ÖZYEĞİN UNIVERSITY

CS 100

Introduction to Engineering Computation

Gonca GURSUN and Cevat BALEK

Slides courtesy of
Prof. M. Oguz Sunay

MATLAB OCTAVE

What is MATLAB

- MATLAB is a high-performance language for technical computing.
- It integrates computation, visualization and programming in an easy to use environment where problems and solutions are expressed in familiar mathematical notation

What is MATLAB?

- Typical uses include
  - Math and computation
  - Algorithm development
  - Data acquisition
  - Modeling, simulation and prototyping
  - Data analysis, exploration and visualization
  - Scientific and engineering graphics
  - Application environment

What is MATLAB?

- The name MATLAB stands for matrix laboratory
- MATLAB is an interactive system whose basic data element is an array that does not require dimensioning
- This allows us to solve many technical problems, especially those with matrix and vector formulations very quickly.

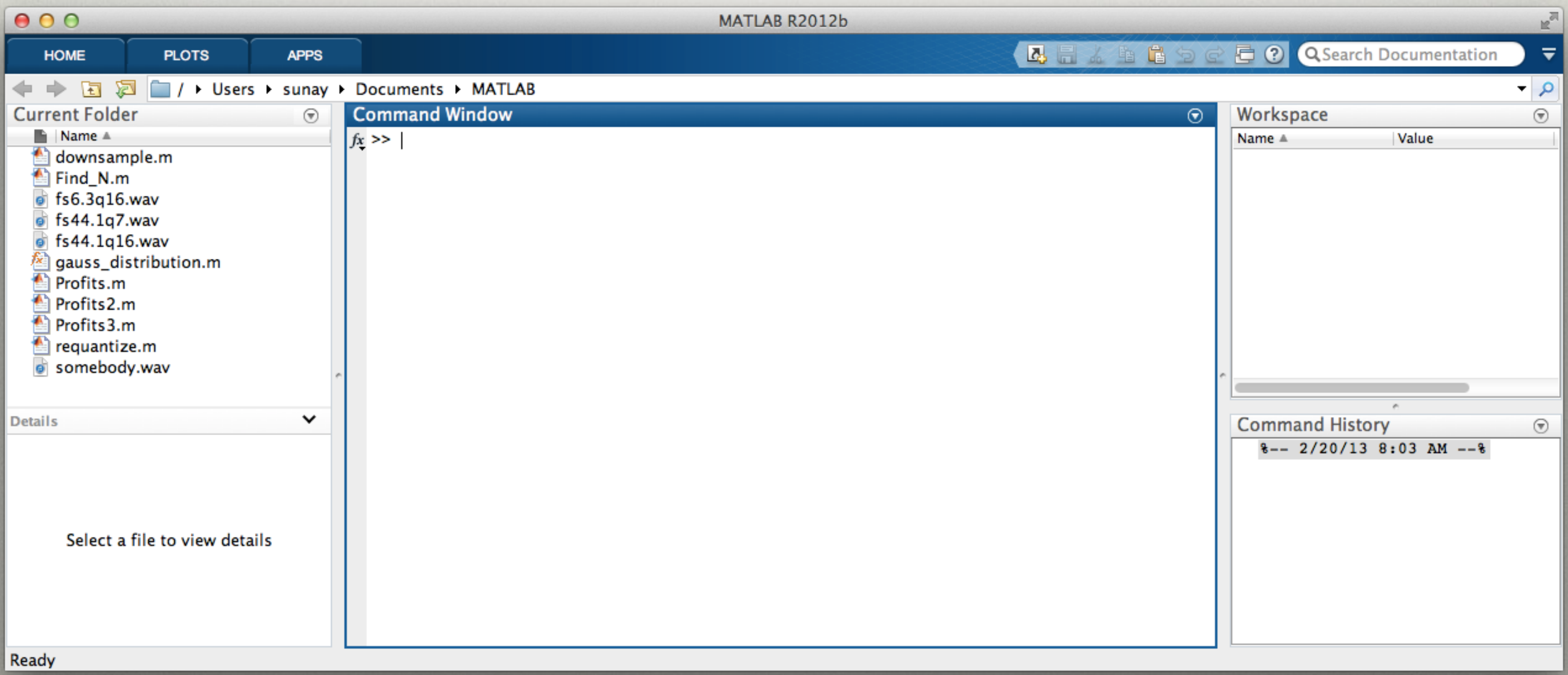
What IS MATLAB?

- MATLAB features a family of add-on application-specific solutions called toolboxes. Toolboxes allow us to learn and apply specialized technology.

Octave

- GNU Octave is a high-level interpreted language, primarily intended for numerical computations.
- Octave is almost identical to MATLAB and almost all programs are directly portable.

