Python

Variables and Data Types

Learning Objectives

By the end of this lesson, you will be able to:

- Describe variables in Python
- Declare variables in Python
- Reference an object
- Define mutable and Immutable variables
- Explain Python sequence

TECHNOLOGY

Variables

Variables

In Python, a variable is a named reference to a value that can be stored in memory.

Variables are used to store data that can be manipulated or used in calculations, and they can be assigned different values at different times during the execution of a program.

To define a variable in Python, you simply need to choose a name for the variable and use the equals sign "=" to assign a value to it.

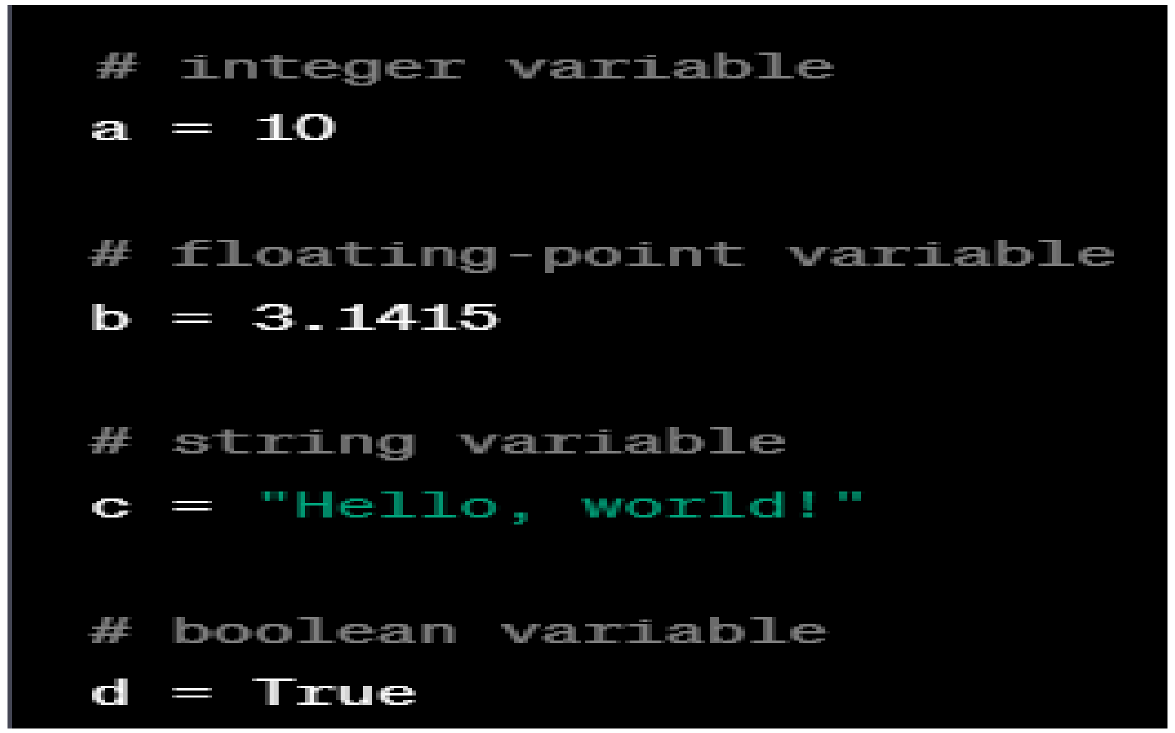
Variables

Python variables can store different types of data, including numbers (integers, floating-point numbers), strings (sequences of characters), and Boolean values (True or False).

Python does not require you to declare the data type of a variable before assigning a value to it, as it will automatically infer the data type based on the value assigned.

Variables

Here are some examples of how to define variables of different types in Python:



```
# integer variable
a = 10

# floating-point variable
b = 3.1415

# string variable
c = "Hello, world!"

# boolean variable
d = True
```

It's important to choose descriptive variable names that reflect the purpose of the data being stored. For example, if you're storing the age of a person, you might use the variable name **age**.

This makes your code more readable and easier to understand for others who might be working with your code.

