

Programming for Data Analytics



Week2: Decision Structures in Python

Dr. Mohammed Hasanuzzaman

Room J102, Melbourne Building (Office)

e-mail: mohammed.hasanuzzaman@cit.ie

Website: <https://mohammedhasanuzzaman.github.io>

Content

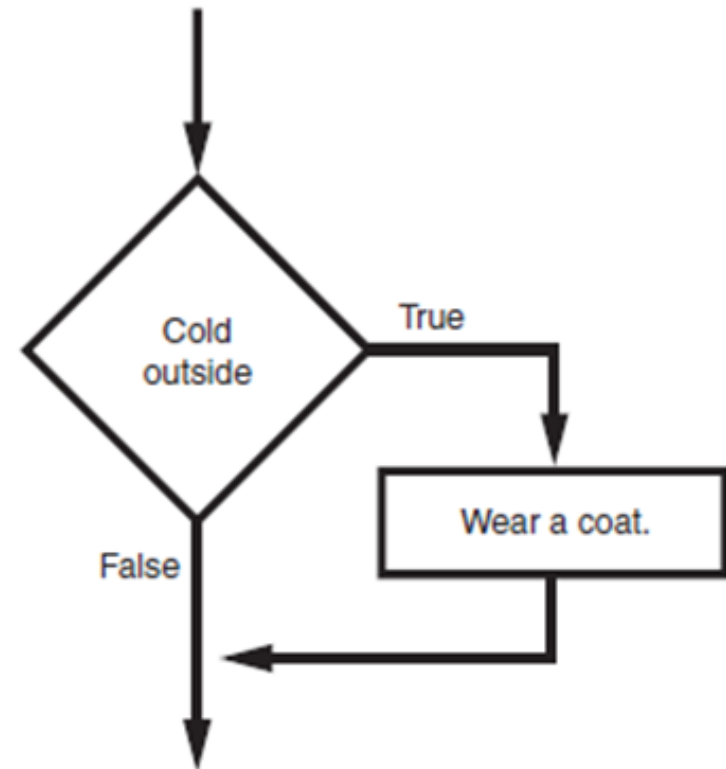


The if Statement

- The if-else Statement
- Nested Decision Structures and the if-elif-else Statement
- Logical Operators
- Boolean Variables

A Decision Structure

- Decision structure: ensures that specific action(s) are performed only if a condition exists
 - Also known as selection structure
 - Example:
 - If a user is ages 18 or less a ticket will cost them €10
 - If the room temperature drops below 10 degree Celsius the heating should be activated



The `if` Statement

- Python syntax:

```
if condition:  
    statement  
    statement
```

Notice the `:` after the condition

Notice the indentation after the `if` condition

- First line is known as the `if` clause
 - Includes the keyword `if` followed by a condition
 - The condition can be true or false
 - When the `if` statement executes, the condition is tested, and if it is true the **indented block statements are executed**. otherwise, block statements are skipped

What is a Condition (Boolean Expression)

- A condition is a boolean expression that is tested by the *if* statement to determine if it is true or false
 - Example: `a > b`
 - This expression returns `True` if *a* is greater than *b*; `False` otherwise

Expression	Meaning
<code>x > y</code>	Is x greater than y?
<code>x < y</code>	Is x less than y?
<code>x >= y</code>	Is x greater than or equal to y?
<code>x <= y</code>	Is x less than or equal to y?
<code>x == y</code>	Is x equal to y?
<code>x != y</code>	Is x not equal to y?

Difference between

`x==y`

`x=y`

Boolean Expressions and Relational Operators (cont'd.)

Employee is rewarded with a bonus of 500 if his/her sales exceed 5000

```
sales= input('Please enter number of sales')
numericalSales = int(sales)
bonus = 0

if numericalSales > 5000:
    bonus = 500

print ("Bonus Value is ", bonus)
```

Boolean Expressions and Relational Operators (cont'd.)

Employee is rewarded with a bonus of 500 if his/her sales exceed 5000

```
sales= input('Please enter number of sales')
numericalSales = int(sales)
bonus = 0

if numericalSales > 5000:
    bonus = 500

print ("Bonus Value is ", bonus)
```

Boolean Expressions and Relational Operators (cont'd.)

