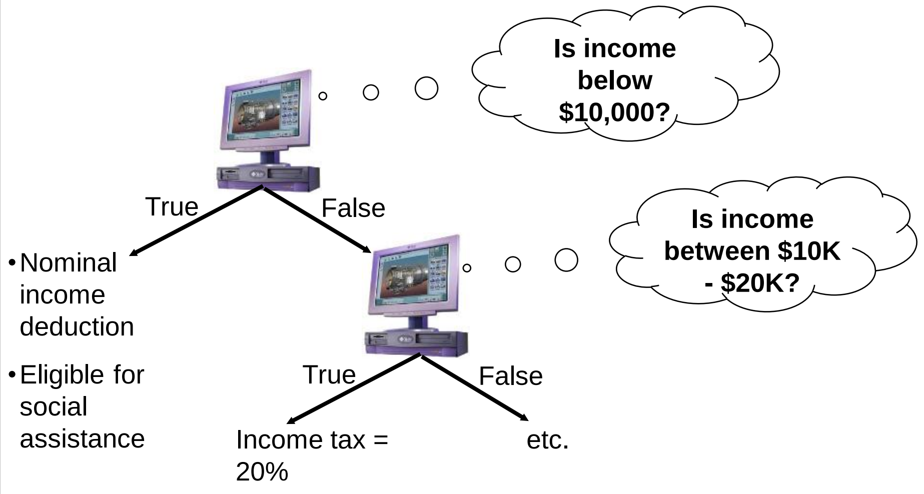
Making Decisions In Python

In this section of notes you will learn how to have your programs choose between alternative courses of action.

Programming: Decision Making Is Branching

- Decision making is choosing among alternates (branches).
- Why is it needed?
  - When alternative courses of action are possible and each action may produce a different result.
- In terms of a computer program the choices are stated in the form of a question that only yield a binary answer (is it true or false that the choice occurred).
  - Although the approach is very simple, modeling decisions in this fashion is a very useful and powerful tool.

High Level View Of Decision Making For The Computer


Is income below $10,000?
True
False
Is income between $10K - $20K?
•Nominal income deduction
•Eligible for social assistance
True
False
Income tax = 20%
etc.

James Tam

How To Determine If Branching Can Be Applied

- Under certain circumstances or conditions events will occur (the program reacts in a certain way if certain conditions have been met).
  - The branch determines if the event occurred and reacts accordingly.
- Examples:
  - If users who don’t meet the age requirement of the website he/she will not be allowed to sign up (conversely if users do meet the age requirement he/she will be allowed to sign up).
  - If an employee is deemed as too inexperienced and too expensive to keep on staff then he/she will be laid off.
  - If a person clicks on a link on a website for a particular location then a video will play showing tourist ‘hot spots’ for that location.
  - If a user enters invalid age information (say negative values or values greater than 114) then the program will display an error message.

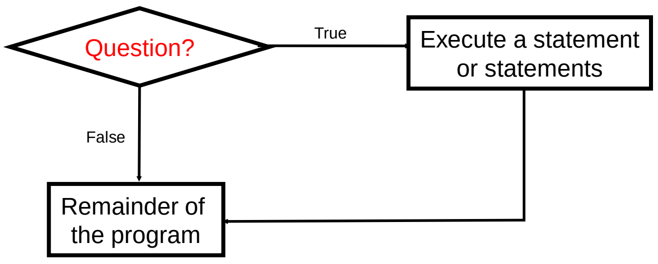
James Tam

Decision-Making In Programming (Python)

- Decisions are questions with answers that are either true or false (Boolean expressions) e.g., Is it true that the variable '`num`' is positive?
- The program may branch one way or another depending upon the answer to the question (the result of the Boolean expression).
- Decision making/branching constructs (mechanisms) in Python:
  - `If` (reacts differently only for true case)
  - `If-else (reacts differently for the true or false cases)`
  - `If-elif-else (multiple cases possible but only one case can apply, if one case is true then it's false that the other cases apply)`

James Tam

Decision Making With An 'If'



James Tam

Terminology

- The ‘body’ of a branch is the part of the program that will execute when the Boolean expression evaluates to true.

