

NANYANG
TECHNOLOGICAL
UNIVERSITY
SINGAPORE

Fundamentals of Computer Science and Programming

BS3033 Data Science for Biologists

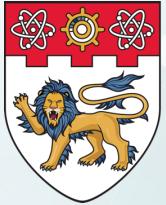
Dr Wilson Goh
School of Biological Sciences

Learning Objectives

By the end of this topic, you should be able to:

- Define algorithms, heuristics, and computer programmes.
- Explain variables, operations, and control statements in the context of programming logic.
- Describe examples of classical tractable and non-tractable algorithms.
- Describe examples of algorithms in genomics and proteomics.
- Describe the process of problem specification, formulation, and evaluation in programming.





NANYANG
TECHNOLOGICAL
UNIVERSITY

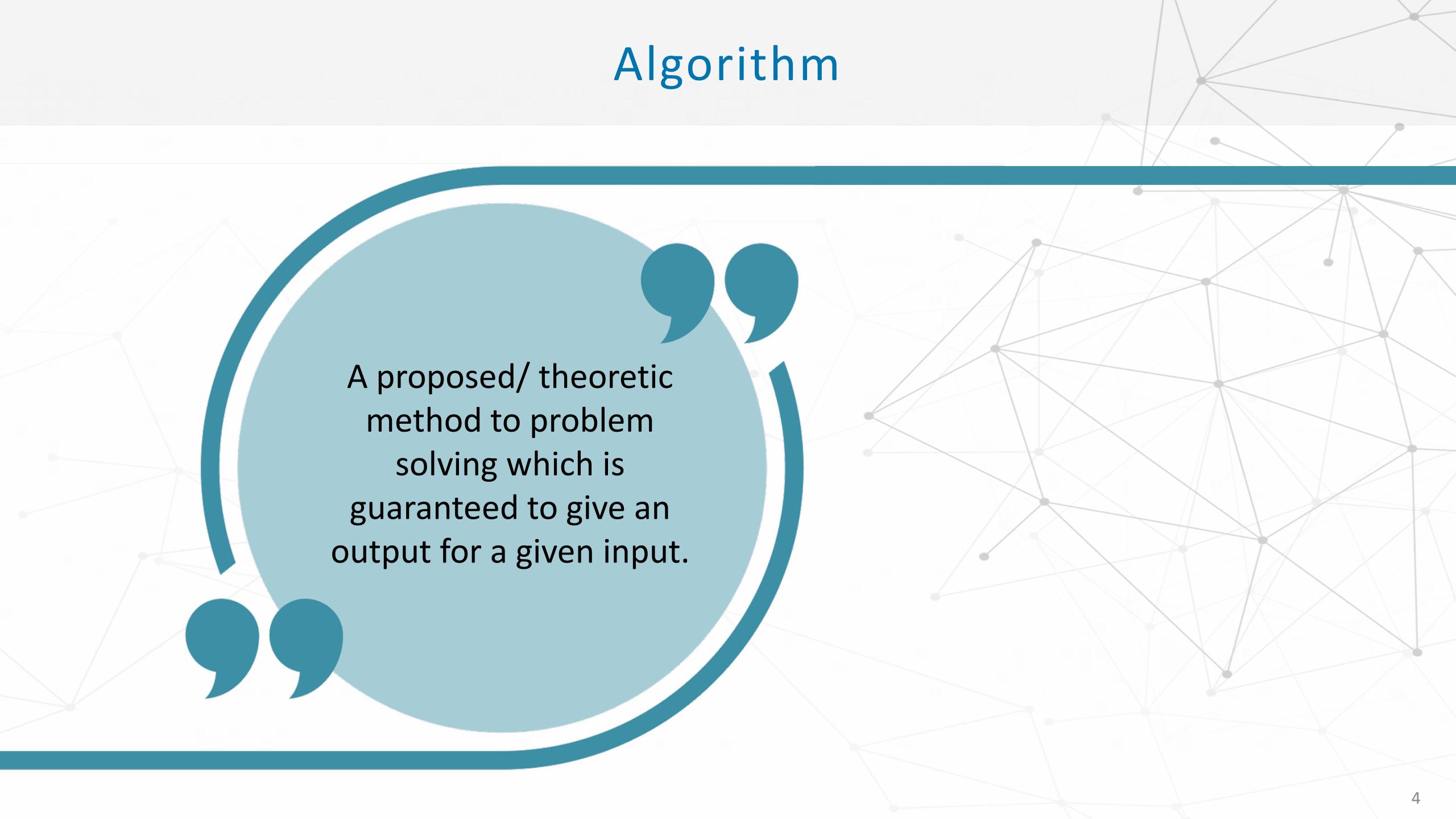
SINGAPORE

What are Algorithms?

BS3033 Data Science for Biologists

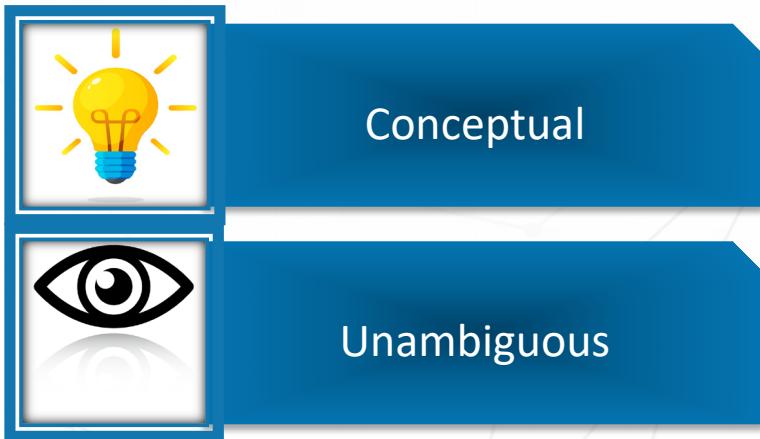
Dr Wilson Goh
School of Biological Sciences