You can use multiple statements as part of an if statement. All indented lines after the if statement will be executed if the condition is evaluated to true.

```
sales= input('Please enter number of sales')
numericalSales = int(sales)
bonus = 0

if numericalSales>5000:
    bonus = 500
    print ("Well done on achieving the bonus")

print ("Bonus Value is ", bonus)
```


Exercise

- Write a simple program that will ask the user to enter a number that is a multiple of 10.
- If the number entered is not a multiple of 10 then the program should display an error message.

```
number = input("Enter a number that is a multiple of ten")
number = int(number)

if number%10 != 0:
    print ("The number you entered is not valid")
```

The `if-else` Statement

- Dual alternative decision structure:

(two alternative paths of execution)

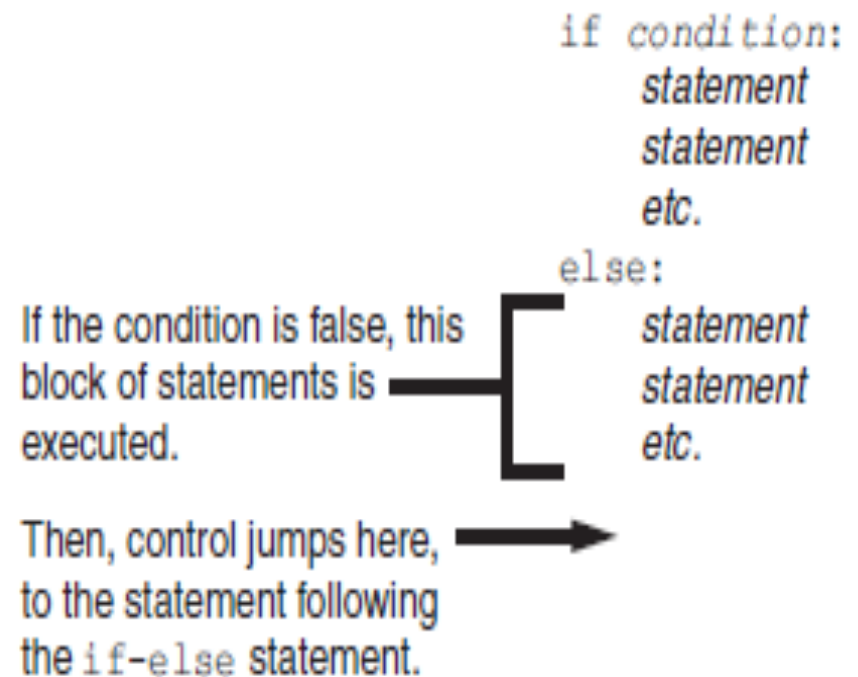
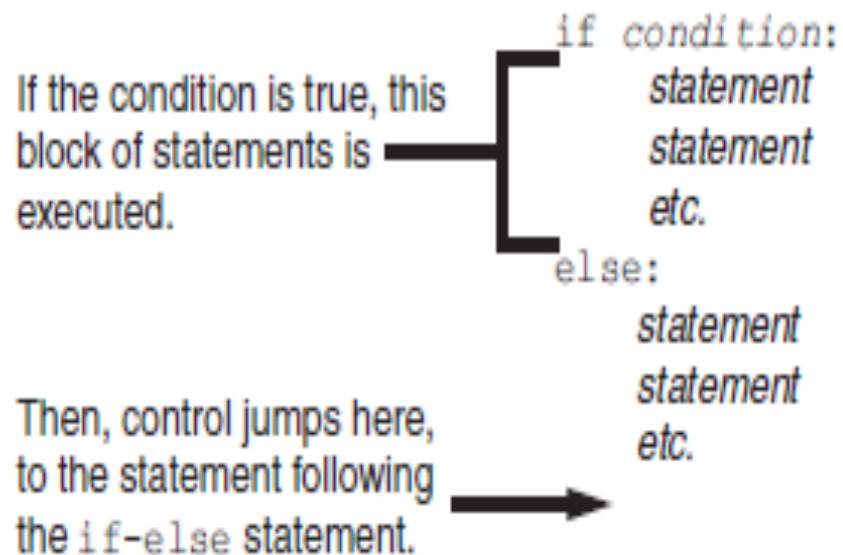
- One is taken if the condition is true, and the other if the condition is false

- Syntax: **if** *condition*:
 statement(s)
 else:
 other statement(s)

- `if` clause and `else` clause must be aligned
- Statements must be consistently indented

The `if-else` Statement (cont'd.)

