**Python Basics**

This module teaches the basics of Python and begins by exploring some of the different data types such as integers, real numbers, and strings. Continue with the module and learn how to use expressions in mathematical operations, store values in variables, and the many different ways to manipulate strings.

**Learning Objectives**

* Demonstrate an understanding of types in Python by converting or casting data types such as strings, floats, and integers.
* Interpret variables and solve expressions by applying mathematical operations.
* Describe how to manipulate strings by using a variety of methods and operations.
* Build a program in JupyterLab to demonstrate your knowledge of types, expressions, and variables.
* Work with, manipulate, and perform operations on strings in Python.

# **About the Course**

## **Course Introduction**

Key Points:

* **Python's Popularity**: It's a top choice for beginners and is widely used in data science and AI.
* **Versatility**: Python can handle various tasks like data analysis, web scraping, and machine learning.
* **Course Structure**:
  + **Module 1**: Basics of Python (types, variables, string operations).
  + **Module 2**: Data structures (lists, tuples, dictionaries, sets).
  + **Module 3**: Programming fundamentals (conditions, loops, functions).
  + **Module 4**: Working with data using libraries like NumPy and Pandas, and using APIs.
* **Hands-on Projects**: You will apply what you learn through real-world projects.

Summary:

The course is designed for beginners and will teach you Python programming from the ground up, covering essential concepts and practical applications. By the end, you'll be able to create basic programs and work with data effectively.

## **About this Course**

This course was designed to provide the building blocks for Python programming and data collection for those choosing a career in Data Science, Data Engineering, AI or Application Development.

Initially conceived as a foundation course for Data Science and AI it has been refreshed several times to keep pace with emerging career options. Additional content has been added which is applicable to Data Science, Data Engineering, AI or Application Development.

After completing this course you will have learned foundational skills in Python programming which you can then go on to apply in the Python Project course for your chosen career.  The Python Project courses involve real world scenarios where you are in charge of a final project as a Data Scientist, a Data Engineer, or in AI and Application Development. By finishing this course and your follow-on Python Project, you will gain the basic skills to continue the steps on your chosen career path.

**Note: This course is a pre-requisite for the Python Project courses and should be completed in full before attempting the appropriate Python Project course.**

## **Course Overview**

Welcome to the Python for Data Science, AI, and Development course. After completing this course, you'll possess the basic knowledge of Python and acquire a good understanding of different data types. You’ll also learn to use lists and tuples, dictionaries, and Python sets. Additionally, you’ll acquire the concepts of condition and branching and will know how to implement loops, create functions, perform exception handling, and create objects. Furthermore, you’ll be proficient in reading and writing files and will be able to implement unique ways to collect data using APIs and web scraping. In addition to the module labs, you'll prove your skills in a peer-graded project and your overall knowledge with the final quiz.

**Course Content**

This course is divided into five modules. You should set a goal to complete at least one module per week.

**Module 1:** Python Basics

* About the Course
* Types
* Expressions and Variables
* String Operations

**Module 2:** Python Data Structures

* Lists and Tuples
* Dictionaries
* Sets

**Module 3:** Python Programming Fundamentals

* Conditions and Branching
* Loops
* Functions
* Exception Handling
* Objects and Classes
* Practice with Python Programming Fundamentals