MATLAB DESKTOP



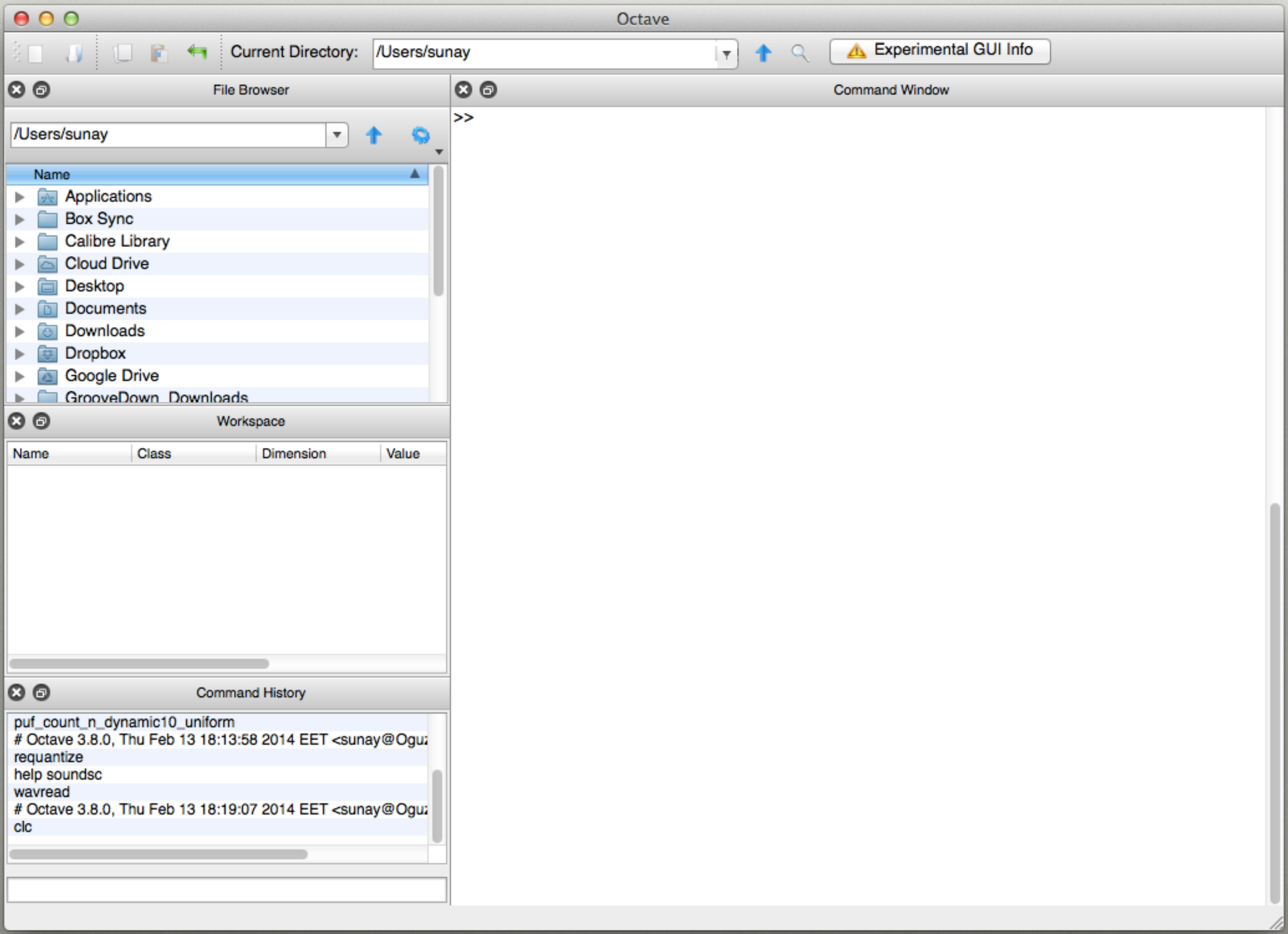
downsample.m
Find_N.m
fs6.3q16.wav
fs44.1q7.wav
fs44.1q16.wav
gauss_distribution.m
Profits.m
Profits2.m
Profits3.m
requantize.m
somebody.wav

Select a file to view details

```
fx >>
```

```
%-- 2/20/13 8:03 AM --%
```

OCTAVE DESKTOP



```
>>
```

```
puf_count_n_dynamic10_uniform
# Octave 3.8.0, Thu Feb 13 18:13:58 2014 EET <sunay@Oguz
requantize
help soundsc
wavread
# Octave 3.8.0, Thu Feb 13 18:19:07 2014 EET <sunay@Oguz
clc
```

MATLAB/OCTAVE Desktop

- The following desktop components are observed:

1. Command Window
2. Command History
3. Launch Button
4. Workspace

  - For the moment, we are only interested in the Command Window.

