Python for Data Management and Analytics

Unit 2

- Definitions
- Machine Learning
- Limitations
- Process
- Application to Healthcare

Python for Data Management and Analytics

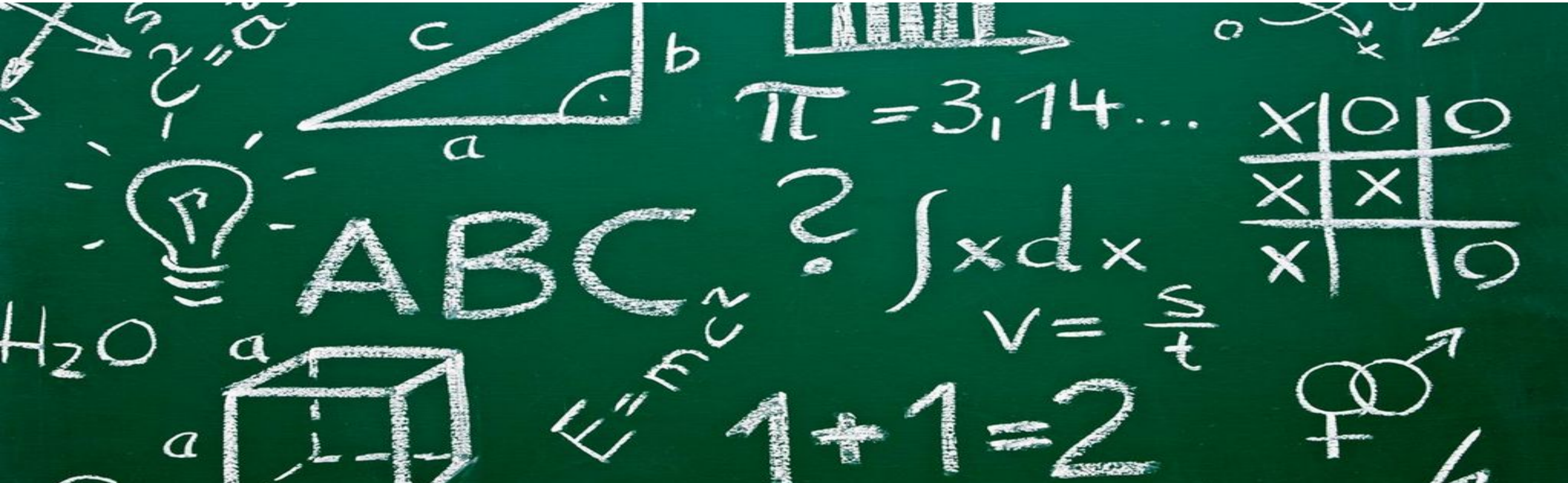
Slides By: Dmitriy Babichenko
Presented By: Ravi Patel

- Variables
  - Data Types
  - Naming Conventions
  - Type Casting
- Arithmetic Operations
- Strings
  - Operations
  - String-number conversions
- Basic Input/Output
- Control structures / conditional statements

Programming v/s math

```
x = 1
```

- In math this means x is equal to 1
- In programming this means the value 1 is stored in variable x



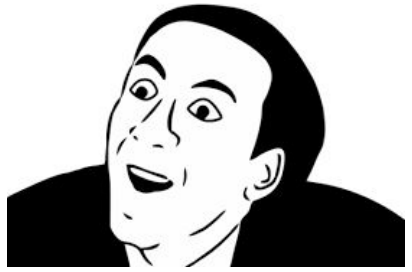
Programming v/s math

```
x = x + 1
```

- In math this is wrong
- In programming this is perfectly valid

Variables in Python

- A **Variable** is a **SPACE in memory** where a **VALUE** can be stored
- The **VALUE** can change - so it is *variable* ;)
- In Python variable CAN change it's **TYPE**



```
x = 5  # x is a number
x = "Hello" # x is a string
```

Memory Allocation – a metaphor

40.432392,-79.922378
(5818 Phillips Avenue, Pittsburgh, PA 15217)

Memory Allocation – a metaphor

this is the *address* of a specific SPACE on earth

who lives in this space is the value stored

if more people moves in, then the space will not be enough

40.432392,-79.922378
(5818 Phillips Avenue, Pittsburgh, PA 15217)

Python Datatypes

- **Numbers**
- **String - Text**
- **Boolean - True, False**
- Dates
- Binary Data
- List
- Tuple
- Dictionary

Declaration and Initialization

- **Declaration:** set type, creates a variable (allocates space in memory)
- **Initialization:** assigns initial value
- In Python, a variable CANNOT be declared without being initialized

A statement like this does both actions in one line:

```
x = 5;
```

Declaration **Initialization**

$x = 5$

The equal sign ***assigns*** the value of the right side to the variable in the left side

Changing the value

- John lives in this house
- John moves out, Amy moves in
- Amy moves out, Shawn moves in

Changing the value

```
x = 5
```

Value of variable “x” is “5”

```
x = 7
```

Now, value of variable “x” is “7”

```
x = x * 2
```

What is the value of “x” now?

id	chol	stab.gl	hdl	ratio	glyhb	location	age	gender	height	weight	frame	bp.1s	bp.1d	waist
1000	203	82	56	3.6	4.31	Buckingham	46	female	62	121	medium	118	59	29
1001	165	97	24	6.9	4.44	Buckingham	29	female	64	218	large	112	68	46
1002	228	92	37	6.2	4.64	Buckingham	58	female	61	256	large	190	92	49
1003	78	93	12	6.5	4.63	Buckingham	67	male	67	119	large	110	50	33
1005	249	90	28	8.9	7.72	Buckingham	64	male	68	183	medium	138	80	44
1008	248	94	69	3.6	4.81	Buckingham	34	male	71	190	large	132	86	36
1011	195	92	41	4.8	4.84	Buckingham	30	male	69	191	medium	161	112	46
1015	227	75	44	5.2	3.94	Buckingham	37	male	59	170	medium			34
1016	177	87	49	3.6	4.84	Buckingham	45	male	69	166	large	160	80	34
1022	263	89	40	6.6	5.78	Buckingham	55	female	63	202	small	108	72	45
1024	242	82	54	4.5	4.77	Louisa	60	female	65	156	medium	130	90	39
1029	215	128	34	6.3	4.97	Louisa	38	female	58	195	medium	102	68	42

Link to complete diabetes dataset: https://drive.google.com/file/d/1SXhHEsiqbU0HVtgTXmH9f9QIw7g7TjMX/view?usp=sharing

Naming Conventions

- Variable names are **case-sensitive**.
- The start character can be the underscore "_" or a capital or lower case letter.
- The letters following the start character can be anything which is permitted as a start character plus the digits.
- An uppercase variable typically represents variables are anticipated to be contant (ex. MY_FAVORITE_NUMBER = 6). This allows you to declare a constant value without having to use the "hard code" throughout.

Naming Conventions

- Python variables are named using lowercase words
- Python variables whose names consist of multiple words should have the words separated with an underscore.
    - Ex: num_of_doors, vehicle_color, patient_weight
- Complete Python coding style guide: https://www.python.org/dev/peps/pep-0008/

Reserved Keywords

No identifier can have the same name as one of the Python keywords:

and, as, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while, with, yield

Typecasting

- Typecasting refers to explicitly **convert** one data type to another

- Variables
  - Data Types
  - Naming Conventions
  - Type Casting
- Arithmetic Operations
- Strings
  - Operations
  - String-number conversions
- Basic Input/Output
- Control structures / conditional statements