TECHNOLOGY

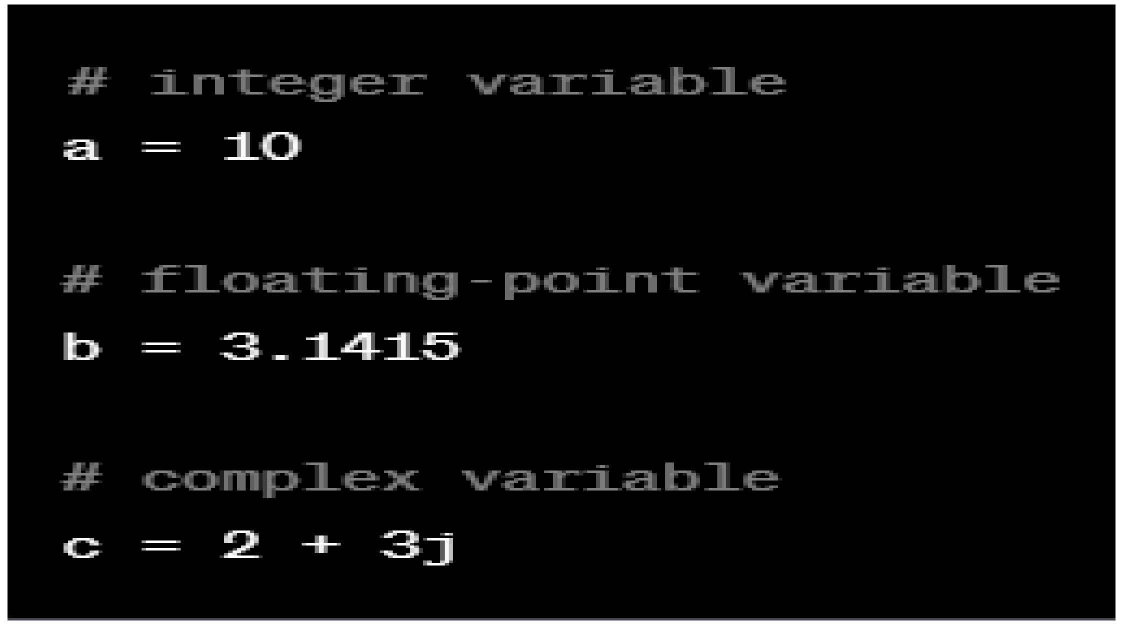
Types of Variables in Python

simplilearn

Types of Variables in Python

In Python, there are several types of variables that you can use depending on the type of data you want to store. Here are the most common types of variables in Python:

1. **Numeric Variables:** These are variables used to store numeric values. Python has three types of numeric variables: integers, floating-point numbers, and complex numbers.



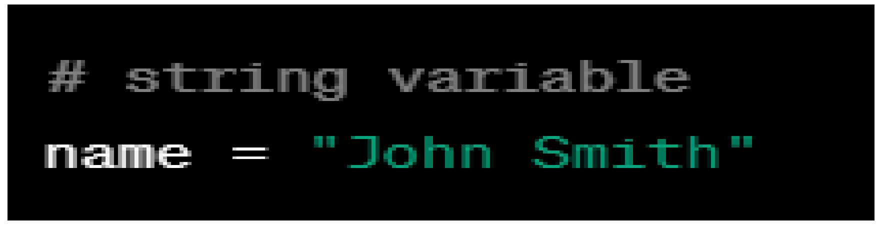
```
# integer variable
a = 10

# floating-point variable
b = 3.1415

# complex variable
c = 2 + 3j
```

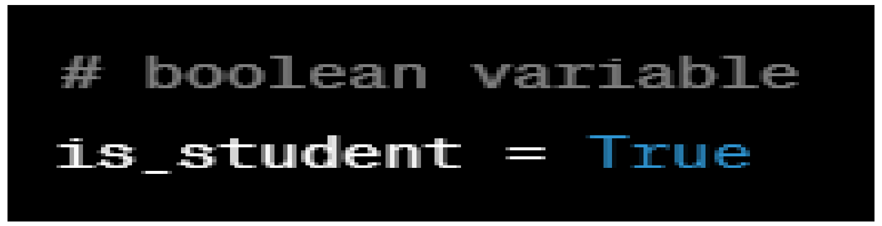
Types of Variables in Python

2. **String Variables:** These are variables used to store text data and are enclosed in quotes (single or double).



```
# string variable
name = "John Smith"
```

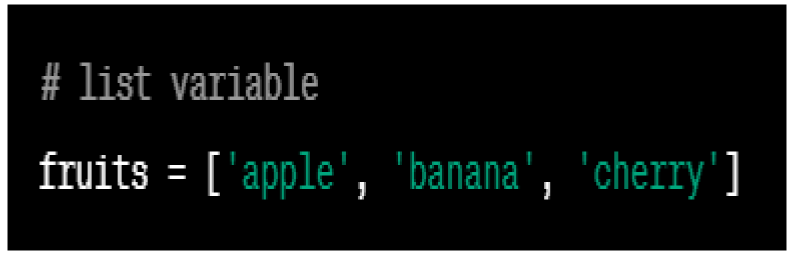
3. **Boolean Variables:** These are variables used to store either True or False values.



```
# boolean variable
is_student = True
```

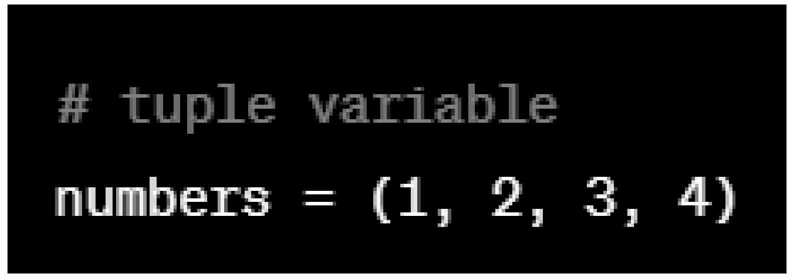
Types of Variables in Python

4. **List Variables:** These are variables used to store a collection of values, which can be of any data type. Lists are ordered and changeable.



```
# list variable
fruits = ['apple', 'banana', 'cherry']
```