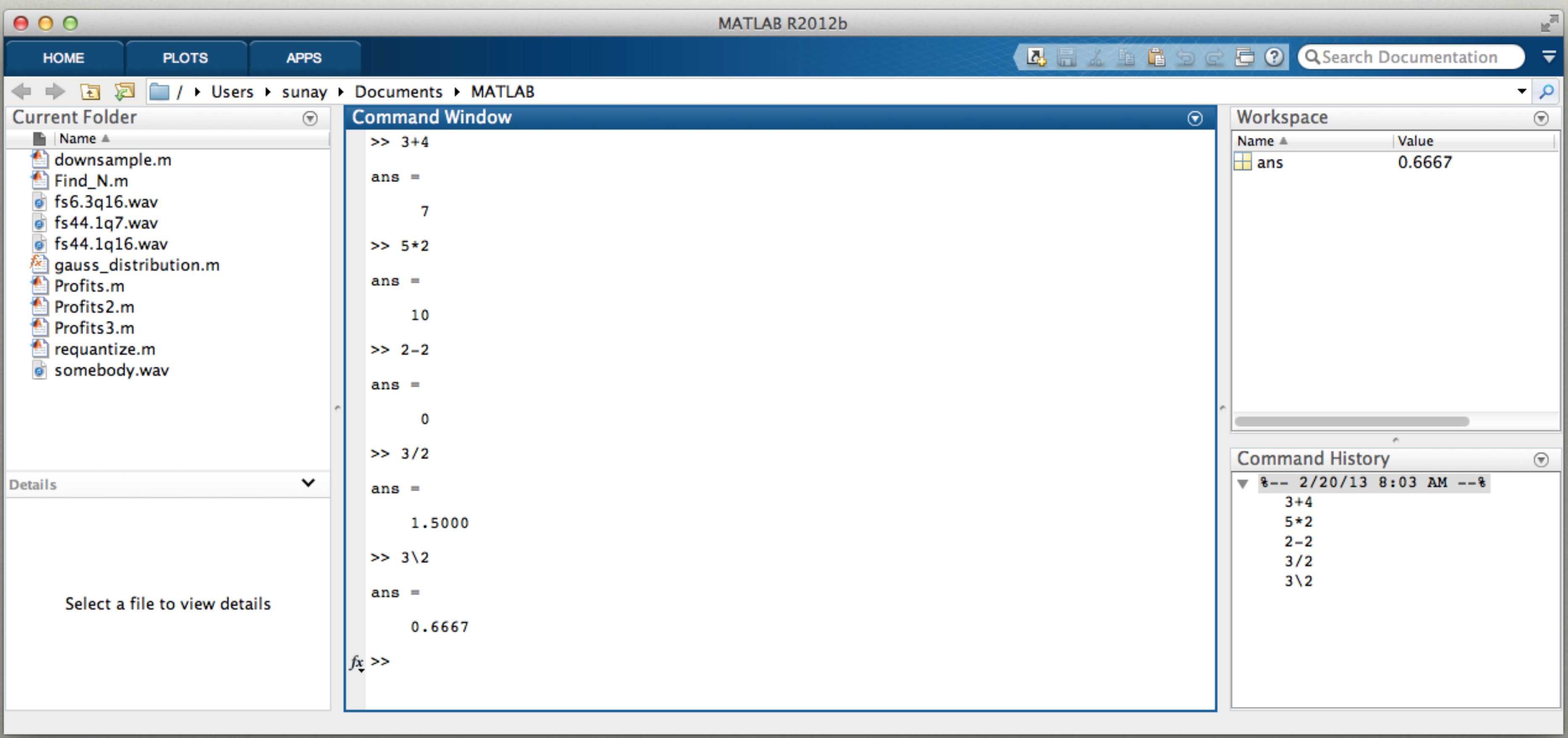
MATLAB/OCTAVE Desktop

- In the command window

  ```
  >>
  ```

  is waiting for you to enter a command. It is possible to use MATLAB as a scientific calculator with a wide-variety of built-in functions.

