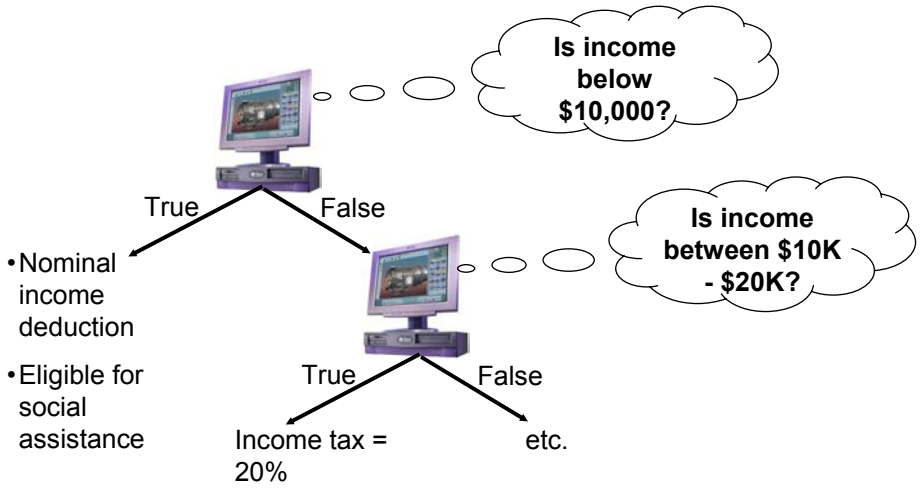
Making Decisions In Python

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

Why Is Branching/Decision Making Needed?

- When alternative courses of action are possible and each action may produce a different result.
- Branching/decision making can be used in a program to structure the alternatives and implement the results for each alternative.

High Level View Of Decision Making For The Computer



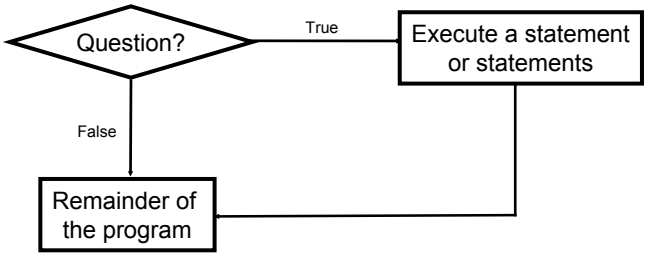
James Tam

Decision-Making In Python

- Decisions are questions with answers that are either true or false (Boolean) e.g., Is it true that the variable 'num' is positive?
- The program branches 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
  - If-else
  - If-elif-else

James Tam

Decision Making With An 'If'



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

If (**operand** relational operator **operand**) then:

Some operands
- integer
- real numbers
- String

Make sure that you are comparing operands of the same type!

Allowable Relational Operators For Boolean Expressions

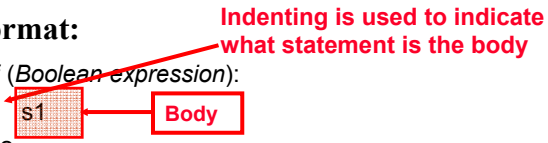
•If (operand **relational operator** operand) then

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	5 <> 5
OR			
!=			5 != 5

If (Simple Body)

•Body of the if consists of a single statement

•**Format:**



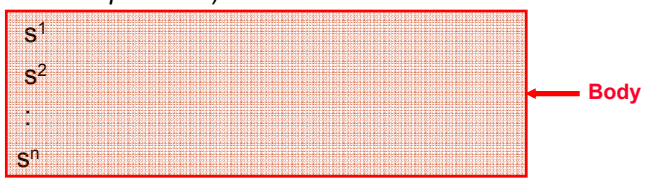
Example:

```
if (num == 1):
    print “Body of the if”
print “After body”
```

If (Compound Body)

- Body of the if consists of multiple statements
- **Format:**

if (*Boolean expression*):



S^{n+1}

End of the indenting denotes the end of decision-making

If (Compound Body(2))

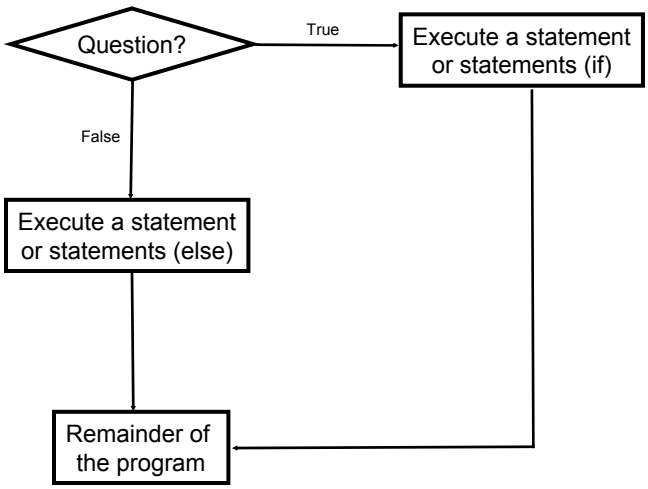
- **Example:**

```
taxCredit = 0
taxRate = 0.2
if (income < 10000):
    print “Eligible for social assistance”
    taxCredit = 100
tax = (income * taxRate) - taxCredit
```

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 a 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 different. It just executes the remainder of the program (which follows the if construct).