Travel website

Click link for Middle East? (T/F)

Click on link for North America? (T/F)

Body: show video for Abu Dhabi

Body: show video for Lethbridge

Images: courtesy of James Tam

James Tam

Terminology

- Operator/Operation: action being performed
- Operand: the item or items on which the operation is being performed.

Examples:

```
2 + 3
2 * (-3)
```

James Tam

The ‘If’ Construct

- Decision making: checking if a condition is true (in which case something should be done).
- **Format:**

```
(General format)
 if (Boolean expression):
        body
(Specific structure)
if (operand relational operator operand):
      body
```

Boolean expression

Note: Indenting the body is mandatory!

The ‘If’ Construct (2)

- **Example:**

```
if (age >= 18):
   print ("You are an adult")
```

Allowable Operands For Boolean Expressions

Format:

```
if (operand     relational operator    operand):
```

Example:

```
if (age >= 18):
```

Some operand types

- integer
- floats (~real)
- String
- Boolean (True or False)

Make sure that you are comparing operands of the same type or at least they must be comparable!

Allowable Relational Operators For Boolean Expressions

```
if (operand     relational operator     operand) then
```

Python operator	Mathematical equivalent	Meaning	Example
<	<	Less than	5 < 3
>	>	Greater than	5 > 3
==	=	Equal to	5 == 3
<=	≤	Less than or equal to	5 <= 5
>=	≥	Greater than or equal to	5 >= 4
!=	≠	Not equal to	x != 5

Note On Indenting (1)

- Indenting can make it easy to see structure (good style)

  1. Magnetic
     -Hard drives (includes older types of drives: floppy, zip)
  2. Optical
     - CD
     - DVD
  3. Solid State
     - USB 'thumb'/'flash' drives
     - Solid state hard drives (SSD)

James Tam

Note On Indenting (2)

- In Python indenting is mandatory in order to determine which statements are part of a body (syntactically required).

```
# Single statement body
if (num == 1):
    print("Body of the if")
print("After body")


# Multi-statement body (program 'if1.py')
taxCredit = 0
taxRate = 0.2
income = float(input("What is your annual income: "))
if (income < 10000):
     print ("Eligible for social assistance")
     taxCredit = 100
tax = (income * taxRate) - taxCredit
```

James Tam

Common Mistake

- Do not confuse the equality operator '==' with the assignment operator '='.
- **Example** (Python) **syntax error**[1]:

`if (num = 1):` Not the same as `(if num == 1):`

To be extra safe some programmers put unnamed constants on the left hand side of an equality operator (which always/almost always results in a syntax error rather than a logic error if the assignment operator is used in place of the equality operator).

- Usually (always?) a syntax error:

```
if (1 = num)
```

1 This not a syntax error in all programming languages so don't get complacent and assume that the language will automatically "take care of things" for you.

James Tam

A Similar Mistake

- **Example** (Python) **logic error**:

`num == 1` Not the same as `num = 1`

James Tam

An Application Of Branches

- Branching statements can be used to check the validity of data (if the data is correct or if the data is a value that's allowed by the program).
- **General structure**:

```
if (error condition has occurred)
    React to the error
```

- **Example**:

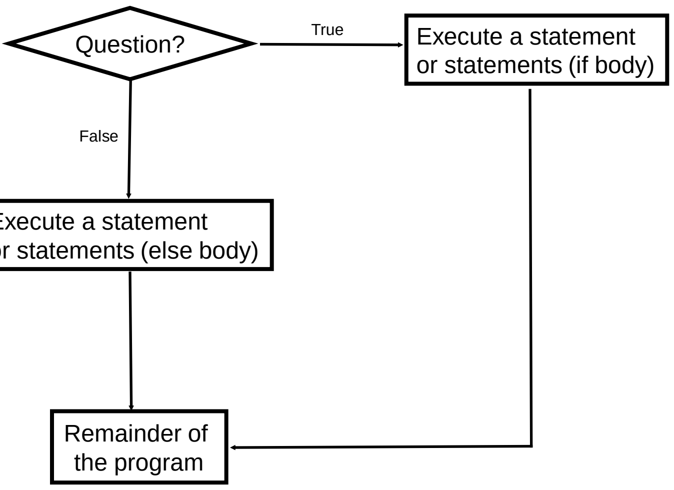
```
if (age < 0):
    print("Age cannot be a negative value")
```

JT's tip: if data can only take on a certain value (or range) do not automatically assume that it will be valid. Check the validity before proceeding onto the rest of the program.