MATLAB/OCTAVE AS A CALCULATOR

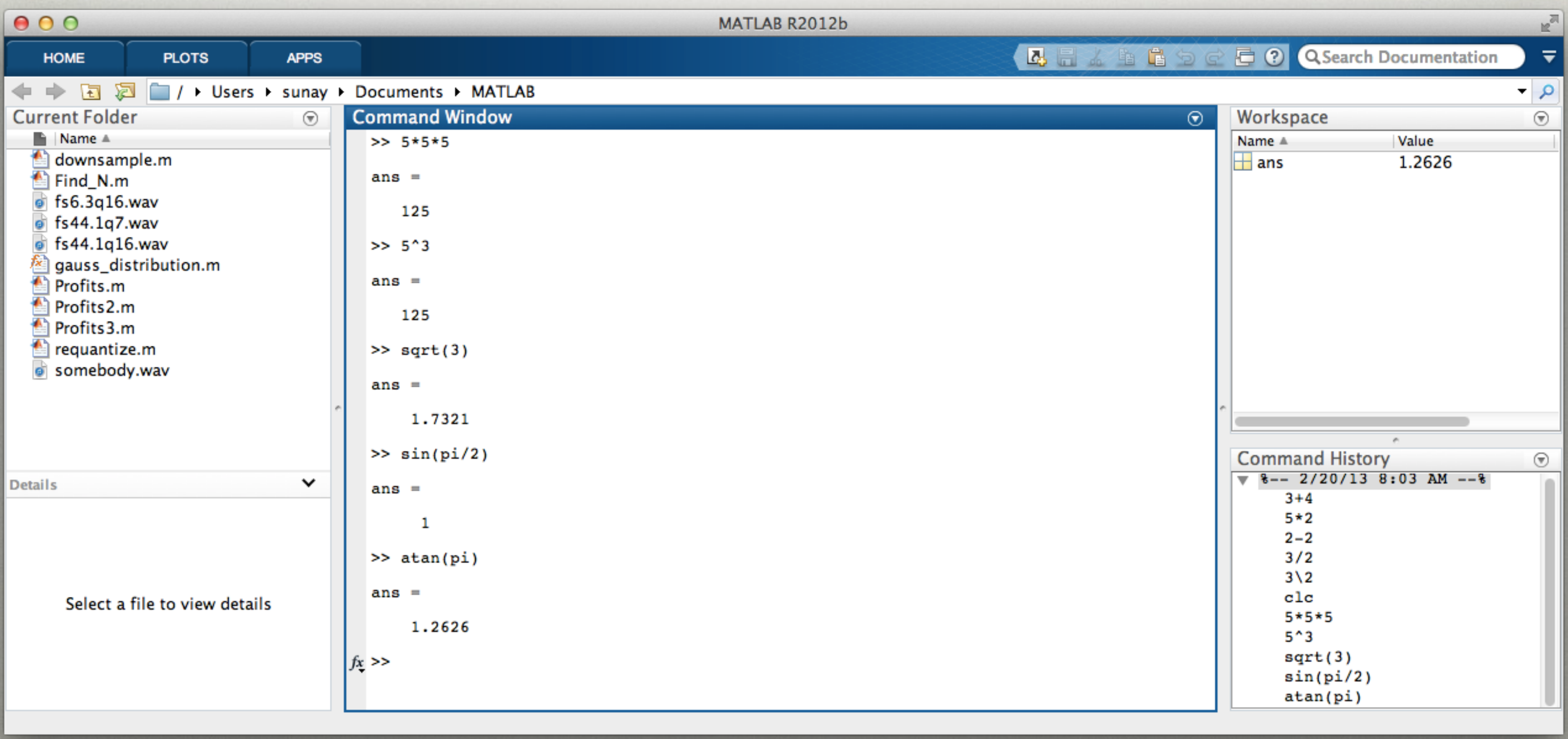


```
>> 3+4

ans =

     7

>> 5*2

ans =

    10

>> 2-2

ans =

     0

>> 3/2

ans =

    1.5000

>> 3\2

ans =

    0.6667

>>
```

Name	Value
ans	0.6667

```
%-- 2/20/13 8:03 AM --%
   3+4
   5*2
   2-2
   3/2
   3\2
```

MATLAB/OCTAVE as a Calculator



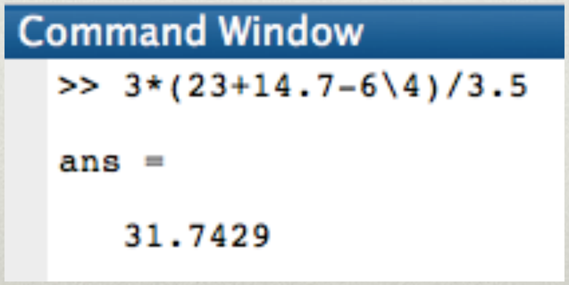
```
>> 5*5*5

ans =

   125

>> 5^3

ans =

   125

>> sqrt(3)

ans =

    1.7321

>> sin(pi/2)

ans =

     1

>> atan(pi)

ans =

    1.2626

>>
```

Name	Value
ans	1.2626

```
%-- 2/20/13 8:03 AM --%
   3+4
   5*2
   2-2
   3/2
   3\2
   clc
   5*5*5
   5^3
   sqrt(3)
   sin(pi/2)
   atan(pi)
```