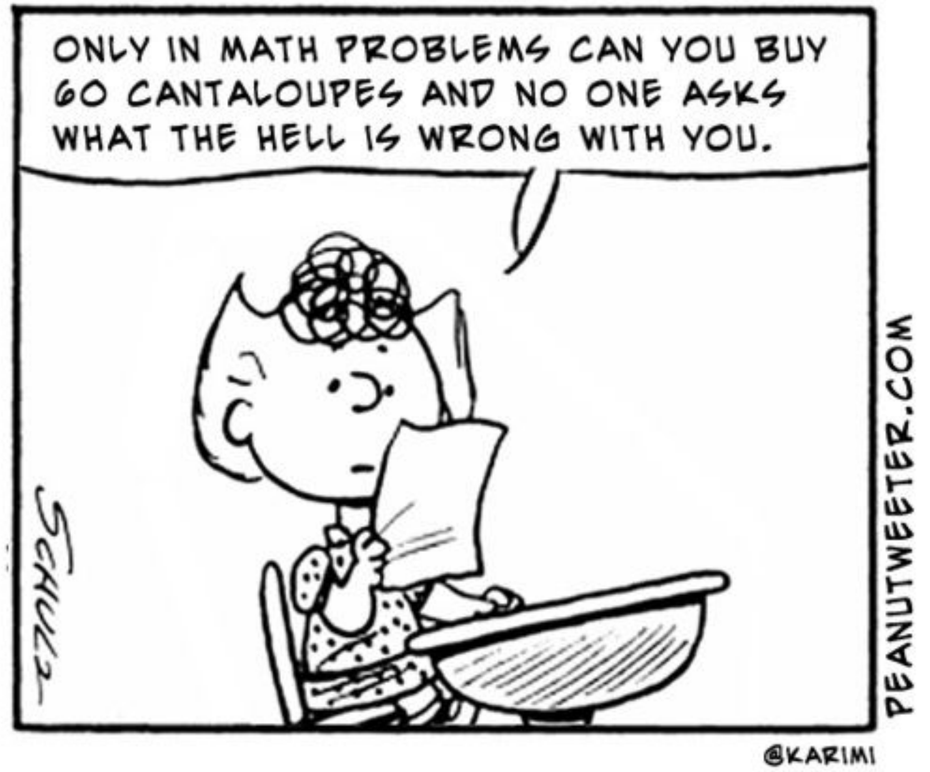
Arithmetic Operators

Five basic operators:

- addition: +
- subtraction: -
- multiplication: *
- division: /
- power: **

One weird operator:

- modulus: %


ONLY IN MATH PROBLEMS CAN YOU BUY 60 CANTALOUPES AND NO ONE ASKS WHAT THE HELL IS WRONG WITH YOU.
SCHULZ
PEANUTWEETER.COM
@KARIMI

Arithmetic Expressions

A combination of operators, values and variables is an EXPRESSION

$$(a + b) / 2$$

Order of Operations

- Parentheses control the order of the computation

(a + b) / 2

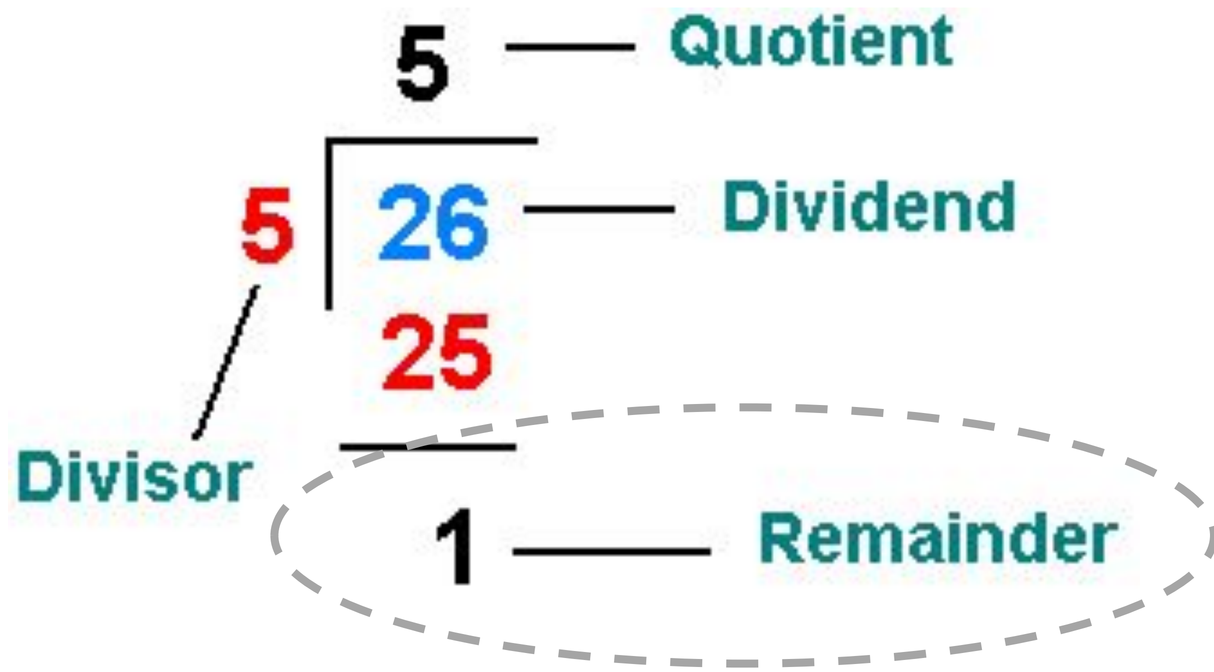
- Multiplication, division and modulus have a higher precedence than addition and subtraction

a + b / 2

Modulus

- Use % operator to get the remainder with (pronounced "modulus", "modulo", or "mod")

7 % 4   is   3



For other operations, use the **math** module

- **math** module contains ***functions*** for common mathematical operations

  y = math.sqrt(4) *https://www.tutorialspoint.com/python/number_sqrt.htm*

  r = math.round(5.743, 2) *https://www.tutorialspoint.com/python/number_round.htm*

- For full **math** class documentation, see **Python API** documentation: https://docs.python.org/3/library/math.html

Pythagorean Theorem Exercise

- **Pythagorean Theorem Example**: https://pitt.box.com/s/ot6yexetia3g0w3x0vljuduoa1315j8e
- **Pythagorean Theorem Challenge**: Modify the program to accept input for the lengths of one adjacent side and the hypotenuse of a right triangle. Calculate the second adjacent side.

- Variables
  - Data Types
  - Naming Conventions
  - Type Casting
- Arithmetic Operations
- Strings
  - Operations
  - String-number conversions
- Basic Input/Output
- Control structures / conditional statements

String Variables

- A string is a sequence of characters
- A string variable is a variable that can hold text
- You can declare variables that hold strings

```
name = "Harry"
```

this is a string variable

this is a string literal

String Type

- String is NOT a primitive data type, but a class
- A String variable has special functionalities attached to it

String Length

- The **len()** function yields the number of characters in a string

```
first_name = "Harry"
n = len(first_name)
```

- A string of length 0 is called the ***empty*** *string*
  - Contains no characters
  - Is written as "" (no space within)

```
first_name = ""
```

first_name = " "
is not the same as
first_ame = "";

Concatenation

- **Concatenating strings** means to put them together to form a longer string
- Use the + operator:

```
first_name = "Walter"
last_name = "White"
full_name = first_name + " " + last_name
```

Result: "Walter White"

Note about Operations and Types

The type also determines how certain operations perform

```
x = 5
x = x + 3


first_name = "Peter"
last_name = "Pan"


name = first_name + last_name
```

operator + do different things depending on the variable (number or String)

Concatenate Strings & Numbers

```
string1 = “Hello, my name is “
string 2 = “Bob”
print(string1 + string2)
```

```
string1 = “My age is “
age = 40
```
~~print(string1 + age)~~
```
print(string1 + str(age))
```

This variable is an integer - concatenating a string and a number will generate an error!

str() function allows us to convert other **datatypes** into **strings**

Concatenation in Print Statements

Useful to reduce the number of *print()* instructions

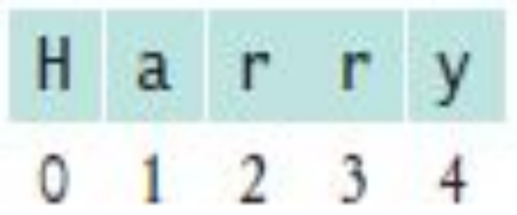
```
print("The total is ")
print(total)
```

versus

```
print("The total is " + str(total))
```

Character Positions Within Strings

- String positions are counted starting with 0.
- The position number of the last character is always one less than the length of the string.



