|  |
| --- |
| Worksheet 5 |

|  |  |
| --- | --- |
| Expression | Meaning |
| x == y | True if x = y (mathematical equality, not assignment; otherwise, false |
| x < y | True if x < y; otherwise, false |
| x <= y | True if x ≤ y; otherwise, false |
| x > y | True if x > y; otherwise, false |
| x >= y | True if x ≥ y; otherwise, false |
| x != y | True if x ≠ y; otherwise, false |

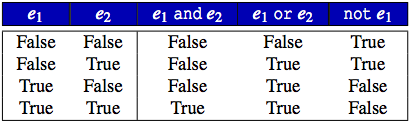
Examples

10 < 20 True

10 >= 20 False

x < 100 True if x Is less than 100; otherwise, False

x != y True unless x and y are equal



Operator **not** has higher precedence than both **and** and **or**.

**and** has higher precedence than **or**.

**and**, **or** are left associative; **not** is right associative.

**and**, **or** have lower precedence than any other binary operator except assignment.

1. Fill in the blank

x = 10

y = 20

b = (x == 10) # assigns True to b

b = (x != 10) # assigns False to b

b = (x == 10 and y == 20) # assigns \_\_\_\_\_\_\_\_ to b

b = (x != 100 and y == 30) # assigns \_\_\_\_\_\_\_\_ to b

b = (x == 10 and y != 20) # assigns \_\_\_\_\_\_\_\_ to b

b = (x != 10 or y != 30) # assigns \_\_\_\_\_\_\_\_ to b

b = (x == 10 or y == 20) # assigns \_\_\_\_\_\_\_\_ to b

b = (x != 100 and y == 30) # assigns \_\_\_\_\_\_\_\_ to b

b = (x == 10 or y != 20) # assigns \_\_\_\_\_\_\_\_ to b

b = (x != 10 and y != 30) # assigns \_\_\_\_\_\_\_\_ to b

1. Given the following definitions

x = 3, y = 2

b1, b2, b3, b4 = False, True, x == 3, y < 3 # This statement assigns Boolean value in

# accordance with the order of the appearance, e.g., # b1 is assigned value of **False**. x == 3 is **True**,

# hence b3 is assigned a Boolean value of **True**.

Evaluation the following Boolean expressions (True or False)

b1 or b2 and b3 ­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_

b1 and b2 and b3 ­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_

b1 or b2 or b3 ­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_

not b1 and b2 and b3 ­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_

not b1 or b2 or b3 ­­­ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

not (b1 and b2 and b3) ­­­ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

not (b1 or b2 or b3) ­­­ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

not b1 and not b2 and not b3 ­­­ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

not b1 or not b2 or not b3 ­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_

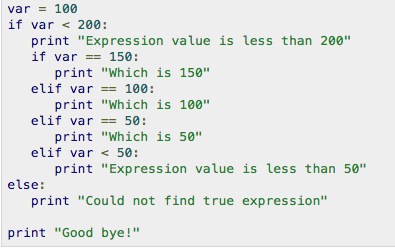
not (not b1 and not b2 and not b3) ­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_

not (not b1 or not b2 or not b3) ­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Nested If-else**

The if...else statement executes two different codes depending upon whether the Boolean expression (if-else condition) is true or false. Sometimes, a choice has to be made from more than 2 possibilities. The nested if...else statement allows you to check for multiple test expressions and execute different codes for more than two conditions.

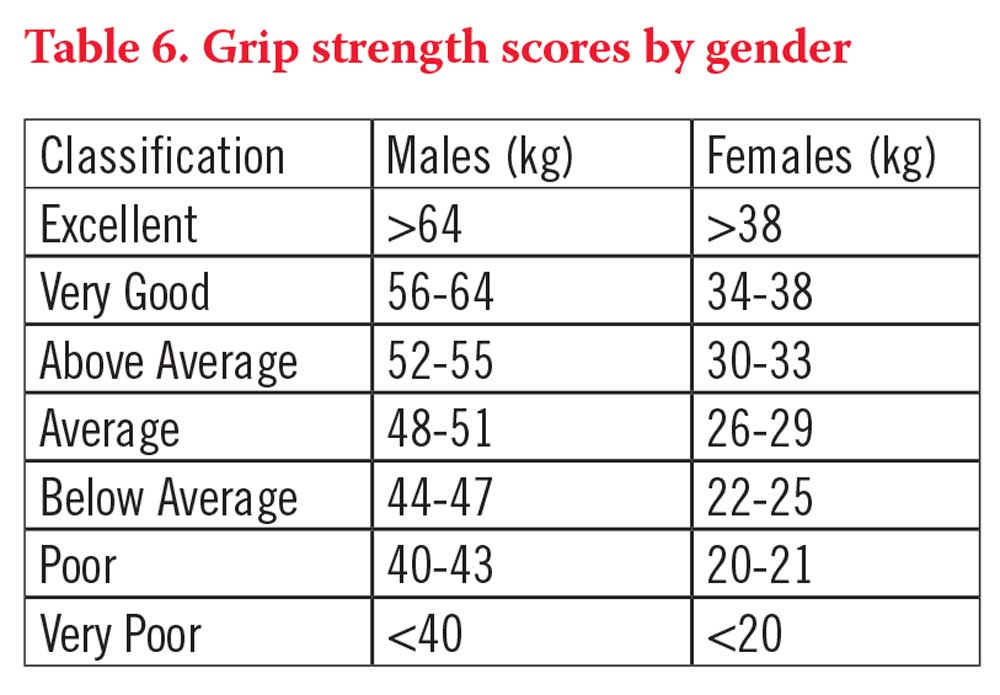
As an example, in Python (or any other programming language), there may be a situation **when** you want to check for another condition after a condition resolves to true. In such a situation, you can use the **nested if** construct. In a **nested if** construct, you can have an **if**...elif...**else** construct inside another **if**...elif...**else** construct.