Conditional execution in an `if-else` statement



The if-else Statement (cont'd)

```
sales = input('Please enter number of sales')
sales = int(sales)
bonus = 0

if sales>5000:
    bonus = 500
else:
    bonus = 100
print ("Bonus Value is ", bonus)
```

Comparing Strings

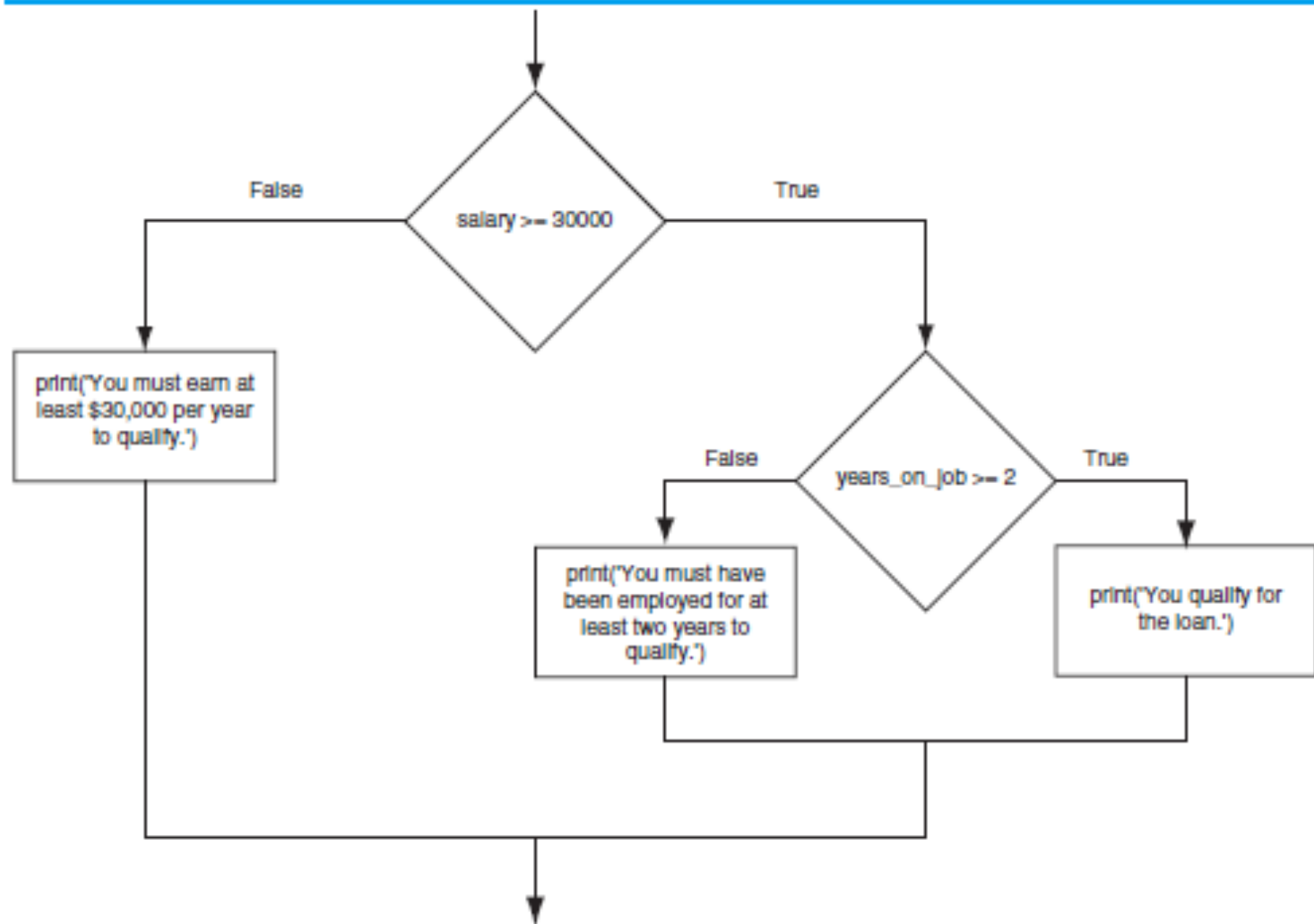
- Strings can be compared using the == and != operators
 - Just like variables String comparisons are case sensitive
 - The below code will print out “The names are NOT the same”

```
name1 = 'Mary'
name2 = 'Abbey'
if name1 == name2:
    print ("The names are the same")
else:
    print ("The names are NOT the same.")
```

