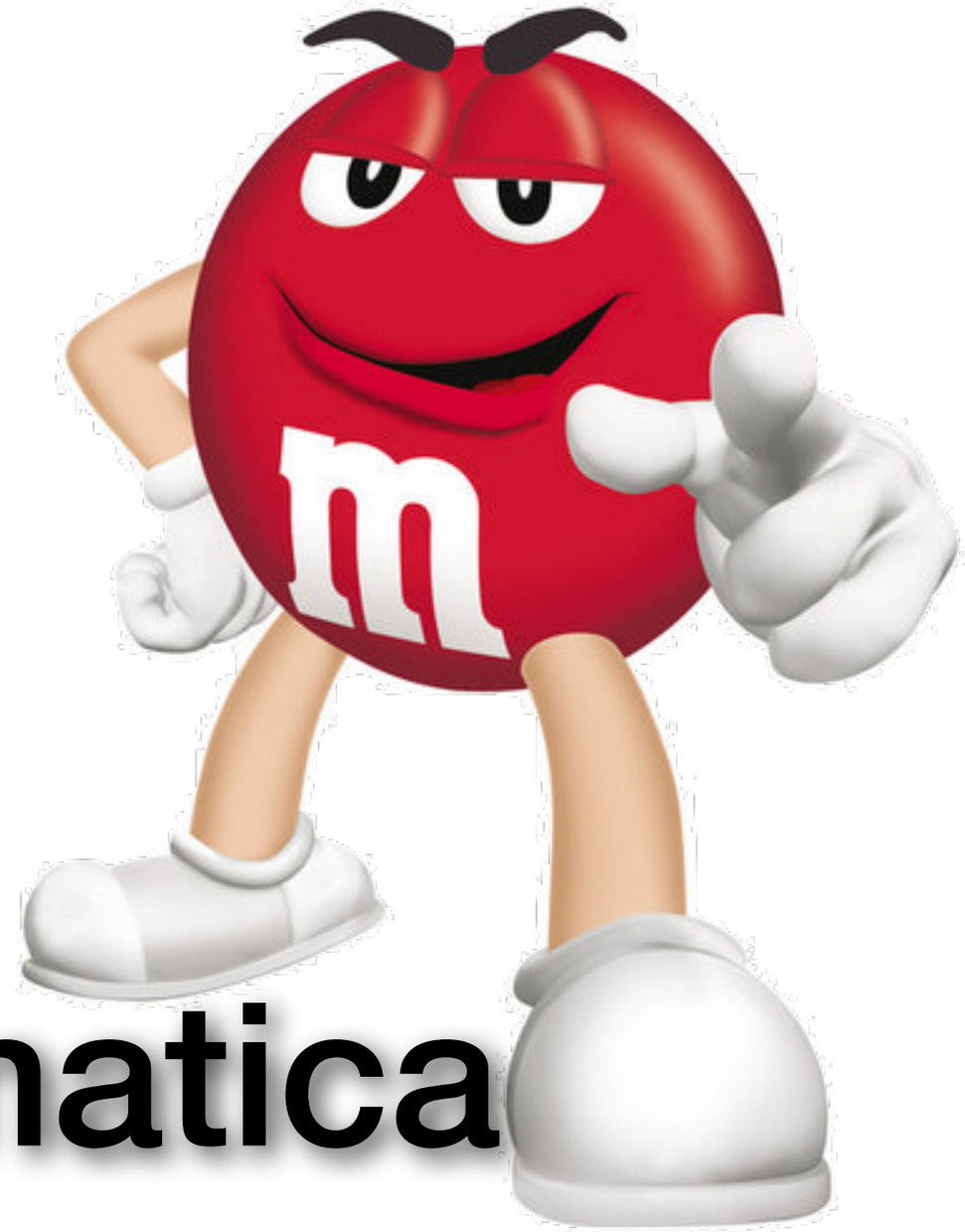
Introduction to



Mathematica

Kuo Kan Liang
RCAS(AS), GSB(AS,NTU), BST(NTU)

Programming Languages

- To ask a computer to work, you need to use a language that computers understand.
- To do a specific kind of works well, you may need a language with specific features.
- To do math...
  - Mathematica
  - MATLAB
  - ...

