

## Branching In Python: Part 1

- IF
- IF-ELSE
- Logic: AND, OR, NOT

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.

```
print ("This program will calculate the area of a  
rectangle")  
length = int(input("Enter the length: "))  
width = int(input("Enter the width: "))  
area = length * width  
print("Area: ", area)
```

Start

End

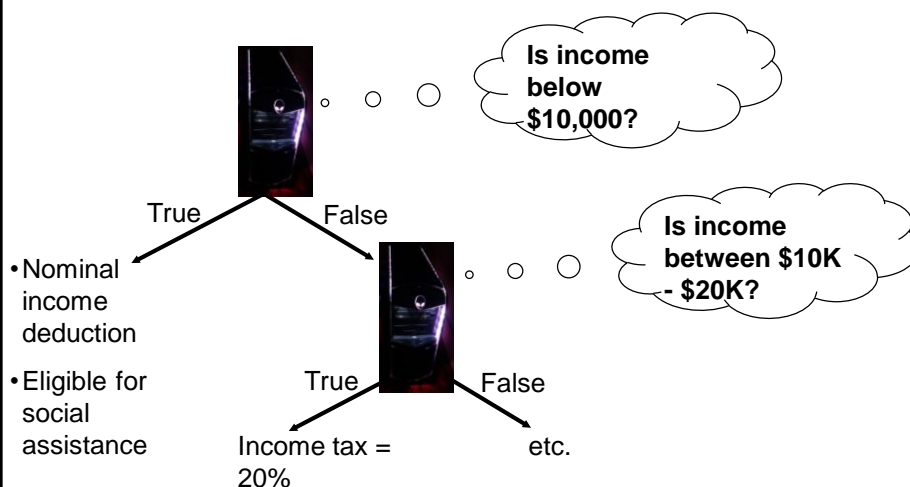
James Tam

## 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

- Under certain **conditions** one or more **actions** (programming instructions) will be taken by a program.
- Examples:
  - If users who don't meet the age requirement of the website the user will not be allowed to sign up (conversely if users do meet the age requirement the person will be allowed to sign up).
  - If an employee is deemed as too inexperienced and too expensive to keep on staff then person 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.

slide 5

James Tam

## 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)

James Tam

## New Terminology

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

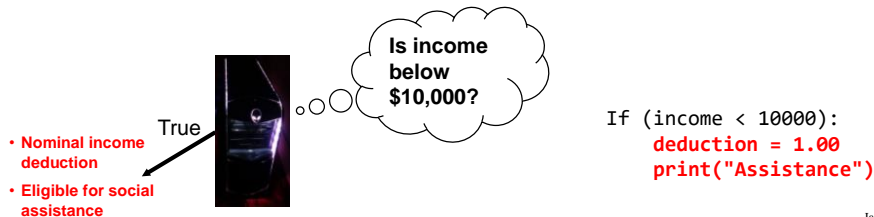
- Body (of a python program)

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

This/these instruction/instructions run when you give the Python interpreter the name of a file, the 'body' of the Python program runs

- **Body of a branch:**

- Specified with indenting in python (4 spaces)
- Don't use tabs (tabs won't consistently indent across computers/programs)
- IDLE typically adds the indenting for you automatically (another 'plus')



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

## Note On Indenting

- With many programming languages indenting is part of good programming style.
- In Python indenting is mandatory in order to determine which statements are part of a body (**syntactically required**<sup>1</sup> in Python).
- A body with multiple instructions simply requires all those statements to be indented.

- 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.")
```

<sup>1</sup> Recall that the syntax of a programming language is analogous to a grammar of human language

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

## 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

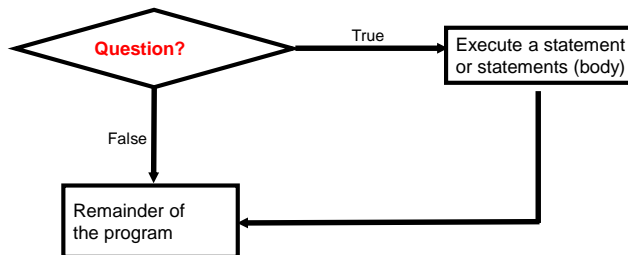
## Allowable **Relational Operators** For Boolean Expressions

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

## 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
```

Boolean expression

Note: 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

## Common Mistake

- Do not confuse the equality operator '==' with the assignment operator '='.

- Mistake: using a single equals sign instead of two when forming a Boolean expression.

- **Example (Python syntax error)<sup>1</sup>:**

```
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).

- A way of producing syntax rather than a logic error:

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

<sup>1</sup> 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 syntax error, used to be a logic error):**

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

- Mistake: including two equals signs instead of one when trying to assign a value to memory location.

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.

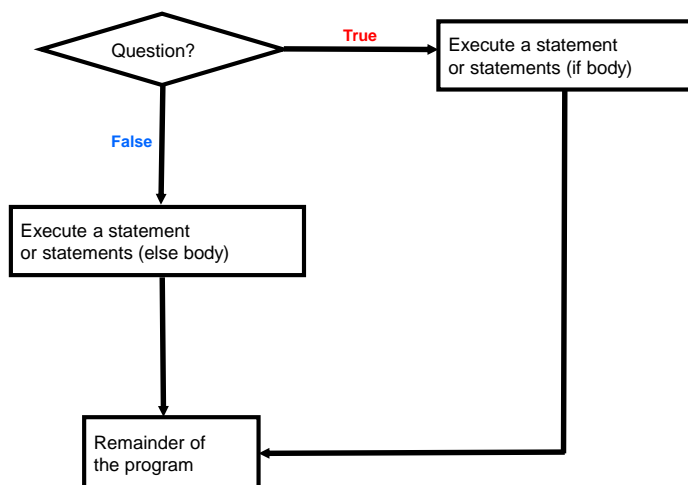
James Tam

## Branching With An 'If': 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 structure).
  - 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 structure).

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

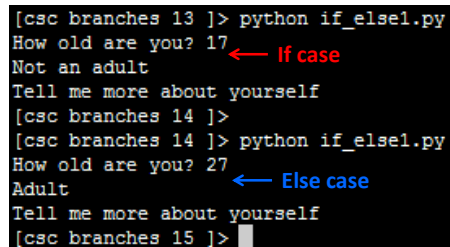
## If-Else Structure (2)

- 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 ← If case  
Not an adult  
Tell me more about yourself  
[csc branches 14 ]>  
[csc branches 14 ]> python if_else1.py  
How old are you? 27 ← Else case  
Adult  
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**.

- **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

## 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

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
<i>True</i>	<i>True</i>	<i>True</i>

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

## 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
<i>False</i>	<i>False</i>	<i>False</i>
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
<i>False</i>	<i>False</i>	<i>False</i>	<i>False</i>
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

## 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 (2<sup>nd</sup> 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

## After This Section You Should Now Know

- What are the three branching mechanisms available in Python:
  - If
  - If-else
  - How does each one work
  - When should each one be used
- Introducing three logical operations
  - AND
  - OR
  - NOT

James Tam

## **After This Section You Should Now Know (2)**

- How the bodies of the branching structures are defined:
  - What is the body of a branching structure
  - What is the difference between branching 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

James Tam

## **Copyright Notification**

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

slide 38

James Tam