Python Operators

Python Operators are special symbols in Python that carry out arithmetic or logical computation.

Comparison (equality) operators are used to compare values and return either `True` or `False` according to the condition.

Comparison Operators

These are the comparison operators in Python:

- `>` : Greater than Operator - True if left operand is greater than the right eg. `x > y`
- `<` : Less than - True if left operand is less than the right eg. `x < y`
- `==` : Equal to - True if both operands are equal eg. `x == y` (We've used this a lot so far!)
- `!=` : Not equal to - True if operands are not equal eg. `x != y`
- `>=` : Greater than or equal to - True if left operand is greater than or equal to the right

`5 > 3 and 5 == 3 + 2` evaluates to `True` but `5 < 3 and 5 == 5` evaluates to `False`.

The `or` operator

Here is a Truth Table for the `or` operator:

The `or` operator evaluates to `True` as long as one of the conditions in the expression is `True`. If both of those conditions are `False`, the entire expression evaluates to `False`. For example:
`5 > 3 or 5 == 3 + 2` evaluates to `True` and `5 < 3 or 5 == 5` also evaluates to `True`, but `5 < 3 or 5 == 6` evaluates to `False`

The `not` operator

Python's not operator allows you to invert the truth value of Boolean expressions and objects.

Here is a Truth Table for the `not` operator:

```
True
>>> 5 > 3
True
>>> 5 < 3
False
>>> 5 != 3
True

>>> [5, 3] == [5, 3] # compare if two lists are the same
True

>>> "hi" == "hello" # compare if two strings are the same
False
```

All these expressions return `True` or `False`, which means they're Boolean expressions. You can combine them using the `and` keyword to create compound expressions that test two—or more–subexpressions at a time:

```
>>> (5 > 3) and (5 == 3 + 2)
True

>>> (5 < 3) and (5 == 5) and ((5 > 3) or (3 > 5))
False
```

- The cards are ranked from Ace to 2; so Ace beats King, which beats Queen, and so on. Jack is assigned a numerical value of 11, Queen is 12, King is 13, and Ace is 14.
- Each suit is given a numerical value based on the numerical equivalent of the letter it begins with. eg. In the Alphabet, A is the first letter and so is 1, Z is the last letter and is 26.
- The number on each card is added to the numerical value of its suit to determine how many points it's worth. eg. 5 of Hearts will be worth 13 points.
- Each player will be dealt a deck of 5 cards and will take turns playing one card at a time, until the winner is determined at the end of the game. (Hint: each game will consist of 5 rounds of playing.)
- The order of play will be reversed if the second player's card beats the first player's card.
- The person who wins the most rounds of a game, wins the game.
- There will be three games. The person who wins the most games out of three is the overall winner of the card game

Tasks

1. Create a new repo and name it: `<your_name>-PyBools`. For example mine will be called: `fiifi-PyBools`
2. Place `card_game.py` in your repo and address all the comments in the file to complete the program.
3. Ensure the code runs with no errors.
4. Do not use any conditionals at all in the code. We haven't covered that yet. We're just using Boolean variables and expressions in this assignment.
5. Good luck! :)