MATLAB/OCTAVE as a Calculator

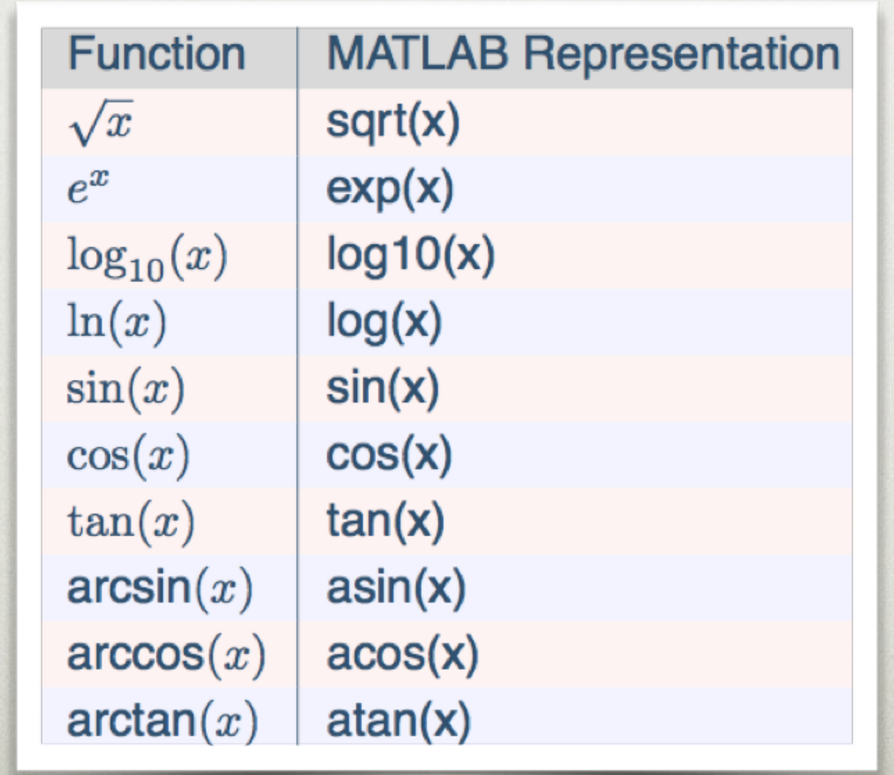
- When you want to calculate a more complex expression use parentheses, in the usual way, to indicate precedence. If parentheses are not used, recall that multiplication and division operators take precedence over the addition and subtraction operators. For example:



```
>> 3*(23+14.7-6\4)/3.5

ans =

   31.7429
```

MATLAB/OCTAVE Built-IN Functions

- MATLAB has a useful set of built-in elementary functions. Here is a list of some of them.



Function	MATLAB Representation
$\sqrt{x}$	sqrt(x)
e^x	exp(x)
$\log_{10}(x)$	log10(x)
$\ln(x)$	log(x)
$\sin(x)$	sin(x)
$\cos(x)$	cos(x)
$\tan(x)$	tan(x)
arcsin(x)	asin(x)
arccos(x)	acos(x)
arctan(x)	atan(x)

Primitive Data Types

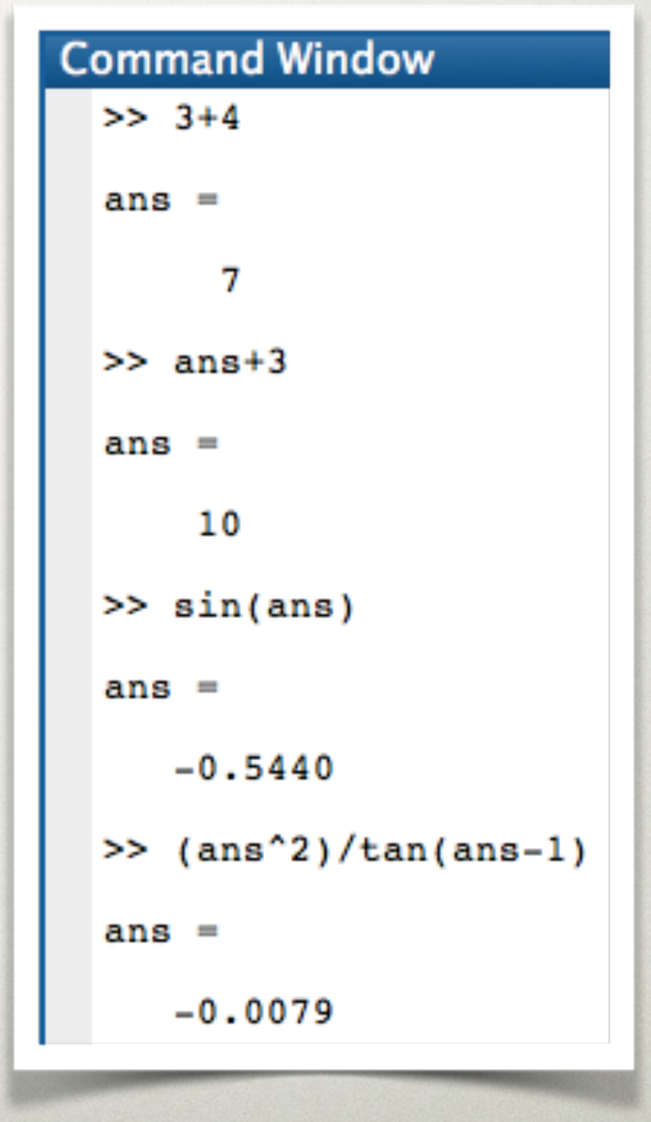
`int`	This type is used to represent integers, which are whole numbers such as 17 or −53.
`double`	This type is used to represent numbers that include a decimal fraction, such as 3.14159265. Such values are called floating-point numbers; the name `double` comes from the fact that the representation uses twice the minimum precision.

