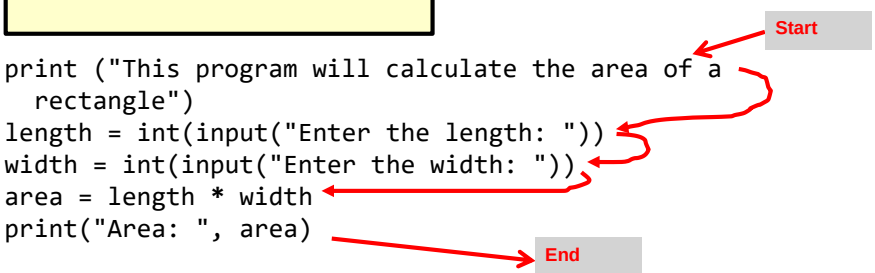
Branching In Python: Part 1

- IF
- IF-ELSE
- Logic: AND, OR, NOT (conceptual and Google search examples)

James Tam

Recap: Programs You've Seen So Far Produces Sequential Execution

Programs you have seen thus far:
- Execute statement after statement one after the other from start to finish.
- There are no options for alternatives (branch in execution).
- Nor are there options to repeat a portion.



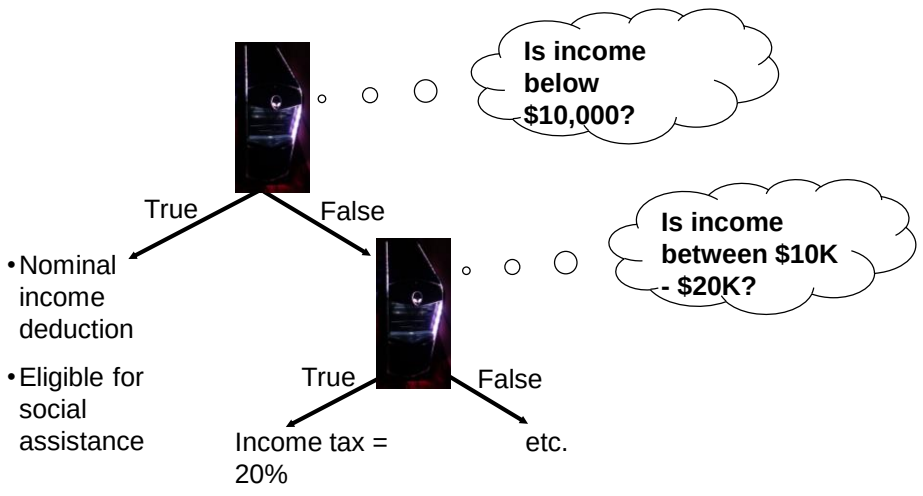
James Tam

Programming: Branching

- 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 an answer that is either true or false
  - Although the approach is very simple, modeling branching in this fashion is a very useful and powerful tool.
- New terminology
  - These true/false questions are referred to "**Boolean expressions**"
    - e.g., **it is over 45 Celsius today**
    - e.g., **the user correctly entered the password**
    - Python code example: `if(income < 10000):`

James Tam

High Level View Of Branching For The Computer



James Tam

How To Determine If Branching Can Be Applied

- When a **condition is true** **action(s)** (programming instructions) will be taken by a program.
- Examples:
  - **If an employee is deemed as too inexperienced and too expensive** to keep on staff then **person will be laid off**.
  - **If a user enters invalid age information** (say negative values or values greater than 114) then the **program will display an error message**.