Algorithm



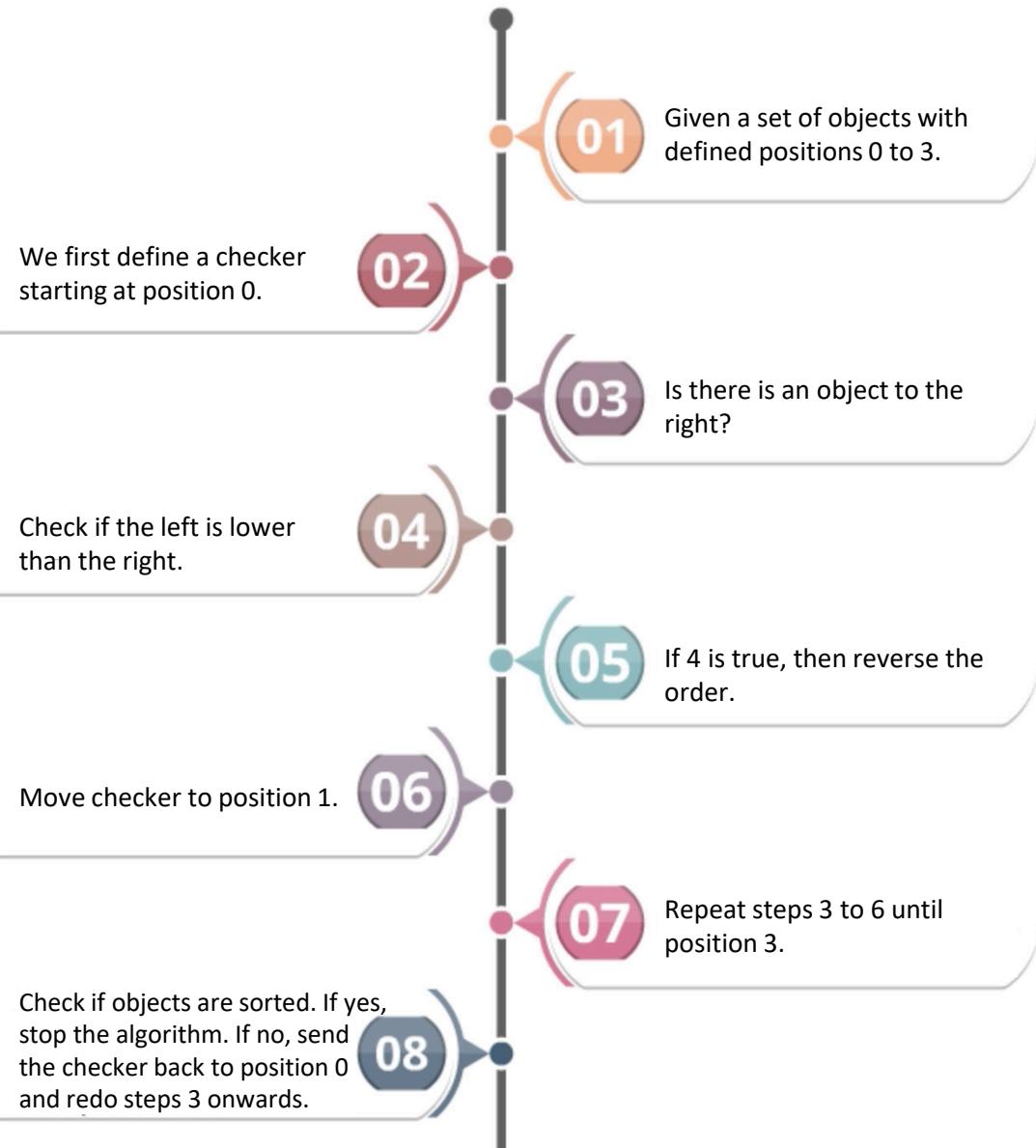
A proposed/ theoretic method to problem solving which is guaranteed to give an output for a given input.

Algorithm



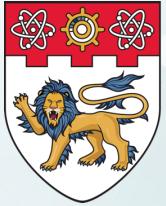
Some definitions suggest that an algorithm should always produce the correct output. But it is arguable when “correct” is subjective, especially if there is no way of knowing if a fully correct answer can ever be achieved. An example being genome sequence assembly problems.

Algorithms in Action



Watch the video lecture to view the animation.





NANYANG
TECHNOLOGICAL
UNIVERSITY

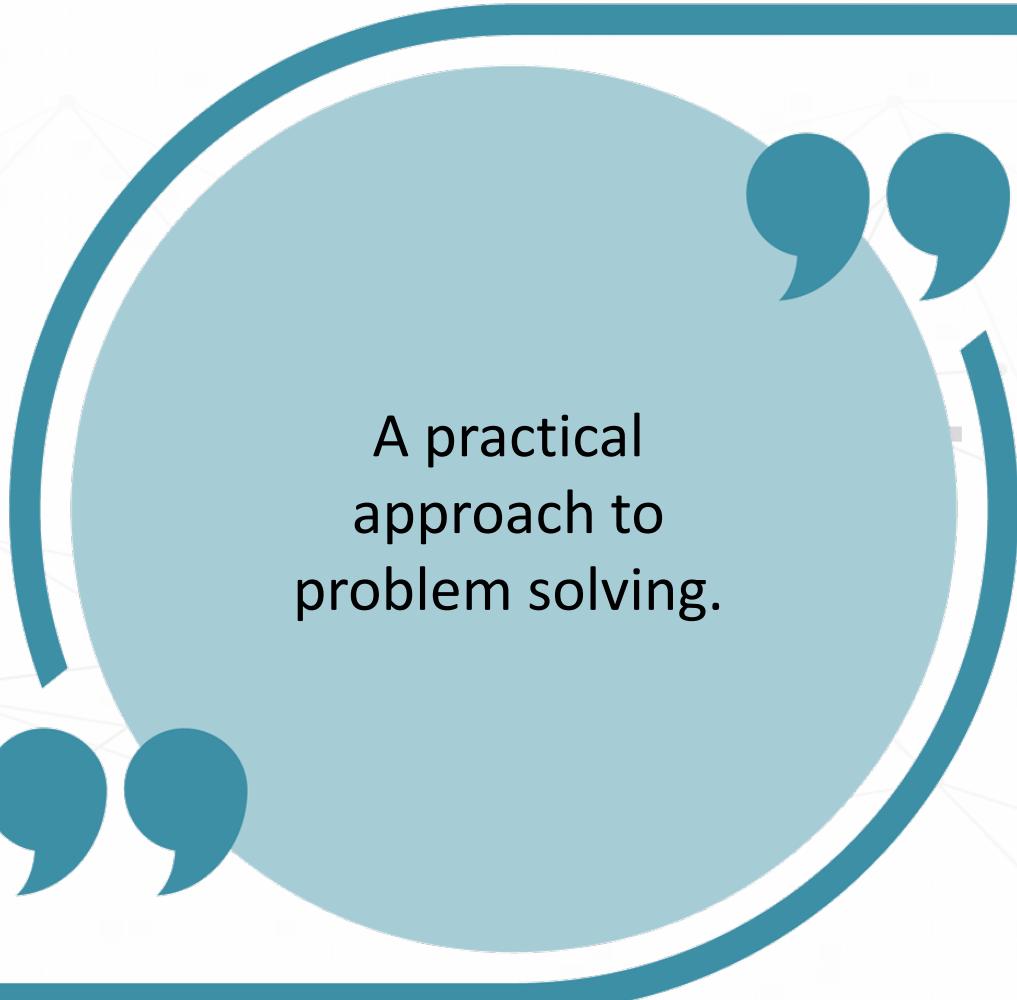
SINGAPORE

What are Heuristics?

BS3033 Data Science for Biologists

Dr Wilson Goh
School of Biological Sciences

Heuristics



A practical approach to problem solving.

Heuristics



Must be fast



Abstraction



Could be wrong

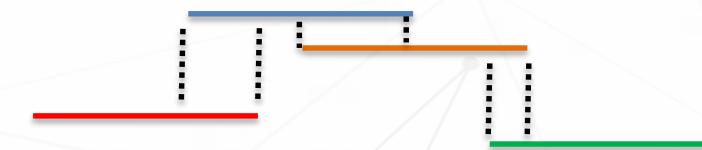


Educated guess

Stereotyping is a form of mental heuristic. While it helps us make snap decisions or form snap opinions. It can be obviously wrong!

