Variables

1. Variables:

A variable is a name that refers to a value in memory. In Python, you don't need to specify the data type of a variable when you declare it, as the interpreter infers the data type from the value assigned to it. Here's an example:

makefile

Copy code

Assigning a string value to a variable called name name = "John"

Assigning an integer value to a variable called age age = 30

Assigning a boolean value to a variable called is_student is_student = True

Print Function

Input Function

Data Types

```
4. Data types:
  In Python, there are several data types, including integers, floats, strings, booleans, and
  lists, among others. You can check the type of a variable using the 'type()' function.
  Here are some examples:
  python
                                                                       Copy code
  # Integers
  x = 5
  print(type(x)) # Output: <class 'int'>
  # Floats
  y = 3.14
  print(type(y)) # Output: <class 'float'>
  # Strings
  name = "John"
  print(type(name)) # Output: <class 'str'>
  # Booleans
  is_student = True
  print(type(is_student)) # Output: <class 'bool'>
  # Lists
 my_list = [1, 2, 3]
  print(type(my_list)) # Output: <class 'list'>
```

Type Conversion

```
5. Type conversion:
  Sometimes you may need to convert a variable from one data type to another. In Python,
  you can use built-in functions to do this. Here are some examples:
  makefile
                                                                       Copy code
  # Converting an integer to a float
 y = float(x)
 print(y) # Output: 5.0
 # Converting a string to an integer
  age = "30"
 age_int = int(age)
 print(age_int) # Output: 30
 # Converting a string to a float
 pi = "3.14"
 pi_float = float(pi)
  print(pi_float) # Output: 3.14
```

First Program

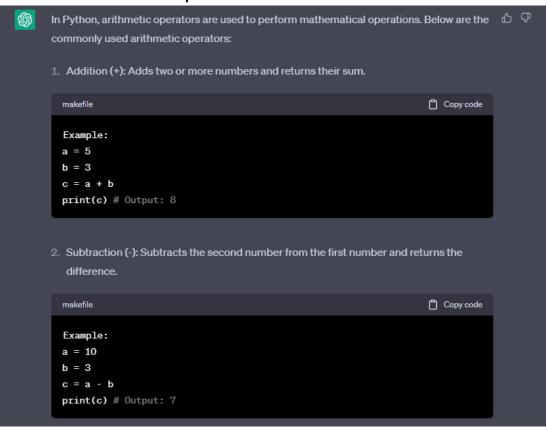
```
# Get the user's name and age
name = input("What is your name? ")
age_str = input("What is your age? ")

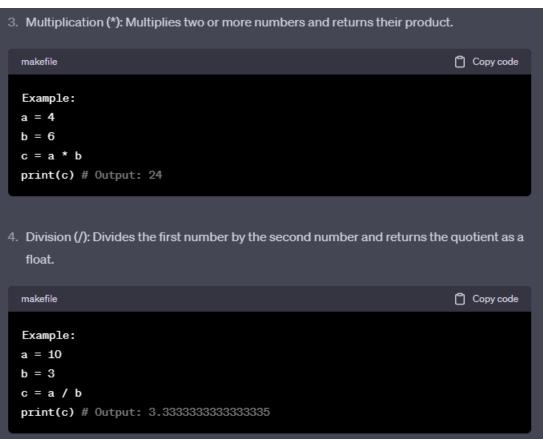
# Convert the age to an integer
age = int(age_str)

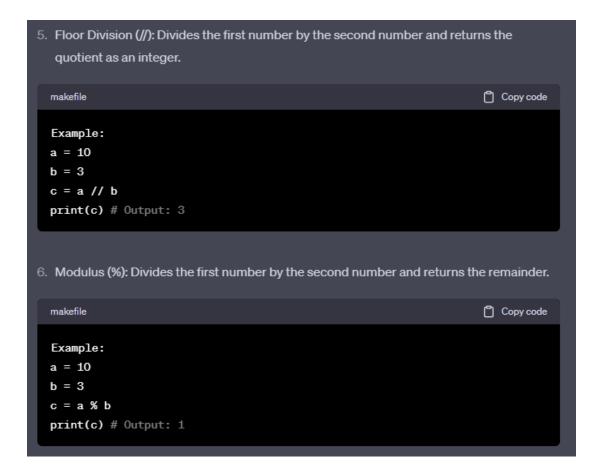
# Calculate the year the user was born
current_year = 2023
birth_year = current_year - age

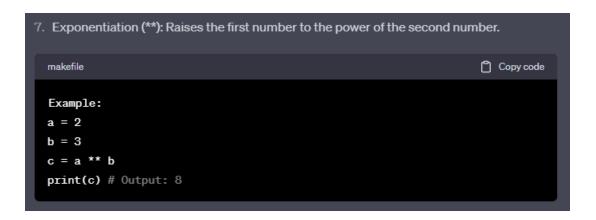
# Print a greeting with the user's name and birth year
print("Hello, " + name + "! You were born in " + str(birth_year) + ".")
```