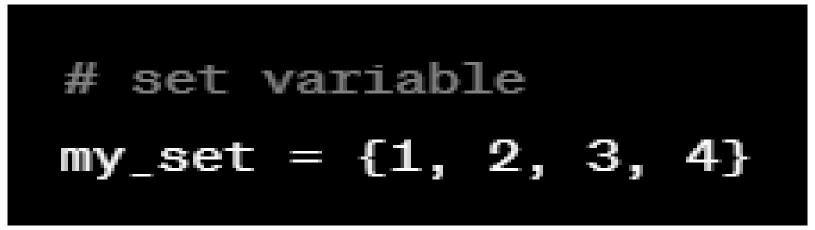
5. **Tuple Variables:** These are variables used to store a collection of values, which can be of any data type. Tuples are ordered and immutable.



```
# tuple variable
numbers = (1, 2, 3, 4)
```

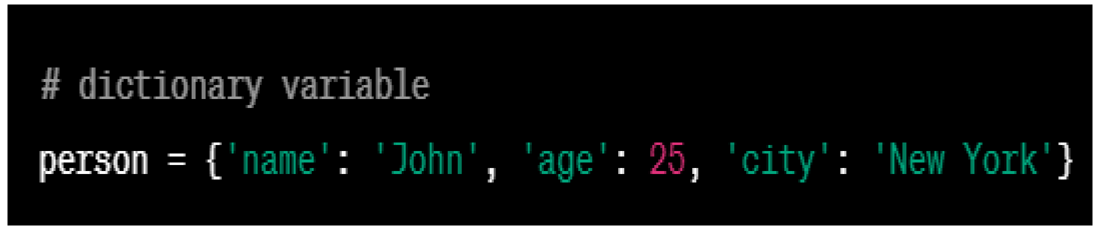
Types of Variables in Python

6. **Set Variables:** These are variables used to store a collection of unique values, which can be of any data type. Sets are unordered and mutable.



```python
# set variable
my_set = {1, 2, 3, 4}
```

7. **Dictionary Variables:** These are variables used to store key-value pairs, where each key is associated with a value. Dictionaries are unordered and mutable.



```python
# dictionary variable
person = {'name': 'John', 'age': 25, 'city': 'New York'}
```

It's important to choose the appropriate type of variable depending on the data you want to store and how you plan to use it in your code.

TECHNOLOGY

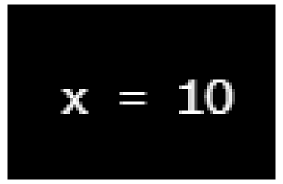
Declaring Variables in Python

Declaring Variables in Python

- In Python, you don't need to declare the data type of a variable before assigning a value to it, as the data type is inferred automatically based on the value assigned.

- To declare a variable in Python, you simply need to choose a name for the variable and use the equals sign "=" to assign a value to it.

- Here's an example:



```
x = 10
```

In this example, "x" is the variable name, and 10 is the value assigned to it.

Python will automatically infer that "x" is an integer variable.

Declaring Variables in Python

If you want to declare multiple variables at once, you can separate them with commas.

Here's an example:

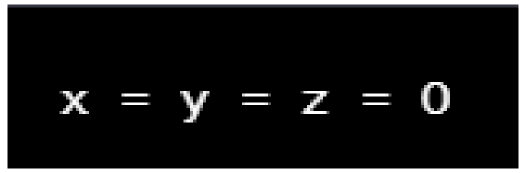
```
x, y, z = 10, 3.1415, "Hello"
```

In this example, "x" is an integer variable with a value of 10, "y" is a floating-point variable with a value of 3.1415, and "z" is a string variable with a value of "Hello".

Declaring Variables in Python

You can also assign the same value to multiple variables at once.

Here's an example:



```
x = y = z = 0
```

In this example, "x", "y", and "z" are all assigned a value of 0.

It's important to choose descriptive variable names that reflect the purpose of the data being stored. This makes your code more readable and easier to understand for others who might be working with your code.

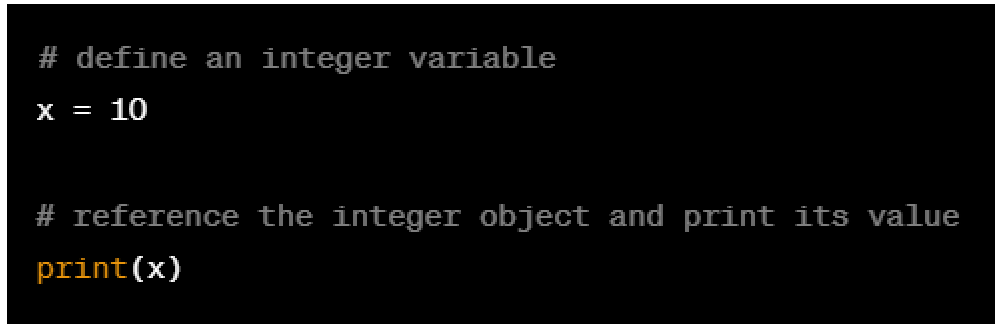
TECHNOLOGY

Referencing an Object in Python

Referencing an Object in Python

In Python, you can reference an object by using its variable name. When you reference an object, you are accessing the memory location where the object is stored, and you can perform operations on the object or retrieve its value.