**Module 4:** Working with Data in Python

* Reading and Writing Files with Open
* Pandas
* Numpy in Python

**Module 5:** APIs and Data Collection

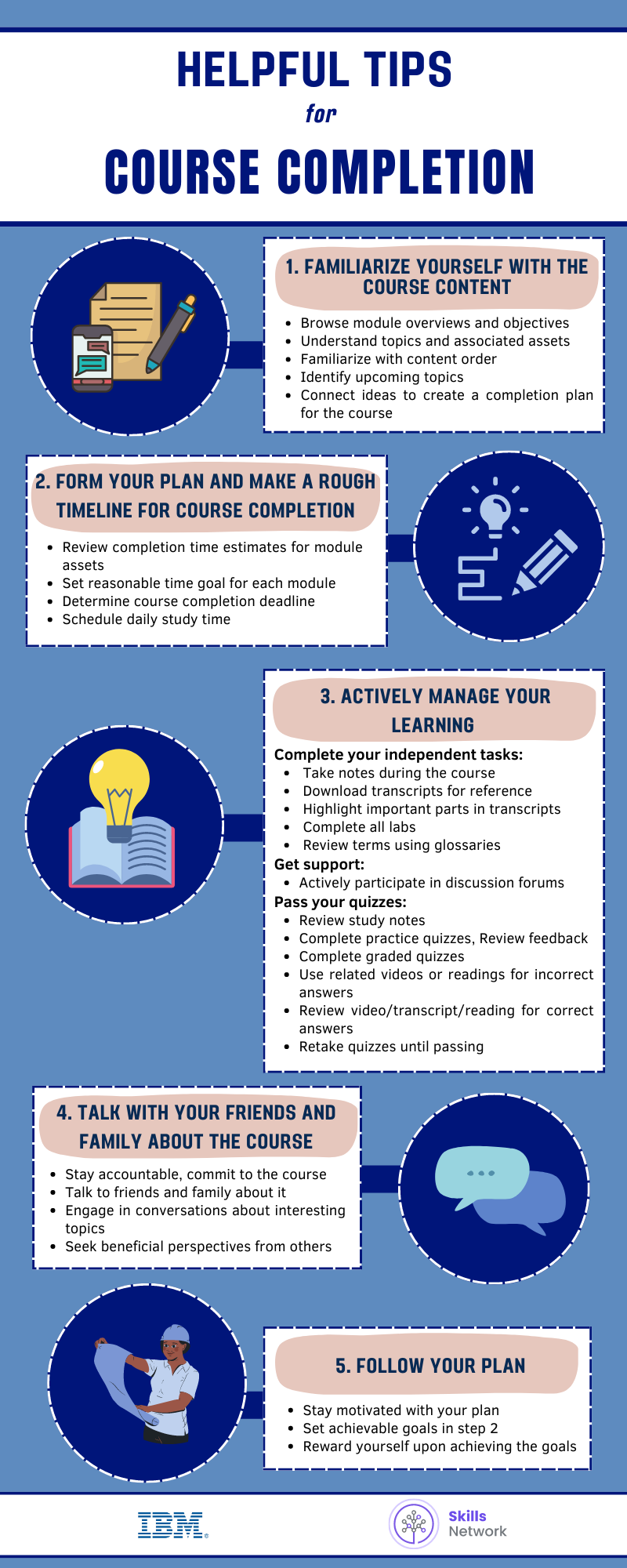
* Simple APIs
* REST APIs, Web Scraping, and Working with Files
* Final Exam

The course contains a variety of learning assets: Videos, activities, labs, projects, practice, graded quizzes, and readings. The videos and readings present the instruction. Labs and activities support that instruction with hands-on learning experiences. Discussions allow you to interact and learn from your peers. A peer-review project that mimics real-world scenarios encourage you to showcase your skills, Practice quizzes enable you to test your knowledge of what you learned. Finally, graded quizzes indicate how well you have learned the course concepts.

Enjoy the course!

## **Helpful Tips for Course Completion**

**Helpful Tips for Course Completion**



# **Getting Started with Python and Jupyter Notebook**

## **Introduction to Python**

Simple Explanation:

* **Who Uses Python?**: Many people, especially in data science, use Python because it's easy to read and write.
* **Benefits of Python**: It's great for beginners and experienced programmers alike. You can do a lot with less code compared to other languages.
* **Where is Python Used?**: Python is used in various fields like data science, artificial intelligence, web development, and more.
* **Community Support**: There’s a large community that helps each other, making it easier to learn.
* **Diversity and Inclusion**: The Python community promotes diversity and inclusion, ensuring everyone feels welcome.