Decision Making With An 'If': Summary

- Used when a question (Boolean expression) evaluates only to a true or false value (Boolean):
  - If the question evaluates to true then the program reacts differently. It will execute the body after which it proceeds to the remainder of the program (which follows the if construct).
  - If the question evaluates to false then the program doesn't react differently. It just executes the remainder of the program (which follows the if construct).

Decision Making With An ‘`If-Else`’


Question?
True
False
Execute a statement or statements (if body)
Execute a statement or statements (else body)
Remainder of the program

James Tam

The `If-Else` Construct

- Decision making: checking if a condition is true (in which case something should be done) but unlike ‘`if`’ *also reacting if the condition is not true (false).*
- **Format:**

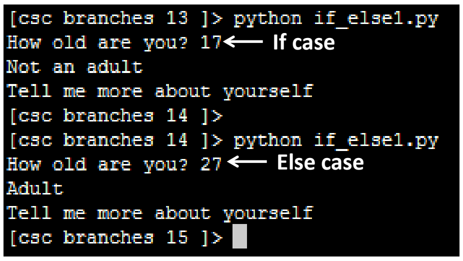
```
if (operand  relational operator  operand):
    body of 'if'
else:
    body of 'else'
additional statements
```

James Tam

If-Else Construct (2)

- **Program name:** `if_else1.py`
- **Example:**

```
if (age < 18):
    print("Not an adult")
else:
    print("Adult")
print("Tell me more about yourself")
```



```
[csc branches 13 ]> python if_else1.py
How old are you? 17
Not an adult
Tell me more about yourself
[csc branches 14 ]>
[csc branches 14 ]> python if_else1.py
How old are you? 27
Adult
Tell me more about yourself
[csc branches 15 ]>
```

James Tam

Lesson: Read Things The Way *They're Actually Stated* (Instead of How You Think They're Stated)

You this read wrong

James Tam

Lesson: Read Things The Way *They're Actually Stated* (Instead of How You Think They're Stated)

- **Example: Actual Code (previous version)**

```
if (age >= 18):
    print ("Adult")
else:
    print ("Not an adult")
print ("Tell me more about yourself")
```

JT's note: this version of the program is logically equivalent (does the same thing) as the previous version. For practice trace by hand both versions to convince yourself that this is the case. Then run both versions to verify.

Lesson: Read Things The Way They're Actually Stated (Instead of How You Think They're Stated)

- **Example:** How One Class Interpreted The Code (Optical Illusion?)

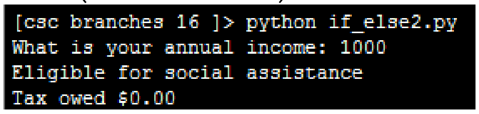
```
if (age >= 18):
    print("Adult")
else:
    print("Not an adult")
    print("Tell me more about yourself")
```

JT's tip: one way of making sure you read the program code the way it actually is written rather than how you think it should be is to take breaks from writing/editing

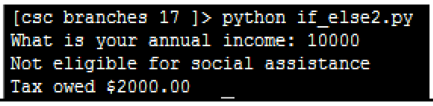
`If-Else` Example

- **Program name:** `if_else2.py`
- **Example:**

```
if (income < 10000):
    print("Eligible for social assistance")
    taxCredit = 100
    taxRate = 0.1
else:
    print("Not eligible for social assistance")
    taxRate = 0.2
tax = (income * taxRate) - taxCredit
```



```
[csc branches 16 ]> python if_else2.py
What is your annual income: 1000
Eligible for social assistance
Tax owed $0.00
```



```
[csc branches 17 ]> python if_else2.py
What is your annual income: 10000
Not eligible for social assistance
Tax owed $2000.00   _
```