Suggestions to Newbies

- Although **mathematical software** are usually also **general-purposed**, try to learn them with **special purposes** in mind

- Try to think about **how you would have done the math by hand** before you plan your programs

- If you do not know how to do the math by hand...

Mathematica

- Developed by Stephen Wolfram
- Distributed via Wolfram Research
- Now more generally named **Wolfram Language**
- Emphasize on **symbolic** and **precise** mathematics, and **knowledge-base computation**

Important References

- Online introduction: https://www.wolfram.com/language/elementary-introduction/2nd-ed/

- Online documents: http://reference.wolfram.com/language/

Essence of a PL

- **Syntax** - Does your code look fine to the software (compiler, interpreter)
- **Semantics** - Does your code behave (seemingly) OK
- **Data types** - Does your code work on the correct things and work correctly on them
- **Library** (built-in functions)- What you do (should?) not write yourself for the code to do its jobs

Syntax: Naming

- Legal characters: alphanumerics and special symbols*
- Cannot begin with number
- Case-sensitive (Built-in symbols begin with upper case)

Syntax: Assignment

- Single equal sign for left assignment
- Result of assignment will be displayed
- Result display is suppressed by trailing semicolon
- No equal sign ⇒ assigned to `%`

Project 1: Hello!
Ask the user for a name and say hello.

- Run
  `reply =.;`
  before the main program. Then, check the contents of reply by
  `?reply`
  and
  `reply`
  before, during and after executing the program cell.

Syntax: Basic Math OP

- +, - : the same as most other languages
- * : can be ignored, or special symbol
- / : can be pretty-typed
- ^ : can be pretty-typed
- ( ) : group ops

Syntax Example

```
In[•]:= x = Pi
```

```
Out[•]= π
```

```
In[•]:= x = 2 Pi;
        x ^ 2
```

```
Out[•]= 4 π^2
```

```
In[•]:= x^2
```

```
Out[•]= 4 π^2
```

```
In[•]:= y = x * r
```

```
Out[•]= 2 π r
```

```
In[•]:= y = x r
```

```
Out[•]= 2 π r
```

Undefined symbols treated as symbols

```
In[•]:= y = x / 2
```

```
Out[•]= π
```

```
In[•]:= y = x/2
```

```
Out[•]= π
```

Syntax: Function Call

- Arguments enclosed in square brackets [ ]
- Arguments separated by **comma**
- Options can be set by using '**rule**' in arbitrary order

Syntax: Array

- List of objects enclosed by **curly braces { }** and separated by **comma** to form **column**
- Objects enclosed by **nested { }** form list in the next dimension
- Undefined symbols can be put in array (List)
- To access one element use `A[[position]]`

Project 2: Arithmetic quiz
Ask the user to subtract two numbers

- Read the help of `RandomInteger`.
- What does it mean by `ToString@q[[2]]`? How to put it in function call form?
- Study and try to use `Differences`. Why do we need to add [[1]] behind?

Data Types Fundamentals

- Being a symbolic language, any data type is possible
- **Sets of rules** are almost fully compatible with JSON
- **Association arrays** are new facility for complex user-defined types (UDT)

Library

- About 5000 built-in functions
- Contributed / third-party codes : http://library.wolfram.com

Boolean OPs

Not	!	Compare equal	== (Equal) === (SameQ)
And	&& And[ ]	Compare unequal etc	>, <, >=, <=, !=
Or	|| Or[ ]	String compare	Many query functions
Xor	Xor[ ]	Array (List) comparison	== (Equal) === (SameQ)
Nand / Nor	Nand[ ] Nor[ ]		

Decision: IF

- **If**[*condition*,
  action if true,
  action if false,
  if undetermined
  ]

- Semicolon is not required right before comma

Project 3: Prime?
Check the user input.

- What happens if you input a negative integer?
  A non-integer? A string which is not a number?

Decision: Switch

- **Switch**[*expression*,
  form1, *action1*,
  form2, *action2*,
  ⋮
  _, *default action*
  ]

Project 4: Vowel?
Check the char user inputs

- We know that `||` means ‘or’.
  What does `|` mean?
- Can we type less characters? What are the alternative ways of assigning the ‘cases’?

Loops

- For and Do
  - **For**[*ptr initialization,*
    ptr test, ptr change,
    actions]
  - **Do**[*actions,*
    ptr range specification,
    next ptr range specification, more ptrs...]
- Do is much more concise than For for more regular operations
- For is much C-like. Ptr operations can be complicated.