The last character of the string "Harry" is at position 4

- Indexes allow us to grab individual characters or substrings from a string

name = "Harry"

start _character = name[0] # H

last_character = name[4] # y

char_sequence = name[1:2] # “ar”

Slicing Strings

It's important to remember that indexing is zero-based; that is, the first item in the sequence is number 0.

```
s = 'Don Quijote'
>>> s[4]  # Get the 5th character
'Q’
```

Slicing Strings

- If you want to start counting from the end of the string, instead of the beginning, use a negative index.
- For example, an index of -1 refers to the right-most character of the string.

```
>>> s[-1]
'e'
>>> s[-7]
'Q'
```

Slicing Strings

```
>>> s = 'Don Quijote'
>>> s[4:8]
'Quij'
>>> s[4:]
'Quijote' # Returns from pos 4 to the end of the string
>>> s[:4]
'Don ' # Returns from the beginning to pos 3
>>> s[:]
'Don Quijote'
```

BMI Calculator Exercise

Body mass index (BMI) is a measure of body fat based on height and weight that applies to adult men and women.

BMI Categories:

- Underweight = <18.5
- Normal weight = 18.5 - 24.9
- Overweight = 25 - 29.9
- Obesity = BMI of 30 or greater

English BMI Formula (Imperial)

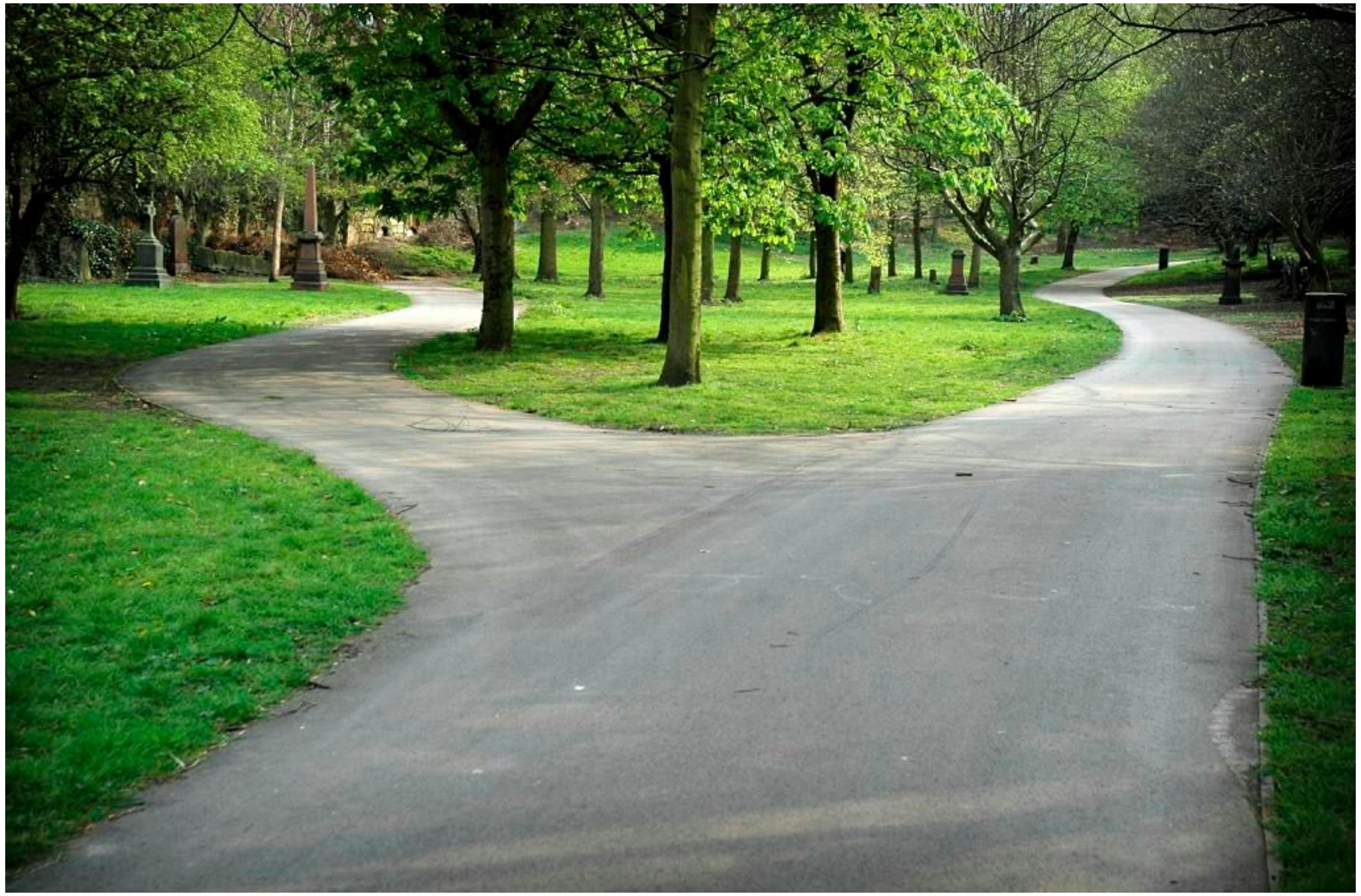
BMI = (Weight in Pounds / (Height in inches x Height in inches)) x 703

BMI Calculator Exercise

- BMI calculator implemented example: https://pitt.box.com/s/t4hbi457bdrkp2ti087be5uvv4xacooo
- BMI calculator challenge 1: BMI results are usually displayed as integers (values without decimal points). Modify this program to round the BMI result to the nearest integer.

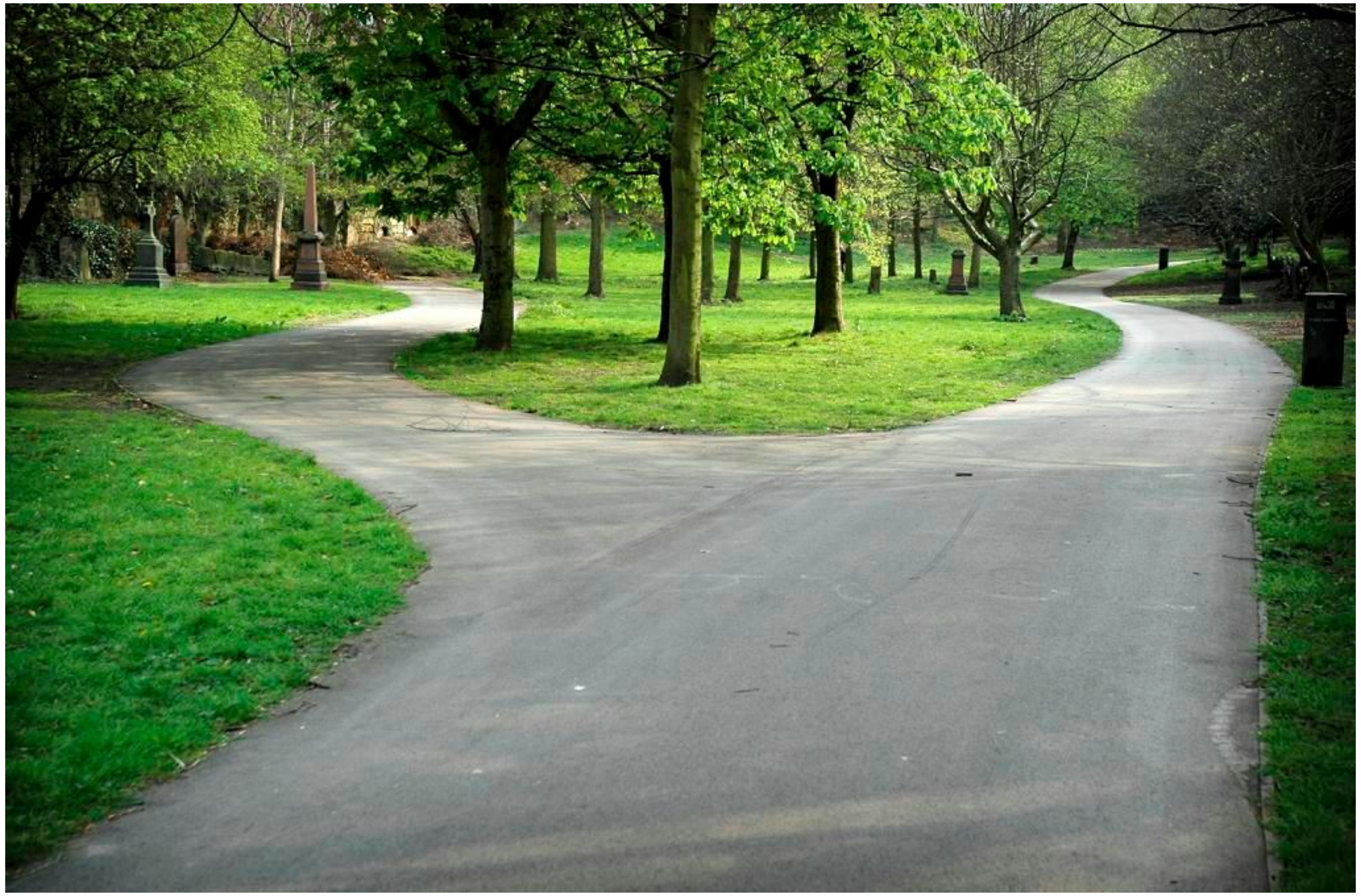
- Variables
  - Data Types
  - Naming Conventions
  - Type Casting
- Arithmetic Operations
- Strings
  - Operations
  - String-number conversions
- Basic Input/Output
- Control structures / conditional statements

The *if* Statement



An if statement is like a fork in the road. Depending upon a decision, different parts of the program are executed.