James Tam

Quick Summary: `If` Vs. `If-Else`

- If:
  - Evaluate a Boolean expression (ask a question).
  - If the expression evaluates to true then execute the ‘body’ of the `if`.
  - No additional action is taken when the expression evaluates to false.
  - Use when your program is supposed to react differently only when the answer to a question is true (and do nothing different if it’s false).
- If-Else:
  - Evaluate a Boolean expression (ask a question).
  - If the expression evaluates to true then execute the ‘body’ of the `if`.
  - If the expression evaluates to false then execute the ‘body’ of the `else`.
  - That is: *Use when your program is supposed to react differently for both the true and the false cases*.

James Tam

Logical Operations

- There are many logical operations but the three most commonly used in computer programs include:
  - Logical AND
  - Logical OR
  - Logical NOT

Logical AND

- The popular usage of the logical AND applies when *ALL* conditions must be met.
- Example:
  - $\underbrace{\text{Pick up your son}}_{\textbf{Condition I}}$ AND $\underbrace{\text{pick up your daughter}}_{\textbf{Condition II}}$ after school today.
- Logical AND can be specified more formally in the form of a true table.

Truth table (AND)		
C1	**C2**	**C1 AND C2**
False	False	False
False	True	False
True	False	False
True	*True*	*True*

Logical AND: Three Input Truth Table

Truth table

C1	C2	C3	C1 AND C2 AND C3
False	False	False	False
False	False	True	False
False	True	False	False
False	True	True	False
True	False	False	False
True	False	True	False
True	True	False	False
True	*True*	*True*	*True*

James Tam

Evaluating Logical AND Expressions

- True **AND** True **AND** True
- False **AND** True **AND** True
- True **AND** True **AND** True **AND** True
- True **AND** True **AND** True **AND** False

James Tam

Logical OR

- The correct everyday usage of the logical OR applies when *ATLEAST* one condition must be met.
- Example:
  - You are using additional recommended resources for this course: the $\underbrace{\text{online textbook}}_{\textbf{Condition I}}$ OR the $\underbrace{\text{paper textbook}}_{\textbf{Condition II}}$ available in the bookstore.
- Similar to AND, logical OR can be specified more formally in the form of a truth table.

Truth table		
C1	**C2**	**C1 OR C2**
False	*False*	*False*
False	True	True
True	False	True
True	True	True

Logical OR: Three Input Truth Table

Truth table			
C1	**C2**	**C3**	**C1 OR C2 OR C3**
False	*False*	*False*	*False*
False	False	True	True
False	True	False	True
False	True	True	True
True	False	False	True
True	False	True	True
True	True	False	True
True	True	True	True

Evaluating Logical OR Expressions

- True **OR** True **OR** True
- False **OR** True **OR** True
- False **OR** False **OR** False **OR** True
- False **OR** False **OR** False **OR** False

James Tam

Logical NOT

- The everyday usage of logical NOT negates (or reverses) a statement.
- Example:
  - I $\underbrace{\text{am finding this class quite stimulating and exciting}}_{\textbf{Statement (logical condition)}}$ ..... $\underbrace{\textit{NOT!!!}}_{\textbf{Negation of the statement/condition}}$
- The truth table for logical NOT is quite simple:

Truth table	
S	**Not S**
False	True
True	False

James Tam

Evaluating More Complex Logical Expressions

- True **OR** True **AND** True
- **NOT** (False **OR** True) **OR** True
- (False **AND** False) **OR** (False **AND** True)
- False **OR** (False **OR** True) **AND** False
- **NOT NOT NOT NOT** True
- **NOT NOT NOT NOT** False
- **NOT NOT NOT** False

James Tam

Extra Practice

- (From "Starting out with Python (2nd Edition)" by Tony Gaddis)

  Assume the variables a = 2, b = 4, c = 6

  For each of the following conditions indicate whether the final value is true or false.