Branching In Programming (Python)

- Branches 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).
- Branching structures 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)

New Terminology

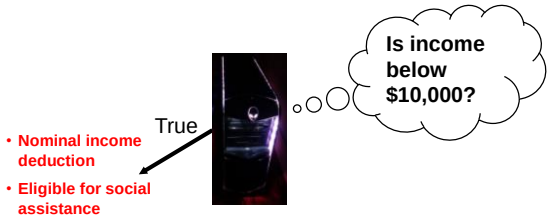
- **New term, body**: A block of program **instructions that will execute when a Boolean expression** evaluates to/works out to **true**)
  - Body (of a python program)

```
name=input("Name: ")
print(name)
```

This/these instruction/instructions run when you run a python program. The whole program (the whole body) executes.

```
if(True):¶
····print("foo")¶
····¶
```

  - **Body of a branch** (sub-part of a program).
    - Specified with indenting in python (4 spaces)
    - Don't use tabs (tabs won't consistently indent across computers/programs)
    - IDLE typically adds the indenting (using spaces) for you automatically.




```
if(income < 10000):
    deduction = 1.00
    print("Assistance")
```

James Tam

New Terminology

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

Math Examples:

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

Relational logic examples (produce a Boolean result)

```
x > 2
username == "tam" (The "equals-equals" operator
checks if the expression on the left and right are
equal)
```

James Tam

Allowable Operands For Boolean Expressions

Format:

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

Example:

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

Some operand types

- Integer e.g. `age = 12`
- floats (~real) e.g. `height = 68.5`
- String e.g. `name = "Tam"`
- Boolean (True or False) E.g. `gameWon = False`

Make sure that you are comparing operands of the same type or at the very least they must be comparable (e.g. integer and float comparison is okay, integer and string is not)!

James Tam

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

James Tam

Note On Indenting

- In Python indenting is mandatory in order to determine which statements are part of a body (**syntactically required** in Python).
- A body with multiple instructions simply requires all those statements to be indented.
  - Typical style requirements specify indenting of 4 spaces.
- Single statement body

```
if(age == 0):
    print("'Gratz' it's your birthday!")
```

- Multi-statement body

```
if(age == 0):
    print("'Gratz' it's your birthday!")
    print("Many happy returns.")
```

James Tam

Note On Indenting (2)

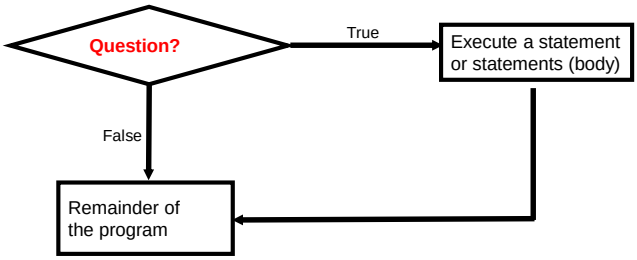
- A "sub-body" (IF-branch) is indented by an additional 4 spaces (8 or more spaces) if one IF-branch is inside the body of another IF-branch (this is called 'nesting' – more details later).
  - Example (just for reference for now, details come later)

```
if(BE 1):
    first body #4 spaces
    if(BE2):
        second (nested body) #4 spaces + 4 spaces
```

- Again you should **NOT use tabs** for indenting what looks neatly and consistently indented with one editor or operating system could be a mess in other cases:
  - If you write programs on different platforms (e.g. using a UNIX editor in the lab and Notepad at home).
  - When your marker views your assignment or project.

James Tam

Branching With An 'If'



James Tam

The 'If' Structure

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

```
(General format)
 if(Boolean expression):
     body
(Detailed structure)
if(<operand> <relational operator> <operand>):
     body
```

Also note: the colon is mandatory.

Boolean expression

Reminder: Indenting the body is mandatory!

James Tam

The 'If' Structure (2)

- **Example (1if1.py):**

  Learning objective of example: program executes a statement when a Boolean expression evaluates to true.