Here's an example of how to reference an integer object:



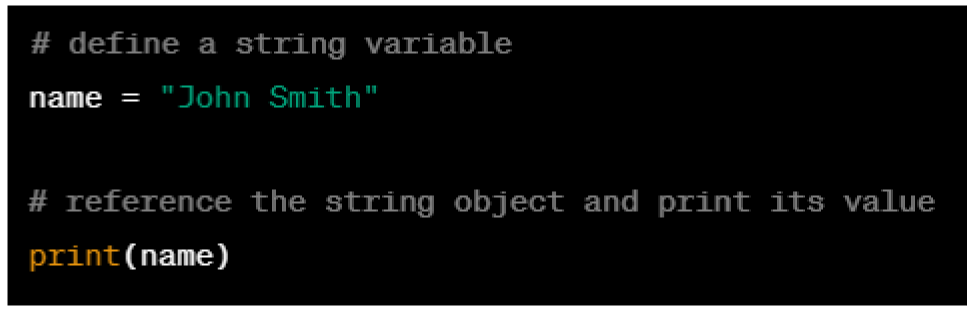
```
# define an integer variable
x = 10

# reference the integer object and print its value
print(x)
```

In this example, "x" is the variable name that references the integer object with a value of 10. When we print "x", we retrieve the value of the integer object and print it to the console.

Referencing an Object in Python

Here's another example of how to reference a string object:



```python
# define a string variable
name = "John Smith"

# reference the string object and print its value
print(name)
```

In this example, "name" is the variable name that references the string object with a value of "John Smith". When we print "name", we retrieve the value of the string object and print it to the console.

Referencing an Object in Python

You can also reference other types of objects, such as lists, dictionaries, or functions, by using their variable names.

The important thing to remember is that when you reference an object, you are referencing the memory location where the object is stored, not just its value.

TECHNOLOGY

Mutable and Immutable Variables in Python

Mutable & Immutable Variables

In Python, variables can be classified as either mutable or immutable depending on whether the value of the variable can be changed or not.

Immutable variables are those whose values cannot be changed once they are created, while mutable variables are those whose values can be changed after creation.

Mutable & Immutable Variables

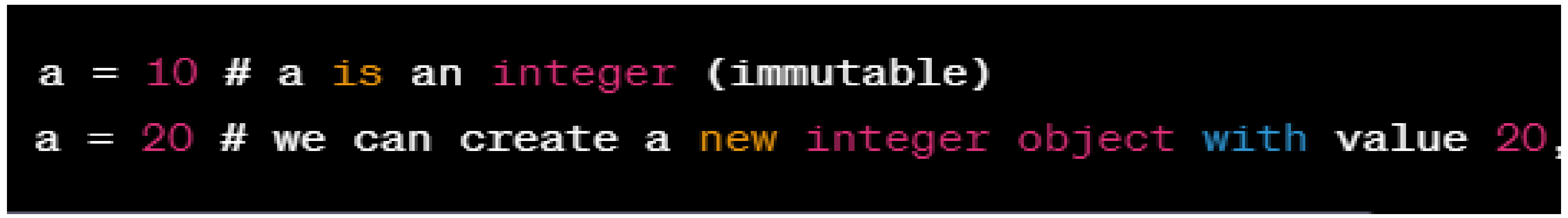
Immutable Variables

Immutable variables are those that cannot be changed once they are created.

Examples of immutable variables in Python include:

- Numbers (integers, floats, complex numbers)
- Strings
- Tuples
- Frozen sets

Once an immutable variable is created, its value cannot be changed. For example:



```
a = 10 # a is an integer (immutable)
a = 20 # we can create a new integer object with value 20,
```