Examples of Heuristics (Genome Assembly)

In genome assembly, short sequence reads are iteratively overlapped to generate a full assembly. This is a very time-consuming process (**all** pairwise comparisons required) and gaps will occur.

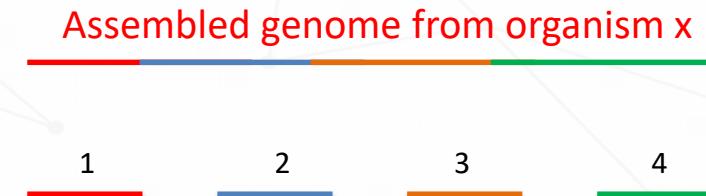


Assembled genome

Heuristic: instead of purely relying on sequence overlaps, use the structural scaffold of a related organism to quickly identify the approximate locations of reads such that less comparisons need to be made.

Pros: Uses homology information; is fast

Cons: May miss critical rearrangement/divergence events during speciation



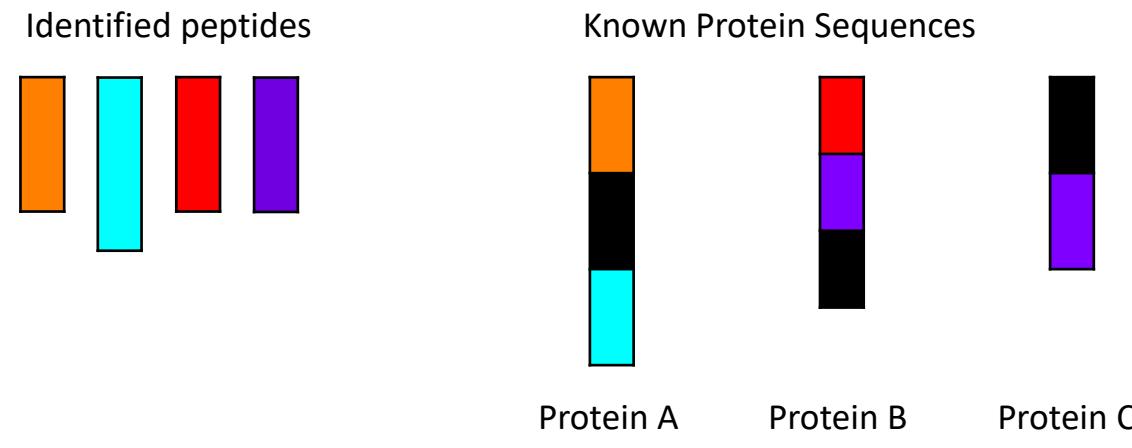
Assembled genome

Order Pre-determination such that we only need to consider position 1 and 2, 2 and 3....

Examples of Heuristics (Proteome Identification)

Parsimony would suggest that Protein A is confidently identified whereas we cannot tell for certain if B or C exists due to the presence of ambiguous peptides. This is a simplifying assumption since...

Protein identification using only unique peptides

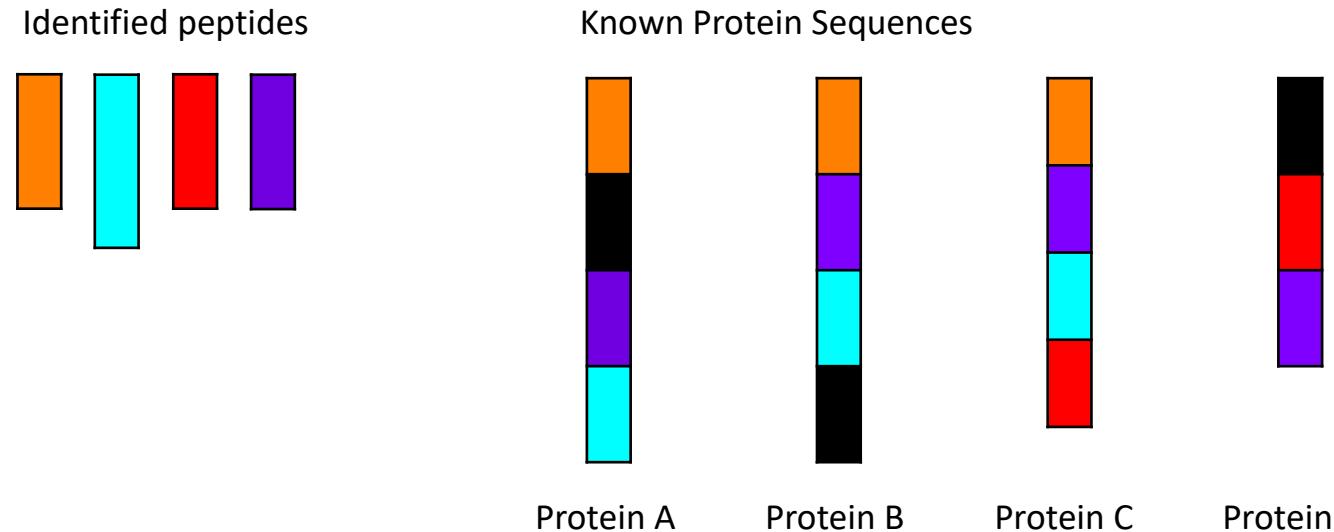


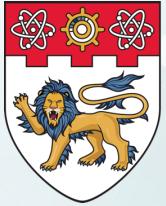
Sequence that is not supported by an spectra

Examples of Heuristics (Proteome Identification)

..there can be meaning in ambiguity. In this case, although all peptides are ambiguous, Protein C is likely present. So disregarding any peptide due to ambiguity gives us a quick set of proteins with minimal computation but we lose useful information consequently.

Protein identification using only unique peptides





NANYANG
TECHNOLOGICAL
UNIVERSITY

SINGAPORE

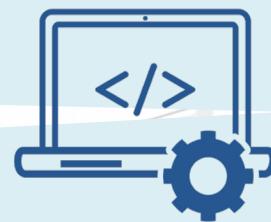
What is Programming?

BS3033 Data Science for Biologists

Dr Wilson Goh
School of Biological Sciences

What is Programming?

Computers cannot solve all problems. They can only help solve well-posed problems.

