Branching In Python: Part 2

- Using logic in conjunction with branching
- Multiple IFs
- IF-ELIF
- Nested branches
- The IF-IN operator

James Tam

Exercise: Boolean Output And Logic

- Student exercise (to do on your own for extra practice): determine the output of the following program.

```
var1 = True
var2 = False
var3 = True

print(var1 and var2)
print(var1 or var2)
print((var1 and not(var2)) or var3)
```

James Tam

Decision-Making With Multiple Boolean Expressions (Connected With Logic)

- **Format:**

    if**(**(*Boolean expression*) *logical operator* (*Boolean expression*)**)**:
        body

- **Example:** 1if_and_positive.py
  - Learning objective of example: applying logical AND (all Boolean expressions must evaluate to true for compound expression to be true).

    if**(**(x > 0) **and** (y > 0)**)**:
        print("All numbers positive")

```
[csc branches 19 ]> python if_and.py
Enter first number: 1
Enter second number: 0
```

```
[csc branches 20 ]> python if_and.py
Enter first number: 0
Enter second number: 111
```

```
[csc branches 21 ]> python if_and.py
Enter first number: 7
Enter second number: 13
All numbers positive
```

James Tam

Forming Compound Boolean Expressions With The "OR" Operator

- **Format:**

    if((*Boolean expression*) **or** (*Boolean expression*)):
        body

- **Name of the online example:** 2if_or_hiring.py
  - Learning objective of example: applying logical OR (at least one Boolean expression must evaluate to true for compound expression to be true).

```
gpa = float(input("Grade point (0-4.0): "))
yearsJobExperience = int(input("Number of years of job
 experience: "))
if((gpa > 3.7) or (yearsJobExperience > 5)):
    print("You are hired")
else:
    print("Insufficient qualifications")
```

```
Grade point (0-4.0): 2
Number of years of job experience: 0
Insufficient qualifications
```

```
Grade point (0-4.0): 2
Number of years of job experience: 25
You are hired
```

```
Grade point (0-4.0): 3.85
Number of years of job experience: 0
You are hired
```

```
Grade point (0-4.0): 4
Number of years of job experience: 6
You are hired
```

James Tam

The "NOT" Operator

- **Format:**

  ```
  if not (Boolean expression):
      body
  ```

- **Name of the online example:** `3if_not.py`
  - Learning objective of example: when an action (IF-body) is applied when a condition is not met.
  - (Alternatives to using Not can sometimes be implemented using the inequality operator '!=')

  ```
  SYSTEM_PASSWORD = "password123"
  userPassword = input("Password: ")
  if not (userPassword == SYSTEM_PASSWORD):
      print("Using logical NOT-operator: Wrong password")
  ```

Don't Bypass Using Negation Just Because "It's Hard"

- Note: the Boolean expression using negation is quite simple but I have seen cases of more complex student code where the person avoided negation with an empty IF-body.

- Using negation:
  ```
  SYSTEM_PASSWORD = "123456"
  userPassword = input("Enter password: ")
  if not(userPassword==SYSTEM_PASSWORD):
      print("Using negation: Wrong password")
  ```

- Negation is too hard to figure out, let's **pretend it doesn't exist** (don't do this).
  ```
  if(userPassword==SYSTEM_PASSWORD):
      print() #empty if-body
  else:
      print("Wrong approach: Wrong password")
  ```

- JT: yes I know you can also check for inequality but that's not the learning objective.

Decision-Making With Multiple Alternatives/Questions

- IF (single question)
    - Checks a condition and executes a body if the condition is true
- IF-ELSE (single question)
    - Checks a condition and executes one body of code if the condition is true and another body if the condition is false
- Approaches for multiple (two or more) questions
    - **Multiple IFs**
    - **IF-ELIF-ELSE**

Decision Making With Multiple Ifs
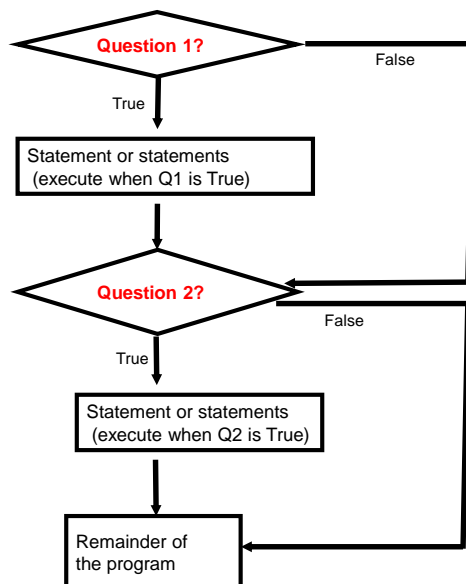
Notice that the 'Ifs' that come later are always evaluated regardless of what earlier 'Ifs' evaluate to.