Nested Decision Structures and the `if-else` Statement

- If you need to evaluate multiple conditions then you can use a nested decision structure (an *if* statement inside another *if*)
 - Commonly needed in programs
 - Example:
 - Determine if someone qualifies for a loan, they must meet two conditions:
 - Must earn at least \$30,000/year
 - Must have been employed for at least two years
 - Check first condition, and if it is true, check second condition

A nested decision structure



Nested Decision Structures and the `if-else` Statement

```
# Get the customer's annual salary.
salary = int(input("Enter your annual salary:"))
# Get the number of years on the current job.
yearsEmployed = int (input("Enter num years employed"))

if salary >= 30000.0:
    if yearsEmployed >= 2:
        print ("You qualify for the loan.")
    else :
        print ("You must have been on your current")
        print ("job for at least two years to qualify.")
else:
    print ("You must earn at least $30,000 per year")
```



Nested Decision Structures and the `if-else` Statement (cont'd.)

- Important to use **proper indentation** in a nested decision structure
 - Important for Python interpreter
 - Makes code more readable for programmer
- Rules for writing nested if statements:
 - else clause should **align** with matching if clause
 - Statements in each block must be **consistently indented**

Nested Decision Structures and the `if-else` Statement

```
# Get the customer's annual salary.
salary = int(input("Enter your annual salary:"))
# Get the number of years on the current job.
yearsEmployed = int (input("Enter num years employed"))


if salary >= 30000.0:
    if yearsEmployed >= 2:
        print ("You qualify for the loan.")
    else :
        print ("You must have been on your current")
        print ("job for at least two years to qualify.")
else:
    print ("You must earn at least $30,000 per year")
```

A blue diagram consisting of two arrows pointing to the right. The top arrow points to the line `if yearsEmployed >= 2:`. The bottom arrow points to the line `else :`. A vertical line connects the start of these two arrows on the left, indicating the scope of the nested decision structure.