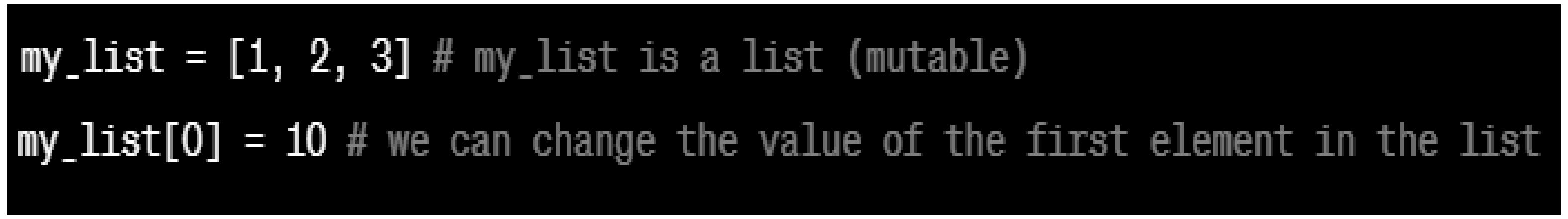
Mutable & Immutable Variables

Mutable Variables

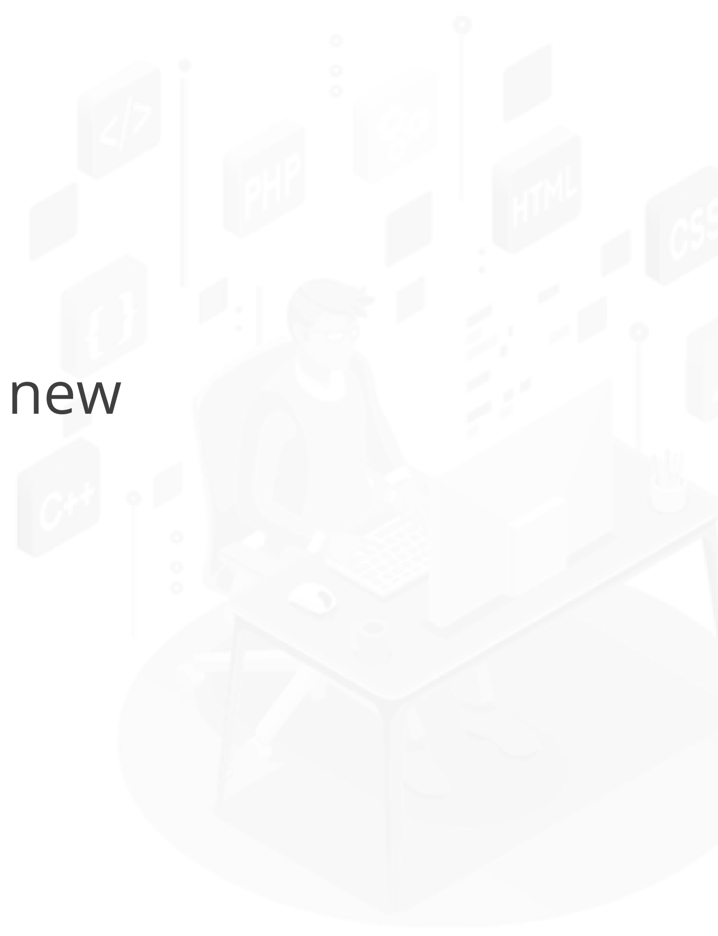
Mutable variables are those that can be changed after they are created. Examples of mutable variables in Python include:

- Lists
- Sets
- Dictionaries

With mutable variables, you can change the value of the variable without creating a new object. For example:



```
my_list = [1, 2, 3] # my_list is a list (mutable)
my_list[0] = 10 # we can change the value of the first element in the list
```



Mutable & Immutable Variables

- Understanding variable mutability or immutability is crucial for handling complex data structures and optimizing code for performance.
- Immutable objects are safer for multi-threaded programs and critical data integrity, while mutable objects can be more efficient for large data sets.
- The mutability or immutability of a variable is an important concept to understand, especially when dealing with complex data structures or when optimizing code for performance.
- In general, immutable objects are safer to use in multi-threaded programs or in situations where data integrity is critical, while mutable objects can be more efficient when dealing with large data sets.

TECHNOLOGY

Python Numerics & Sequence

Python Numerics & Sequence

Python provides several built-in data types to represent numerical and sequential data.

Numerical Data Types

Python provides the following built-in numerical data types:

- Integers (int): Integers are whole numbers, such as 1, 2, 3, and -4. In Python, integer values have no limit in size except for the amount of memory available on the computer.
- Floating-point numbers (float): Floating-point numbers are real numbers with a decimal point, such as 1.0, 2.5, and -3.14159.
- Complex numbers (complex): In Python, complex numbers are represented as a pair of floating-point numbers, where the real part is represented by the first number and the imaginary part is represented by the second number with a "j" suffix.

Python Numerics & Sequence

Sequential Data Types

Python provides the following built-in sequential data types:

- **Strings (str):** Strings are used to represent text data. They are created by enclosing a sequence of characters within single or double quotes, such as "hello", 'world', and "42".

- **Lists (list):** Lists are used to represent a collection of items. They are created by enclosing a sequence of values within square brackets and separating them with commas, such as [1, 2, 3], ['apple', 'banana', 'cherry'], and [1, 'hello', 3.14].