Multiple `Ifs`: Non-Exclusive Conditions

- Use when more than one condition can be true.
  - Alternatively worded: 0+ conditions can be true.
  - Employ when a series of independent questions will be asked

- **Format:**

```
if(Boolean expression 1):
    body 1 (executes when BE1 true)
if(Boolean expression 2):
    body 2 (executes when BE2 true)
          :
statements after the conditions
```

Multiple `Ifs`: Non-Exclusive Conditions (Example)

- **Example:**

```
if(ableGPA >= 1):
    print("Passed course")
if(bakerGPA >= 1):
    print("Passed course")
if(foxTrotGPA >= 1):
    print("Passed course")
```

Use multiple Ifs for non-exclusive conditions: independent branches
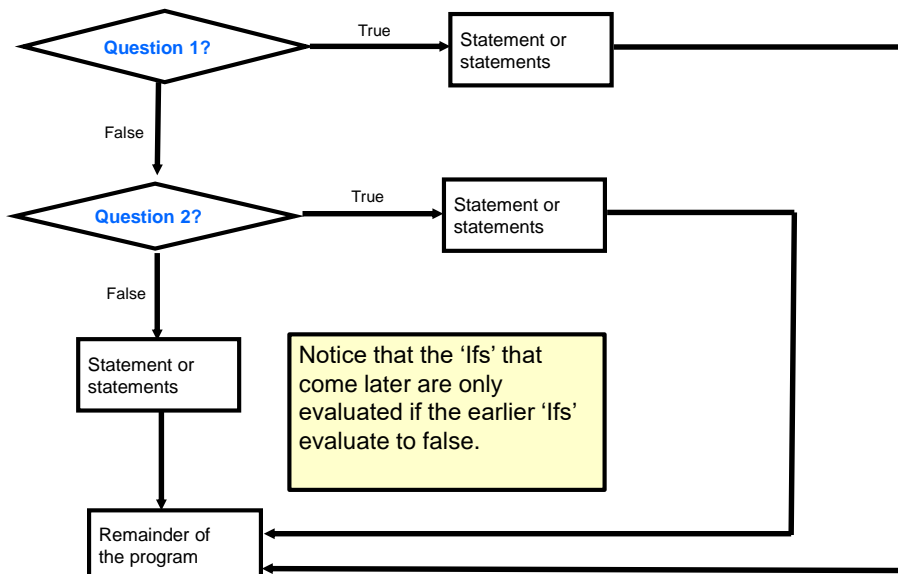
Multiple `Ifs`: Mutually Exclusive Conditions

- At most *only one* of many (i.e. 0 or 1) conditions can be true
- Can be implemented through multiple `ifs` ← **Inefficient combination!**
- **Example**: The name of the complete online program is:
  "4grades_inefficient.py"
  - Learning objective of example: illustrating how specifying a sequence of independent conditions can be less than optimal when at most only one condition can be true.

```
if(letter == "A"):
    gpa = 4
if(letter == "B"):
    gpa = 3
if(letter == "C"):
    gpa = 2
if(letter == "D"):
    print("Min pass")
    gpa = 1
if(letter == "F"):
    print ("Failing grade")
    gpa = 0
```

DO NOT use multiple Ifs for mutually exclusive conditions (interdependent branches) because you cannot easily implement a "none of the above" case.

James Tam

Decision Making With `If-Elif-Else`

Notice that the 'Ifs' that come later are only evaluated if the earlier 'Ifs' evaluate to false.

James Tam

Multiple `If-Elif-Else`: Use With Mutually Exclusive Conditions

•**Format:**

```
if(Boolean expression 1):
        body 1
elif(Boolean expression 2):
        body 2
                :
else
        body n ("none of the above")
statements after the conditions
```

Mutually exclusive
•One condition evaluating to true excludes other conditions from being true (no need to check)
•Example: having your current location as 'Calgary' excludes the possibility of the current location as 'Edmonton', 'Toronto', 'Medicine Hat'
•This means that only one of the 'bodies' 1, 2…n will execute.