Arithmetic Operators

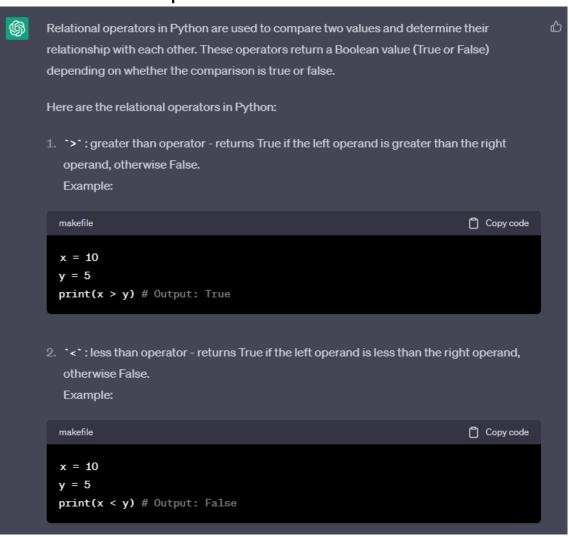


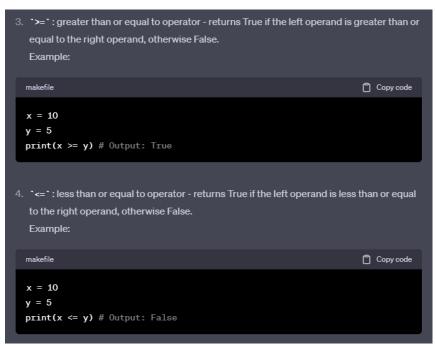






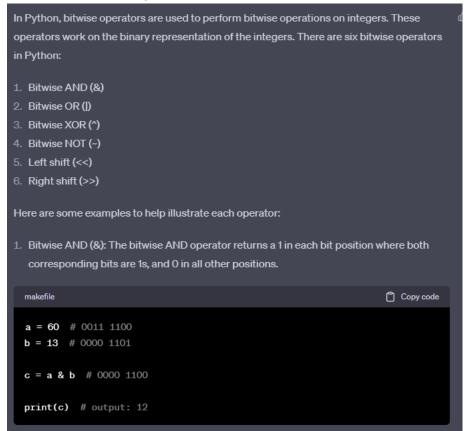
Relational Operators





5. `==`: equal to operator - returns True if the left operand is equal to the right operand,	
otherwise False.	
Example:	
makefile	Copy code
 x = 10 y = 5 print(x == y) # Output: False 6. `!=`: not equal to operator - returns True if the left operand is reoperand, otherwise False. Example: 	not equal to the right
makefile	Copy code
<pre>x = 10 y = 5 print(x != y) # Output: True</pre>	

Bitwise Operators



2. Bitwise OR (I): The bitwise OR operator returns a 1 in each bit position where at least one corresponding bit is a 1.

makefile

a = 60 # 0011 1100
b = 13 # 0000 1101

c = a | b # 0011 1101

print(c) # output: 61

3. Bitwise XOR (^): The bitwise XOR operator returns a 1 in each bit position where exactly one corresponding bit is a 1.