Expression	Final result
a == 4 or b > 2	
6 <= c and a > 3	
1 != b and c != 3	
a >-1 or a <= b	
not (a > 2)	

James Tam

Logic Can Be Used In Conjunction With Branching

- Typically the logical operators AND, OR are used with multiple conditions/Boolean expression:
  - If multiple conditions *must all be met* before the body will execute. (AND)
  - If *at least one condition* must be met before the body will execute. (OR)
- The logical NOT operator can be used to check for inequality (not equal to).
  - E.g., If it's true that the user *did not* enter an invalid value the program can proceed.

James Tam

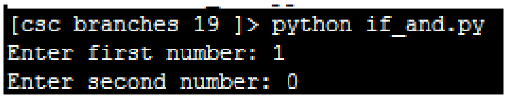
Decision-Making With Multiple Boolean Expressions (Connected With Logic)

- **Format:**

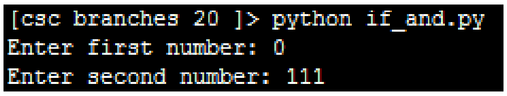
```
if (Boolean expression) logical operator (Boolean expression):
    body
```

- **Example:** `if_and.py`

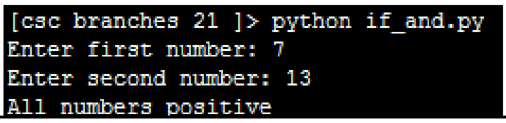
```
if (x > 0) and (y > 0):
    print ("All numbers positive")
```



```
[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

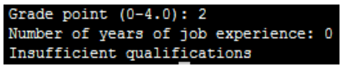
Forming Compound Boolean Expressions With The "OR" Operator

- **Format:**

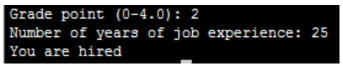
```
if (Boolean expression) or (Boolean expression):
    body
```

- **Name of the online example:** `if_hiring.py`

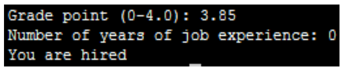
```
gpa = float(input("Grade point (0-4.0): "))
yearsJobExperience = int(input("Number of years of job
  experience: "))
if (gpa > 3.7) or (yearsJobExperience > 5):
    print("You are hired")
else:
    print("Insufficient qualifications")
```



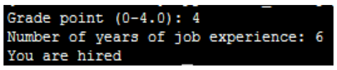
```
Grade point (0-4.0): 2
Number of years of job experience: 0
Insufficient qualifications
```



```
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

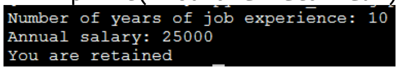
Forming Compound Boolean Expressions With The "AND" Operator

- **Format:**

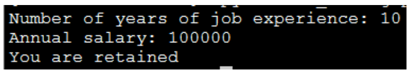
```
if (Boolean expression) and (Boolean expression):
    body
```

- **Name of the online example:** `if_firing.py`

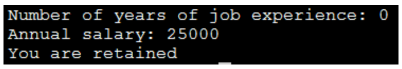
```
yearsOnJob = int(input("Number of years of job experience:
   "))
salary = int(input("Annual salary: "))
if (yearsOnJob <= 2) and (salary > 50000):
    print("You are fired")
else:
    print("You are retained")
```



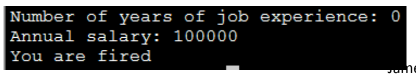
```
Number of years of job experience: 10
Annual salary: 25000
You are retained
```



```
Number of years of job experience: 10
Annual salary: 100000
You are retained
```



```
Number of years of job experience: 0
Annual salary: 25000
You are retained
```



```
Number of years of job experience: 0
Annual salary: 100000
You are fired
```

James Tam

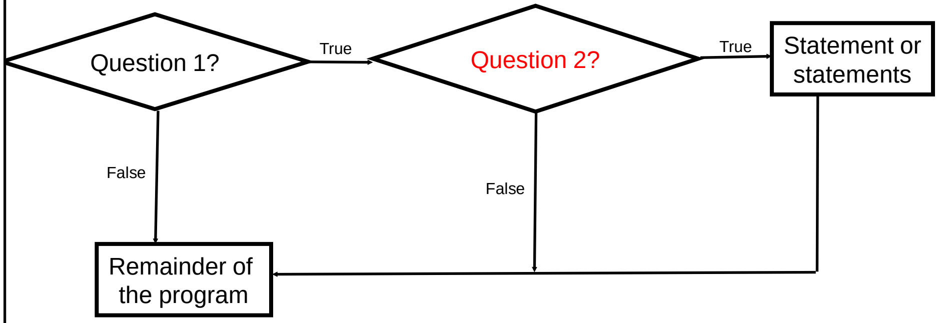
Quick Summary: Using Multiple Expressions

