ICDT 1201Y Computer progarmming

Week4_part2

Decision Structures (Continued)

Learning Objectives

- Understand the nested if statement in Python.
- Use of the Logical Operators `and`, `or`, and `not` to chain together simple conditions.

The nested if statement

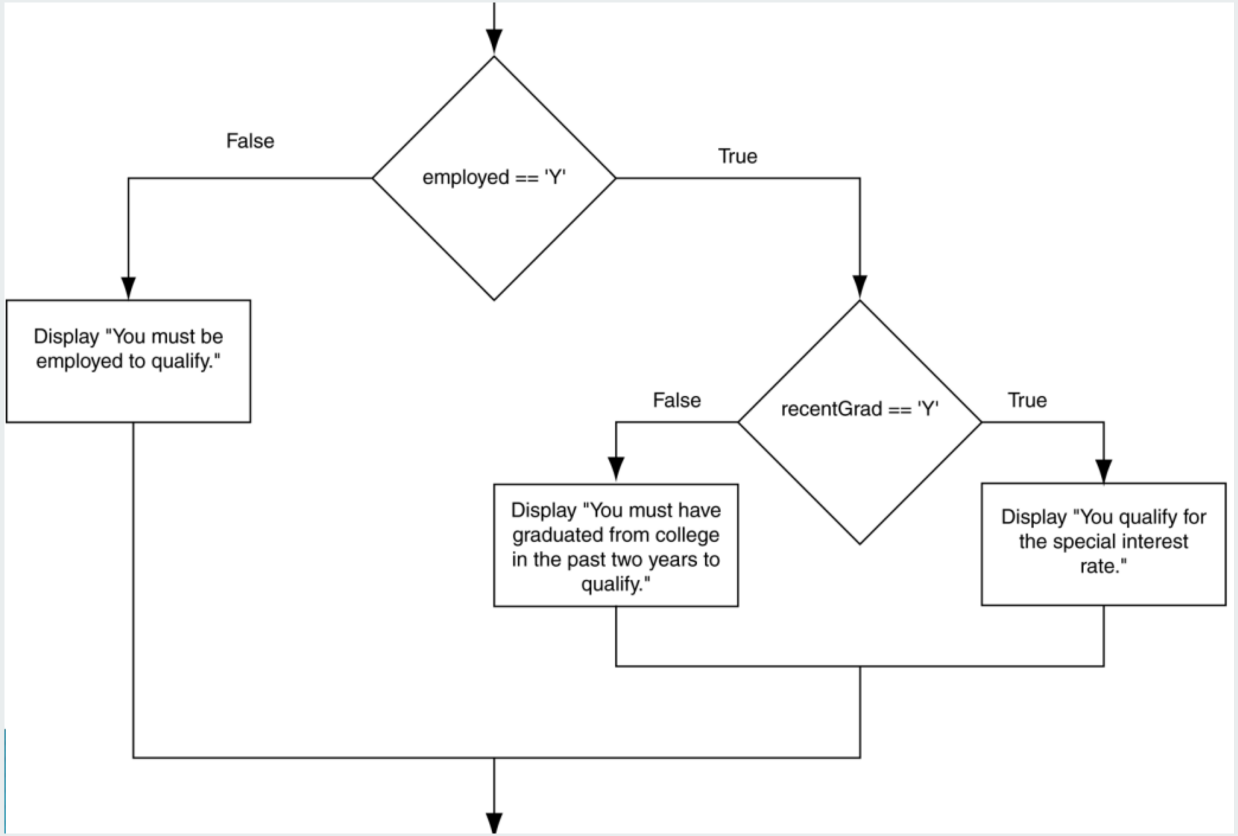
- There may be a situation when you want to check for another condition after a condition resolves to true.
- In such a situation, you can use the nested if construct.
- In a nested if construct, you can have an if... elif ...else construct inside another if ...elif ...else construct.

Example 4.1

Consider the following case:

- A person qualifies for a loan with a special interest rate, if he/she is employed provided he/she has recently graduated.

Flowchart for a nested if statement



employed == 'Y'
False
True
Display "You must be employed to qualify."
recentGrad == 'Y'
False
True
Display "You must have graduated from college in the past two years to qualify."
Display "You qualify for the special interest rate."

Pseudocodes

```
if (employed==‘Y’) then
        if (recently_graduated==‘Y) then
                Output "Qualified for loan with special interest rate"
        else
                Output "Not qualified for loan (not recently graduated)"
else
        Output "Not qualified for loan (not employed)"
End
```

Nested if statements

# Check eligibility for special interest rate loan

```
if employment_status == "Y":
        if graduation_status == "Y":
                print("You qualify for the special interest rate")
        else: #Not a recent graduate but employed
                print("Only those who have graduated within the last two years to
                        qualify")
else: #Not Employed
        print("You must be employed to qualify")
```

Proper Indentation

- Proper indentation in Python is crucial for maintaining code clarity and ensuring that the structure of nested if statements (and other control structures) is visually clear and logically correct.
- Python uses indentation to define the scope of code blocks, unlike languages that use curly braces {} or keywords like begin and end

Nested if example 4.2

Consider the following problem.