The *if* Statement



An if statement is like a fork in the road. Depending upon a decision, different parts of the program are executed.

The *if* statement

- ***if*** is a statement, and *reserved word*
- The general form is:

```
if condition true or false:
    # executed when condition is true
```

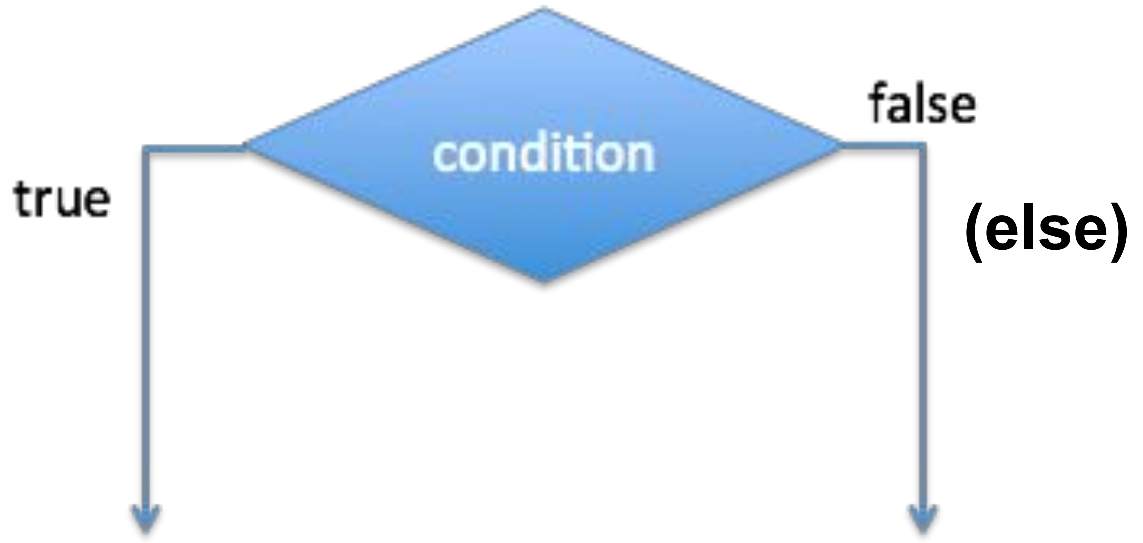
```
if x < 0:
   print("negative number");
```

*but, what "**if not**"?* ***the else statement***

Else statement creates an alternative branch

The general form is:

```
if condition true or false:
    # executed when condition is true
else:
    # executed when condition is false
```



```
if x < 0:
   print("negative number")
else:
   print("positive number")
```

A condition that’s either true or false – a **Boolean Expression**

```
if  x > 1:
    x = x * x
else:
    x = x / 2
```

If the condition is true, the statement(s) in this branch are executed in sequence; if the condition is false, they are skipped.

Omit the *else* branch if there is nothing to do

If the condition is false, the statement(s) in this branch are executed in sequence; if the condition is true, they are skipped.

Relational Operators in Python

Python	Math Notation	Description
>	>	Greater than
>=	≥	Greater than or equal
<	<	Less than
<=	≤	Less than or equal
==	=	Equal
!=	≠	Not equal

Even/Odd Example

Using an if statement and the modulus (%) operator write a program that asks users to enter a number and determines whether the number entered is **even** or **odd**.

- Ask user an input
- Convert input to number
- if reminder of number / 2 is 0, print is ***even***, otherwise print ***odd***

Equivalent conditions

```
if age >= 21:
    print("CAN buy beer")
else:
    print("CAN'T buy beer")
```

```
if age < 21:
    print("CAN'T buy beer")
else:
    print("CAN buy beer");
```

But, be careful, sometimes this can induce logic error

Lucky Number Exercise

Many cultures consider number 7 to be a lucky number. Write a Python program that takes a numeric input from a user and checks if the input is a "lucky" number.

Full implementation example:

https://pitt.box.com/s/4g5lm8h2h483zqs53wk19rd3xbp3gwut

Lucky Number Exercise

Challenge 1:

Some users will try to submit a blank input. When user submits input without entering a value, input string will be empty, or equal to a blank string (""). Make sure to validate user inputs

Lucky Number Exercise

Challenge 2:

In Italy number **17** is also considered unlucky. The unluckiness of seventeen in Italian culture dates back to the Roman times. Seventeen in Roman numnerals is XVII, which is an anagram for VIXI, which is Latin for "I Lived" and is a common marking on Roman tombstones. Modify the program below to not only check for lucky number 7, but also for unlucky numbers 13 and 17 and to display appropriate messages.

Equivalent conditions

Referring to Even/Odd Example, would it make any difference if your if statement is checking for remainders of 0 or remainders of 1?

```
if(number % 2 == 0){
System.out.println("Even");
}else{
System.out.println("Odd");
}
```

```
if(number % 2 == 1){
System.out.println("Odd");
}else{
System.out.println("Even");
}
```

The second version gives errors for negative numbers!

Assignment v/s comparison

= v/s ==

boolean expressions

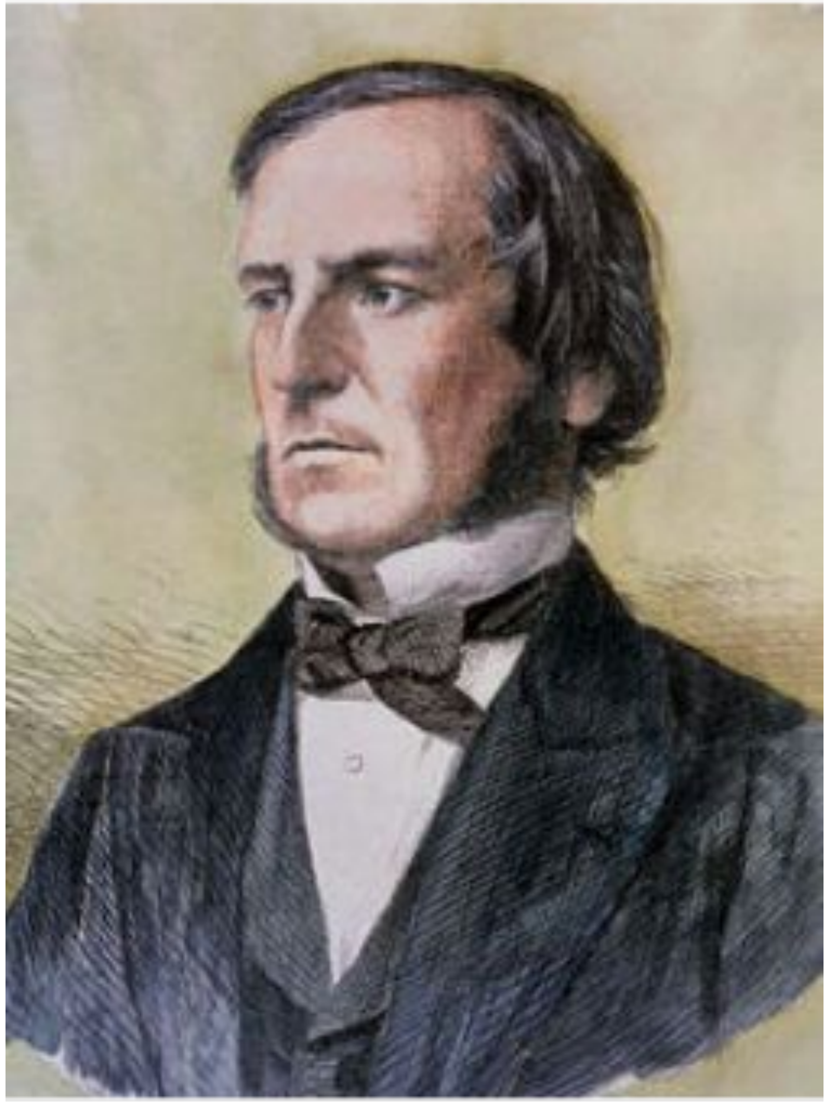
A boolean expression results in either ***true*** or ***false*** value