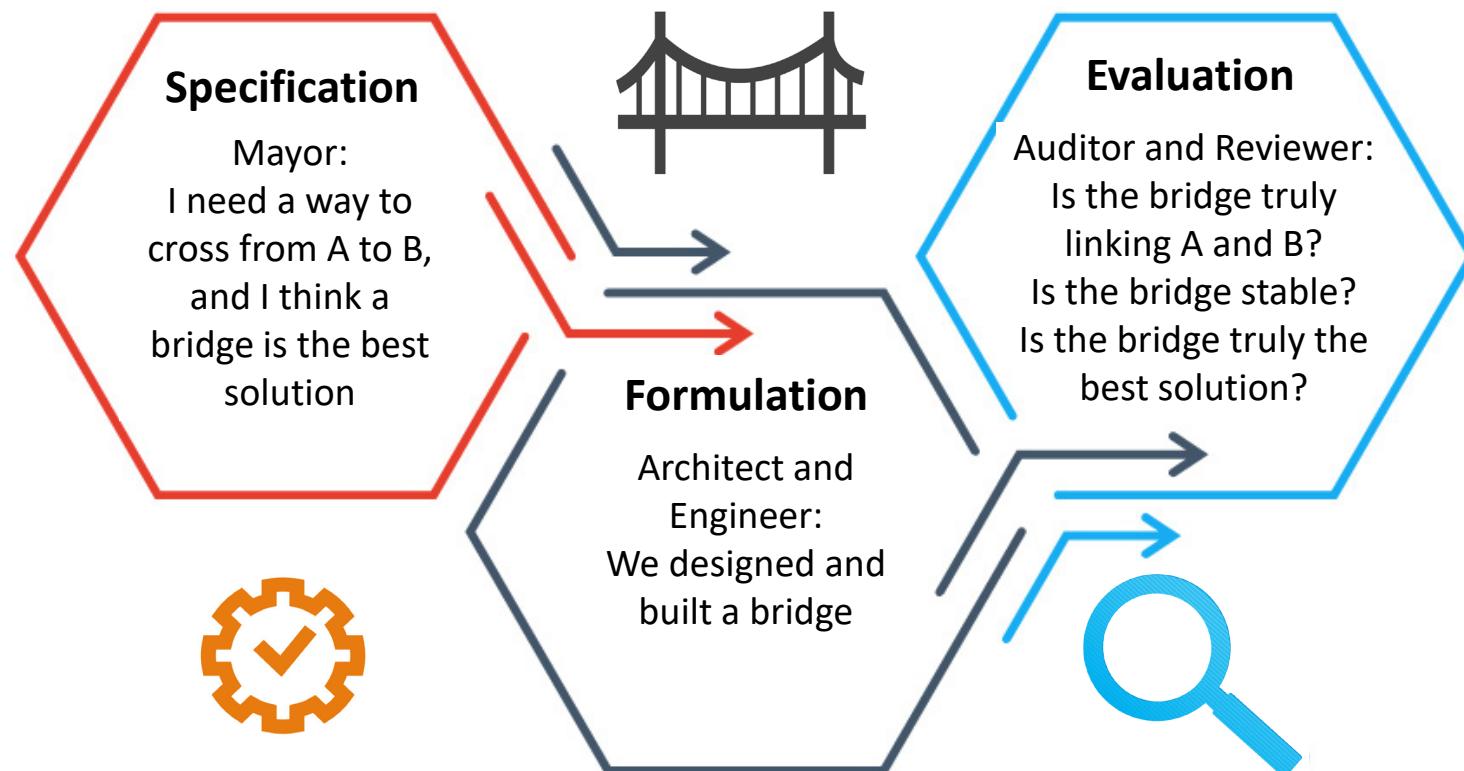


Programming is the process of commanding the computer to solve problems. It is an elaborate process which involves:

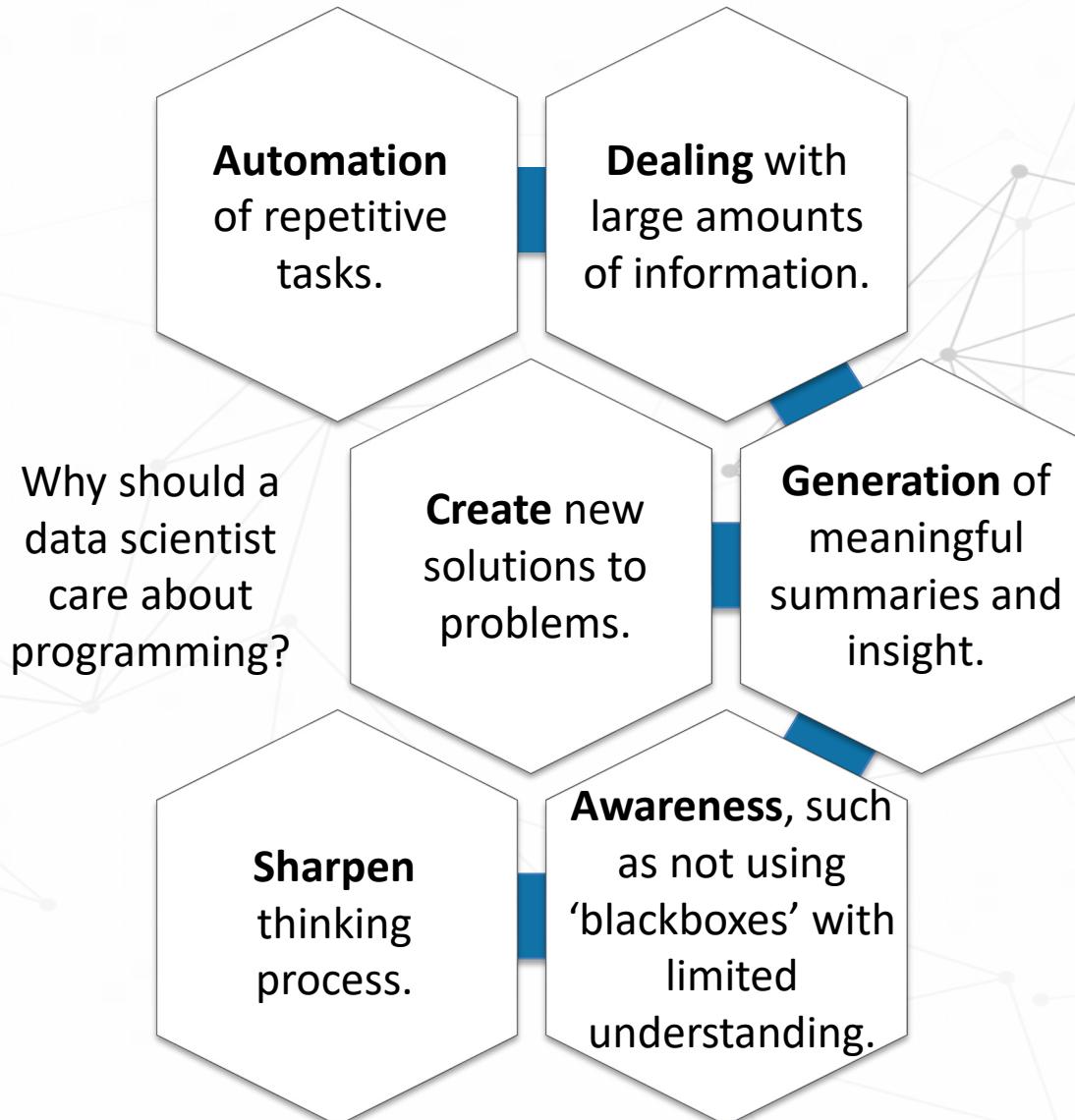
- ✓ **Specification:** Stating precisely the nature of the problem that needs to be solved
- ✓ **Formulation:** Designing and implementing an appropriate solution
- ✓ **Evaluation:** Check if the solution is appropriate. Are there bugs?

Specification, Formulation, and Evaluation (Simple Example)

This process is not so different from carrying out a project.



Programming and Data Science



Specification Exemplifies the Problem

Examples:

Does gene A cause disease X?

I need to compare two sequences to find out how similar they are.