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

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 (at least display an error message)
```

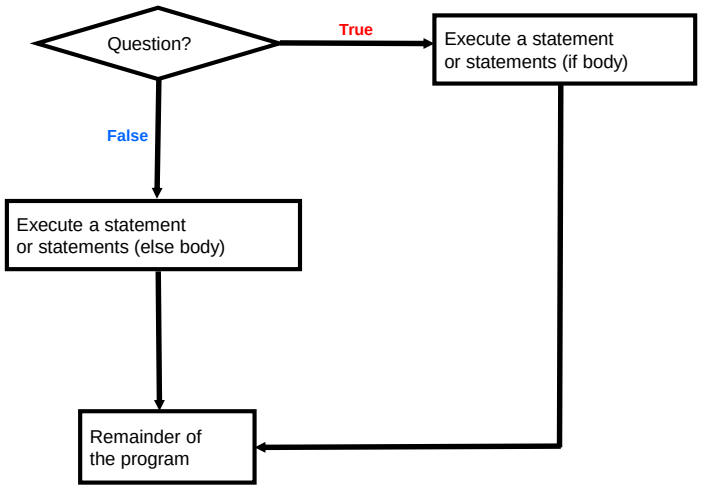
- **Example**:

```
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 of range before proceeding onto the rest of the program.

James Tam

Branching With An 'If-Else'



James Tam

The If-Else Structure

- Branching: 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 Structure (2)

- **Program name:** `2if_else1.py`
  - Learning objective of example: program executes one body when a Boolean expression evaluates to true and another when it's false.
- **Partial 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                ← If case
Tell me more about yourself
[csc branches 14 ]>
[csc branches 14 ]> python if_else1.py
How old are you? 27
Adult                       ← Else case
Tell me more about yourself
[csc branches 15 ]>
```

James Tam

If-Else Example

- **Program name:** `3if_else2.py`
  - Learning objective of example: defining the bodies of an IF-case and an ELSE-case with multiple statements.
    - Make sure you properly indent the statements that are a part of the body.
- **Partial 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

Logical Operations

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

James Tam

Online Searches, Multiple Words: All Words Must Appear In Results

- For instance you not only want to search for 'CPSC' courses but also ones with the course number '203' in 'Calgary'
- Format (searches for all words):
  - <*First word*> <*Second word*> (There is an implicit AND, but there is no need include a specific 'AND' operator with Google)
- Example

```
CPSC 203 Calgary
```

James Tam

Logical AND

- The popular usage of the logical AND applies when *ALL* conditions must be met.
- Logical AND can be specified more formally in the form of a truth table in order to cover all cases of true/false.

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

James Tam

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

- In class:
  - False **AND** True **AND** True
- Extra for you to do:
  - True **AND** True **AND** True
  - True **AND** True **AND** True **AND** False

James Tam

Searching Among Alternatives

- As you just saw when multiple words are typed into the search box Google will try to find web pages that include **all** of those words.
- To specify that each word (or phrase) is an alternative separate each search criteria using the **OR** operator (capitalization is important).
- Example (searches include alternatives):