`If-Elif-Else`: Mutually Exclusive Conditions (Example)

•**Example**: The name of the complete online program is: "5grades_efficient.py"

Use an If-elif structure for mutually exclusive conditions (interdependent branches).

- Learning objective of example: illustrating how ELIF is more efficient than multiple IFs when at most 1 condition (0 or 1) can be true.

```
if(letter == "A"):
    gpa = 4
elif(letter == "B"):
    gpa = 3
elif(letter == "C"):
    gpa = 2
elif(letter = "D"):
    gpa = 1
elif(letter = "F"):
    gpa = 0
else:
    print("GPA must be one of 'A', 'B', 'C', 'D' or 'F'")
```

This approach is more efficient when at most only one condition can be true.

Extra main benefit:

The body of the else executes only when all the Boolean expressions are false. (Useful for error checking/handling).

When To Use `Multiple-Ifs`

- When all conditions must be checked (<u>more than one</u> Boolean expressions for each 'if' <u>can be true</u>).
  - Non-exclusive conditions.

- Example:
  - Some survey questions:
    - When <u>all</u> the questions must be asked
    - The answers to previous questions will not affect the asking of later questions
      - E.g.,
      - Q1: Is your height above 172 cm?
      - Q2: Is your age 18 years or more?
      - Q3: Is Canada your country of birth?

When To Use `If,  ElIfs`

- When all conditions may be checked but <u>at most</u> only one Boolean expression <u>can evaluate to true</u>.
  - Exclusive conditions

- Example:
  - Survey questions:
    - When only some of the questions will be asked
    - The answers to previous questions WILL determine if later questions will be asked
      - E.g.,
      - Q1: Were you born in BC?
      - Q2 (ask only if the person answered 'no' to the previous): Were you born in AB?
      - Q3 (ask only if the person answered 'no' to the previous questions): Were you born in SK?
      - …

Rules Of Thumb With IF, ELSE, ELIF

- IF:
  - (As you have seen the ELSE, ELIF is optional.
- ELSE, ELIF
  - They are associated with the nearest IF at the same level of nesting.
    - Example:
      ```
      IF(BE1):
      ELIF(BE2):
      ELSE:
          IF(BE3):
          ELSE:
      ```
  - You cannot have more than one ELSE associated with an IF (there's only two states for the Boolean (True/False).
  - (As you have surmised): there is no syntax limit to the number of ELIF expressions that can be associated with an IF.
    - Any limits are tied to the problem (e.g. number of letter grades) not the language.

"Deal Breakers"

- A term that is used in everyday usage:
  - Business negotiations
    - "someone or something that prevents a deal or arrangement from being reached or fulfilled" [1]
  - Job applications (for both the person applying for the job and for the organization doing the hiring)
  - Personal relationships ('dating')
    - "a reason for rejecting someone…"[1]
      - E.g. if the person isn't 'local' then nothing else matters.
    - Top 10 relationship deal breakers (not required reading for developing your programming skills but seeing everyday examples of actual/common deal breakers can help if you are not familiar with the term "deal breaker")
      - https://www.psychologytoday.com/us/blog/close-encounters/201510/the-top-10-relationship-deal-breakers

[1] Definitions from: https://www.merriam-webster.com/dictionary/deal%20breaker

Programming Deal Breakers

- They are a condition or conditions that must be met.
- If the deal breaker condition/conditions are not met then other conditions are not relevant.
- Deal breakers can be implemented in the form of nesting (or nested branches/conditions).

Nesting In Programming (~"Deal Breaker")

- Nesting refers to an item that is "inside of" (or "nested in") some other item.
  - E.g. (**2+2**)*3, int(**input("Age: ")**)
- Nested branches: an '**IF-branch' that is inside of another** 'IF-branch'
  - Example (assume that the respondent previously indicated the birthplace was an Alberta city).
  - **Select the AB city in which you were born**
    1. Airdrie
    2. Calgary
    3. Edmonton

  Only when the user specifies the residence as Alberta does the program ask for which Alberta city is the residence
  - The prompt for which Alberta city is nested within (only asked) within a positive response indicating Alberta is the province.
  - Is province Alberta? Yes province is Alberta
    - **Which Alberta city?**