I need a smart program that can select the best statistical test given my data.

The problem must be stated as specific as possible so that an appropriate formulation (solution) can be designed.

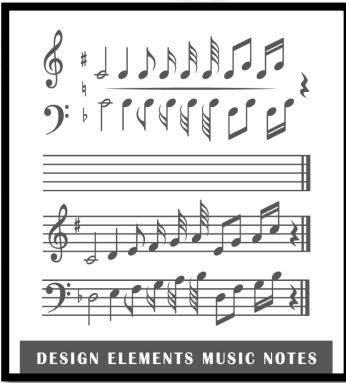
Formulation is Achieved via Algorithms

An **algorithm** is an ordered series of steps for problem solving.

It should be **exact** and **unambiguous**.

It can be **expressed** by a programming language (where it becomes a program).

Algorithm



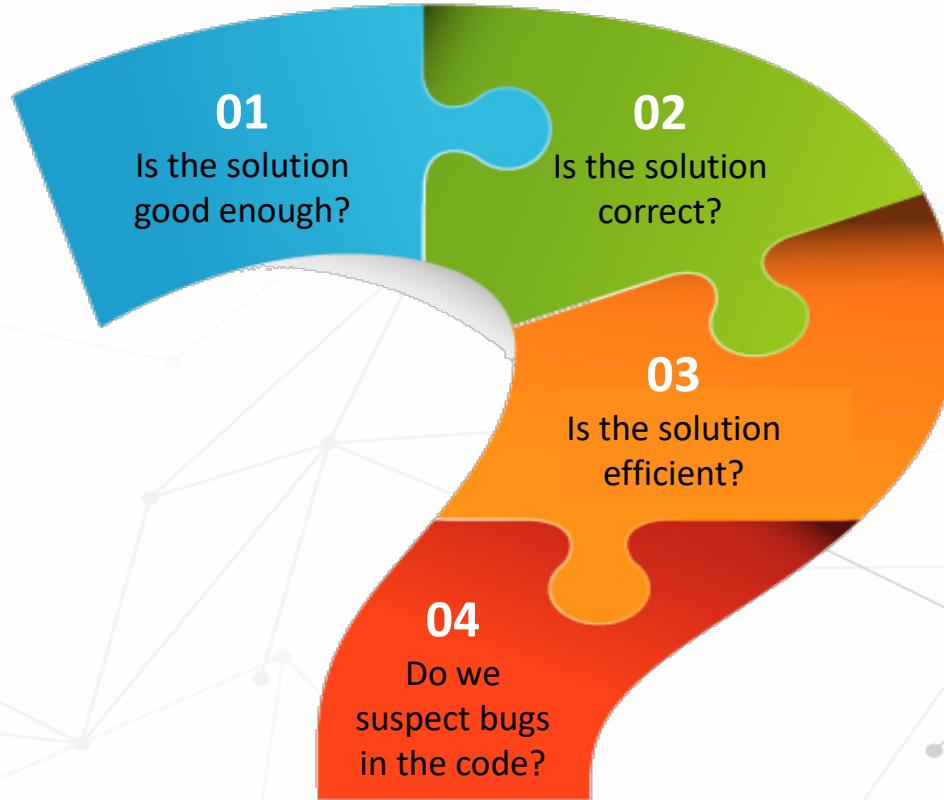
Implementation/
Expression



Program

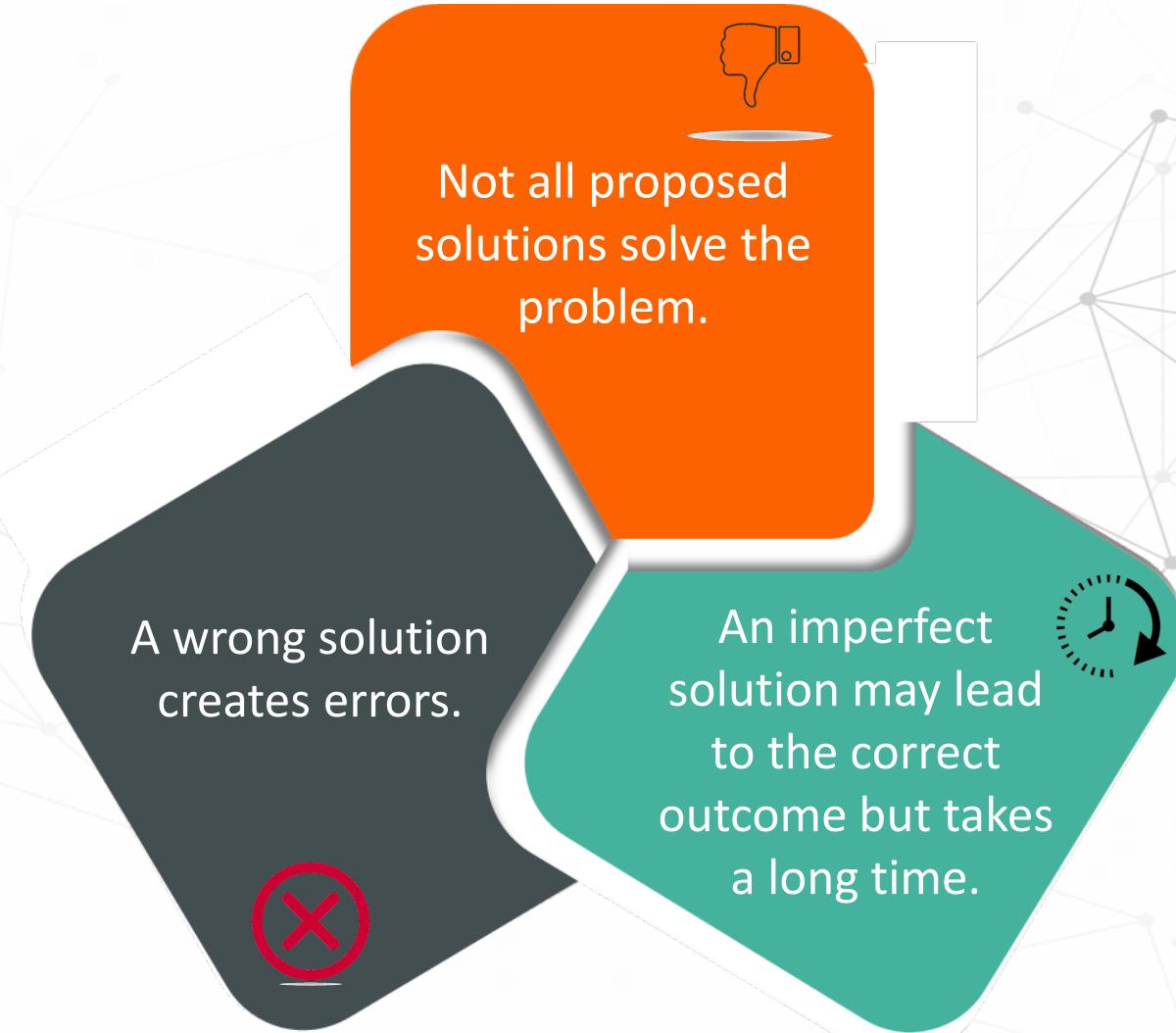


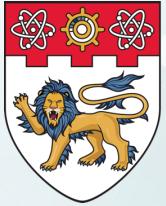
Evaluation



After we write the program and run the analysis, we should not stop there. We must always ask ourselves:

Evaluation





NANYANG
TECHNOLOGICAL
UNIVERSITY

SINGAPORE

Variables, Operations and Control Statements

BS3033 Data Science for Biologists

Dr Wilson Goh
School of Biological Sciences

Variables, Operations, and Control Statements

Algorithms can be written in “English” but we do need some degree of **precision** and **consistency**.