Comparison operators

- equal, greater than, not equal, etc.
- compare any type (numbers, strings, etc)

Boolean operators

- And, Or, Not



George Bool
Founder of Boolean Algebra
http://en.wikipedia.org/wiki/George_Boole

10 > 4	**true**
10 > 20	**false**
0 < -2	**false**
'a' == 'a'	**true**

Exercise

Which of the following conditions are true, provided a = 3 and b = 4?

```
a + 1 <= b

a + 1 >= b

a + 1 != b
```

The type ***boolean***

- 2 possible values (or states):

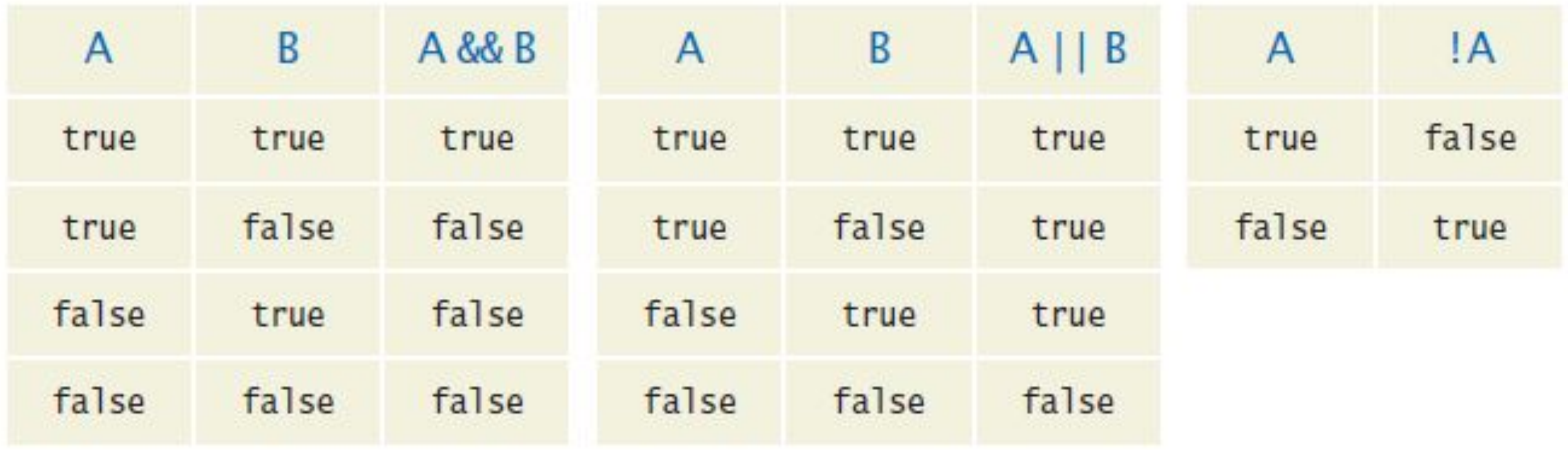
  True (1, yes, positive)

  False (0, no, negative)

Boolean Variables and Operators

- You often need to combine Boolean values when making complex decisions
- An operator that combines Boolean conditions is called a Boolean operator.
- The **and** operator Yields true only when both conditions are true.
- The or operator
  - Yields the result true if at least one of the conditions is true.