makefile

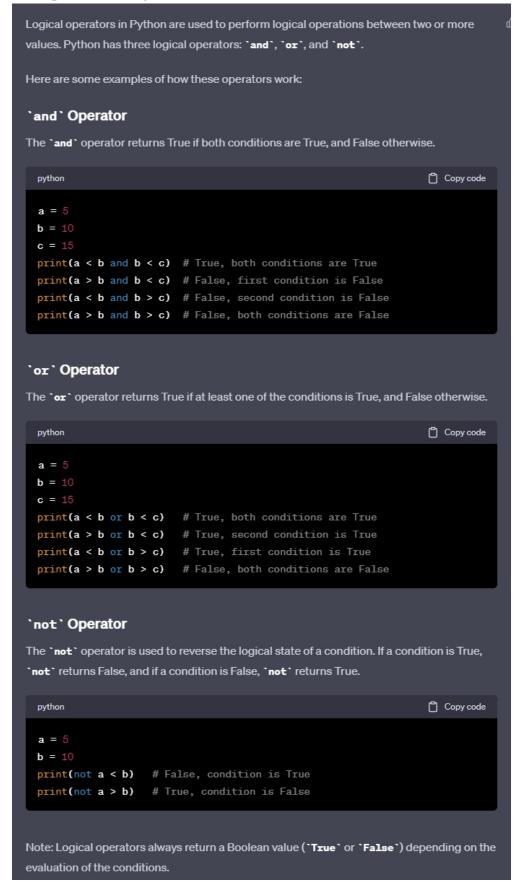
a = 60 # 0011 1100
b = 13 # 0000 1101

c = a ^ b # 0011 0001

print(c) # output: 49

4. Bitwise NOT (~): The bitwise NOT operator returns the complement of a number by changing all the bits from 0 to 1, and from 1 to 0. makefile Copy code a = 60 # 0011 1100 c = ~a # 1100 0011 print(c) # output: -61 Note that the bitwise NOT operator returns the two's complement of the input integer, which can be negative in Python. 5. Left shift (<<): The left shift operator shifts the bits of a number to the left by a specified number of positions, filling in the empty spaces with Os. makefile Copy code a = 60 # 0011 1100 c = a << 2 # 1111 0000 print(c) # output: 240

Logical Operators



Assignment Operators

In Python, assignment operators are used to assign a value to a variable. These operators combine the assignment and arithmetic operations in a single statement. Here are some common assignment operators in Python: 1. = (Assignment Operator): Assigns the value on the right-hand side to the variable on the left-hand side. python Copy code x = 52. += (Addition Assignment Operator): Adds the value on the right-hand side to the variable on the left-hand side and assigns the result to the left-hand side variable. Copy code python x += 2# same as x = x + 23. -= (Subtraction Assignment Operator): Subtracts the value on the right-hand side from the variable on the left-hand side and assigns the result to the left-hand side variable. Copy code python x -= 2# same as x = x - 24. *= (Multiplication Assignment Operator): Multiplies the value on the right-hand side with the variable on the left-hand side and assigns the result to the left-hand side variable. python Copy code x *= 2 # same as x = x * 25. /= (Division Assignment Operator): Divides the variable on the left-hand side by the value on the right-hand side and assigns the result to the left-hand side variable. python Copy code x /= 2# same as x = x / 2

6. %= (Modulus Assignment Operator): Computes the modulus of the variable on the lefthand side with the value on the right-hand side and assigns the result to the left-hand side variable. Copy code python x %= 2 # same as x = x % 27. #= (Floor Division Assignment Operator): Performs floor division on the variable on the left-hand side with the value on the right-hand side and assigns the result to the left-hand side variable. python Copy code # same as x = x // 2x //= 28. **= (Exponentiation Assignment Operator): Raises the variable on the left-hand side to the power of the value on the right-hand side and assigns the result to the left-hand side variable. python Copy code x **= 2 # same as x = x ** 2These assignment operators help to write concise and readable code in Python.

Compound Operators

In Python, compound operators are shorthand notations that combine an arithmetic or bitwise operation with an assignment operation. These operators allow you to perform an operation on a variable and store the result back into the same variable in one line of code.

Here are some examples of compound operators in Python:

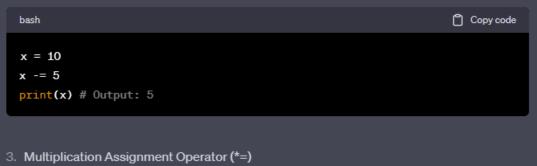
1. Addition Assignment Operator (+=)

The addition assignment operator adds the value on the right side of the operator to the value of the variable on the left side and assigns the result back to the same variable.



2. Subtraction Assignment Operator (-=)

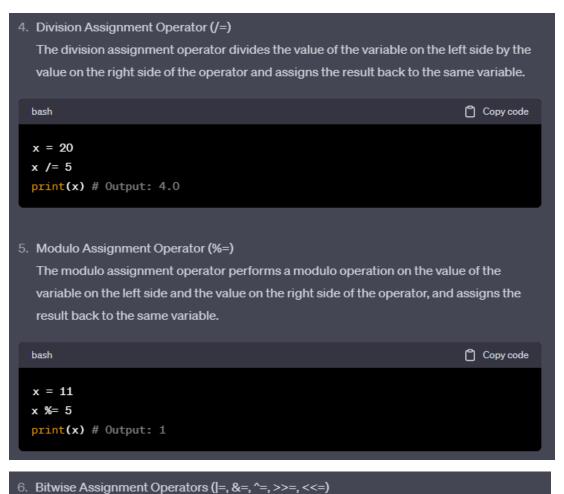
The subtraction assignment operator subtracts the value on the right side of the operator from the value of the variable on the left side and assigns the result back to the same variable.