Some standard elements – variables, operations and control statements – are required.

Variables

Algorithms use variables to input and output data and compute and store data.

A variable can be local or global.

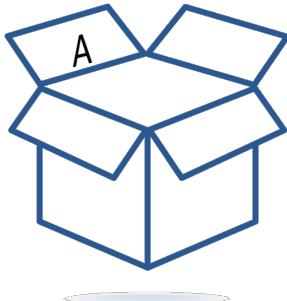
String, numeric, binary, logical are types of variables.

Variables are simple (one variable; one value). More sophisticated structures exist for complex or multi-dimensional data (data structures).

Variables

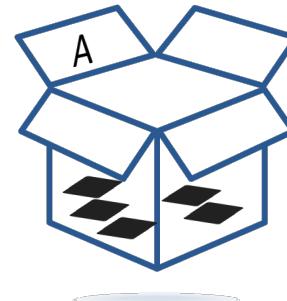
A variable is a container that can be filled with data.

`A = ()`
#nothing in it



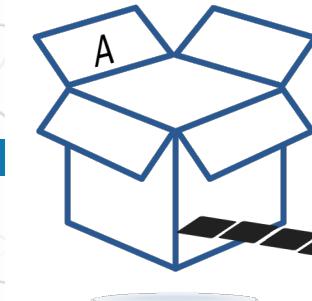
Create an empty container.

`A = 5`
#now the variable A stores the value 5



Fill up the container with some data using an assignment.

`print(A)`
#output the value of A



Output from this variable can be received in several ways. A simple way to output is to print it.

Variables

Variables can take many forms:

Example:
Alice, Paul,
Wilson

Example:
10, 100, 1000

Example:
1 or 0

Example:
True or False

String

Numeric

Binary

Logical

Variables



Global

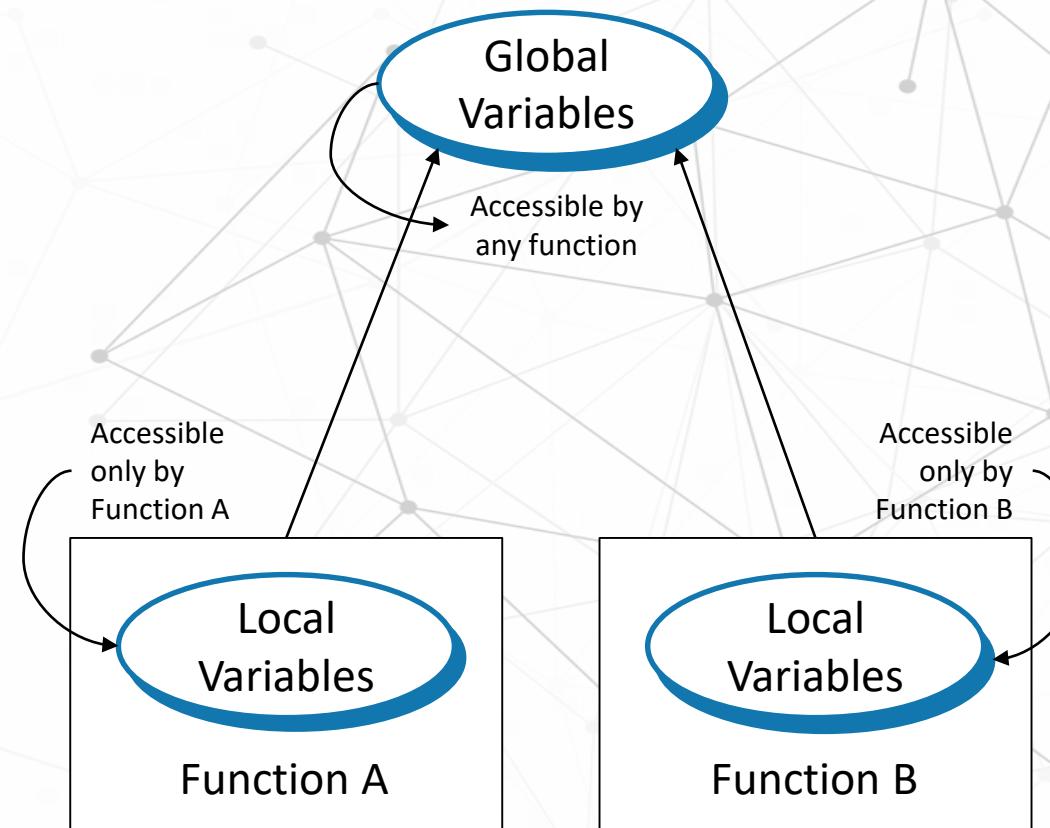
Variables can be made accessible to any part of the program.



Local

Variables are only made accessible to a fixed part of the program.

A function is an autonomous segment of code that performs a specific role; its internal processes are kept separate from the rest of the program.



Operations

An operation evaluates a line of code to see if it meets some condition. If the condition is met, then it is true. If the condition is not met, then it is false.

The most common operations are **if** and **else**.

Operations (if and else)

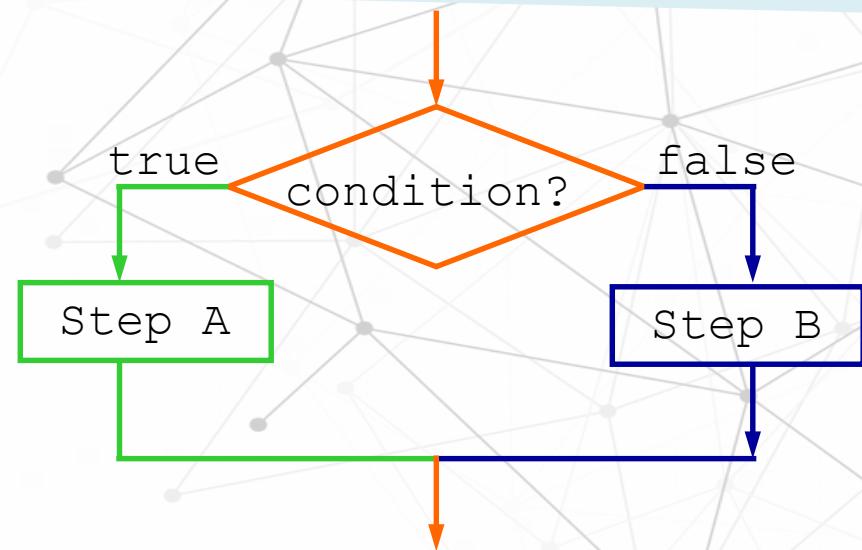
if and **else** conditional statement is used to take different actions based on a condition.