Boolean Truth Tables



A	B	A && B
true	true	true
true	false	false
false	true	false
false	false	false

A	B	A || B
true	true	true
true	false	true
false	true	true
false	false	false

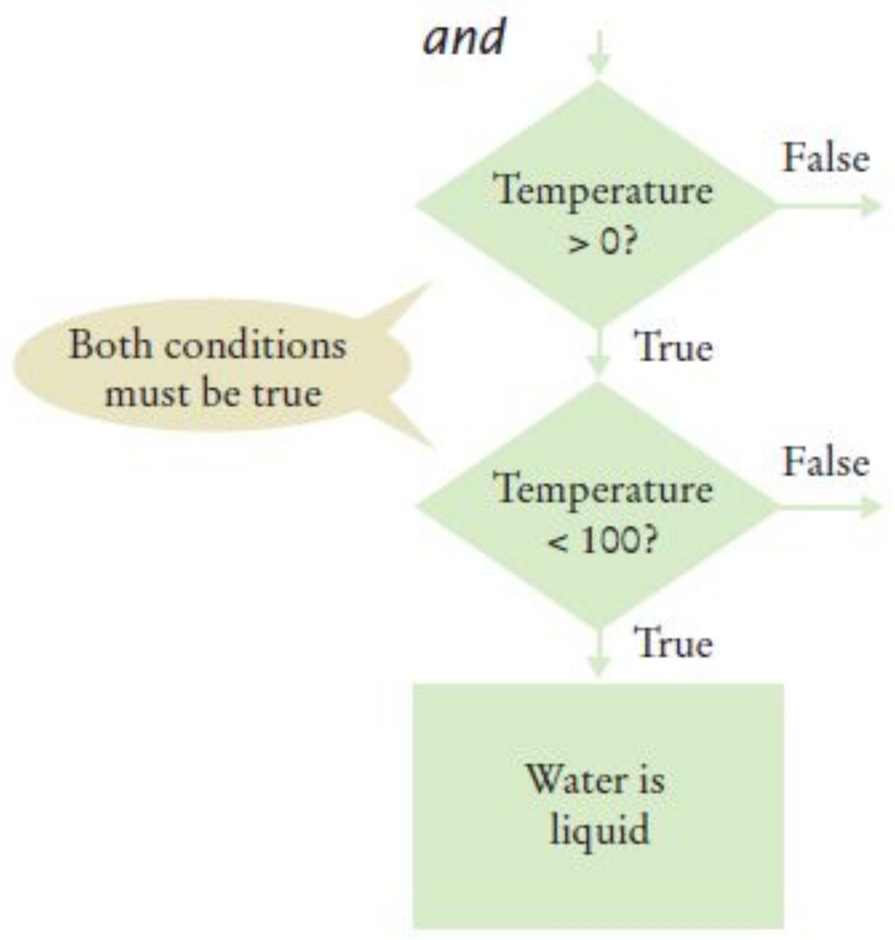
A	!A
true	false
false	true

Figure 9 Boolean Truth Tables

Boolean Variables and Operators

To test if water is liquid at a given temperature

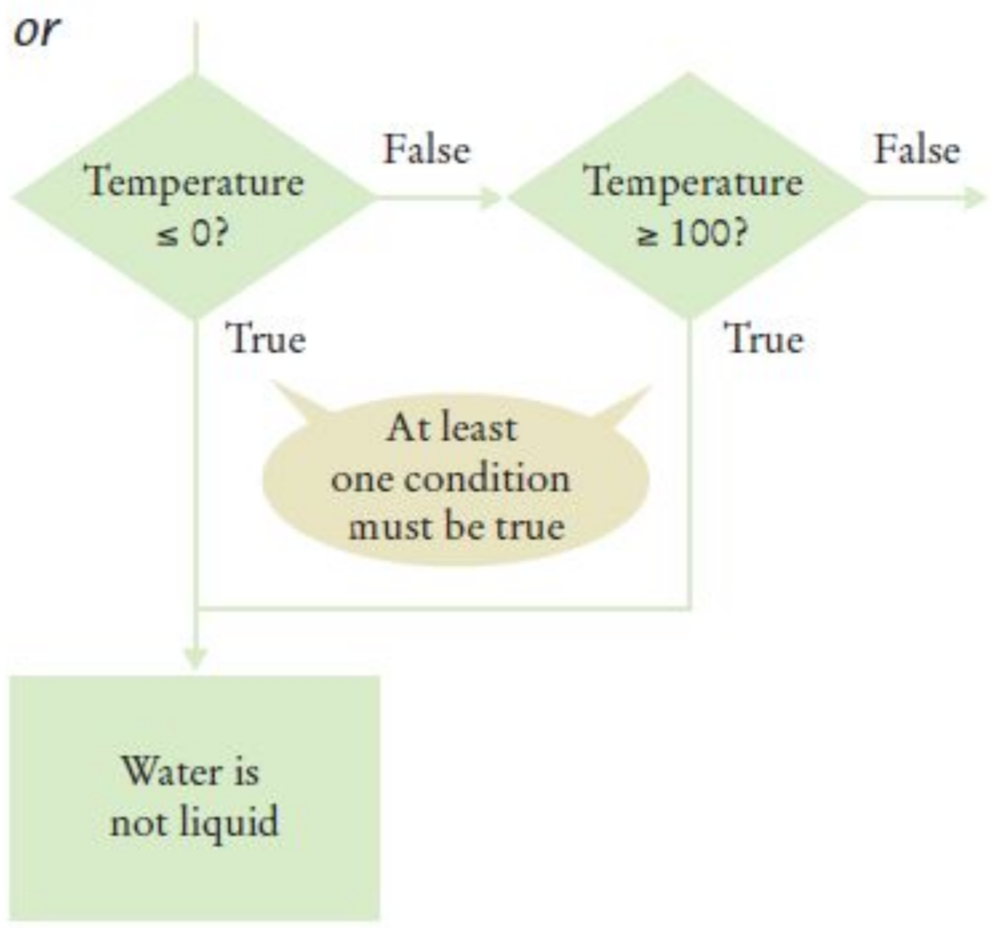
```
if temp > 0 AND temp <100:
    print ("Water is liquid")
```



Boolean Variables and Operators

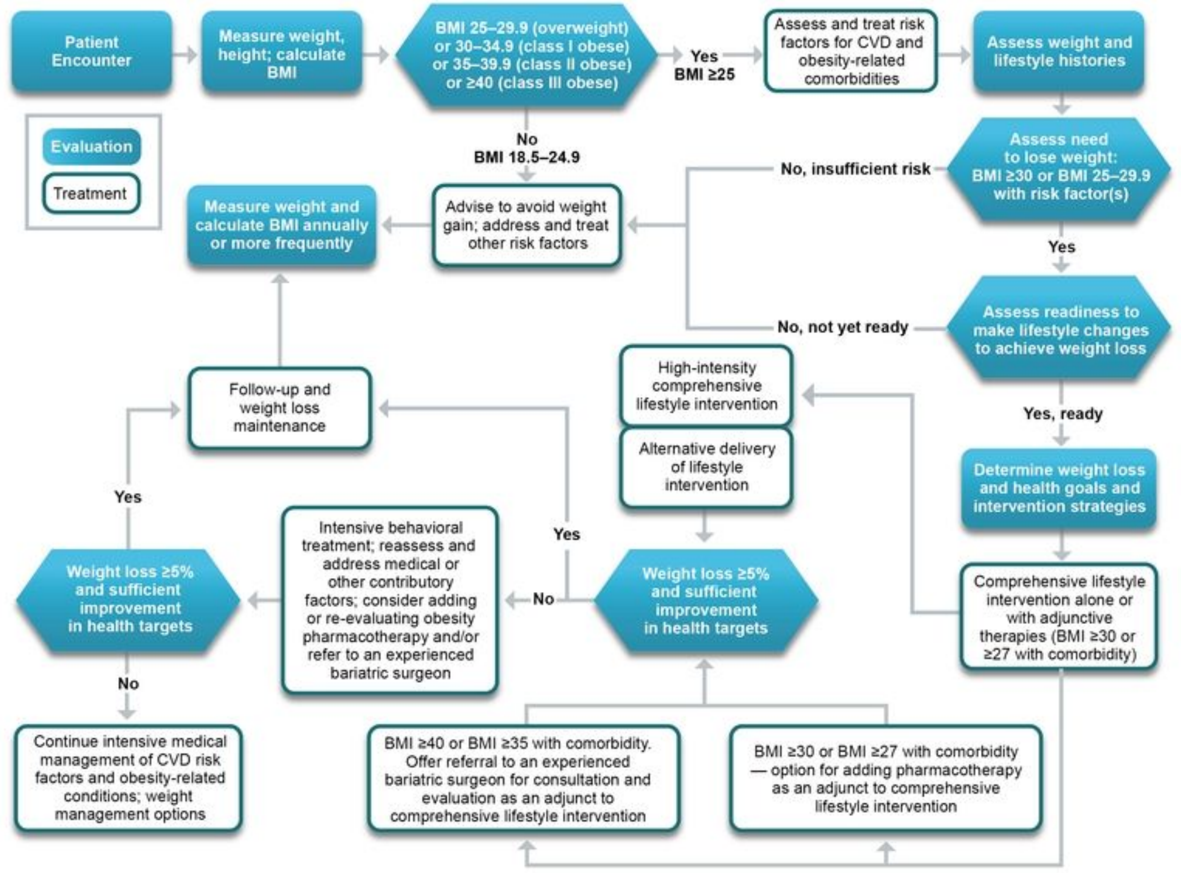
To test if water is **NOT** liquid at a given temperature

```
if temp <= 0 OR temp >= 100)
      print ("Water is not liquid")
```