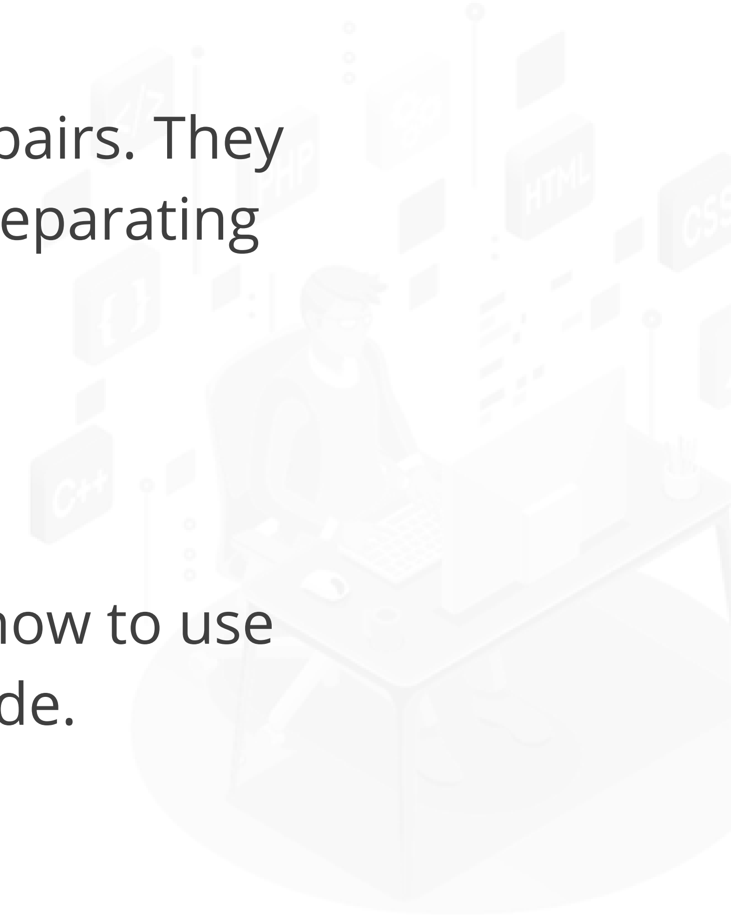
- **Tuples (tuple):** Tuples are similar to lists, but they are immutable, meaning that their values cannot be changed after they are created. They are created by enclosing a sequence of values within parentheses and separating them with commas, such as (1, 2, 3), ('apple', 'banana', 'cherry'), and (1, 'hello', 3.14).

Python Numerics & Sequence

- **Sets (set):** Sets are used to represent a collection of unique items. They are created by enclosing a sequence of values within curly braces and separating them with commas, such as {1, 2, 3}, {'apple', 'banana', 'cherry'}, and {1, 'hello', 3.14}.

- **Dictionaries (dict):** Dictionaries are used to represent a collection of key-value pairs. They are created by enclosing a sequence of key-value pairs within curly braces and separating them with commas, such as {'name': 'John', 'age': 30, 'city': 'New York'}.

These data types are fundamental to programming in Python, and understanding how to use them and their respective methods is essential for writing effective and efficient code.



TECHNOLOGY

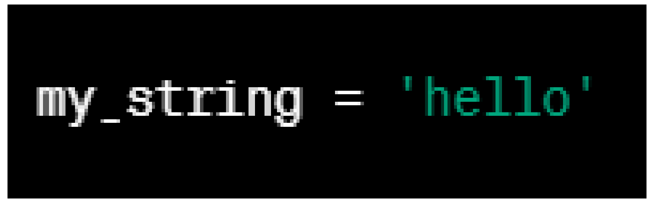
Python Sequence

Python Sequence

In Python, a sequence is an ordered collection of elements. The built-in sequence types in Python are strings, lists, tuples, and range objects.

Strings

A string is a sequence of characters. Strings can be created by enclosing a sequence of characters in single quotes, double quotes, or triple quotes. For example:

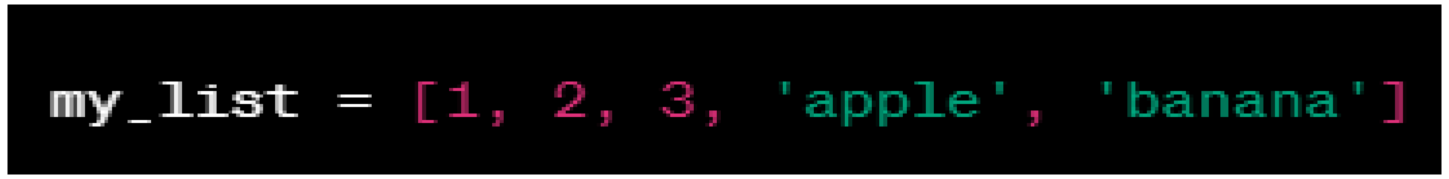


```python
my_string = 'hello'
```

Python Sequence

Lists

A list is a sequence of values, separated by commas and enclosed in square brackets. Lists can contain elements of different data types. For example:



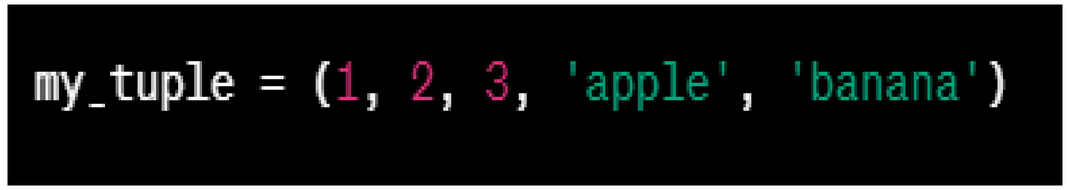
```
my_list = [1, 2, 3, 'apple', 'banana']
```

Python Sequence

Tuples

A tuple is similar to a list, but its elements are enclosed in parentheses instead of square brackets. Tuples can also contain elements of different data types.