If and Else

```
if (condition)
    then (Step A)
    else (Step B)
endif
```

If

```
if (condition)
    then (Step A)
endif
```



Endif is used to end the **if** condition

Operations (if and else)

A simple example:

```
Let mark be the total-mark obtained  
if (mark < 40)  
    then (print "Student fail")  
    else (print "Student pass")  
endif  
...
```

```
read in mark (*from a list*)  
if (mark < 40) then (Grade ← "F")  
    else if (mark < 50) then (Grade ← "D")      if else  
    else if (mark < 60) then (Grade ← "C")      statements  
    else if (mark < 70) then (Grade ← "B")      can be  
    else if (mark < 80) then (Grade ← "A");      chained  
endif  
print "Student grade is", Grade  
...
```

Control Statements

Control statements allow us to repeat a block of code many times over:



Until some condition is met



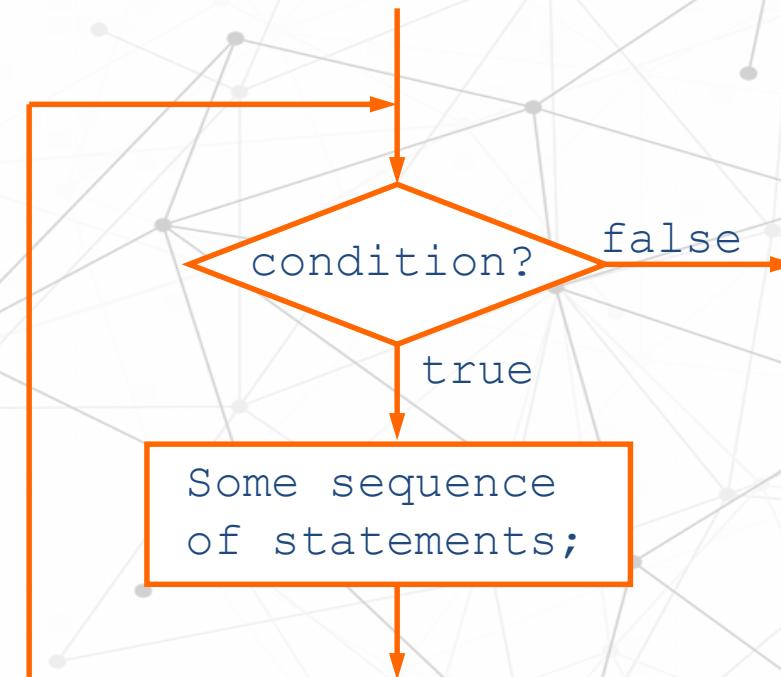
Otherwise, it will run for eternity or until the computer crashes

The most common control statements are **For** and **While** loops.

Control Statements (while-loop)

The **while-loop** control statement loops a “variable” number of times.

```
while (condition) do  
    (some sequence  
     of statements)  
endwhile
```

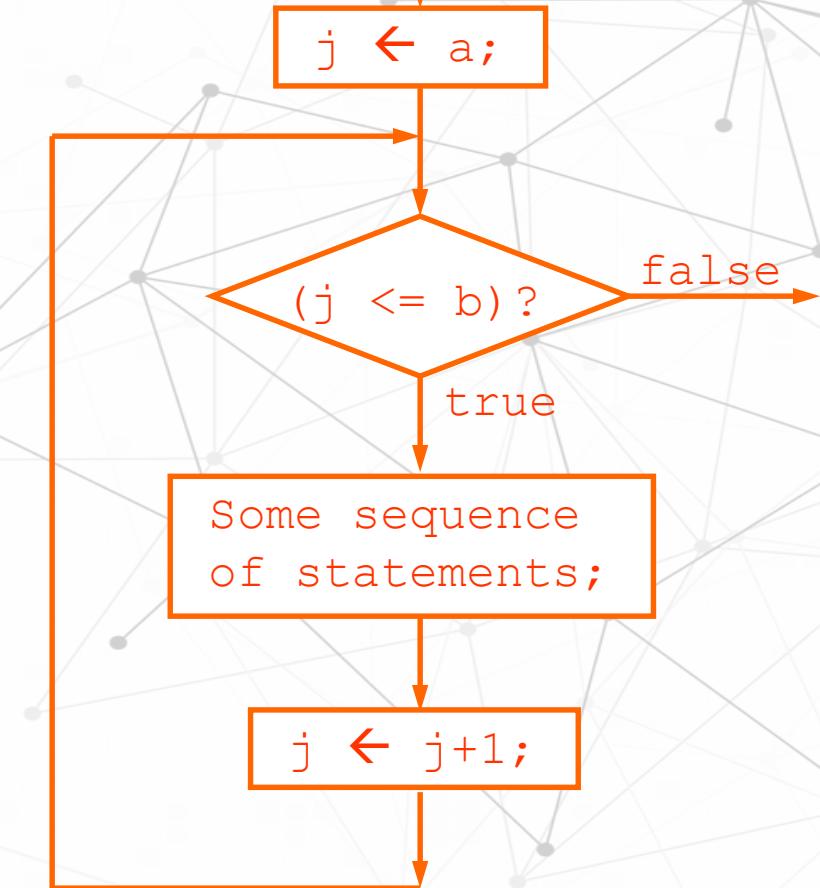


If the statement that has been evaluated in the previous round is true (ith-1), continue round i.
What do you think will happen if the statement is never false?

Control Statements (for-loop)

The **for-loop** control statement loops a specific or pre-determined number of times.

```
for j <- a to b do  
    (some sequence  
     of statements)  
endfor
```



Control Statements (for and while)

```
for j ← 1 to 4 do
    print 2*j;
endfor
print "--- Done ---"
```

Output:

2
4
6
8
--- Done ---

```
j ← 1;
while (j <= 4) do
    print 2*j;
    j ← j + 1;
endwhile
print "--- Done ---"
```

Output:

2
4
6
8
--- Done ---

Putting it together: Can you guess what this algorithm does?

```
MysteryAlgo(a, b, c)
if a > b
    if b > c
        return c
else
    return b
else
    if a < c
        return a
else
    return c
```

- Now that you know what it does, can you think of a smarter way of rewriting this?
- **Hint:** Recall the “else if” statements in earlier example.
- **Hint:** Also, you may chain several evaluation statements together using “And” statements.

Putting it together: Can you guess what this algorithm does?

Computer code for identifying the minimum given a set of numbers:

```
MysteryAlgo(a, b, c)
if a > b
    if b > c
        return c
    else
        return b
else
    if a < c
        return a
    else
        return c
```

Longer version

```
MysteryAlgo(a, b, c)
if a > c and b > c
    return c
else if a > b and c > b
    return b
else return a
```

Shorter version

Putting it Together: The Hamming Distance

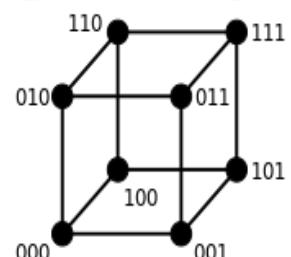
Hamming Distance is a number used to denote the difference between two binary strings.

It is a small portion of a broader set of formulas used in information analysis.

Specifically, Hamming's formulas allow computers to detect and correct errors on their own.

