4.
Нашел ответ прямо в документации:
Результатом выполнения оператора присваивания является само присвоенное значение.

5.
1 == 1.0 после преобразования **1** в **1.0** (как к более широкому типу) производится сравнение и результат **true**, но доверять такому сравнению нельзя;
1 === 1.0 операнды разных типов, проверка это учитывает, поэтому **false**;
'02' == 2 строка **'02'** преобразуется в число **2** и далее сравнивается с числом **2**, результат будет **true**;
'02' === 2 операнды разных типов, проверка это учитывает, поэтому **false**;
'02' == '2' строки преобразуются в числа и затем сравниваются, результат **true**.

6.
$x = true xor true;
Это выражение будет вычисляться на основании приоритетов операторов. Так как нет скобок, которые могут явно указать порядок выполнения операций.
У оператора присваивания приоритет выше, чем у **xor**, поэтому он будет вычислен в первую очередь. Нашел это вот здесь.

Можно сказать, что такая запись будет эквивалентна исходной: **($x = true) xor true;**
В переменную **$x** будет записано **true**. Вторая часть выражения тоже будет вычислена, результатом будет **false**, но это нигде не сохраняется.
Поэтому **var_dump($x);** вернет **true**.

Очень хитрое задание, сначала кажется, что выражение **(true xor true)** вычислится первым и результату будет дано имя **$x**. Я подставлял в **var_dump** разные вариации выражения из задания и именно это меня натолкнуло на мысль о приоритетах. Полез в мануал и оказалось, что есть операции **and**, **xor**, **or** у которых приоритет ниже чем у оператора присваивания.