1. In the above example, what would be printed on the screen? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. Write a Python code to take 2 numbers for variable names, *var1* and *var2*. If the value of *var1* is equal to that of *var2*, print “(value of *var1*) is equal to (value of *var2*)”. For example if you enter 10 and 10 for *var1* and *var2*, the code will print “8 is equal to 8”.

The code will print “(value of var1) is greater than (value of var2)” if var1 > var2, and vice versa. **Solve this problem using nested if-else. Hint: you should first check if the two values are not the same, then you can if another if-else to check if var1 > var2 or var2 > var 1 under the first if statement.**

1. The following table illustrates the grip strengths (on average) based on Gender. In general, the dominant hand will often score 10 percent higher than the non-dominant hand. Write a Python code to take a gender (capital M or F) and grip strength in kg., and then print out the strength level. For example, if you enter M and 50 as inputs, the code should print “Your grip strength is average.”



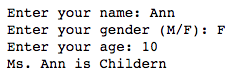
Examples:

 and 

1. Write a Python program to take name, gender, age as inputs and print out the growth stage of that person.

|  |  |
| --- | --- |
| Age | Growth Stage |
| < 11 | Children |
| 11 to 17 | Adolescence |
| 18 to 40 | Young adult |
| 41 to 65 | Adult |
| > 65 | Elderly |

Examples of inputs and outputs

 and 

1. Write a Python code that takes two integers, *a* and *b*. Then print all integer between these two numbers (including *a* and *b*). Note that any integer ended with 3 and 7 will be replaced by \*. For example, if you enter 2 and 12 for *a* and *b*, the code will print



If you enter 2 and 28, the code will print

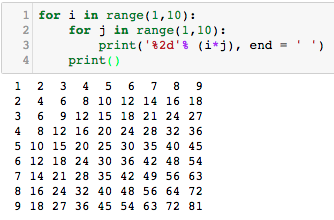


1. Write a Python code to print all integer numbers between 0 and 30 with the following conditions:
   1. All printed numbers must be divisible by 3 (otherwise, \* symbol is printed), and
   2. the number between 10 – 15 and 20 – 25 will be replaced by an underscore symbol.

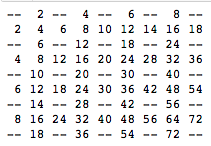
Expected output is given below.



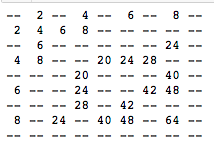
1. Write a Python code to print integers between 100 and 200 with the following conditions:
   1. The right-most digits must be even number,
   2. the integers that is divisible by 5 will not be printed,
   3. 0 is considered as an even number, and
   4. all the printed integers between 130 – 140 and 160 – 170 will be replaced by @ sign.
2. Based on the given Python code, modify the code to print the number where all digits are even number. Otherwise, -- will be printed as shown in the expected output below.



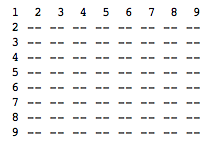
**1st expected output** (only even numbers):



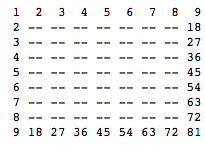
**2nd expected output** (only even numbers and both digits must be even numbers):



**3rd expected output** (only first row and first column printed)



**4th expected output** (only first and last rows, and first and last columns printed)



**5th expected output** (based on 4th output but integers appeared in diagonal lines printed)