Summary:

* Python is a widely used programming language, especially in data science.
* It has a clear syntax, making it beginner-friendly.
* Many big companies like Google and NASA use Python.
* The Python community is supportive and focuses on diversity and inclusion.

## **Introduction to Jupyter**

Jupyter is a freely available web application that enables creation and sharing of documents containing equations, live coding, visualizations, and narrative text. Jupyter provides an interactive computing environment that supports multiple programming languages, including Python, R, Julia, and more, but it shines brightest when used with Python. Jupyter revolves around notebooks, documents containing a mix of code, visualizations, narrative text, equations, and multimedia content. These notebooks allow users to create, share, and collaborate on computational projects seamlessly.

**Why Jupyter?**

Jupyter's popularity stems from its flexibility and ease of use. Regardless of your level of programming expertise, whether you're an experienced coder or embarking on your data science journey, Jupyter offers an intuitive platform for writing, testing, and sharing code. Its interactive interface enables data exploration, algorithm experimentation, and result visualization—all seamlessly integrated within a unified environment.

**Key Features of Jupyter**

Here are some key features and advantages of Jupyter:

1. **Interactive Computing**: Jupyter notebooks enable users to write and execute code interactively. This means you can run code cells individually and see the output immediately, fostering an iterative approach to coding and experimentation.
2. **Support for Multiple Languages**: While Jupyter was initially developed for Python (hence the name, which stands for Julia, Python, and R), it now supports various programming languages through its kernel system. This flexibility makes Jupyter suitable for various computational tasks and interdisciplinary collaboration.
3. **Rich Output**: Jupyter Notebooks support rich media integration, allowing users to generate interactive plots, charts, images, videos, and more directly within the document. This makes visualizing data, communicating findings, and creating compelling narratives easier.
4. **Integration with Data Science Libraries**: Jupyter seamlessly integrates with popular libraries and frameworks used in the data science ecosystem, such as NumPy, Pandas, Matplotlib, sci-kit-learn, TensorFlow, and PyTorch. This allows users to leverage the full power of these tools within the notebook environment for tasks like data manipulation, visualization, machine learning, and deep learning.
5. **Collaboration and Sharing**: Jupyter promotes collaboration and reproducibility by allowing users to share their notebooks with others via email, GitHub, or the Jupyter Notebook Viewer. This facilitates knowledge sharing, peer review, and interdisciplinary collaboration, as users can easily exchange ideas, code snippets, and best practices.

**Jupyter in Data Science**

Jupyter has become an indispensable tool for researchers, analysts, and developers in data science. Its seamless integration with popular libraries such as NumPy, pandas, and sci-kit-learn makes it the go-to choice for data manipulation, analysis, and machine learning. Jupyter provides a user-friendly interface, interactive capabilities, and robust collaboration features, making it an essential tool for anyone involved in data analysis, scientific research, education, or software development. Whether you're exploring data, building machine learning models, teaching a class, or conducting research, Jupyter empowers you to work more efficiently and share your insights with others.

**Getting Started with Jupyter**

Now that you can glimpse what Jupyter offers, it's time to dive in and experience its capabilities firsthand. Our Getting Started with Jupyter video will walk you through the basics of setting up and using Jupyter, empowering you to unleash the full potential of Python and embark on your data science journey with confidence.

So, let's jump into the world of Jupyter and unlock a world of possibilities in Python and data science!

## **Getting Started with Jupyter**

Key Points:

* **Running Cells**: You can run code in small sections called cells. You can insert new cells, delete them, or run all cells at once.
* **Multiple Notebooks**: You can open and work with more than one notebook at the same time.
* **Presenting Results**: You can use Markdown to add text and titles to your notebooks, making it easier to present your work.
* **Shutting Down**: After finishing, you can shut down your notebooks to free up memory.

Summary:

In this video, you learned how to:

* Run, insert, and delete code cells in Jupyter Notebooks.
* Work with multiple notebooks simultaneously.
* Present your results using Markdown.
* Shut down your notebook sessions when done.