Decision Making With An 'If-Else'



The If-Else Construct

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

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


If-Else Construct (2)

- **Example:**

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

If-Else (Compound Body(2))

•**Example:**

```
taxCredit = 0
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
```


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.
- Use when your program is supposed to react differently for both the true and the false case.

Logical Operations

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

Logical AND

- The popular usage of the AND applies when *ALL* conditions must be met.
  - Example:
  - Pick up your son AND pick up your daughter after school today.

Condition I **Condition II**

- Logical AND can be specified more formally in the form of 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
- False **AND** True **AND** False **AND** True **AND** True **AND** False **AND** False **AND** True **AND** True

James Tam

Logical OR

- The correct everyday usage of the OR applies when *ATLEAST* one condition must be met.
- Example:
  - You are using additional recommended resources for this course: the online textbook OR the paper textbook available in the bookstore.

Condition I **Condition II**

- Similar to AND, logical OR can be specified more formally in the form of true table.

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

James Tam

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

James Tam

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

Logical NOT

- The everyday usage of logical NOT negates (or reverses) a statement.
- Example:
  - I am finding this class quite stimulating and exciting.....*NOT!!!*

Statement **Negation of the condition**

- The truth table for logical NOT is quite simple:

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

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

Logic Can Be Used In Conjunction With Branching

- Typically the logical operators AND, OR are used with multiple conditions:
  - If multiple conditions *must all be met* before a statement will execute. (AND)
  - If *at least one condition* must be met before a statement 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.

Decision-Making With Multiple Expressions

- **Format:**

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

  body

- **Example:**

```
if (x > 0) and (y > 0):
    print "X is positive, Y is positive"
```


Forming Compound Boolean Expressions With The "OR" Operator

- **Format:**

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

  body

- **Example:**

```
if (gpa > 3.7) or (yearsJobExperience > 5):
    print "You are hired"
```

Forming Compound Boolean Expressions With The "AND" Operator

- **Format:**

  if (*Boolean expression*) and (*Boolean expression*):

  body

- **Example:**

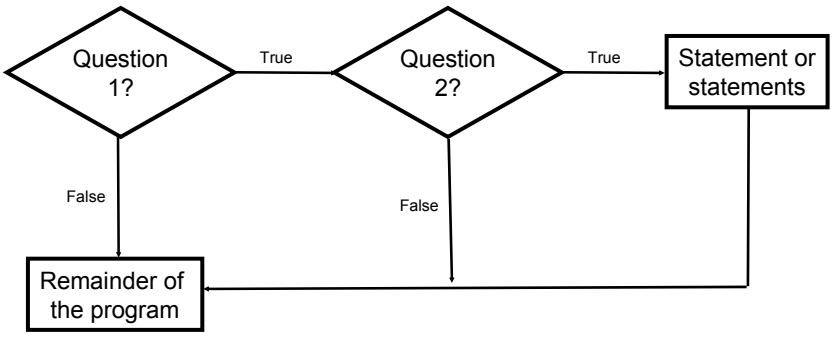
```
if (yearsOnJob <= 2) and (salary > 50000):
    print "You are fired"
```


Quick Summary: Using Multiple Expressions

- Use multiple expressions when multiple questions must be asked and the result of each expression may have an effect on the other expressions:
- AND:
  - 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:
  - 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.

Nested Decision Making

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



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
```

Outer body

Inner body

Nested Decision Making (2)

- **Example:**

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

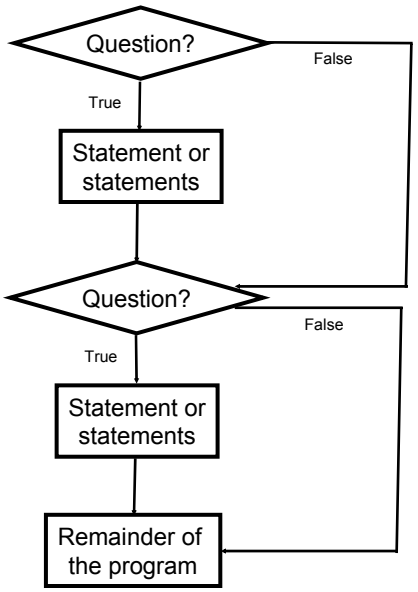

Question

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

Decision-Making With Multiple Alternatives

- IF
  - Checks a condition and executes the body of code if the condition is true
- IF-ELSE
  - 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) alternatives
  - Multiple IF's
  - IF-ELIF-ELSE

Decision Making With Multiple If's



Multiple If's: Non-Exclusive Conditions

- Any, all or none of the conditions may be true (independent)
- **Format:**

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


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

- **Example:**