- Use multiple expressions when multiple questions must be asked and the result of all expressions must be considered together:
- AND (strict: all must apply):
  - All Boolean expressions must evaluate to true before the entire expression is true.
  - If any expression is false then whole expression evaluates to false.
- OR (at least one must apply):
  - If any Boolean expression evaluates to true then the entire expression evaluates to true.
  - All Boolean expressions must evaluate to false before the entire expression is false.

James Tam

Nested Decision Making

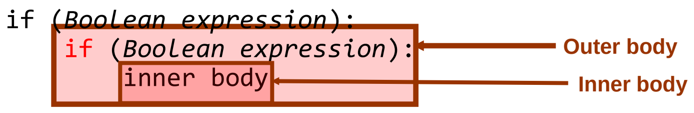
- Decision making is dependent.
- The first decision must evaluate to true ("gate keeper") before successive decisions are even considered for evaluation.



James Tam

Nested Decision Making

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

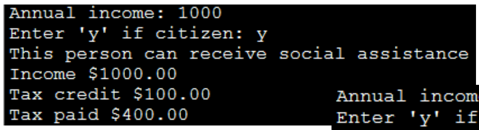


```
if (Boolean expression):
    if (Boolean expression):
        inner body
```

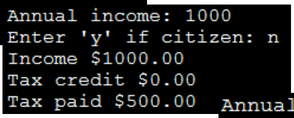
Nested Decision Making (2)

- **Example:** `nesting.py`

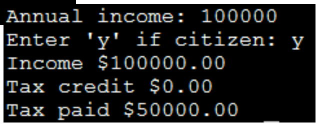
```
if (income < 10000):
    if (citizen == 'y'):
        print ("This person can receive social assistance")
        taxCredit = 100
tax = (income * TAX_RATE) - taxCredit
```



```
Annual income: 1000
Enter 'y' if citizen: y
This person can receive social assistance
Income $1000.00
Tax credit $100.00
Tax paid $400.00
```



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



```
Annual income: 100000
Enter 'y' if citizen: y
Income $100000.00
Tax credit $0.00
Tax paid $50000.00
```

Question

- What's the difference between employing nested decision making and a logical AND?

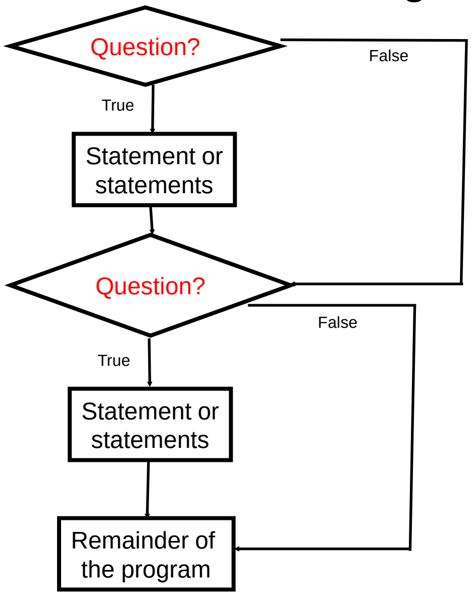
James Tam

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 IF's
  - IF-ELIF-ELSE

James Tam

Decision Making With Multiple If's



James Tam

Multiple If's: Non-Exclusive Conditions

- Any, all or none of the conditions may be true (independent)
- Employ when a series of independent questions will be asked
- **Format:**

```
if (Boolean expression 1):
    body 1
if (Boolean expression 2):
    body 2
          :
statements after the conditions
```

James Tam

Multiple If's: Non-Exclusive Conditions (Example)

- **Example:**

```
if (ableAge > 0):
     print ("Happy birthday!")
if (bakerAge > 0):
     print ("Happy birthday!")
if (foxtrotAge > 0):
     print ("Happy birthday!")
```