Nested Decision Structures and the `if-else` Statement

```
# Get the customer's annual salary.
salary = int(input("Enter your annual salary:"))
# Get the number of years on the current job.
yearsEmployed = int (input("Enter num years employed"))

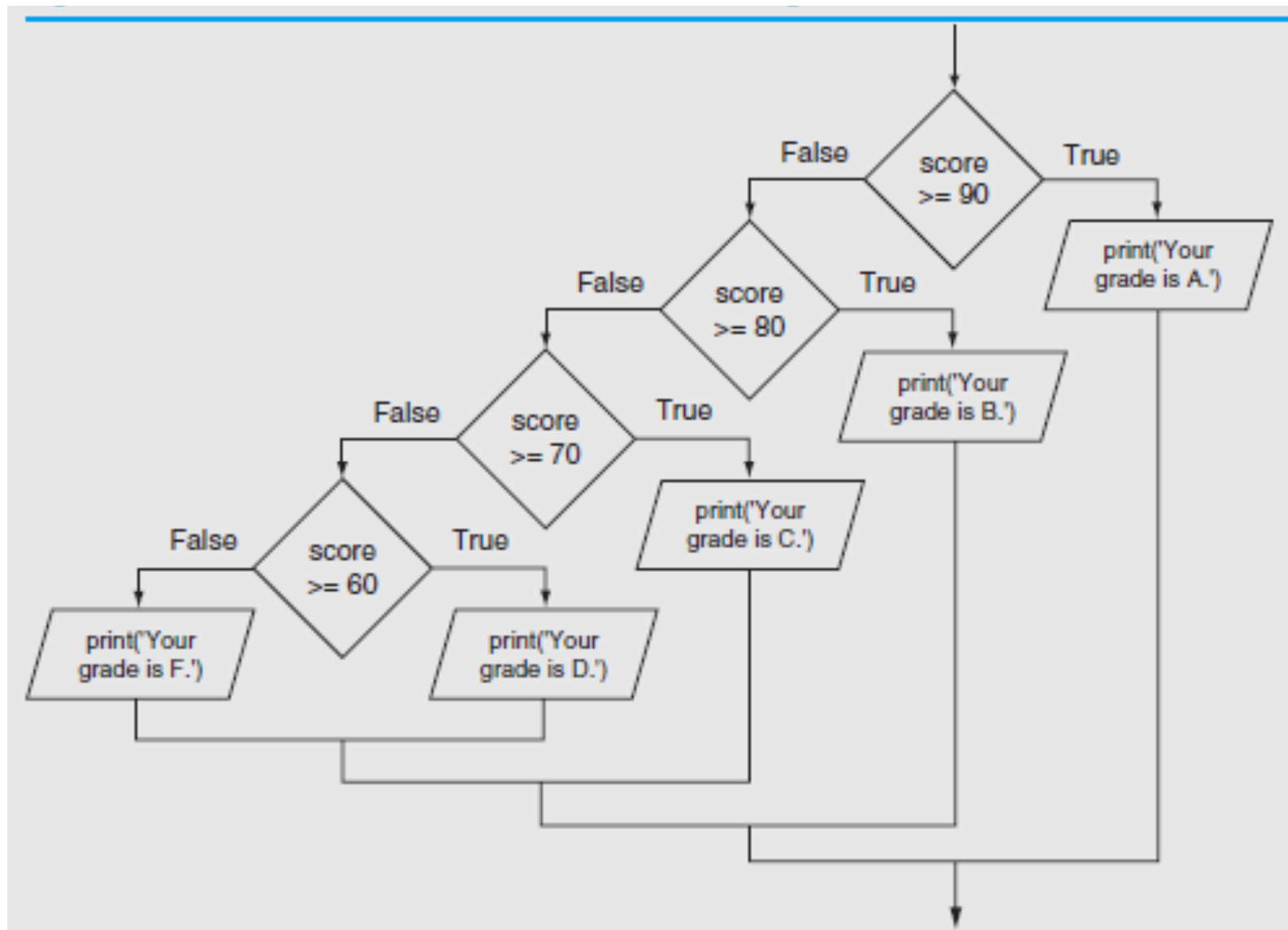
if salary >= 30000.0:
    if yearsEmployed >= 2:
        print ("You qualify for the loan.")
    else :
        print ("You must have been on your current")
        print ("job for at least two years to qualify.")
else:
    print ("You must earn at least $30,000 per year")
```

A blue line with arrows at both ends highlights the nested structure. One arrow points to the outer 'if salary >= 30000.0:' line, and the other points to the 'else:' line at the bottom. This indicates the flow of the decision process.

The `if-elif-else` Statement

- `if-elif-else` statement: special version of a decision structure
 - Makes logic of nested decision structures simpler to write
 - Can include multiple `elif` statements
 - Syntax:

```
if condition1
    statements
elif condition2
    statements
...
else
    statements
```



Nested Decision Structures and the `if-elif-else` Statement

```
# Get the students overall grade.  
score = int(input("Enter your exam score:"))  
if score < 60:  
    print "Your grade is F."  
elif score < 70:  
    print "Your grade is D."  
elif score < 80:  
    print "Your grade is C."  
elif score < 90:  
    print "Your grade is B."  
else :  
    print "Your grade is A."
```

Programming Task

- Write a program that will provide a user with pricing information on airline tickets. A user should select the type of ticket they wish to purchase and the program should print out the price (see below for pricing information).

Seat Type	Price
First Class	600
Premium	350
Economy	200

```
print ("The following are the category of tickets you can
purchase.")
print (" 1. First Class \n 2. Premium    \n 3. Economy")

userChoice = int(input("Please input an option 1, 2 or 3"))

if userChoice == 1:
    print ("Price is 600")
elif userChoice == 2:
    print ("Price is 350")
elif userChoice == 3:
    print ("Price is 200")
else:
    print ("Invalid Choice")
```


Logical Operators

- Logical operators: operators that can be used to create complex conditional statements
 - **and** operator and **or** operator: binary operators, connect two Boolean expressions into a single compound Boolean expression
 - **not** operator: unary operator, reverses the truth of its Boolean operand

The and operator

```
# Get the students exam score and output the grade.  
score = int(input("Enter your exam score:"))  
  
if score >= 60 and score <70:  
    print ("Your grade is a D.")
```

Notice for the first time that the if statement now contains two conditions. Only when both conditions are true will the subsequent line of indented code be executed.