- A student is asked to input his marks. If his score is >= 70, he passes with Grade A. Else If his score is below 70, but >= 40 he passes with Grade B. Else if his score < 40 he fails with Grade F.
- Additionally there is another check if the score is >=70. If the score is above 80, the message "Excellent" gets displayed. If the score is above 90, the message "Outstanding" gets displayed.
- Exercise: Write the pseudocode for the above problem to output the Grade of the student depending on his marks and the message "Excellent" or "Outstanding" if applicable

Pseudocode nested if example 4.2

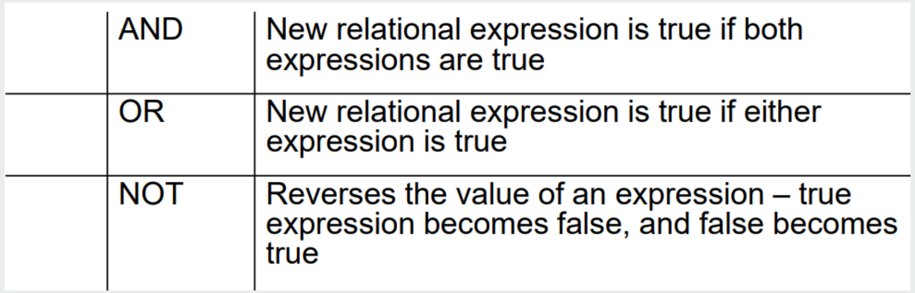
```
if student_marks >= 70 then
        Output "Grade A"
        if student_marks > 90 then
                Output "Outstanding"
        elif student_marks > 80 then
                Output "Excellent"
        End
elif student_marks >= 40 then
        Output "Grade B"
else
        Output "Grade F"
End
```

Exercise 4.1

- Convert the previous Pseudocode into a Python program

Logical Operators

- Used to create relational expressions from other relational expressions
- Operators, meaning, and explanation:



	AND	New relational expression is true if both expressions are true
	OR	New relational expression is true if either expression is true
	NOT	Reverses the value of an expression – true expression becomes false, and false becomes true

- The relational operators can be used to form compound condition

Compound Conditional Statements
Logical AND

Let’s reconsider example 4.1.

- The problem can be reformulated as :
- A person qualifies for a loan with special interest, if he/she is employed and has recently graduated.
- The conditional statement can be written, in pseudocodes, as below:

```
If ((employed==‘Y’) And (recentGrad==‘Y’))
        Output “You qualify for the special interest rate
Else
        Output “You must be employed and have recently graduated to
                qualify for the special interest rate”
End
```

The Logical AND Operator in Python

- In Python, the logical operator used for “AND" is the keyword “and”.
- Syntax:

expression1 **and** expression2

Thus, in Python the code becomes:

```
if employment_status == “Y“ and  graduation_status == “Y":
        print("You qualify for the special interest rate”)
```

NOTE.:It's important to use parentheses to clarify the order of operations when combining multiple conditions with and, especially in complex expressions.

Logical OR operator example

- Consider that a person qualifies for a loan if his income exceeds a threshold (minimum) income or has worked for more than a minimum number of years.
- The conditional statement can be written as below (pseudocode):

```
If ((income >=MIN_INCOME) OR (years >= MIN_YEARS))
        output "You qualify for loan"
Else
        output message specifying terms to qualify
End
```

The Logical OR operator in Python

- In Python, the logical operator used for OR is “or”
- Syntax:

expression1 or expression2

```
if income >=MIN_INCOME or years >= MIN_YEARS :
        print(“You qualify for loan”)
else:
        print(“Sorry, you are not qualified for the loan ”)
```

Logical not operator

- The `not` operator computes the opposite of a Boolean expression.
- `not` is a *unary* operator, meaning it operates on a single expression.

Example of not operator

```
#Example using the not operator with a variable
a = False
print("Original value of 'a':", a)
print("Using 'not' operator on 'a':", not a)

Output:
Original value of 'a': False
Using 'not' operator on 'a': True
```

Logical Operator -Order of preference

- We can put these operators together to make arbitrarily complex Boolean expressions.
- In Python, when multiple logical operators (and, or, not) are used in a single expression, they are evaluated based on their precedence and associativity rules. Here's the order of precedence from highest to lowest:

- Parentheses (): Parentheses can be used to explicitly specify the order of operations.

