for cleanup (e.g., closing files or releasing resources).
* Test your code thoroughly to minimize errors in the first place.
* Avoid overusing try-except

With proper error handling, your programs become more robust and user-friendly

# Introduction to Jupyter

Jupyter is a free web application that ***allows you to create and share documents combining live code, equations, visualizations, and text***. It ***provides an interactive environment supporting multiple programming languages, including Python***, R, and Julia, but is especially effective with Python. ***Jupyter focuses on "notebooks,"*** which are documents that mix code, charts, explanations, equations, and multimedia. These notebooks make it easy to work on, share, and collaborate on computational projects.

## Why Choose Jupyter?

Jupyter is popular because it’s flexible and easy to use, catering to both beginners and experts. Whether you're new to coding or experienced in data science, Jupyter offers a simple platform for writing, testing, and sharing code. Its interactive interface lets you explore data, try out algorithms, and visualize results all in one place.

**Key Features of Jupyter**

* **Interactive Coding**: Write and run code in sections (cells) and see results instantly, making it great for experimenting and refining your work.
* **Supports Many Languages**: Originally for Python, now supports R, Julia, and more, thanks to its kernel system.
* **Rich Output**: You can include interactive graphs, charts, images, and even videos within Jupyter notebooks, which makes data visualization and storytelling much easier.
* **Data Science Integration**: Works with libraries like NumPy, Pandas, Matplotlib, scikit-learn, TensorFlow, and PyTorch for data analysis and machine learning.
* **Collaboration and Sharing**: You can easily share Jupyter notebooks via email, GitHub, or Jupyter Notebook Viewer. This promotes collaboration, peer review, and sharing of ideas, making it perfect for group projects or interdisciplinary work.