Recognizing When Nesting Is Needed

- **Scenario 1 (IF inside of another IF, other scenarios are described in the next section)**: A second question is asked if a only if the first question answers true:
  - Example (evaluating eligibility for social assistance): If the person's income is below a certain amount then check **if it's true the applicant is a Canadian citizen**.
  - Type of nesting: an IF-branch nested inside of another IF-branch
  - **Basic structure**
    ```
    If (Boolean):
         If (Boolean):
              ...
    ```
  - **Example**
  - ```
    If (Income less than cut off):
         If (Canadian citizen?):
              Receive social assistance
    ```

Nested Decision Making: Flowchart

- Decision making is dependent.
- The **first decision** must evaluate to true ("**deal breaker**"/"**gate keeper**") before **successive decisions** are even considered for evaluation.
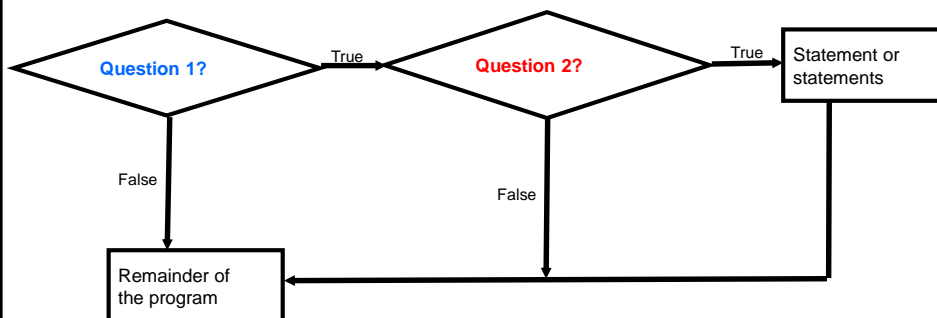
Nested Decision Making: Code Like (Pseudo-Code) Representation

- One **decision is made inside another**.
- **Outer decisions** must evaluate to true before inner decisions are even considered for evaluation.
- **Format:**

  ```
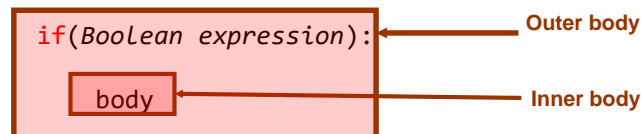  if(Boolean expression):
  ```

```
if(Boolean expression):          ← Outer body

     body          ← Inner body
```

Nested Decision Making: Illustrative Example

- **File containing the full example:** 6nesting.py
  - Learning objective of example: specifying an outer/initial gatekeeper condition for other conditions.

  ```
  if(income < 10000):
      if(citizen == 'y'):
          print("This person can receive social assistance")
          taxCredit = 100
  tax = (income * TAX_RATE) - taxCredit
  ```

```
Annual income: 10000
Income $10000.00
Tax credit $0.00
Tax paid $5000.
```

```
Annual income: 1000
Enter 'y' if citizen: n
Income $1000.00
Tax credit $0.00
Tax paid $500.00
```

```
nnual income: 1000
ter 'y' if citizen: y
is person can receive social assistance
Income $1000.00
Tax credit $100.00
Tax paid $400.00
```

Extra Practice

- (From "Starting out with Python" by Tony Gaddis).

  Write a program that prompts the user to enter a number within the range of 1 through 10. The program should display the Roman numeral version of that number. If the number is outside the range of 1 through 10, the program should display an error message. The table on the next screen shows the Roman numerals for the numbers 1 through 10.

Extra Practice (2)

Number	Roman Numeral
1	I
2	II
3	III
4	IV
5	V
6	VI
7	VII
8	VIII
9	IX
10	X

Recap: What Decision Making Mechanisms Are Available /When To Use Them

Mechanism	When To Use
`If`	Evaluate a Boolean expression and execute some code (body) if it's **true**
`If-else`	Evaluate a Boolean expression and execute some code (first body: 'if') if it's **true**, execute alternate code (second body: 'else') if it's **false**.
`Multiple ifs`	Multiple Boolean expressions need to be evaluated with the answer for **each expression being independent** of the answers for **the others (non-exclusive)**. Separate instructions (bodies) can be executed for each expression.
`If-elif-else`	Multiple Boolean expressions need to be evaluated but **zero or at most only one of them can be true (mutually exclusive)**. Zero bodies or exactly one body will execute. Also it allows for a separate body (`else`-case) to execute when all the `if-elif` Boolean expressions are false.

Recap: When To Use Compound And Nested Decision Making

Mechanism	When To Use
Compound decision making	There may have to be more than one condition to be considered before the body can execute. **All expressions must evaluate to true (AND)** or **at least one expression must evaluate to true (OR)**.
Nested decision making	The **outer Boolean expression ("gate keeper") must be true before the inner expression will be evaluated**. (Inner Boolean expression is part of the body of the outer Boolean expression).

Extra Practice

- (From "Starting out with Python" by Tony Gaddis)

  The following code contains several nested if-else statements.
  Unfortunately it was written without proper alignment and
  indentation. Rewrite the code and use the proper conventions of
  alignment and indentation.

Extra Practice (2)