Loops

- While
  - **While**[*conditions test*, *actions*]
- While does not 'return' anything unless there are **Return**[ ] statements in the actions.
- Loops fine control: **Break**[ ], **Continue**[ ], **Return**[ ]

Project 5: Factors
List the factors of user input

- Read the help of `Do`. When would it be more convenient than `For`?

Vectors and Matrices

- Loops are useful for Performing the same action according to, or on, a sequence of values. But…

- If the operation is ‘linear’, it is even better if one can perform the action over a ‘vector’

- Linear algebra, by itself, is one of the most important field of applied mathematics

- List in Mathematica is useful mathematical and programming constructs.

Vectors

- Vectors are ordered lists of scalar quantities.

- Vectors can be expressed as a row of quantities or a column of quantities.

$$(x_1, x_2, \cdots, x_N)$$

A row vector

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix}$$

A column vector

Vectors

- Mathematica List
  - Default: column
  - Creating column vectors:
    ```
    v = { 1, 2, 3 };
    ```
  - Creating row vectors:
    Basically you cannot create a row vector.
  - Converting between row and column:
    Basically you do not convert vectors.

Matrix

- A matrix can be thought of as:
  - An ordered list of several column vectors, of the same length, in a row, or
  - an ordered list of several row vectors, of the same length, in a column.
- An N by M matrix has N rows and M columns. The first subscript is the row number and the second the column number.

$$\begin{pmatrix} x_{1,1} \\ x_{2,1} \\ \vdots \\ x_{N,1} \end{pmatrix} \begin{pmatrix} x_{1,2} \\ x_{2,2} \\ \vdots \\ x_{N,2} \end{pmatrix} \cdots \begin{pmatrix} x_{1,N} \\ x_{2,N} \\ \vdots \\ x_{N,N} \end{pmatrix}$$

or

$$\begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,N} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,N} \\ & & \vdots & \\ x_{N,1} & x_{N,2} & \cdots & x_{N,N} \end{pmatrix}$$

Matrix

- Create matrix:
  row-by-row

In[1]:=
```
m = { {1, 2, 3}, {4, 5, 6}}
m // MatrixForm
```
[矩陣格式

Out[1]=
```
{{1, 2, 3}, {4, 5, 6}}
```

Out[2]//MatrixForm=

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

Part of a Matrix

- Some operations should be done on part of a matrix. There are ways to access part of a matrix (vector)

Single Element in Matrix

- Vector indexing:
  `x = {3,5,7};`
  `x[[2]]` gives you 5.

- No linear indexing in matrices. Only 2D indexing: x(row, col)
  `x = {{1,2},{3,4}};`
  `x[[2,1]]` gives you 3.

Row or Column

- All rows in a column —
  `x[[ ;;, col ]]`
- All columns in a row —
  `x[[ row, ;; ]]`
- `;;` means Span and is only used in indexing.

Range in a Row or Column

- The form of ‘span’ is
  start;;end or
  start;;end;;step

- ```
  v = Range[5];
  Print[v[[1;;4]]];
  Print[v[[1;; ;;2]]];
  {1,2,3,4}
  {1,3,5}
  ```