We expand this to create a if statement with many conditions.

The and Operator

- Takes two Boolean expressions as operands
 - Creates a compound Boolean expression that is True only when **both sub expressions are true**
 - Can be used to simplify nested decision structures
- Truth table for the **and** operator

Expression	Value of the Expression
false and false	false
false and true	false
true and false	false
true and true	true

The or operator

If a user enters a value that is less than 0 or greater than 100 then they should be informed that it is invalid otherwise our code should say invalid.

Notice that as long as one of the conditions below is true then the subsequent indented code is executed.

```
# Inform use if value received is invalid
grade = int(input(" Please enter exam grade: "))

if grade < 0  or grade > 100:
    print ("Invalid Grade ")
else:
    print ("Valid Grade ")
```

The `or` Operator

- Takes two Boolean expressions as operands
 - Creates compound Boolean expression that is true when **either of the sub expressions is true**
 - Can be used to simplify nested decision structures
- Truth table for the `or` operator

Expression	Value of the Expression
false and false	false
false and true	true
true and false	true
true and true	true

Short-Circuit Evaluation

- Short circuit evaluation: deciding the value of a compound Boolean expression after evaluating only one sub expression
 - Performed by the `or` and `and` operators
 - **For `and` operator**: If left operand is false, compound expression is false. Otherwise, evaluate right operand
 - **For `or` operator**: If left operand is true, compound expression is true. Otherwise, evaluate right operand

The `not` Operator

- Takes one Boolean expressions as operand and reverses its logical value
 - Sometimes it may be necessary to place parentheses around an expression to clarify to what you are applying the not operator
- Truth table for the *not* operator

Expression	Value of the Expression
true	false
false	true

This code checks the current temperature value and outputs a message if it has not exceeded 30 degrees

```
temperature = int(input(" Please enter current temperature: "))
if not(temperature > 30):
    print ("This is below the maximum temperature")
```

Boolean Variables

- Boolean variable: references one of two values, `True` or `False`
 - Represented by `bool` data type
 - `discountAvailable = True`
 - `isValid = False`

```
salary = int(input("Enter your annual salary:"))
```

```
result = salary > 30000
```

```
print (result)
```

```
print (type(result))
```

Enter your annual salary: 700

False

<class 'bool'>

Boolean Variables

- Boolean variables are commonly used as flags:
 - Flag: variable that signals when some condition exists in a program
 - Flag set to `False` → condition does not exist
 - Flag set to `True` → condition exists

Sample Program using Booleans

- A salesperson has a quota of 50,000 in sales.

```
sales = int(input("Please input total sales"))

if sales >= 50000.0:
    salesQuotaMet = True
else:
    salesQuotaMet = False

if salesQuotaMet == True:
    print ("Congratulations. You will receive a bonux ")
```

Sample Program using Booleans

- A salesperson has a quota of 50,000 in sales.

```
sales = int(input("Please input total sales"))

if sales >= 50000.0:
    salesQuotaMet = True
else:
    salesQuotaMet = False

if salesQuotaMet:
    print ("Congratulations. You will receive a bonux")
```

Programming Task

- Write a revised and extended version of the loan approval program using logical operators (use a single if statement).
- This version of the program should tell the user if they are approved or not for the loan.
 - A user will be approved for a loan if at least one of A or B below are met.
 - A) User has a salary greater than 30,000 and they have been working for more than two years
 - B) User has a bank account balance of greater than 70,000

Programming Task

```
salary = int(input("Please your annual salary"))
yearsWorking = int(input("Please enter number of years you have been working"))
bankBalance = int(input("Please enter current bank balance"))

if (salary >= 30000 and yearsWorking > 2) or bankBalance>70000:
    print ("Loan Approved")
else:
    print ("Loan Denied")
```

Summary

- Decision structures, including:
 - Single alternative decision structures (if)
 - Dual alternative decision structures (if/else , if/elif)
 - Nested decision structures
- Relational operators and logical operators as used in creating Boolean expressions
- Boolean variables

Discussion



Thank you