By default, MATLAB stores all numeric variables as double-precision floating-point values

`boolean`	This type represents a logical value (`true` or `false`).
`char`	This type represents a single character.

Constants & Variables

- The contents of the variable change with each calculation.
- We can use this property when we want to carry a sequence of calculations.
- Observe the following:

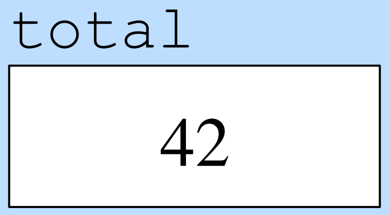


```
>> 3+4

ans =

     7

>> ans+3

ans =

    10

>> sin(ans)

ans =

   -0.5440

>> (ans^2)/tan(ans-1)

ans =

   -0.0079
```

Constants and Variables

- The simplest terms that appear in expressions are constants and variables. The value of a constant does not change during the course of a program. A variable is a placeholder for a value that can be updated as the program runs.

- A variable in Matlab is most easily envisioned as a box capable of storing a value.



(contains an `int`)

- Each variable has the following attributes:
  - A name, which enables you to differentiate one variable from another.
  - A type, which specifies what type of value the variable can contain.
  - A value, which represents the current contents of the variable.

- The value of the variable changes whenever you assign a new value to it.

Defining Variables

- In Matlab, you must define a variable before you can use it. This means establishing the name of the variable and, in most cases, specifies the initial value as well.

- The most common form of a variable declaration is

```
name = value;
```

where *name* is an identifier that indicates the name of the variable, and *value* is an expression specifying the initial value.

Matlab Identifiers

- Names for variables (and other things) are called identifiers.
- Identifiers in Matlab conform to the following rules:
  - A variable name must begin with a letter, followed by letters, digits, or underscore character.
  - The name must not be one of Matlab's reserved words: *if, else, for* etc.
  - Identifiers should make their purpose obvious to the reader.
  - Identifiers should adhere to standard conventions. Variable names, for example, should begin with a lowercase letter.

Use of Variables

```
(30 + 40) * (30-40+2*(30*30*30*40))

OR

x = 30
y = 40
(x + y) * (x-y+2*(x*x*x*y))
```

Operators and Operands

- As in most languages, Matlab programs specify computation in the form of arithmetic expressions that closely resemble expressions in mathematics.
- The most common operators are the ones that specify arithmetic computation:

+	Addition	*	Multiplication
–	Subtraction	/	Division

- Operators usually appear between two subexpressions, which are called its operands. Operators that take two operands are called binary operators.
- The - operator can also appear as a unary operator, as in the expression `-x`, which denotes the negative of `x`.

Expressions

- You can change the value of a variable in your program by using an assignment statement, which has the general form:

```
variable = expression;
```

- An *expression* is any legal combination of symbols that represents a *value*.

  It can be a combination of explicit values, constants, variables, operators, and functions.

- Each programming language (PL) has its own rules for what is legal and illegal.

Assignment Statements

- You can change the value of a variable in your program by using an assignment statement, which has the general form:

```
variable = expression;
```

- The effect of an assignment statement is to compute the value of the expression on the right side of the equal sign and assign that value to the variable that appears on the left. Thus, the assignment statement

```
    total = total + value;
```

adds together the current values of the variables `total` and `value` and then stores that sum back in the variable `total`.

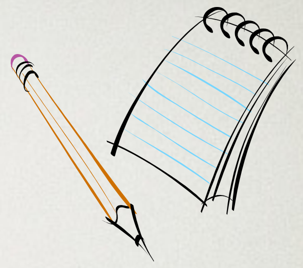
- When you assign a new value to a variable, the old value of that variable is lost.