```
# Grade cut-offs
A_SCORE = 90
B_SCORE = 80
C_SCORE = 70
D_SCORE = 60
```

**Common student question: If
there isn't a pre-created solution
then how do I know if I "got this
right"?**

```
if (score >= A_SCORE):
print("Your grade is A")
else:
if (score >= B_SCORE):
print("Your grade is B")
else:
if (score >= C_SCORE):
print("Your grade is C")
else:
if (score >= D_SCORE):
print("Your grade is D")
else:
print("Your grade is F")
```

Rule Of Thumb: Branches

• Be careful that your earlier cases don't include the later cases if each case is supposed to be handled separately and exclusively.

Example 1 (Wrong)

```
if (num >= 0):
elif (num >= 10):
elif (num >= 100):
```

Example 2 (Right)

```
if (num >= 100):
elif (num >= 10):
elif (num >= 0):
```

Extra Practice: Grades

• Write a program that converts percentages to one of the following letter grades: A (90 – 100%), B (80 – 89%), C (70 – 79%), D (60 – 69%), F (0 – 59%).

```
# First approach
if ((percentage <= 100) or (percentage >= 90)):
    letter = 'A'
elif ((percentage <= 89) or (percentage >= 80)):
    letter = 'B'
Etc.

# Second approach
if ((percentage <= 100) and (percentage >= 90)):
    letter = 'A'
elif ((percentage <= 89) and (percentage >= 80)):
    letter = 'B'
Etc.
```

Decision Making: Checking Matches

•Python provides a quick way of checking for matches within a set.
   - E.g., for a menu driven program the user's response is one of the values in the set of valid responses.

Format:
```
(Strings)
if <string variable> in ("<string_1> <string_2>...<string_n>"):
     body

(Numeric)
if <numeric variable> in (<number_1>, <number_2>,...<number_n>):
     body
```

Decision Making: Checking Matches (2)

Example:
```
(String):
if "the" in ("thetheretheir"):
    print("the is a sub-string of thetheretheir ")
else:
    print("not sub-string")

answer = input("Selection (any but 1, 2, 7): ")
if answer in ("one two seven"):
    print("selection taken")
else:
    print("selection available")

(Numeric):
if num in (1, 2, 3):
    print("in set")
```

Checking Matches: Another Example

• **Complete example**: `7inOperator.py`

• Learning objective of example: illustrating the use of the `IF-IN` operator in conjunction with strings as well showing the use of the NOT operator.

```
print("Menu options")
print("(a)dd a new player to the game")
print("(r)emove a player from the game")
print("(m)odify player")
print("(q)uit game")
selection = input("Enter your selection: ")
if selection not in ("a", "A", "r",  "R", "m", "M", "q", "Q"):
    print("Please enter one of 'a', 'r', 'm' or 'q' ")
```

Exercise: Nesting

• Student exercise (to do on your own for extra practice): determine the output of the following program.

```
num1 = int(input("First number: "))
num2 = int(input("Second number: "))
num3 = int(input("Third number: "))
print("1",end="")
if(num1 > 0):
    print("2",end="")
    if((num2 > 0) or (num3 < 0)):
        print("3",end="")
    else:
        print("4",end="")
print("5",end="")
```

> Determine output when the user enters:
>
> 0,0,0
> 1,2,3

After This Section You Should Now Know

- How to use the logical operators in conjunction with branching:
  - AND
  - OR
  - NOT
- How to test decision making constructs
- When/how to employ multiple IF structures
- When/how to employ an IF-ELIF structure
- When to employ nested IF structures
- How to trace nested IF structures
- The IF-IN operator

Copyright Notification

- "Unless otherwise indicated, all images in this presentation are used with permission from Microsoft."