The Hamming Code earned Richard Hamming the Eduard Rheim Award of Achievement in Technology in 1996, two years before his death. Hamming's additions to information technology have been used in innovations, such as modems and compact discs.

Let's go through the steps and work out how to implement this.



Putting it Together: The Hamming Distance

1

Ensure the two strings are of equal length. The Hamming distance can only be calculated between two strings of equal length.

3

Compare each bit in succession and record either "1" or "0" as appropriate.

2

Compare the first bits in each string. If they are the same, record a "0" for that bit. If they are different, record a "1" for that bit.

4

Add all the ones and zeros in the record together to obtain the Hamming distance.



Hint: You can move from one part of the string to another part by indexing its position from 1 to n, where n is the length of the string

Putting it Together: The Hamming Distance

1

String 1: "1001 0010 1101"
String 2: "1010 0010 0010"

3

String 1: "1001 0010 1101"
String 2: "1010 0010 0010"
Record: "0011 0000 1111"

2

First bit of both strings: "1"
Record: "0" for the first bit

4

Hamming distance:
0+0+1+1+0+0+0+1+1+
1+1 = 6

Putting it Together: The Hamming Distance

```
begin
let A be string 1
let B be string 2
let hamming be 0

if length of A is equal to length of B
    for  x from position 1 to position n equals to the length of A
        if position x in A equals position x in B
            hamming = hamming + 0
        else
            hamming = hamming + 1
    else
        exit
return hamming
exit
end
```

Case Study: The Hamming Distance

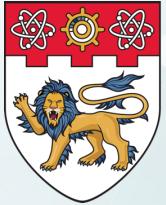
The Hamming distance compares two strings for differences.

It can also be used in biology. Consider the following example involving 2 DNA sequences:

Sequence 1: G A G C C T A C T A A C G G G A T

Sequence 2: C A T C G T A A T G A C G G C C T

The Hamming distance is 7 out of 17 positions. So we may say that this pair of DNA sequences are not very similar.



NANYANG
TECHNOLOGICAL
UNIVERSITY
SINGAPORE

Pseudocode

BS3033 Data Science for Biologists

Dr Wilson Goh

School of Biological Sciences

Pseudocode Comes to the Rescue

All different programming languages use different syntax but there are common elements such as operators, variables and control elements.

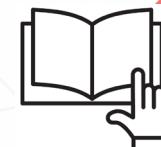


There is a more relaxed method of code logic expression → Pseudocode.



The programmer will then follow the recipe and convert the pseudocode into a **program** (think of the programmer as a cook).

Pseudocode is a series of instructions (statements) written in English that tells the programmer what to do (in a more human language) (it is like a rather loose recipe).



What if a Python programmer wants to share ideas for a code with a fellow R programmer. Can they still understand each other?

Representing Algorithms: Pseudocode

Pseudocode is “almost” code, but not quite. It needs to be properly encoded in specific syntax to become a program.

Algorithms in pseudocode will almost always take the below form:

```
begin
    statement 1;
    ...
    statement n;
end
```

The begin and end are optional – just to make things REALLY obvious.



Expressing Pseudocode

A very loose way of expressing pseudocode:

Specification:

What is the largest integer?

INPUT: All the integers { ... -2, -1, 0, 1, 2, ... }

OUTPUT: The largest integer

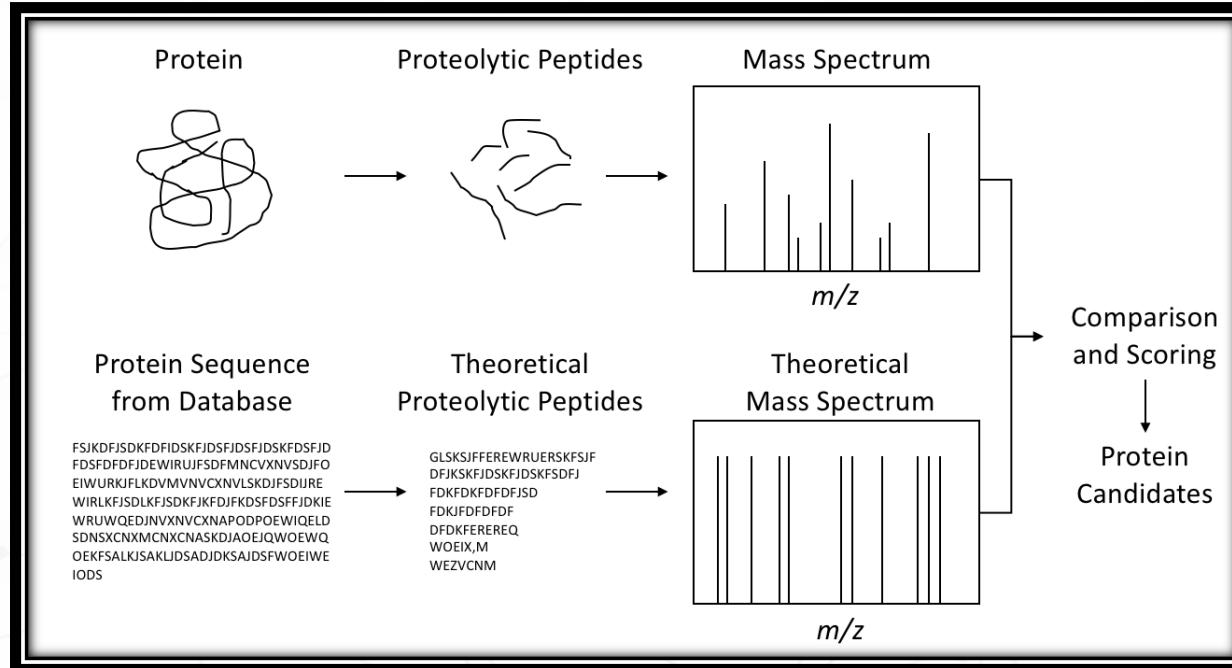
Formulation:

Arrange all the integers in a list in decreasing order;

MAX = first number in the list;

Print out MAX;

Expressing Pseudocode



Peptide Mass Fingerprinting

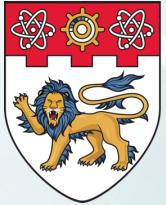
Objective:

To identify the protein based on its mass peaks (this is effectively a pattern matching problem).

A more specific way of expressing pseudocode:

```
Get the experimental mass list L
For each sequence s in the database do
    digest s and obtain a set of peptides P
    for each peptide p in P do
        compute mass (p)
        Push mass (p) in M
    x <- score(M, L)
    store score x for protein s
Compute p-values for each score
Return the n best proteins /* highest score or lowest p-value */
```

Corresponding Pseudocode



NANYANG
TECHNOLOGICAL
UNIVERSITY

SINGAPORE

Summary

BS3033 Data Science for Biologists

Dr Wilson Goh

School of Biological Sciences

Key Takeaways from this Topic

An **algorithm** is conceptual.

A **pseudocode** is written in plain English to express the algorithm. It is human readable but less exact.

Conversion of the pseudocode or algorithm into the formal instructions (syntax) used in a programming language is a **Program**. It has machine readable instruction (very exact).