The multiplication assignment operator multiplies the value on the right side of the operator with the value of the variable on the left side and assigns the result back to the

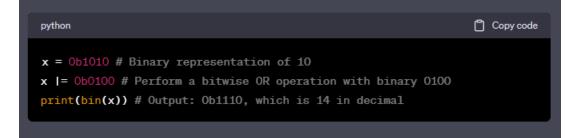
same variable.





These operators perform bitwise operations on the value of the variable on the left side and the value on the right side of the operator, and assigns the result back to the same variable.

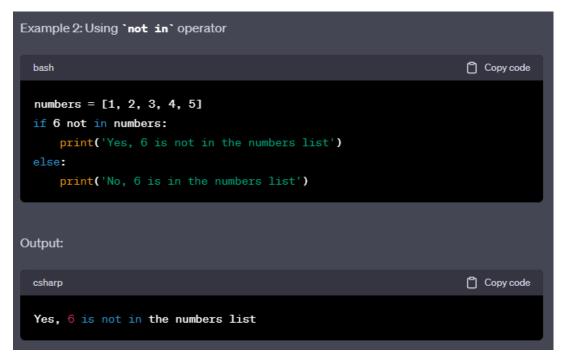
Example:

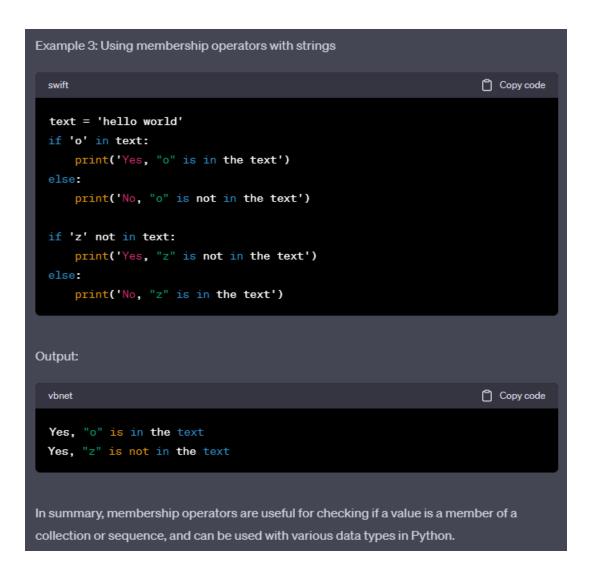


Note that the above examples are just a few of the many compound operators available in Python. They make code more concise and readable by reducing the amount of code needed to perform simple operations on variables.

Membership Operators

In Python, membership operators are used to test whether a value or variable is a member of a sequence or collection. There are two types of membership operators in Python: 1. 'in' operator: Returns 'True' if a value is found in the sequence or collection, otherwise 2. 'not in' operator: Returns 'True' if a value is not found in the sequence or collection, otherwise 'False'. Here are some examples of how to use membership operators in Python: Example 1: Using `in` operator bash Copy code fruits = ['apple', 'banana', 'orange'] if 'banana' in fruits: print('Yes, banana is in the fruits list') print('No, banana is not in the fruits list') Output: Copy code csharp Yes, banana is in the fruits list





Identity Operators

In Python, identity operators are used to compare the memory locations of two objects. These operators are used to check whether the two objects have the same identity or not. There are two identity operators in Python: is operator: It returns True if two variables refer to the same object, otherwise False. 2. 'is not' operator: It returns True if two variables do not refer to the same object, otherwise False. Here are some examples: Example 1: Copy code python x = [1, 2, 3]y = [1, 2, 3]z = xprint(x is y) # False, x and y refer to two different lists print(x is z) # True, x and z refer to the same list print(x is not y) # True, x and y do not refer to the same list Example 2: Copy code CSS a = 5b = 5print(a is b) # True, a and b refer to the same integer object in memo print(a is not b) # False, a and b refer to the same integer object in mem Example 3: Copy code vbnet s1 = "hello" s2 = "hello" s3 = "world" print(s1 is s2) # True, s1 and s2 refer to the same string object in mem print(s1 is not s3) # True, s1 and s3 do not refer to the same string object In general, identity operators should be used only when we need to check whether two objects refer to the same object in memory. For checking the equality of two objects, we should use the equality operator ('==').