The main difference between lists and tuples is that tuples are immutable, meaning that their values cannot be changed after they are created. For example:



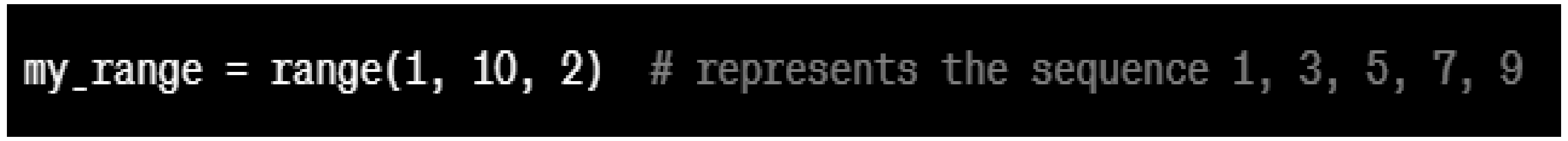
```python
my_tuple = (1, 2, 3, 'apple', 'banana')
```

Python Sequence

Range Objects

A range object represents an immutable sequence of numbers. It is often used for looping a specific number of times in a for loop.

The range function takes two or three arguments, the start value, the stop value, and an optional step value. For example:

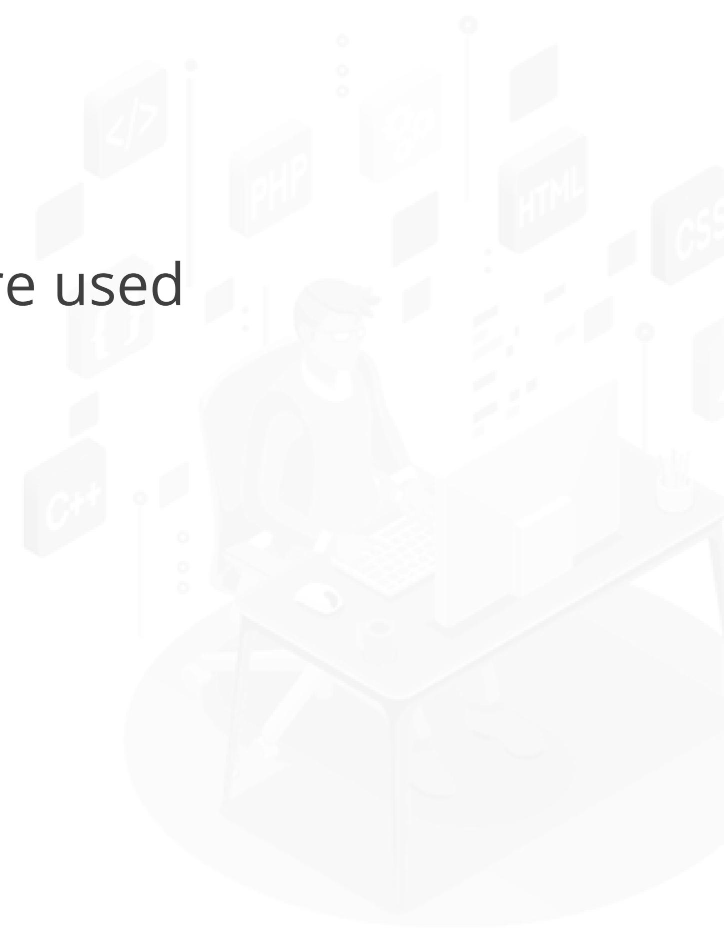


```
my_range = range(1, 10, 2)  # represents the sequence 1, 3, 5, 7, 9
```

Python Sequence

Sequences in Python have many built-in methods that allow you to manipulate, concatenate, and iterate over them.

Understanding sequences is an essential part of programming in Python, as they are used extensively in many different applications.



String in Python

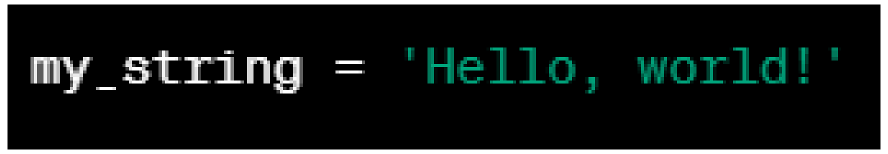
String in Python

In Python, a string is a sequence of characters enclosed in either single quotes ('...') or double quotes ("..."). Strings are a fundamental data type in Python, and they are used to represent textual data.

Here are some basic operations that can be performed on strings in Python:

String Creation

A string can be created by enclosing a sequence of characters in single quotes or double quotes. For example:

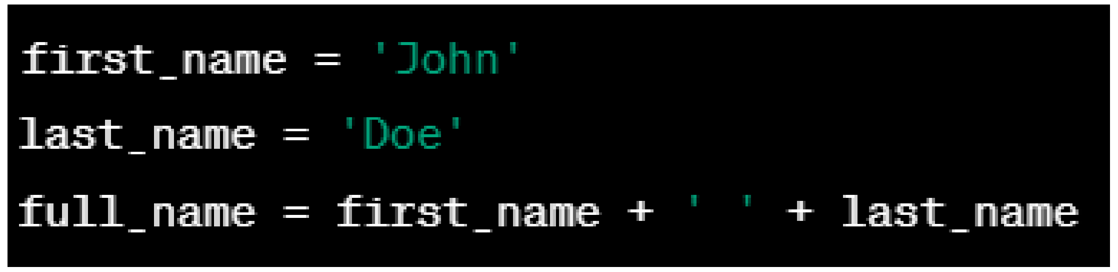


```
my_string = 'Hello, world!'
```

String in Python

String Concatenation

Two or more strings can be concatenated using the "+" operator. For example:



```
first_name = 'John'
last_name = 'Doe'
full_name = first_name + ' ' + last_name
```