James Tam

Multiple If's: Mutually Exclusive Conditions

- At most *only one* of many conditions can be true
- Can be implemented through multiple if's

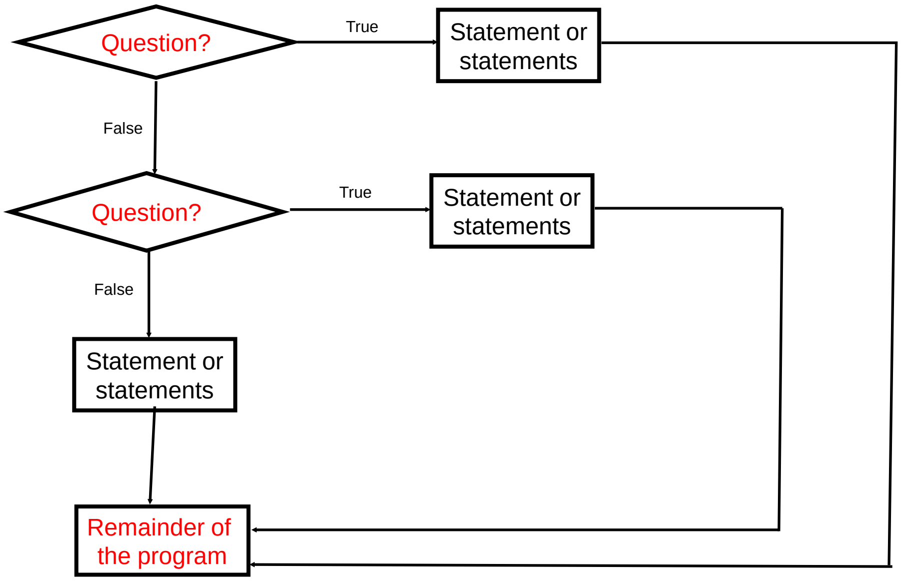
Inefficient combination!

- **Example**: The name of the complete online program is: "inefficient.py"

```
if (gpa == 4):
    letter = 'A'
if (gpa == 3):
    letter = 'B'
if (gpa == 2):
    letter = 'C'
if (gpa == 1):
    letter = 'D'
if (gpa == 0):
    letter = 'F'
```

James Tam

Decision Making With If-Elif-Else



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
statements after the conditions
```

James Tam

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

- **Example**: The name of the complete online program is: “`efficient.py`”

```
 if (gpa == 4):
     letter = 'A'

elif (gpa == 3):
     letter = 'B'

elif (gpa == 2):
     letter = 'C'

elif (gpa == 1):
     letter = 'D'

elif (gpa == 0):
    letter = 'F'

else:
   print ("GPA must be one of '4', '3', '2', '1' or '1'")
```

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

Extra benefit:

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

James Tam

When To Use `Multiple-If’s`

- When all conditions must be checked (more than one Boolean expressions for each ‘if’ can be true).
  - Non-exclusive conditions
- Example:
  - Some survey questions:
    - When all the questions must be asked
    - The answers to previous questions will not affect the asking of later questions

James Tam

When To Use `If, Else-If's`

- When all conditions must be checked (at most only one Boolean expression can evaluate to true).
  - Non-exclusive conditions
- Example:
  - Survey questions:
    - When only some of the questions will be asked
    - The answers to previous questions WILL affect the asking of later questions
      - E.g.,
      - Q1: Are you an immigrant?
      - Q2 (ask only if the person answered ‘no’ to the previous): In what Canadian city were you born?

James Tam

Extra Practice

- (From “Starting out with Python (2nd Edition)” 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 slide shows the Roman numerals for the numbers 1 through 10.

James Tam

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 if's`	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 even be evaluated. (Inner Boolean expression is part of the body of the outer Boolean expression).

Testing Decision Making Constructs

- Make sure that the body of each decision making mechanism executes when it should.
- Test:
  1) Obvious true cases
  2) Obvious false cases
  3) Boundary cases

Testing Decisions: An Example

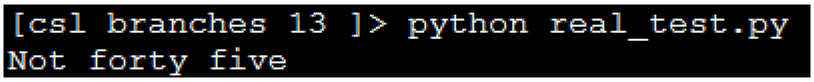
```
num = int(input("Type in a value for num: "))
if (num >= 0):
    print ("Num is non-negative. ")
else:
    print ("Num is negative. ")
```

James Tam

Lesson: Avoid Using A Float When An Integer Will Do

Program name: `real_test.py`

