|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **What Are Assignment Operators?**  Assignment operators assign the value on the right side of the operator to the left side of the operator.  In Python, some assignment operators are special forms that are made up of an arithmetic operator as well as an assignment operator. These can be used as shortcuts during programming.  For example, the += operator is a shortcut for a + operation and = operator.  y = y + 3  is the same as this shortcut  y += 3   The assignment operators are:   |  |  |  | | --- | --- | --- | | **Operator** | **Usage** | **Description** | | = | x = y | Assigns value of y to x. | | += | x += y | Adds y to x and assigns the result to x. Same as x = x + y. | | -= | x -= y | Subtracts y from x and assigns the result to x. Same as x = x - y. | | \*= | x \*= y | Multiplies x and y and assigns the result to x. Same as x = x \* y. | | /= | x /= y | Divides x and y and assigns the result to x. Same as x = x / y. | | %= | x %= y | Assigns the modulus of x and y to x. Same as x = x % y. | |
| **What Are Comparison Operators?**  Comparison operators compare the values on the left and right side of the operator and return a boolean value of True or False based on the comparison.  The comparison operators in Python are:   |  |  |  | | --- | --- | --- | | **Operator** | **Usage** | **Description** | | == | x == y | Equal Returns True if *x* and *y* are equal. Otherwise it returns False. | | != | x != y | Not Equal Returns True if *x* and *y* are not equal. Otherwise it returns False. | | > | x > y | Greater than Returns True if *x* is greater than *y*. Otherwise it returns False. | | >= | x >= y | Greater than or Equal to Returns True if x is greater than or equal to y. Otherwise it returns False. | | < | x < y | Less than Returns True if x is less than y. Otherwise it returns False. | | <= | x <= y | Less than or Equal to Returns True if x is less than or equal to y. Otherwise it returns False. | |
| **What Are Logical Operators?**  Logical operators take boolean values True or False and return a boolean value.  **Logical Operators**  In Python, there are logical 'and,' 'or,' and 'not' operators.   |  |  |  | | --- | --- | --- | | **Name** | **Operator** | **Description** | | Logical And | and | Logical **"and"** operator  expr1 and expr2 Returns *True* if both *expr1* and *expr2* are true. Otherwise, it returns *False*. | | Logical Or | or | Logical **"or"** operator  expr1 or expr2 Returns *True* if either *expr1* or *expr2* is true. Returns *False* if both *expr1* and *expr2* are false. | | Logical Not | not | Logical **"not"** operator  not expr Returns *False* if *expr* is true, returns *True* if *expr* is false. |   For and, if one of the expressions is false, it will immediately return false without evaluating later expressions.  For or, if one of the expressions is true, it will immediately return true without evaluating later expressions. |
| What Is an "Else"? Let's look at other conditional statements we can use in Python.  The **"else"** statement executes a segment of code when certain conditions are false. Else Statement In Python, the "**else**" statement has the following syntax:    if(condition):  #commands  else:  #commands  **Condition**  The condition decides which segment of code is executed.  If the **condition** is true, the code inside the "**if**" segment of the "**if**" statement will be executed.  If the **condition** is false, the code inside the **"else"** segment will be executed.  For example:   * If the condition is i < 10,  the code inside the "**if**" statement will be executed if the value of i is less than 10. * If the value of i is greater than or equal to 10, the code inside the "**else**" statement will be executed.  Example 1 To output the odd and even numbers from 1 to 25, the example below uses an "**if else**" statement. Click on the **Play** button to try it out.    for i in range(1,25):  if ((i % 2) == 1):  print(str(i) + " - Odd")  else:  print(str(i) + " - Even")  ​ Elif Statement In Python, you can use the **"elif"** statement to specify more conditions if there are more conditional statements that need to be executed.  The "**elif**" statement has the following syntax:   * "**if**" * "**elif**" * "**else**"     if (condition1):  #commands1  elif (condition2):  #commands2  else:  #commands3  **Condition1**  The code inside the "**if**" segment is executed when condition1 is true.  **Condition2**  The code inside the "**elif**" segment is executed when condition1 is false and condition2 is true.  If both condition1 and condition2 are false, then the code inside the "**else**" segment is executed. Example 2 Now let's try to use an "**if-elif-else**" statement to group the numbers from 1 through 25. Click on the **Play** button to try it out.    for i in range(1,25):  if (i < 10):  print(str(i) + " - Between 1 and 10, exclusive of 10.")  elif (i < 20):  print(str(i) + " - Between 10 and 20, exclusive of 20.")  else:  print(str(i) + " - Between 20 and 25, exclusive of 25.") |