```
cute wallpapers cats OR dogs
"Bruce Lee" OR "Little Dragon" OR "Lee Siu Lung"
"Wayne Gretzky" OR "The Great One" OR "Number 99" OR
 "Number ninety nine"
```

James Tam

Logical OR

- The correct everyday usage of the logical OR applies when *ATLEAST* one condition must be met.

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

- In class:
  - False **OR** True **OR** True
- Extra for you to do:
  - True **OR** True **OR** True
  - False **OR** False **OR** False **OR** True

James Tam

Excluding Words

- Return search results excluding (or "not including" specified words)
- There may be times when you want Google to exclude from results certain words or phrases.
- This can be done with the **subtraction operator** (subtract the words that follow the operator from search results).
- Example:

  "James Tam"

  Vs.

  "James Tam" -Calgary
- An alternate approach is to use the 'NOT' operator

  "James Tam" **NOT** Calgary

James Tam

Logical NOT

- The everyday usage of logical NOT negates (or reverses) a statement.
- The truth table for logical NOT is quite simple:

Truth table	
C	**Not C**
False	True
True	False

James Tam

Evaluating More Complex Logical Expressions

- Order of operation (left to right evaluation if the 'level' is equal)
  1. Brackets (inner first)
  2. Negation
  3. AND
  4. OR

James Tam

Evaluating More Complex Logical Expressions

- In class:
  - True **OR** False **AND** False
  - (True **OR** False) **AND** False
  - **NOT** False
  - **NOT NOT** False
- Extra for you to do:
  - **NOT** (False **OR** True) **OR** True
  - (False **AND** False) **OR** (False **AND** True)
  - **NOT NOT NOT NOT** True
  - **NOT NOT NOT** False

James Tam

Student Exercise: 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

Precedence: Order Of Operation

- Table from the lecture notes of Richard Zhao and Jonathan Hudson.
- Determining order from the operators you should now know.

Order	Operations	Precedence
1	`()`	Highest
2	`x ** y`	
3	`-x, +x`	
4	`x * y, x / y, x % y, x // y`	
5	`x + y, x - y`	
6	`<, <=, >, >=`	
7	`!=, ==`	
8	`not`	
9	`and`	
10	`or`	
11	`=`	Lowest

James Tam

Constructing An Arbitrary Truth Table

- Given some description of a logical expression you need to be able:
  - Construct the truth table (implementation in class notes)
  - Write a python statements with equivalent implementation (exercise for students to do)
- The following truth table example comes from the lecture notes of Richard Zhao and Jonathan Hudson

James Tam

Boolean Logic

- Example:
  - Construct a truth table for A and (B or not C):

UNIVERSITY OF CALGARY

Boolean Logic

- Example:
  - Construct a truth table for **A and (B or not C)**:

A	B	C
F	F	F
F	F	T
F	T	F
F	T	T
T	F	F
T	F	T
T	T	F
T	T	T

UNIVERSITY OF CALGARY

Copyright © 2024. Do not distribute outside of the CPSC 231 Fall 2024 class.

Boolean Logic

- Example:
  - Construct a truth table for **A and (B or not C)**:

A	B	C	not C
F	F	F	T
F	F	T	F
F	T	F	T
F	T	T	F
T	F	F	T
T	F	T	F
T	T	F	T
T	T	T	F

UNIVERSITY OF CALGARY

Boolean Logic

- Example:
  - Construct a truth table for **A and (B or not C)**:

A	B	C	not C	B or not C
F	F	F	T	T
F	F	T	F	F
F	T	F	T	T
F	T	T	F	T
T	F	F	T	T
T	F	T	F	F
T	T	F	T	T
T	T	T	F	T

UNIVERSITY OF CALGARY

Boolean Logic

- Example:
  - Construct a truth table for **A and (B or not C)**

A	B	C	not C	B or not C	A and (B or not C)
F	F	F	T	T	F
F	F	T	F	F	F
F	T	F	T	T	F
F	T	T	F	T	F
T	F	F	T	T	T
T	F	T	F	F	F
T	T	F	T	T	T
T	T	T	F	T	T

Python Implementation

- Your practice exercise is to form the equivalent logical expression for *A and (B or not C)*, you don't have to literally draw out entire truth table (which is of little value).
- To that end you have a program which sets 3 Boolean variables to some starting values.
- Starting program:
- Both the starting program and the solution can be found in the same location in D2L & the course website under the 'exercise' link.

One way to initialize the 3 Booleansz

```
answer = input("t or f for 1st Boolean 'A': ")
if((answer=="T")or(answer=="t")):
    booleanA = True
elif((answer=="F")or(answer=="f")):
    booleanA = False
else:
    errorState = True

answer = input("t or f for 2nd Boolean 'B': ")
if((answer=="T")or(answer=="t")):
    booleanB = True
elif((answer=="F")or(answer=="f")):
    booleanB = False
else:
    errorState = True

answer = input("t or f for 3rd Boolean 'C': ")
if((answer=="T")or(answer=="t")):
    booleanC = True
elif((answer=="F")or(answer=="f")):
    booleanC = False
else:
    errorState = True

if(errorState==True):
    print("End program: Entered a non-Boolean value
```

Learning objective: defining these in python

not C	B or not C	A and (B or not C)

James Tam

Copyright Notification

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