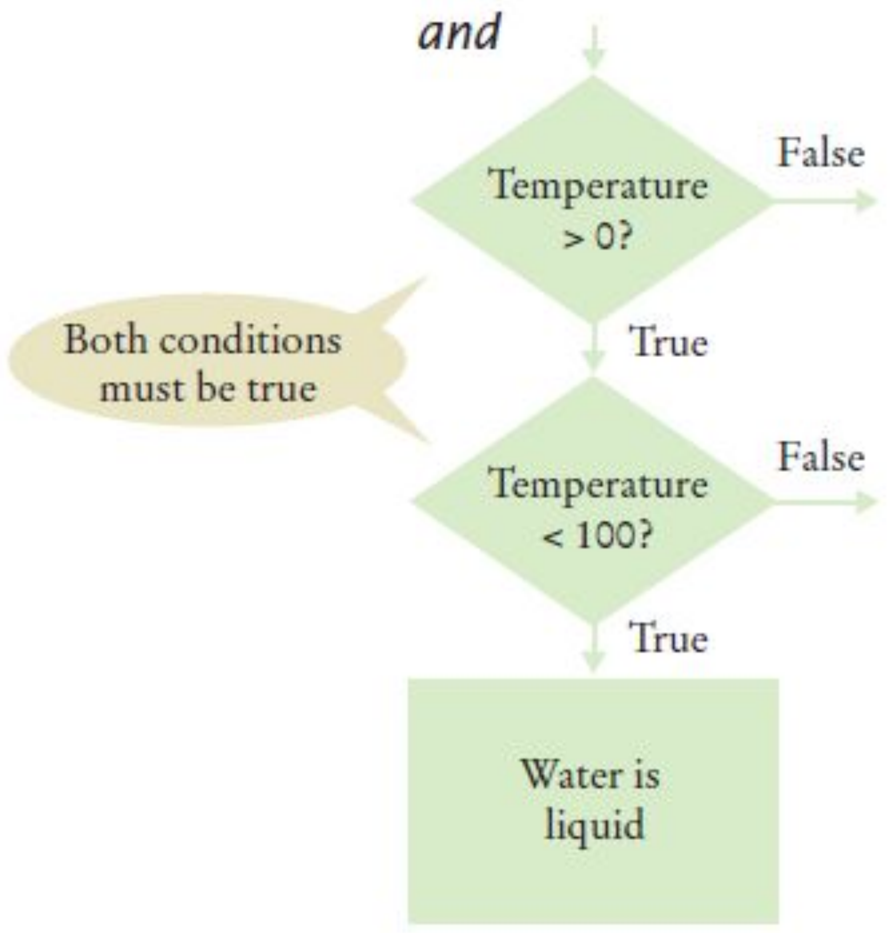


Boolean Variables and Operators

Consider how Boolean operators would fit into clinical flow considerations


Patient Encounter
Measure weight, height; calculate BMI
BMI 25–29.9 (overweight) or 30–34.9 (class I obese) or 35–39.9 (class II obese) or ≥40 (class III obese)
Yes BMI ≥25
Assess and treat risk factors for CVD and obesity-related comorbidities
Assess weight and lifestyle histories
Evaluation
Treatment
No BMI 18.5–24.9
No, insufficient risk
Assess need to lose weight: BMI ≥30 or BMI 25–29.9 with risk factor(s)
Measure weight and calculate BMI annually or more frequently
Advise to avoid weight gain; address and treat other risk factors
Yes
No, not yet ready
Assess readiness to make lifestyle changes to achieve weight loss
High-intensity comprehensive lifestyle intervention
Follow-up and weight loss maintenance
Alternative delivery of lifestyle intervention
Yes, ready
Determine weight loss and health goals and intervention strategies
Yes
Yes
Intensive behavioral treatment; reassess and address medical or other contributory factors; consider adding or re-evaluating obesity pharmacotherapy and/or refer to an experienced bariatric surgeon
Weight loss ≥5% and sufficient improvement in health targets
No
Weight loss ≥5% and sufficient improvement in health targets
Comprehensive lifestyle intervention alone or with adjunctive therapies (BMI ≥30 or ≥27 with comorbidity)
No
Continue intensive medical management of CVD risk factors and obesity-related conditions; weight management options
BMI ≥40 or BMI ≥35 with comorbidity. Offer referral to an experienced bariatric surgeon for consultation and evaluation as an adjunct to comprehensive lifestyle intervention
BMI ≥30 or BMI ≥27 with comorbidity — option for adding pharmacotherapy as an adjunct to comprehensive lifestyle intervention

https://www.epgonline.org/cf/image-popup.cfm?id=7238



Boolean Variables and Operators

- To *invert* a condition use the ***not*** Boolean operator
- To test if the Boolean variable frozen is false:

```
if NOT frozen:
    print("Not frozen")
```

Question

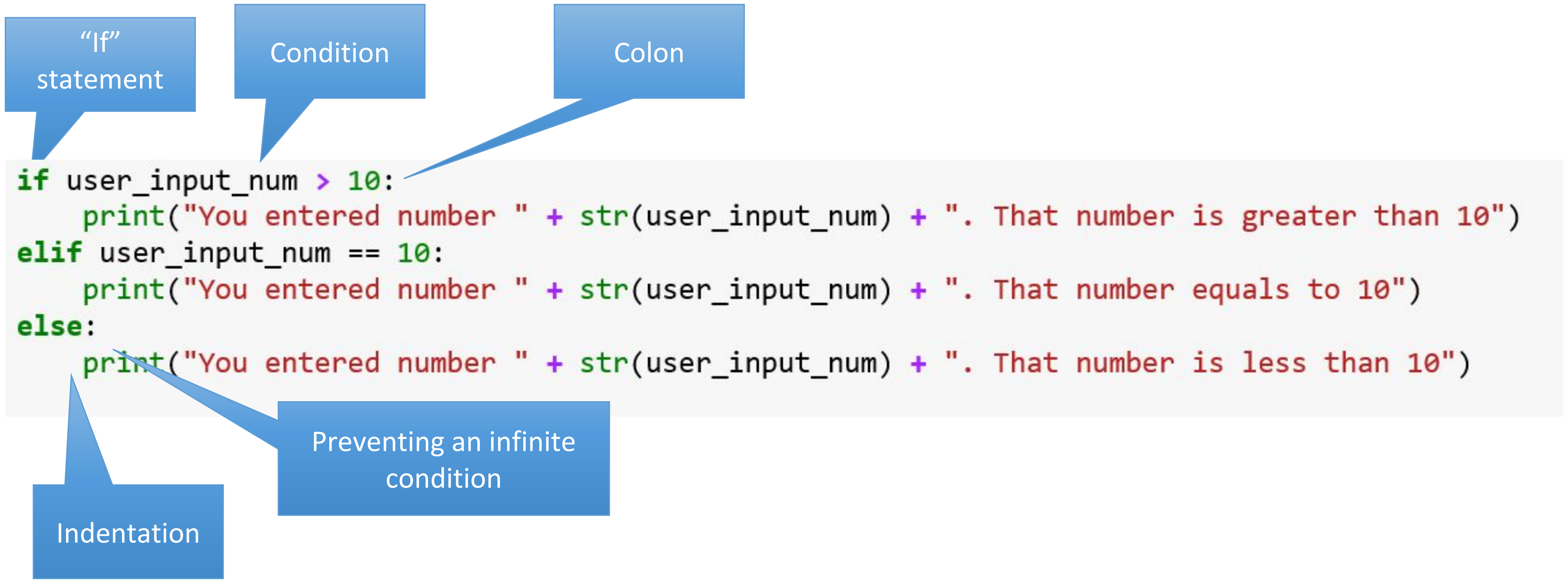
What is the value of NOT NOT frozen?

when frozen is True, NOT NOT frozen is ____

when frozen is False, NOT NOT frozen is ____

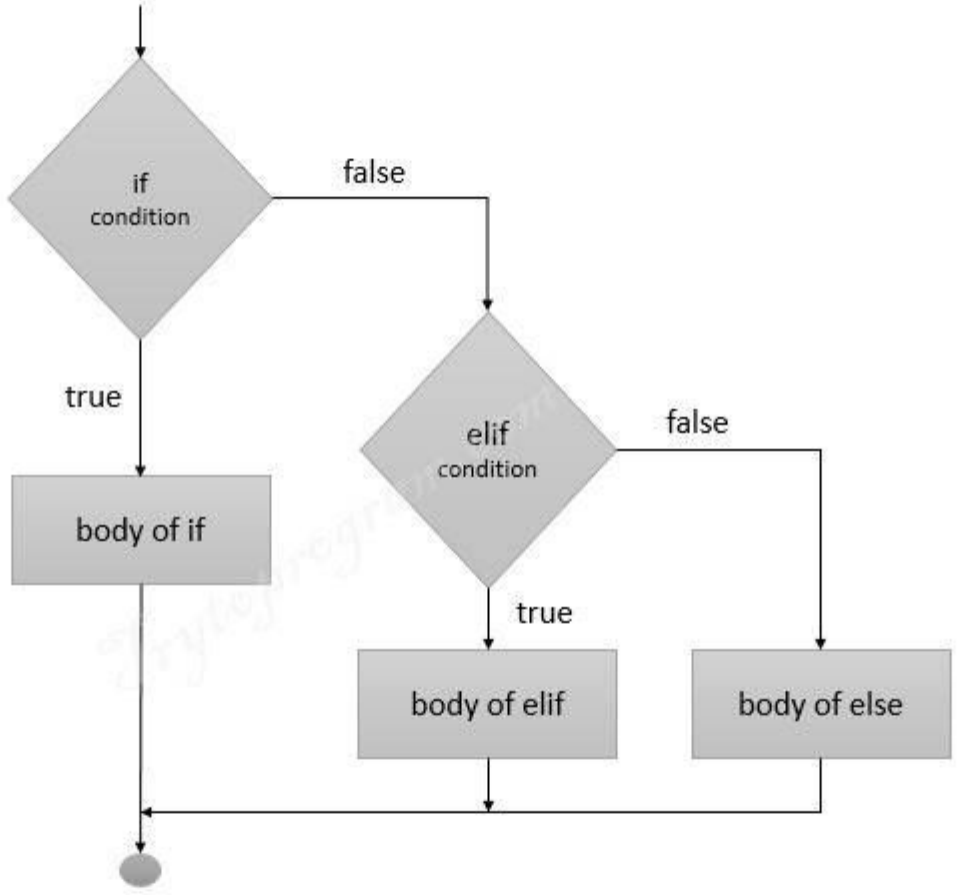
- Variables
    - Data Types
    - Naming Conventions
    - Type Casting
- Arithmetic Operations
- Strings
    - Operations
    - String-number conversions
- Basic Input/Output
- Control structures / conditional statements

Anatomy of a Conditional Statement



```python
if user_input_num > 10:
    print("You entered number " + str(user_input_num) + ". That number is greater than 10")
elif user_input_num == 10:
    print("You entered number " + str(user_input_num) + ". That number equals to 10")
else:
    print("You entered number " + str(user_input_num) + ". That number is less than 10")
```