| What Are Nested Ifs? You can add conditionals within conditionals. These are called nested conditional statements.  Each conditional statement will execute when the condition is true. Nested "if"s are hard to read and debug if they are not properly indented. Example 1 In the following example, we will use nested "if"s to output a letter grade. Change the value of the variable *grade* below and press **Play** to see the letter grade output below.    # grade  grade = 84  ​  # letter grade.  # "A" - 90 and above  # "B" - 80 to 89  # "C" - 70 to 79  # "D" - 60 to 69  #"FAIL" - less than 59  #  ​  # each letter grade has a nested "if" statement  print(grade)  if (grade < 90):  ​  # not A  if (grade < 80):  ​  # not B  if (grade < 70):  ​  # not C  if (grade < 60):  ​  # not D  print("FAIL")  else:  print("D")  ​  else:  print("C")  ​  else:  print("B")  ​  else:  print("A") Example 2 An "elif" statement can be used instead of the nested ifs. Let's change the conditions a little bit and use a nested **elif** for the same output.    grade = 70  ​  # letter grade.  # "A" - 90 and above  # "B" - 80 to 89  # "C" - 70 to 79  # "D" - 60 to 69  #"FAIL" - less than 59  #  ​  # using else-if for each each letter grade  print(grade)  if (grade >= 90):  print("A")  ​  elif (grade >= 80):  print("B")  ​  elif (grade >= 70):  print("C")    elif (grade >= 60):  print("D")    else:  print("FAIL") |
| Review Let's review what you've learned so far in this lesson. Use this as a guide for the quiz coming up next. Conditionals In Python, you can use conditional statements to execute different segments of code based on a **condition**. If The "if" statement executes a segment of code if its condition evaluates to true.  In Python, the "if" statement has the following syntax:  if (condition):  commands()  The code inside the "if" segment is executed when condition is true.  Example: i < 10  When the value of *i is less than 10*, the code inside the "if" section is executed. Else The "else" statement executes a segment of code when certain conditions are false.  if (condition):  commands()  else:  commands()   * The code inside the **"if"** segment is executed when condition is true. * The code inside the **"else"** segment is executed when condition is false.  Elif In Python, you can use the "elif" statement to specify more conditions if there are more conditional statements that need to be executed.  if (condition1):  commands1()  elif (condition2):  commands2()  else:  commands3()   * The code inside the **"if**" segment is executed when condition1 is true. * The code inside the **"elif**" segment is executed when condition1 is false and condition2is true. * If both condition1 and condition2 are false, then the code inside the **"else"** segment is executed.  Nested Ifs You can add conditionals within conditionals. This is called nested conditionals statements.  Each conditional statement will execute when the condition is true.  Example:  # outer if  if (condition1):     # inner if     if (conditional2):  commands() Assignment Operators Assignment operators assign the value on the right side of the operator to the left side of the operator.  Some assignment operators are special and are made up of an arithmetic operator as well as an assignment operator. These are used as shortcuts during programming.  For example, the += operator is a shortcut for a + operation and = operator.  y = y + 3  is the same as this shortcut  y += 3   The assignment operators are:   |  |  |  | | --- | --- | --- | | **Operator** | **Usage** | **Description** | | = | x = y | Assigns value of y to x. | | += | x += y | Adds y to x and assigns the result to x. Same as x = x + y. | | -= | x -= y | Subtracts y from x and assigns the result to x. Same as x = x - y. | | \*= | x \*= y | Multiplies x and y and assigns the result to x. Same as x = x \* y. | | /= | x /= y | Divides x and y and assigns the result to x. Same as x = x / y. | | %= | x %= y | Assigns the modulus of x and y to x. Same as x = x % y. |  Comparison Operators Comparison operators compare the left and right side of the operator and return a boolean value of True or False based on the comparison.  The comparison operators in Python are:   |  |  |  | | --- | --- | --- | | **Operator** | **Usage** | **Description** | | == | x == y | Equal Returns True if *x* and *y* are equal. Otherwise it returns False. | | != | x != y | Not Equal Returns True if *x* and *y* are not equal. Otherwise it returns False. | | > | x > y | Greater than Returns True if *x* is greater than *y*. Otherwise it returns False. | | >= | x >= y | Greater than or Equal to Returns True if x is greater than or equal to y. Otherwise it returns False. | | < | x < y | Less than Returns True if x is less than y. Otherwise it returns False. | | <= | x <= y | Less than or Equal to Returns True if x is less than or equal to y. Otherwise it returns False. |  Logical Operators Logical operators take boolean values True or False as expressions and return a boolean value.  In Python, there are logical 'and,' 'or,' and 'not' operators.   |  |  |  | | --- | --- | --- | | **Name** | **Operator** | **Description** | | Logical And | and | Logical **"and"** operator  expr1 and expr2 Returns true if both *expr1* and *expr2* are true. Otherwise, it returns false. | | Logical Or | or | Logical **"or"** operator  expr1 or expr2 Returns true if either *expr1* or *expr2* is true. Returns false if both *expr1* and *expr2* are false. | | Logical Not | not | Logical **"not"** operator  not expr Returns false if *expr* is true, returns true if *expr* is false. | |
|  |
|  |