- Unary not: Unary not operator has the highest precedence.

- Binary and: Binary and operator (and) is evaluated next.

- Binary or: Binary or operator (or) has the lowest precedence.

Logical Operator -Order of preference

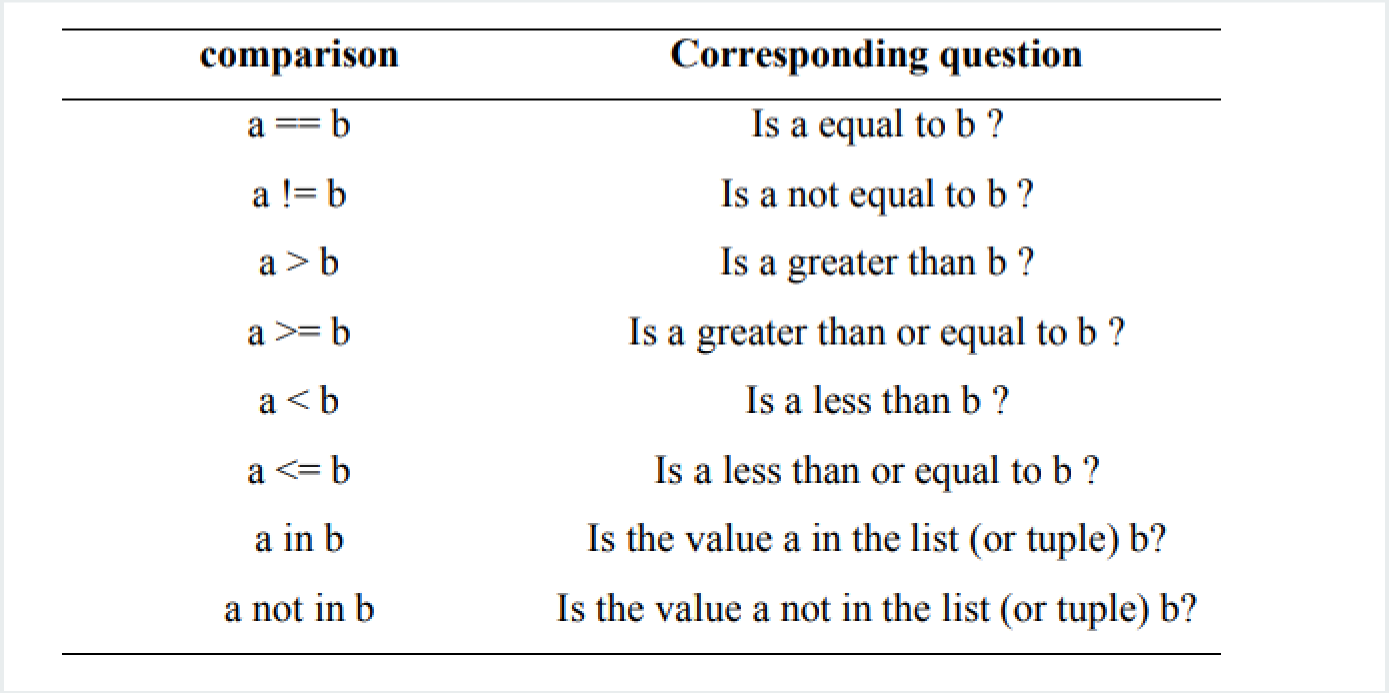
Example:

- Consider the expression A and B or C:
  - *and* has higher precedence than *or*, so the expression is evaluated as (A and B) or C.
  - If A is True, B is True, and C is False, then (A and B) evaluates to True, so the entire expression evaluates to True.
  - If A is False, B is False, and C is True, then (A and B) evaluates to False, so the entire expression evaluates to True.

Logical Operator -Order of preference

- Use of Parentheses:
- To avoid confusion and to explicitly specify the order of operations, it's recommended to use parentheses in expressions involving multiple operators. For example:
- A and (B or C) specifies that B or C should be evaluated first, and then A and ... should be evaluated based on that result.

Python Comparison Operators



comparison	Corresponding question
a == b	Is a equal to b ?
a != b	Is a not equal to b ?
a > b	Is a greater than b ?
a >= b	Is a greater than or equal to b ?
a < b	Is a less than b ?
a <= b	Is a less than or equal to b ?
a in b	Is the value a in the list (or tuple) b?
a not in b	Is the value a not in the list (or tuple) b?

Exercise 4.2

- To be eligible to graduate from ABC University, you must have 120 credits and a CPA of at least 40.
- Write a Python program that allows the input of number of credits and CPA of a student and checks and displays the eligibility to graduate from ABC University.