Use of Variables

```
x = 10
y = 50
x * h
```

```
x = x + 10
x = x + 10

x * y
```

Exercise: Assignment Statements



- What would the value of total be at the end of the following set of statements?

```
num = 20;
total = num - 5;
total = total * num;
num = num - 1;
total = total - num;
total = total * 2;
```

```
A) 320

B) -4

C) 30

D) Err… I don't know.

E)  562

F)  None of the above
```

Designing for Change

- While it is clearly necessary for you to write programs that the compiler can understand, good programmers are equally concerned with writing code that *people* can understand.
- The importance of human readability arises from the fact that programs must be maintained over their life cycle. Typically, as much as 90 percent of the programming effort comes *after* the initial release of a system.
- There are several useful techniques that you can adopt to increase readability:
  - Use names that clearly express the purpose of variables and methods
  - Use proper indentation to make the structure of your programs clear
  - Use named constants to enhance both readability and maintainability

Precedence

- If an expression contains more than one operator, Matlab uses precedence rules to determine the order of evaluation. The arithmetic operators have the following relative precedence:

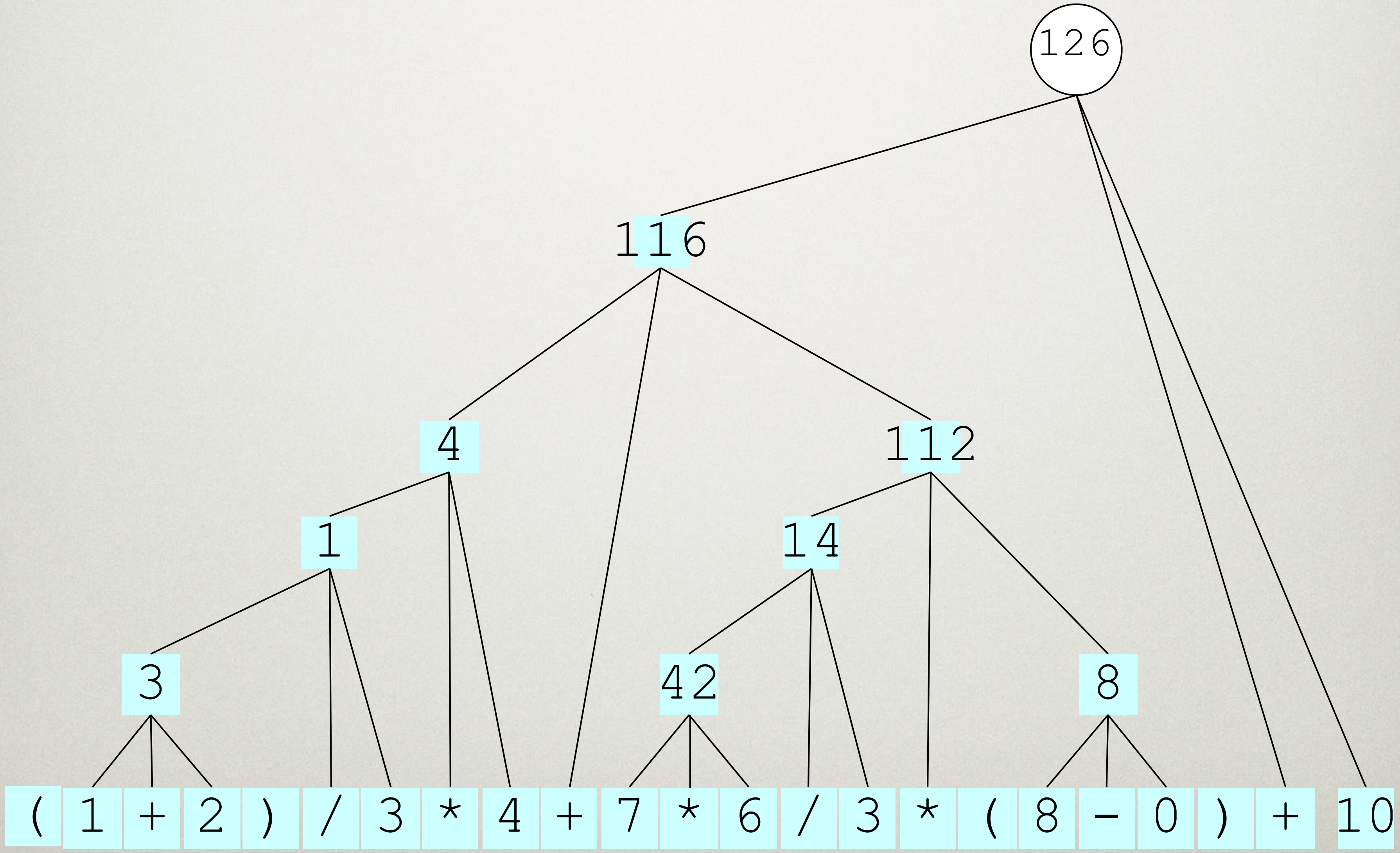
unary −	*highest*
* /	↕
+ −	*lowest*

Thus, Matlab evaluates unary − operators first, then the operators *, /, and then the operators + and −.

