01: Introduction to MATLAB

Built-In Functions

- Type functions:
  - Real numbers: **single**, **double** (default for all numbers)
  - Integers: **int8**, **int16**, **uint32**, **uint64**
  - Unsigned integers: **uint8**, **uint16**, **uint32**, **uint64**
  - Single characters and character vectors: **char**
  - Strings of characters: **string**
  - True/false: **logical**
- Some elementary math functions:
  - Absolute value **abs(x)**
    >> abs(-4) => 4
  - Trig functions e.g., **sin(x)**, **cos(x)**, **tan(x)**, **asin(x)**, **sinh(x)**, **asinh(x)**, **sind(x)**, **asind(x)**
  - Rounding and remainder functions **fix(x)**, **floor(x)**, **ceil(x)**, **round(x)**, **rem(x)**, **mod(x)**, **sign(x)**
    - **fix(x)**: Round toward zero
    - **floor(x)**: Round toward negative infinity
    - **ceil(x)**: Round toward positive infinity
    - **round(x)**: Round to nearest decimal or integer
    - **rem(a,b)**: Remainder after division
      >> rem(23,5) => 3
    - **mod(a,m)**: Remainder after division (Modulo operation)
      >> mod(23,5) => 3
    - **sign(x)**: Return sign as -1, 0, or 1
      >> sign(-5) => -1
  - Root functions: **sqrt(x)**, **nthroot(x, N)**
    - **nthroot(x, N)**: Real nth root of real number
      >> nthroot(-27,3) => (-27)^(-3) => -3
  - Conversion between degrees and radians: **deg2rad(x)**, **rad2deg(x)**
    - **deg2rad(x)**: Convert angle from degrees to radians
      >> deg2rad(90) => 1.5708
    - **rad2deg(x)**: Convert angle from radians to degrees
      >> rad2deg(pi) => 180
  - Log functions: **log(x)**, **log2(x)**, **log10(x)**, **exp(n)**
    - **log(x)**: Returns the normal logarithm
    - **log2(x)**: Returns the base 2 logarithm
    - **log10(x)**: Returns the base 10 logarithm
    - **exp(n)**: Returns the constant e^n
- Other functions:
  - Functional form of operators, e.g., **plus(x)** for +
  - Constant functions: **pi**, **i**, **j**, **inf**, **NaN**
    - **NaN**: Not a number
      >> 0/0 => NaN
  - Random number generating functions: **rand*N, rand(m,n)**, **randn(m,n)**, **randi(imax)**, **randi([m,n], x)**
    - **rand*N**: Generate random real number in the interval (0, N)
      >> rand*5 => 4.0837
    - **rand(m,n)**: Returns m*n matrix of uniformity distributed random numbers
    - **randn(m,n)**: Returns m*n matrix of normally distributed random numbers

        - **randi(imax)**: Generate random integer in the range from 1 to imax
        - **randi([m,n], x)**: Generate one integer in the range from m to n in x*x matrix
    - **class**: returns the type
    - **format**: has many formatting options
        - **long**, **short**: Control number of decimal places
            >> format short
        - **loose**, **compact**: Control spacing between lines
    - **help**: find help topics, lists of functions in help topics, or descriptions of a function
    - **xor(A,B)**: exclusive or; true if only one argument is true
        >> xor(true false) => 1
    - Type range for integers: **intmin(x)**, **intmax(x)**
        >> intmin('int8') => -128
        >> intmax('int8') => 127
    - Types Converts: **double(x)**, **char(x)**
        - **double(x)**: If x is a character, it converts to its integer
            >> double('a') => 97
        - **char(x)**: If x is a integer, it converts to its character
            >> char(97) => 'a'

A. Variables and Assignments

Variables and Assignments

- To store a value, use a *variable*
- One way to put a value in a variable is with an *assignment statement*
- General form:
    - variable = expression
- The order is important
    - Variable name on the left
    - The assignment operator "=" (Note: this does NOT mean equality)
    - Expression on the right

Modifying Variables

- Initialize a variable (put its first value in it)
    >> mynum = 5;
- Change a variable (e.g. by adding 3 to it)
    >> mynum = mynum + 3;
- Increment by one
    >> mynum = mynum + 1;
- Decrement by two
    >> mynum = mynum - 2;

  NOTE: after this sequence, mynum would have the value 7 (5+3+1-2)

Variable names

- Names must begin with a letter of the alphabet
- After that names can contain letters, digits, and the underscore character _
- MATLAB is case-sensitive
- The built-in function **namelengthmax** tells what the limit is for the length of a variable name
- Names should be mnemonic (they should make sense!)

- The commands **who** and **whos** will show variables
- To delete variables: **clearvars**
- **clear** clears out variables and also functions

Types

- Every expression and variable has an associated type, or class
  - Real numbers: **single**, **double**
  - Integer types: numbers in the names are the number of bits used to store a value of that type
    - Signed integers: **int8**, **int16**, **int32**, **int64** ← Storing sign first
    - Unsigned integers: **uint8**, **uint16**, **uint32**, **uint64** ← Not storing the sign, so all numbers are positive.
  - Single characters and character vectors: **char**
  - Strings of characters: **string** ← "x"
  - True/false: **logical**
- The default type is **double**

'x'

Use Single quotes in character vectors, and Double quotes in the string.

Range and Type Casting

- Range of integer types found with **intmin**/**intmax**
  - e.g. **intmin('int8')** is -128, **intmax('int8')** is 127
- Converting from one type to another, using any of these names as a function, is called casting or type casting, e.g.:

```
>> num = 6+3;
>> numi = int32(num);
>> whos
      Name          Size              Bytes  Class     Attributes

      num           1*1                   8  double
      numi          1*1                   4  int32
```

- The **class** function returns the type of a variable

B. Expressions

Expressions

- Expressions can contain values, variables that have already been created, operators, built-in functions, and parentheses
- Operators include:

+	addition
–	negation, subtraction
*	multiplication
/	division (divided by e.g. 10/5 is 2)
\	division (divided into e.g. 5\10 is 2)
^	exponentiation (e.g. 5^2 is 25)
e	exponent of 10 raised to a power (e.g. 3e5 is $3*10^5$)

Relational Expressions

>	greater than
<	less than
>=	greater than or equals
<=	less than or equals

==	equality
~=	inequality

Logical Operators

||	or for scalars
&&	and for scalars
~	not

Operator Precedence

- Some operators have precedence over others
- Precedence list (highest to lowest) so far:

( )	parenthesis
^	exponentiation
-, ~	negation, not
*, /, \	all multiplication and division
+, -	addition and subtraction
<, >, <=, >=, ==, ~=	relational expression
&&	and
||	or
=	assignment

- Nested parenthesis: expressions in inner parentheses are evaluated first