```
num = 1.0 - 0.55
if (num == 0.45):
    print ("Forty five")
else:
    print ("Not forty five")
```



```
[csl branches 13 ]> python real_test.py
Not forty five
```

James Tam

Epsilon

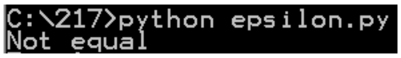
- Because floating point numbers are only approximations of real numbers when performing a comparison “seeing if two numbers are ‘close’ to each other” sometimes an Epsilon is used instead of zero.
- Epsilon is a very small number.
- If the Absolute difference between the numbers is less than the Epsilon then the numbers are pretty close to each other (likely equal).

James Tam

Not Using Epsilon: Floating Point Error

```
a = 0.15 + 0.15
b = 0.2 + 0.1
if (a == b):
   print("Equal")
else:
    print("Not equal")
```

Result



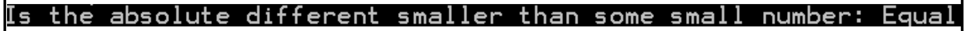
```
C:\217>python epsilon.py
Not equal
```

James Tam

Using Epsilon: Better Approach

```
EPSILON = 0.00001
a = 0.15 + 0.15
b = 0.2 + 0.1
if (abs((a - b)/b) < EPSILON):
   print("Is the absolute different smaller than some
small number: Equal")
else:
    print("Not equal")
```

Result



```
Is the absolute different smaller than some small number: Equal
```

James Tam

Extra Practice

- (From "Starting out with Python (2nd Edition)" 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.

James Tam

Extra Practice (2)

```
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")
```

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

James Tam

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

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

Example 2

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

James Tam

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.
```

James Tam

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 "<string1> <string2>...<stringn>":
     body

(Numeric)
if <numeric variable> in (<number1>, <number2>,...<numbern>):
     body
```

James Tam

Decision Making: Checking Matches (2)

Example:

```
(String):
if answer in ("userName1 userName2 userName3"):
    print ("User name already taken")
else:
    print ("User name is allowed")

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

James Tam

Checking Matches: Another Example

- **Complete example**: `userNames.py`

```
userNames = ""
print("User names already been taken [%s]" %userNames)
userName = input("Enter a user name: ")
if (userName in userNames):
     print("Name %s has already been taken" %userName)
else:
     userNames = userNames + userName + " "
print()
```

James Tam

Checking Matches: Another Example (2)

```
print("User names already been taken [%s]" %userNames)
userName = input("Enter a user name: ")
if (userName in userNames):
    print("Name %s has already been taken" %userName)
else:
     userNames = userNames + userName + " "
print()
```

James Tam

Checking Matches: Another Example (3)

```
print("User names already been taken [%s]" %userNames)
userName = input("Enter a user name: ")
if (userName in userNames):
    print("Name %s has already been taken" %userName)
else:
     userNames = userNames + userName + " "
print()

print("User names [%s]" %userNames)
```

James Tam

After This Section You Should Now Know

- What are the three decision making constructs available in Python:
  - `If`
  - `If-else`
  - `If-elif-else`
  - How does each one work
  - When should each one be used
- Three logical operations:
  - AND
  - OR
  - NOT
- How to evaluate and use decision making constructs:
  - Tracing the execution of simple decision making constructs
  - How to evaluate nested and compound decision making constructs and when to use them

You Should Now Know (2)

- How the bodies of the decision making constructs are defined:
  - What is the body of decision making construct
  - What is the difference between decision making constructs with simple bodies and those with compound bodies
- What is an operand
- What is a relational operator
- What is a Boolean expression
- How multiple expressions are evaluated and how the different logical operators work
- How to test decision making constructs