- Allow blanks and negative indices
  ```

Project 6: Factors again
Find factors of user input in 1 step

- Try to further simplify the example program (reduce the number of lines of codes). Do you like the simplified version?

Furthermore

- Make use of the documents of Mathematica as much as possible. Read and try the examples.
- **See Also** are very important.
- Use your training in mathematics and your imagination.

Functional Programming

- Make use of pieces of codes repeatedly.
    - Save time in coding.
    - Reduce risk of bugs by localizing problems.
    - Give meaning to a set of codes.
    - Apply more advanced programming skills.

Facilities

- Notebook with cells
- Head-body structure of symbols
- Pure functions
- Iterations
- List operations
- Libraries

Mathematica Notebook

- Notebook makes Mathematica fancy to use
- Notebook is composed of cells (of various types)
- Cells do not have to be executed in order — may be buggy and misleading
- Cells do not have independent kernel — may be buggy
- A cell is like a box of chocolate, you never know what you shall get, unless you intentionally prevent your codes from mixing up
  - Various scoping constructs are provided for this purpose

Mathematica Function

- In Mathematica, there are fake functions and pure functions.

- Everything in Mathematica is a symbol. A symbol, if ever defined, has a head and a body.

- Defining fake functions is pretty similar to function definition in other languages such as MATLAB.

- Pure functions can be defined by using `Function[]` (or `&`) with slots (#) and so on.

Mathematica Function

- You need to read the Overview and Guide of Patterns to get mastered in passing input to the fake functions.

- You need to read the Tutorial of Pure Functions and the help on `Function`, `Slot` and `SlotSequence` to master pure functions

Project 8: Mathematica Fake and Pure Functions

- What does `_` (`Blank`) mean?
- What does `:=` (`SetDelayed`) mean?
- What does `#` (`Slot`) mean?

Project 9: Mathematica Function of Circle Area

- In the one-liner version, what does `@` (`Prefix`) do? Since you are not familiar with it, how can you change it to a form that you are (a bit more) familiar with?

Function over List / Array

- Linear functions automatically apply to all elements.
- For better control of the operations, `Map` series of functions and `Scan` are available. See Tutorial on Applying Functions to Parts of Expressions.

Project 10: Plot
Plot the cube of a vector

- Check the document of `ListPlot`. Why do we have to use `MapThread` to prepare data for plotting? Can you find other ways out?
- How to use `Function` instead of `Slot` (#) in the second part?

Special Topic: Differential Equations

- Mathematica solves differential equations analytically or numerically as we desired.

- To solve as many differential equations, and to solve them well, still take a lot of human knowledge and effort.

Mathematica Differential Equations

- Mathematica can solve differential equations either analytically or numerically

- Analytic solution of a DE can be used for numerical calculation

- Basic functions

  - `DSolve` : analytic solution

  - `NDSolve` : numerical solution

Mathematica Differential Equations

- Steps of working on a differential equation
  - Write down as many equations you need to solve as possible, but no more than that.
  - If necessary, provide boundary (initial) conditions
  - Solve it!

Mathematica
Defining DE

- There are three kinds of ‘differentiation’ in Mathematica
  - `D` : partial (explicit) differentiation, and total in simple cases
  - `Dt` : total differentiation, with or not with respect to specific variables
  - `Derivative` : More rigorous way of writing partial or perhaps total differentials. Without having to refer to names of variables.

Project 11: Mathematica Defining DE

- As an example, work on this equation:

  $\frac{\mathrm{d}\, y(x)}{\mathrm{d}x} = y \ln x$ and $y^{(1)}(x = e) = 1$

  - First try to solve the DE without assigning the boundary condition.

    Try to make the solution more comprehensible.

  - With the correct BC, one obtains a particular solution.

    Can you plot it?

  - In what other ways can you define BC in Mathematica?

Mathematica Numerical DE

- The DEs can still be solved numerically by `DSolve` function.
- There may be some problems using DSolve. In general, using NDSolve avoids some analytic problem.
- The format of the function call is similar to `DSolve`, but a range of the independent variable(s) needs to be given.
- All of the parameters should have values assigned.
- The return from `NDSolve` is an interpolation function, while the result of `DSolve` is a general mathematical expression.

Project 13: Mathematica Numerical integration of DE

- Use the result from the previous project as the DE for substrate concentration.

- Learn a little bit about Module.

- Define parameter values and apply `NDSolve`.

- What if you simply use `DSolve`?

- Use parameters v_{max} and K instead and try to solve the differential equation by yourself.

More on Plotting

- Visualization in MATLAB and Mathematica are fancy.
- User can customize nearly all details of the visualization.
- Mathematica uses options (as rules) to manipulate the output. Unfortunately the manipulations can only be decided before plotting.
- MATLAB uses graphics handles to manipulate the graphics, and can be done before (in plotting functions) or after the plotting is done.

Project 16: Mathematica Plotting and Manipulation

- Consider a wave $x(t) = \sin t$ with an exponentially decaying envelope $a(t) = \exp(-t/16)$. That results in a decaying wave $y(t) = \exp(-t/16)\sin t$.
- Use Plot to plot all three function in one figure.
- Make the plot in a frame instead of an open axes.
- Make some more decorations.

Conclusion

- This is a good place for me to stop, but a good place for you to start.

- What you can do is only limited by your imagination, and your knowledge about your problems.

- Good luck!