# **Types**

## **Data types**

Key Points:

* **Data Types**: Python has various data types, including:
  + **Integers**: Whole numbers (e.g., 11).
  + **Floats**: Real numbers that can have decimals (e.g., 21.213).
  + **Strings**: Sequences of characters (e.g., words).
  + **Booleans**: True or False values.
* **Type Command**: You can check the type of a value using the type() function in Python.
* **Typecasting**: You can change one data type to another:
  + Convert an integer to a float (e.g., 2 to 2.0).
  + Be careful when converting a float to an integer, as you may lose information (e.g., 1.1 becomes 1).
* **Boolean Values**:
  + True (1) and False (0) can also be converted to integers or floats.

Summary:

The video explains that Python has different data types like integers, floats, strings, and booleans. You can check the type of a value and convert between types using typecasting. Understanding these data types is essential for programming in Python.

# **Expressions and Variables**

**expressions** and **variables** in Python.

Simple Explanation:

* **Expressions**: These are operations that Python performs, like basic math. For example:
  + **Addition**: 80 + 80 = 160
  + **Subtraction**: 5 - 10 = -5
  + **Multiplication**: 5 \* 5 = 25
  + **Division**: 25 / 5 = 5.0 (results in a float)

Python follows specific rules for the order of operations, meaning it does multiplication before addition unless parentheses are used.

* **Variables**: These are like containers that store values. You can assign a value to a variable using the equal sign (=). For example:
  + my\_variable = 1 assigns the value 1 to my\_variable.
  + You can change the value later, like my\_variable = 10, and the old value is replaced.

Summary of the Video:

* The video explains how to perform basic arithmetic operations in Python using expressions.
* It introduces variables, showing how to assign and change values.
* It emphasizes the importance of using meaningful variable names for better code readability.
* The video also mentions that you can perform calculations using variables and store results in new variables.

# **String Operations**

summary of the key points:

* **Definition**: A string is a sequence of characters, which can include letters, numbers, spaces, and special characters.
* **Indexing**: You can access specific characters in a string using an index. The first character is at index 0, and you can also use negative indexing to access characters from the end.
* **Slicing**: You can extract parts of a string using slicing, which allows you to select a range of characters.
* **Length**: The len() function gives you the total number of characters in a string.
* **Concatenation**: You can combine two strings using the + operator.
* **Replication**: You can create multiple copies of a string by multiplying it with a number.
* **Immutability**: Strings cannot be changed directly; instead, you create new strings based on modifications.
* **Escape Sequences**: Special characters can be included in strings using escape sequences, like \n for a new line or \t for a tab.
* **String Methods**: Strings have methods that allow you to manipulate them, such as converting to uppercase or finding substrings.

This summary captures the main concepts discussed in the video. If you have any specific questions about these points, feel free to ask!

## **Reading**

**Reading: Format Strings in Python**

**Estimates effort: 5 mins**

Format strings are a way to inject variables into a string in Python. They are used to format strings and produce more human-readable outputs. There are several ways to format strings in Python:

**String interpolation (f-strings)**

Introduced in Python 3.6, f-strings are a new way to format strings in Python. They are prefixed with 'f' and use curly braces {} to enclose the variables that will be formatted. For example:

1. name = "John"
2. age = 30
3. print(f"My name is {name} and I am {age} years old.")

Copied!

This will output:

1. My name is John and I am 30 years old.

Copied!

**str.format()**

This is another way to format strings in Python. It uses curly braces {} as placeholders for variables which are passed as arguments in the format() method. For example:

1. name = "John"
2. age = 50
3. print("My name is {} and I am {} years old.".format(name, age))

Copied!

This will output:

1. My name is John and I am 50 years old.

Copied!

**% Operator**

This is one of the oldest ways to format strings in Python. It uses the % operator to replace variables in the string. For example:

1. name = "Johnathan"
2. age = 30
3. print("My name is %s and I am %d years old." % (name, age))

Copied!