- Precedence applies only when two operands compete for the same operator. If the operators are independent, Matlab evaluates expressions from left to right.
- Parentheses may be used to change the order of operations.

Exercise: Precedence Evaluation

What is the value of the expression at the bottom of the screen?



Boolean Expressions

George Boole (1791-1871)

In many ways, the most important primitive type is `boolean`, even though it is by far the simplest. The only values in the `boolean` domain are `true` and `false`, but these are exactly the values you need if you want your program to make decisions.

The name `boolean` comes from the English mathematician George Boole who in 1854 wrote a book entitled *An Investigation into the Laws of Thought, on Which are Founded the Mathematical Theories of Logic and Probabilities*. That book introduced a system of logic that has come to be known as *Boolean algebra*, which is the foundation for the `boolean` data type.

Boolean Operators

- The operators used with the `boolean` data type fall into two categories: relational operators and logical operators.
- There are six relational operators that compare values of other types and produce a `boolean` result:

`==`	Equals	`~=`	Not equals
`<`	Less than	`<=`	Less than or equal to
`>`	Greater than	`>=`	Greater than or equal to

For example, the expression `n <= 10` has the value `true` if `n` is less than or equal to 10 and the value `false` otherwise.

Assume x is 3, y is 5.

Expression	Value
`x > 0`	`true`
`x < 3`	`false`
`x > 3`	`false`
`x >= 3`	`true`
`x <= 3`	`true`
`x <= y`	`true`
`x == y`	`false`
`x ~= y`	`true`

Assume x is 3, y is 5.

Can use arithmetic operations.

Expression	Value
`x - y > 0`	`false`
`(x - 2) == 1`	`true`
`((x+1) * 2) == 0`	`FALSE`
`x + y >= 8`	`true`

Boolean Operators

- There are also three logical operators:

`&&`	Logical AND	`p && q` means both `p` and `q`
`||`	Logical OR	`p || q` means either `p` or `q` (or both)
`~`	Logical NOT	`~p` means the opposite of `p`

Boolean Operators

`true && true`	`true`
`false && true`	`false`
`true && false`	`false`
`false && false`	`false`

`true || true`	`true`
`false || true`	`true`
`true || false`	`true`
`false || false`	`false`

`~ true`	`false`
`~ false`	`true`

Assume x is 3, y is 5.

Expression	Value
`x > 0 && x < 5`	[redacted]
`x < 3 || x > 3`	[redacted]
`x < 3 || x >= 3`	[redacted]
`~(x > y)`	[redacted]
`~(y - x > 0)`	[redacted]

Boolean Expressions

```
x = 3;
y = 5;

b = x - y > 0;


b = x > 0 && x < 5;

b = x < 3 || x > 3;

b = ~(x > y);

b = ~(y - x > 0);
```

Notes on the Boolean Operators

- Remember that Matlab uses = to denote assignment. To test whether two values are equal, you must use the == operator.
- It is legal in Matlab to use more than one relational operator in a single comparison as is often done in mathematics. To express the idea embodied in the mathematical expression

$$0 \leq x \leq 9$$

but not a good practice. Instead use the following:

```
0 <= x && x <= 9
```

- The `||` operator means *either or both*, which is not always clear in the English interpretation of *or*.
- Be careful when you combine the `~` operator with `&&` and `||` because the interpretation often differs from informal English.

Short-Circuit Evaluation

- Matlab evaluates the `&&` and `||` operators using a strategy called short-circuit mode in which it evaluates the right operand only if it needs to do so.
- For example, if `n` is 0, the right hand operand of `&&` in

```
n ~= 0 && x/n == 1
```

is not evaluated at all because `n ~= 0` is `false`. Because the expression

`false &&` *anything*

is always `false`, the rest of the expression no longer matters.

- One of the advantages of short-circuit evaluation is that you can use `&&` and `||` to prevent execution errors. If `n` were 0 in the earlier example, evaluating `x / n` would cause a “division by zero” error.

Short-Circuit Evaluation

```
x = 10;
y = 0;
b1 = (y!= 0) && x/y > 1;

b2 = false && x/y == 4;

b3 = true || x/y == 4;

b4 = x/y > 1;
```