```
if (num1 > 0):
    print "num1 is positive"
if (num2 > 0):
    print "num2 is positive"
if (num3 > 0):
    print "num3 is positive"
```

Multiple If's: Mutually Exclusive Conditions

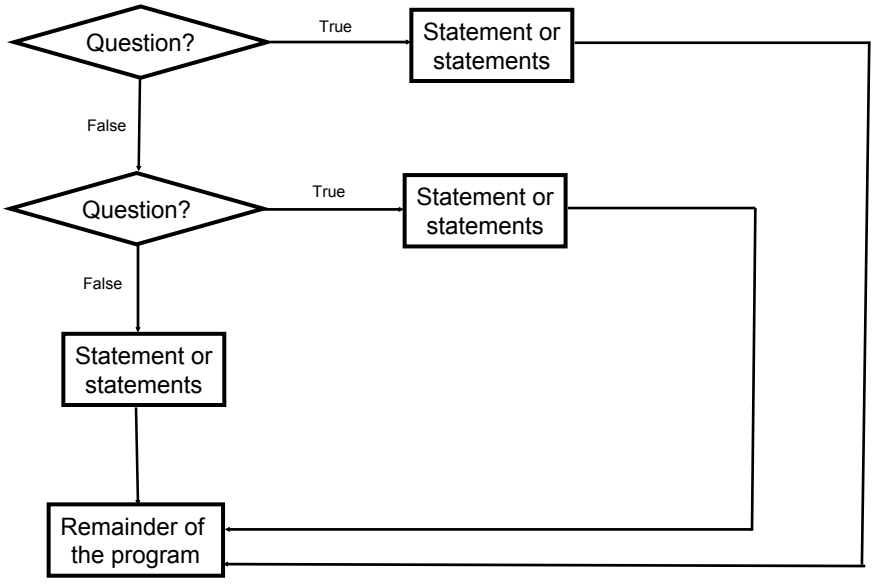
- At most *only one* of many conditions can be true
- Can be implemented through multiple if's

Inefficient combination!

- **Example** (for full example look in UNIX under /home/231/examples/decisions/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'
```


Decision Making With If-Elif-Else



Multiple If-Elif-Else: Mutually Exclusive Conditions

- **Format:**

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


Multiple If, Else-If's: Mutually Exclusive Conditions (Example)

- **Example** (the full version can be found in UNIX under /home/231/examples/decisions/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.

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

Recap: What Decision Making Constructs Are Available In Pascal/When To Use Them

Construct	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 code (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) to execute when all the if-elif Boolean expressions are false.

Recap: When To Use Compound And Nested Decision Making Constructs (2)

Construct	When To Use
Compound decision making	More than one Boolean expression must be evaluated before some code (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 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 construct executes when it should.

•Test:

1) Obvious true cases
2) Obvious false cases
3) Boundary cases

Testing Decisions: An Example

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

Avoid Using Real Values When An Integer Will Do

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


Problem Solving: Branches

- 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%).
- The percentage score should come from the user.
- After determining the letter grade, the original percentage and it's corresponding letter should be displayed.
- The program should display an error message for percentages outside of the above ranges.

Outline Of Solution

- Get the percentage score.
- Determine the letter grade
- Display the result

Developing A Solution: Start With The Easier Parts

```
percentage = 0.0
letter = ' '
percentage = input ("Enter the percentage score: ")

# Determine letter grade: don't look at the solution until you've tried to
# come up with a solution yourself.

print "Percentage: ", percentage, "%\t Letter: ", letter
```

Determining The Correct Ranges

- Before directly implementing a solution (i.e., writing Python code) make sure that you have a clear idea of what's entailed.
- Depending upon the complexity of the problem this process may be formal (e.g., drawing diagrams, writing text descriptions, using detailed and specific notations etc.) or informal (e.g., going over the solution in your head).
- Also if your solution is not working (contains errors) then return back to the process of specifying what's entailed but do it more formally and in a more detailed form.

Determining Ranges: A Solution (Don't Look Until You've Tried It Yourself)

```
if (percentage <= 100) and (percentage >= 90):
    letter = 'A'
elif (percentage <= 89) and (percentage >= 80):
    letter = 'B'
elif (percentage <= 79) and (percentage >= 70):
    letter = 'C'
elif (percentage <= 69) and (percentage >= 60):
    letter = 'D'
elif (percentage <= 59) and (percentage >= 0):
    letter = 'F'
else:
    print "Percent score is outside the allowable range (0 - 100%)"
    letter = 'Error'
```

Question: What happens if logical "OR" is employed instead of "AND"

Decision Making: Checking Matches (2)

Example:

(String):

```
if answer in "password1 password2 password3":
    print "correct"
else:
    print "incorrect“
```

(Numeric):

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


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

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 construct 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