This will output:

1. My name is Johnathan and I am 30 years old.

Copied!

“**My name is %s and I am %d years old.**“: This is a string that includes format specifiers:

* **%s**: This is a placeholder for a string.
* **%d**: This is a placeholder for an integer.

**% (name, age)**: This is a tuple containing the variables name and age. The values of these variables will replace the placeholders in the string.

Each of these methods has its own advantages and use cases. However, f-strings are generally considered the most modern and preferred way to format strings in Python due to their readability and performance.

**Additional capabilities**

F-strings are also able to evaluate expressions inside the curly braces, which can be very handy. For example:

1. x = 10
2. y = 20
3. print(f"The sum of x and y is {x+y}.")

Copied!

This will output:

1. The sum of x and y is 30.

Copied!

**Raw String (r’’)**

In Python, raw strings are a powerful tool for handling textual data, especially when dealing with escape characters. By prefixing a string literal with the letter ‘r’, Python treats the string as raw, meaning it interprets backslashes as literal characters rather than escape sequences.

Consider the following examples of regular string and raw string:

**Regular string:**

1. regular\_string = "C:\new\_folder\file.txt"
2. print("Regular String:", regular\_string)

Copied!

This will output:

1. Regular String: C:
2. ew\_folderile.txt

Copied!

In the regular string regular\_string variable, the backslashes (\n) are interpreted as escape sequences. Therefore, \n represents a newline character, which would lead to an incorrect file path representation.

**Raw string:**

1. raw\_string = r"C:\new\_folder\file.txt"
2. print("Raw String:", raw\_string)

Copied!

This will output:

1. Raw String: C:\new\_folder\file.txt

Copied!

However, in the raw string raw\_string, the backslashes are treated as literal characters. This means that \n is not interpreted as a newline character, but rather as two separate characters, ‘’ and ‘n’. Consequently, the file path is represented exactly as it appears.

# **Module 1 Summary**

Congratulations! You have completed this module. At this point, you know that:

* Python can distinguish among data types such as integers, floats, strings, and Booleans.
* Integers are whole numbers that can be positive or negative.
* Floats include integers as well as decimal numbers between the integers.
* You can convert integers to floats using typecasting and vice-versa.
* You can convert integers and floats to strings.
* You can convert an integer or float value to True (1) or False (0).
* Expressions in Python are a combination of values and operations used to produce a single result.
* Expressions perform mathematical operations such as addition, subtraction, multiplication, and so on.
* We can use // to perform integer division, which results in an integer value by discarding the fractional part.
* Python follows the order of operations (BODMAS) to perform operations with multiple expressions.
* Variables store and manipulate data, allowing you to access and modify values throughout your code.
* The assignment operator "=" assigns a value to a variable.
* ":" denotes the value of the variable within the code.
* Assigning another value to the same variable overrides the previous value of that variable.
* You can perform mathematical operations on variables using the same or different variables.
* Modifying the value of one variable will affect other variables only if they reference the same mutable object.
* Python string operations involve manipulating text data using tasks such as indexing, concatenation, slicing, and formatting.
* A string is usually written within double quotes or single quotes, including letters, white space, digits, or special characters.
* A string attaches to another variable and is an ordered sequence of characters.
* Characters in a string identify their index numbers, which can be positive or negative.
* We use strings as a sequence to perform sequence operations.
* You can input a stride value to perform slicing while operating on a string.
* Operations like finding the length of the string, combining, concatenating, and replicating, result in a new string.
* You cannot modify an existing string; they are immutable.
* You can use escape sequences with a backslash (\) to change the layout of a string. (For example, \n for a new line, \t for a tab, and \\ for a backslash, etc.)
* In Python, you perform tasks such as searching, modifying, and formatting text data with its pre-built string methods functions.
* You apply a method to a string to change its value, resulting in another string.
* You can perform actions such as changing the case of characters in a string, replacing items in a string, finding items in a string, and so on using pre-built string methods.

