

# **Compound Boolean Expressions**

Riya Jacob K

Assistant Professor on Contract  
Dept of Computer Applications

# Boolean logic and logical operators

There are three **logical operators** that let us combine Boolean expressions. They have **lower** precedence than the relational operators (<, >, ...)

- `not A`: True if A is False, False if A is True
  - A is any Boolean expression:  
`if not is_finished:`  
`do_more_work()`
- `A and B`: True if **both** A and B are True  
`in_range = size >= 0 and size <= 100`
- `A or B`: True if **either** A or B is True or Both!  
`if snow_inches > 6 or temperature < 0:`  
`print("Class is cancelled")`

# Complex Boolean expressions

- `not` has the highest precedence (but still lower than relational)
- `and` has the next highest
- `or` has the lowest of the three
- So `not A or B and C or D` means  
`((not A) or (B and C)) or D`
- People often forget the order of **and** and **or** operators
  - It's not a bad idea to always use parentheses when they are both in an expression  
`not A or (B and C) or D`

# The OR operator

- Using the OR operator, we can create a compound expression that is true when *either* of two conditions are true.
- Imagine a program that determines whether a student is eligible to enroll in AP CS A.
- The school's requirement is that the student must either have earned at least 75% in AP CSP *or* in Intro to programming.
- One way to implement that logic is with two separate if statements, like in the code below:

```
if cspGrade >= 75:  
    print("You're eligible for AP CS A!")  
if progGrade >= 75 :  
    print("You're eligible for AP CS A")
```

That code is problematic, however: we've repeated the same instructions in 2 places, and that means we now have to update 2 places whenever we want to change the instructions. It's very likely for one to get out of sync with the other;

A much better approach is to use an **OR** operator to combine those two conditions.

main.py

```
1   csp_grade = 92
2   prog_grade = 74
3
4   if csp_grade >= 75 or prog_grade >= 75:
5       print("You're eligible for AP CS A!")
```

Output :

You're eligible for AP CS A!

- Our new code implements the same logic, but is much shorter and easier to maintain.
- If the school adds a third way to be eligible, like completing a summer camp, we can easily add that to the compound expression using an additional OR operator and condition:

main.py

```
1  | csp_grade = 92
2  | prog_grade = 74
3  | summer_camp = True
4
5  | if csp_grade >= 75 or prog_grade >= 75 or summer_camp
   | == True:
6  |     print("You're eligible for AP CS A!")
```

Output :

You're eligible for AP CS A!

- Since that expression is made entirely of OR operators, the expression is true as long as *any* of the conditions are true.
- It will only be false if every single condition is false.



# The **AND** operator

- Using the **AND** operator, we can create a compound expression that is true only when *both* of the conditions are true.
- Let's make a program that determines whether a student meets a university's graduation requirements.
- The university requires that the student has a cumulative GPA higher than 2.0 *and* that the student has completed at least 120 units.

- One way to implement that logic is with nested conditionals:  
    if cumulativeGPA > 2.0:  
        if totalUnits >= 120:  
            print("You can graduate!")
- We can shorten that code significantly by using an AND operator.

We can shorten that code significantly by using an AND operator.

main.py

```
1 cumulative_gpa = 2.9
2 total_units = 135
3
4 if cumulative_gpa > 2.0 and total_units >= 120:
5     print("You can graduate!")
6
```

Output

You can graduate!

This code is logically equivalent to the nested conditional, but it's both shorter and easier to read;