✓ **Congratulations! You passed!**

**TO PASS** 70% or higher

Keep Learning

**GRADE** 100%

# Logic and Conditionals

**LATEST SUBMISSION GRADE**

100%

Try again

**1. Which of the following are Boolean values in Python?** 1 / 1 point

- [ ] `Grade`
- [ ] `"True"`

100%

View Feedback

We keep your highest score

- [x] `False`

✓ **Correct**

Correct.

- [x] `True`

✓ **Correct**

Correct.

**2. Consider the Boolean expression `not (p or not q)`. Give the four following values in order, separated only by spaces:** 1 / 1 point

**the value of the expression when `p` is `True`, and `q` is `True`,**

**the value of the expression when `p` is `True`, and `q` is `False`,**

**the value of the expression when `p` is `False`, and `q` is `True`,**

**the value of the expression when `p` is `False`, and `q` is `False`,**

**Remember, each of the four results you provide should be `True` or `False` with the proper capitalization.**

False False True False

✓ **Correct**

**3. Given the following initialization:** 1 / 1 point

```
1  bool1 = True
2  bool2 = False
```

**Which of the expressions below evaluate to `True`?**

- [x] `not False`

✓ **Correct**

- [ ] `bool2 == True`
- [x] `not (bool1 == bool2)`

✓ **Correct**

- [x] `bool1 != False`

✓ **Correct**

**4. Two expressions are *logically equivalent* if they have the same value for all possible values of the variables that comprise the expression.** 1 / 1 point

**Given two numbers `num1` and `num2`, which one of the expressions below is logically equivalent to the following arithmetic comparison:**

```
num1 >= num2
```

- ○
```
not (num1 <= num2)
```
- ◉
```
(num1 > num2) or (num1 == num2)
```
- ○
```
(num1 > num2) and (num1 != num2)
```
- ○
```
num2 < num1
```

✓ **Correct**

Correct.

**5. An `if` statement can have at most how many `elif` parts?** 1 / 1 point

- ○ 0
- ◉ Unlimited, i.e., 0 or more
- ○ 1

✓ **Correct**

Correct.

**6. In Python, conditional statements may be nested. Consider the following function that takes two Boolean values as input and returns a Boolean value.** 1 / 1 point

```
def nand(bool1, bool2):
    """
    Take two Boolean values bool1 and bool2
    and return the specified Boolean values
    """

    if bool1:
        if bool2:
            return False
        else:
            return True
    else:
        return True
```

**Which Boolean expression below is logically equivalent to the function call `nand(bool1, bool2)` where `bool1` and `bool2` are Boolean variables?**

- ○ `(bool1 and bool2)`
- ○ `(bool1 or bool2)`
- ○ `not (bool1 or bool2)`
- ◉ `not (bool1 and bool2)`

✓ **Correct**

Correct. The function name `nand` should have been a give away since it is short for "not and".

**7. The Collatz conjecture is an example of a simple computational process whose behavior is so unpredictable that the world's best mathematicians still don't understand it.** 1 / 1 point

**Consider the simple function $f(n)$ (as defined in the Wikipedia page above) that takes an integer $n$ and divides it by two if $n$ is even and multiplies $n$ by 3 and then adds one to the result if $n$ is odd. The conjecture involves studying the value of expressions of the form $f(f(f(\ldots f(f(n)))))$ as the number of calls to the function $f$ increases. The conjecture is that, for any non-negative integer $n$, repeated application of $f$ to $n$ yields a sequence of integers that always includes 1.**

**Your task for this question is to implement the Collatz function $f$ in Python. The key to your implementation is to build a test that determines whether $n$ is even or odd by checking whether the remainder when $n$ is divided by 2 is either zero or one. Hint: You can compute this remainder in Python using the remainder operator `%` via the expression `n % 2`. Note you will also need to use integer division `//` when computing $f$.**

**Once you have implemented $f$, test the your implementation on the expression $f(f(f(f(f(f(f(674)))))))$. This expression should evaluate to 190. Finally, compute the value of the expression $f(f(f(f(f(f(f(f(f(f(f(f(1071)))))))))))$ and enter the result below as an integer. Remember to use copy and paste when moving the expressions above into your Python environment. Never try to retype expressions by hand.**

3053

✓ **Correct**