# **Cheatsheet**

<https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-PY0101EN-SkillsNetwork/labs/Cheat_Sheet_Week-1.md.html>  
  
Glossary

| **Term** | **Definition** |
| --- | --- |
| AI | AI (artificial intelligence) is the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings. |
| Application development | Application development, or app development, is the process of planning, designing, creating, testing, and deploying a software application to perform various business operations. |
| Arithmetic Operations | Arithmetic operations are the basic calculations we make in everyday life like addition, subtraction, multiplication and division. It is also called as algebraic operations or mathematical operations. |
| Array of numbers | Set of numbers or objects that follow a pattern presented as an arrangement of rows and columns to explain multiplication. |
| Assignment operator in Python | Assignment operator is a type of Binary operator that helps in modifying the variable to its left with the use of its value to the right. The symbol used for assignment operator is "=". |
| Asterisk | Symbol "\* " used to perform various operations in Python. |
| Backslash | A backslash is an escape character used in Python strings to indicate that the character immediately following it should be treated in a special way, such as being treated as escaped character or raw string. |
| Boolean | Denoting a system of algebraic notation used to represent logical propositions by means of the binary digits 0 (false) and 1 (true). |
| Colon | A colon is used to represent an indented block. It is also used to fetch data and index ranges or arrays. |
| Concatenate | Link (things) together in a chain or series. |
| Data engineering | Data engineers are responsible for turning raw data into information that an organization can understand and use. Their work involves blending, testing, and optimizing data from numerous sources. |
| Data science | Data Science is an interdisciplinary field that focuses on extracting knowledge from data sets which are typically huge in amount. The field encompasses analysis, preparing data for analysis, and presenting findings to inform high-level decisions in an organization. |
| Data type | Data type refers to the type of value a variable has and what type of mathematical, relational or logical operations can be applied without causing an error. |
| Double quote | Symbol “ “ used to represent strings in Python. |
| Escape sequence | An escape sequence is two or more characters that often begin with an escape character that tell the computer to perform a function or command. |
| Expression | An expression is a combination of operators and operands that is interpreted to produce some other value. |
| Float | Python float () function is used to return a floating-point number from a number or a string representation of a numeric value. |
| Forward slash | Symbol “/“ used to perform various operation sin Python |
| Foundational | Denoting an underlying basis or principle; fundamental. |
| Immutable | Immutable Objects are of in-built datatypes like int, float, bool, string, Unicode, and tuple. In simple words, an immutable object can’t be changed after it is created. |
| Integer | An integer is the number zero (0), a positive natural number (1, 2, 3, and so on) or a negative integer with a minus sign (−1, −2, −3, and so on.) |
| Manipulate | Is the process of modifying a string or creating a new string by making changes to existing strings. |
| Mathematical conventions | A mathematical convention is a fact, name, notation, or usage which is generally agreed upon by mathematicians. |
| Mathematical expressions | Expressions in math are mathematical statements that have a minimum of two terms containing numbers or variables, or both, connected by an operator in between. |
| Mathematical operations | The mathematical “operation” refers to calculating a value using operands and a math operator. |
| Negative indexing | Allows you to access elements of a sequence (such as a list, a string, or a tuple) from the end, using negative numbers as indexes. |
| Operands | The quantity on which an operation is to be done. |
| Operators in Python | Operators are used to perform operations on variables and values. |
| Parentheses | Parentheses is used to call an object. |
| Replicate | To make an exact copy of. |
| Sequence | A sequence is formally defined as a function whose domain is an interval of integers. |
| Single quote | Symbol ‘ ‘ used to represent strings in python. |
| Slicing in Python | Slicing is used to return a portion from defined list. |
| Special characters | A special character is one that is not considered a number or letter. Symbols, accent marks, and punctuation marks are considered special characters. |
| Stride value | Stride is the number of bytes from one row of pixels in memory to the next row of pixels in memory. |
| Strings | In Python, Strings are arrays of bytes representing Unicode characters. |