String in Python

String Length

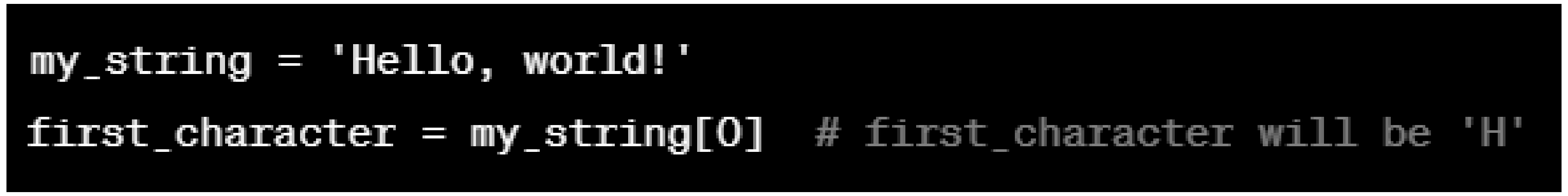
The **len()** function can be used to get the length of a string. For example:

```
my_string = 'Hello, world!'
length = len(my_string)  # length will be 13
```

String in Python

String Indexing

Individual characters in a string can be accessed using their index. In Python, the first character of a string has an index of 0. **For example:**

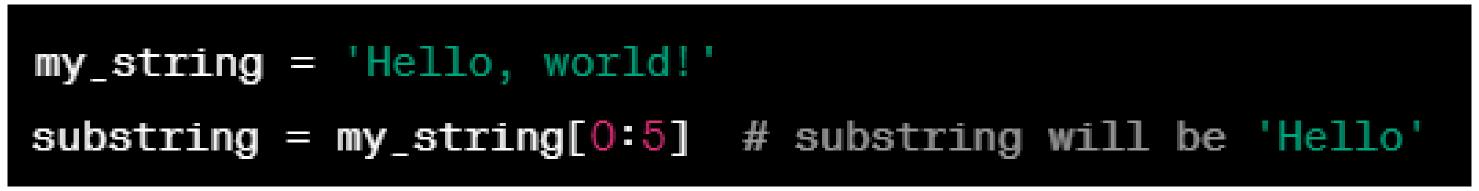


```
my_string = 'Hello, world!'
first_character = my_string[0]  # first_character will be 'H'
```

String in Python

String Slicing

A substring of a string can be extracted using slicing. Slicing is done using the colon (:) operator. **For example:**

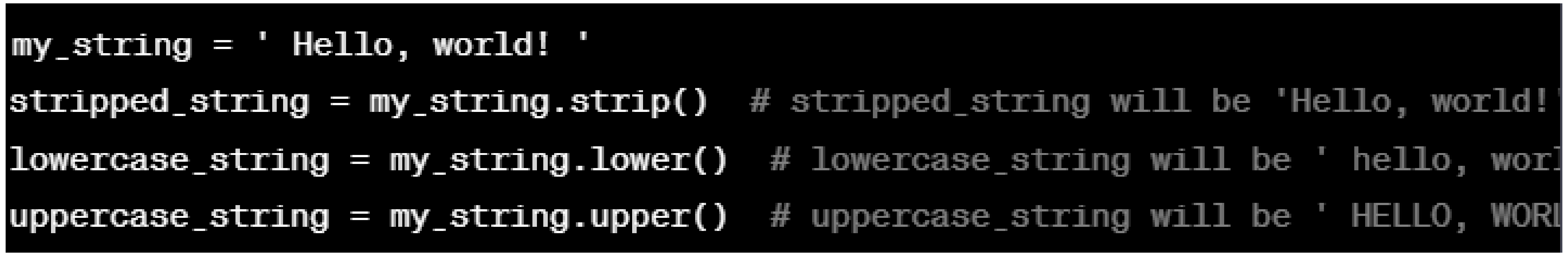


```
my_string = 'Hello, world!'
substring = my_string[0:5]  # substring will be 'Hello'
```

String in Python

String Methods

Python provides many built-in methods for working with strings. These include methods for converting the case of the string, stripping whitespace, replacing substrings, and more. For example:



```
my_string = ' Hello, world! '
stripped_string = my_string.strip()  # stripped_string will be 'Hello, world!'
lowercase_string = my_string.lower()  # lowercase_string will be ' hello, worl
uppercase_string = my_string.upper()  # uppercase_string will be ' HELLO, WORL
```

These are just a few of the many operations that can be performed on strings in Python.

Understanding how to work with strings is an essential part of programming in Python, as strings are used extensively in many different applications.

Key Takeaways

- In Python, a variable is a named reference to a value that can be stored in memory.
- To declare a variable in Python, you simply need to choose a name for the variable and use the equals sign "=" to assign a value to it.
- In Python, you can reference an object by using its variable name.
- In Python, variables can be classified as either mutable or immutable depending on whether the value of the variable can be changed or not