Flow of “If” “Elif” and “Else”



http://www.trytoprogram.com/python-programming/python-if-else-statement/

if With Multiple Conditions

A store accepts payments in US dollars and Euros, but not in any other currency:

“OR” operator

```
if (currency.equals("dollars") or currency.equals("euro")){
    // accept payment
}
```

Lucky number example

if With Multiple Conditions

You are checking whether or not user has permissions to login to the system and whether or not he can view a particular page:

“AND” operator

```
If (isLoggedIn == true and canViewPage == true){
    // can login and view page
}
```

doing x == true is redundant, better do just this

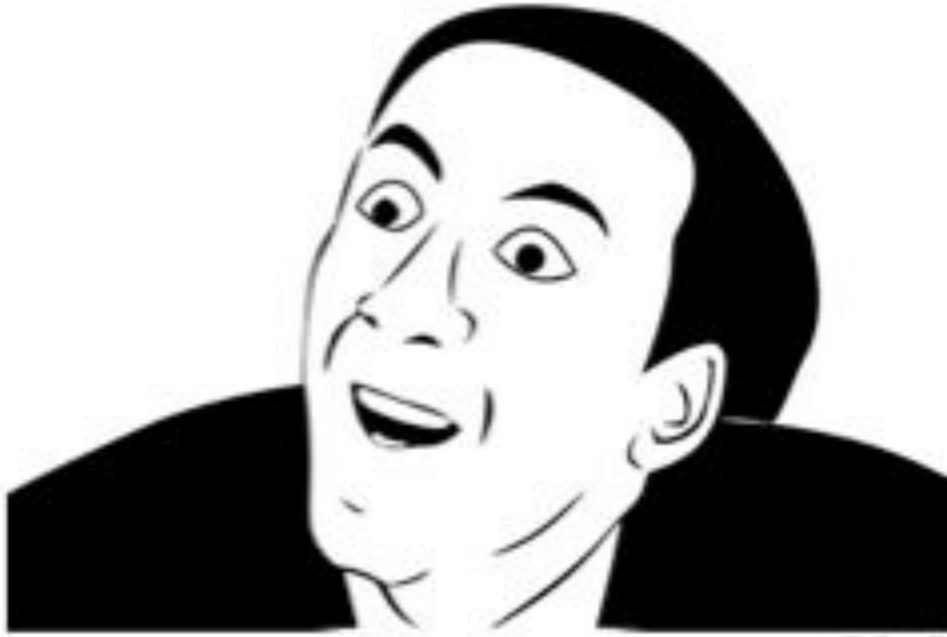
```
If (isLoggedIn && canViewPage){
    // can login and view page
}
```

No matter value of a, this is **always false**:

a && !a

No matter value of a, this is **always true**:

a || !a



Exercise

Suppose ***x*** and ***y*** are two integers.

How do you test whether both of them are zero?

How do you test whether at least one of them is zero?

How do you test whether exactly one of them is zero?

Multiple Alternatives - Example

The 1989 Loma Prieta earthquake that damaged the Bay Bridge in San Francisco and many buildings measured 7.1 on the Richter scale.

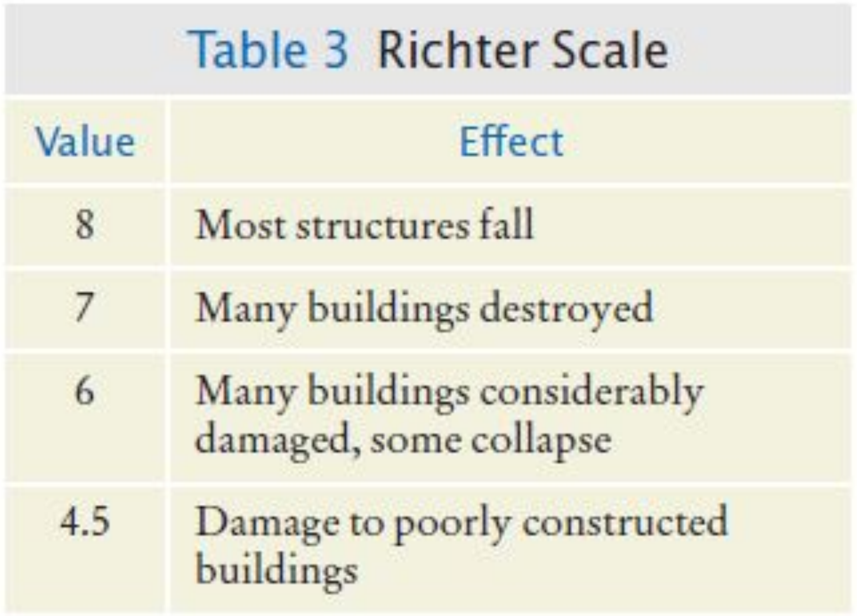


Table 3 Richter Scale

Value	Effect
8	Most structures fall
7	Many buildings destroyed
6	Many buildings considerably damaged, some collapse
4.5	Damage to poorly constructed buildings

```
if (richter >= 8.0)
{
        description = "Most structures fall";
}
elif (richter >= 7.0)
{
        description = "Many buildings destroyed";
}
elif (richter >= 6.0)
{
        description = "Many buildings considerably damaged, some collapse";
}
elif (richter >= 4.5)
{
        description = "Damage to poorly constructed buildings";
}
else
{
        description = "No destruction of buildings";
}
```

Multiple Alternatives - Error

In this example, must use if/else if/else sequence, not just multiple independent if statements

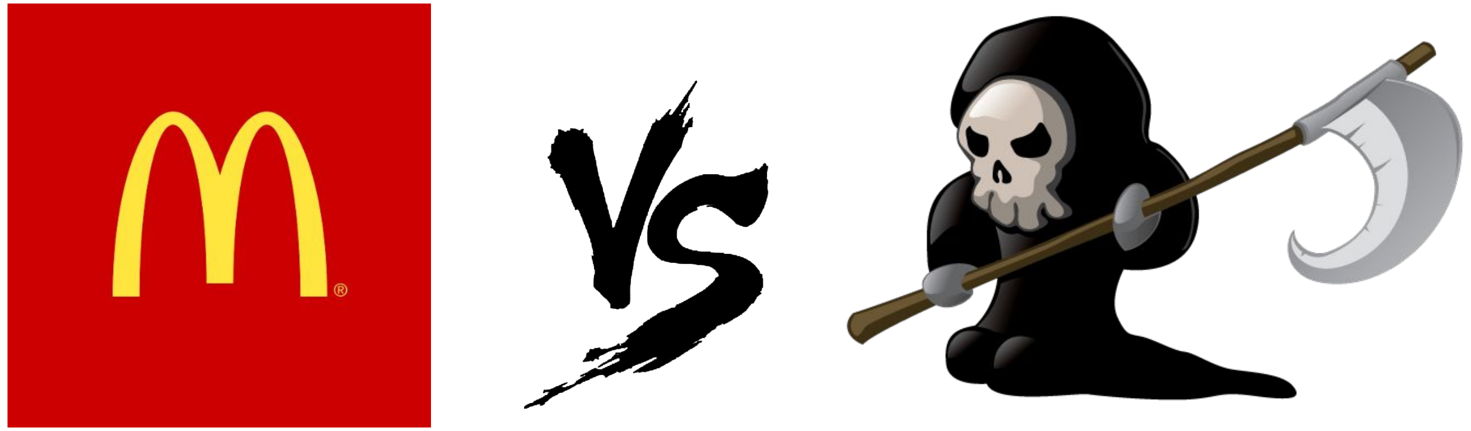
```
if (richter >= 8.0) // Didn't use else
{
    description = "Most structures fall";
}
if (richter >= 7.0)
{
    description = "Many buildings destroyed";
}
if (richter >= 6.0)
{
    description = "Many buildings considerably damaged, some collapse";
}
```

Multiple Alternatives

- Multiple if statements can be combined to evaluate complex decisions.
- Use multiple if statements to implement multiple alternatives.
- As soon as one of the tests succeeds:
  - The effect is displayed
  - No further tests are attempted.
- If none of the cases applies
  - The final else clause applies

Spurious Correlation of the Day

Customer satisfaction with McDonald's is highly correlated with **deaths caused by obstruction of the respiratory tract following ingestion of food**



Vigen, Tyler. Spurious Correlations (p. 10). Hachette Books. Kindle Edition.