Basic python practice

1. Identify methods from the integer class that can recapitulate all the listed mathematical operators above. E.g., x + 2 can be recapitulated as x.__add__(2). Do the same for '-', '*', '/', '%', '**', and '//'
2. Predict the results of the following expressions. Verify you are correct using the Python interpreter.
    a. 2 + 3 * 5
    b. (2 + 3) * 5
    c. 15 // 4
    d. 15 / 4
    e. 15 % 3
    f. 8 ** 2
    g. 4 ** .5
    h. (6 + 25) % 5
    i. (5 // 2) ** 3
    j. (2.3 % 2) * 2
3. Type var = 2 in your interpreter. Predict the values of var of the following expressions. Verify using the interpreter.
    a. var += 3
    b. var -= 2
    c. var *= 4
    d. var /= 3
    e. var **= 2
    f. var %=13
    g. var //= 2
4. In lectures we discussed how operators rely on magic methods to run. However, there is not a 1:1 correspondence between how an operator works and directly calling a magic method. To see this, let's try a few examples. Example 1: Set x = 2
    a. Try to create a one line expression using magic methods that is the same as x * 4 + 3
    b. Try to create a one line expression using magic methods that is the same as x + 4 * 3. Why is there an issue here?
5. Example 2: Set x = 2 and y = 3.0
    a. Add x + y
    b. Now try to use the magic method to run this. What is the issue? The + operator actually can call multiple magic methods. If there is a NotImplemented exception thrown by the magic method, it next tries a second magic method of the 2nd variable (y) in this case. Try to figure out which magic method is called next (hint it is close to the 1st magic method)
6. Identify the magic method/attribute used by the type() and dir() functions
7. z = (3 + 4j). What attributes hold the real data and imaginary data for z?
8. Set w = True, x = 2, y = 3.0, z = (3 + 4j).
    a. What type are the four variables?

   b. What type is created when you do:
      i. w + w, x + x, y + y, or z + z?
      ii. w + x, w + y, w + z?
      iii. x + y, x + z?
      iv. y + z?

9. Set x = 3.0. What objects are created by the __int__ and __str__ magic methods?
10. isinstance is a built in function that takes in two arguments, an object and a class name and returns True if the object belongs to the class and False if it doesn't. Create an integer and float object and verify that you can use isinstance to test whether the object classes are integers or floats.