



Variables

Content

- What is a variable?
- Data types
- Mutability (constant vs variable)
- Scope and visibility
- Preallocation

What is a variable

- **Location** where a value is stored

Variable: a 

Address	Value
000	NULL
001	5
002	8
003	NULL

What is a variable

- **Location** where a value is stored
- Every variable has:
 - Type (statically typed languages)
 - Variable name (identifier)
 - Value

```
int a = 5;
```

Data types

Name	Shorthand	What
Integer	int	Whole numbers
Floating-point number	float	32-bit number with digits after decimal place
Double precision float	double	64-bit number with digits after decimal place
Boolean (logical in MATLAB)	bool (logical in MATLAB)	Value that's either true or false (0 or 1)
Array (1D is a vector)	~	A list of variables, depending on the language this can be further divided into lists, arrays, vectors, queues, etc.
Character	char	A single (ASCII) character
String	string or str	A list or array of characters

Typing

- **Statically typing**
 - Languages: C (Igor), Java, PEARL
 - Variable has one type
 - Type preassigned

```
int a = 5;
```

Typing

- **Statically typing**
 - Languages: C (Igor), Java, PEARL
 - Variable has one type
 - Type preassigned
- **Dynamically typing**
 - Languages: MATLAB, Python, R, Perl
 - Value has one type
 - Type of variable may change

```
a = 5
```

Type casting

- Explicitly converting type of a variable or value
- Only in dynamically typed languages
- Especially relevant for numerical types
- Reasons for casting:
 - Increase precision/memory
 - Reduce precision
 - Cast base type to derived type

Mutability

- Variable
 - Value dynamically assigned
 - Value may change during run-time
- Constant
 - Value assigned at initialization
 - Value cannot be changed

```
const float PI = 3.1415927;
```

Mutability

- Depending on the language more subtypes available
- Other languages don't have constant modifiers
 - Workarounds if needed
 - Use naming conventions: ALL_CAPS

```
MY_CONSTANT = 3.1415927
```

Mutability: why use constants?

- Users are stupid!
- Reminder for yourself
- Physical constants (π , N of Avogadro, etc)

Scope

- Lifetime of a variable
- MATLAB is an exception
- Important for memory allocations

Scope

```
1  for(int i = 0; i < 10; i++) {  
2      System.out.println(i);  
3  }  
4  
5  System.out.println(i);
```

Output: ?



Output: ?



What is happening?

Initialization:
i is created

Memory for an int
allocated

```
1  for(int i = 0; i < 10; i++) {  
2      System.out.println(i);  
3  }  
4  
5  System.out.println(i);
```

Memory cleared:
Space in memory
available for new
assignments

i no longer exists

Function example

```
1 void main () {
2     /* Local variable definition */
3     int a = 100;
4     res = myFunction(a);
5     print(res);
6     print(a);
7 }
8
9 int myFuntion(int a) {
10     a = a + 100;
11     return a;
12 }
```

← Output: ?
← Output: ?

Levels of scope

- Blocks (not all languages!)
- Functions
- Module/Class
- Global

Scope modifiers

- Most language allow you to make a variable global
 - Automatic if outside function
 - Only use in dire need!

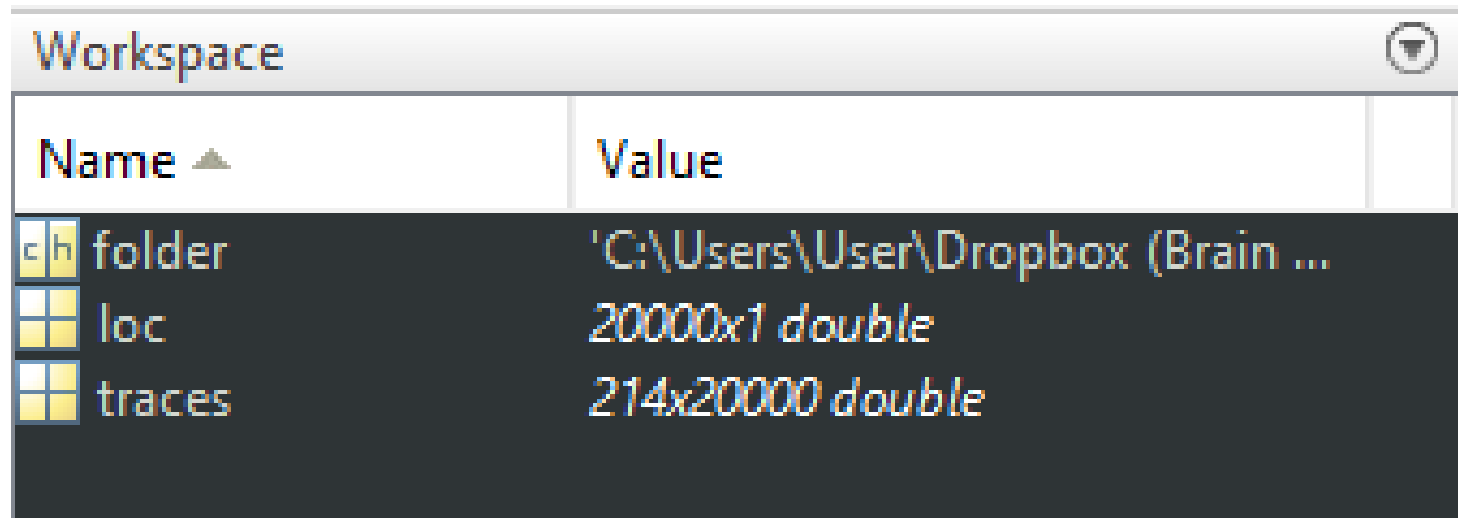
```
global i;  
i = 0;
```

```
addValue(5);  
disp(i)  
addValue(3);  
disp(i)
```

```
function addValue(v)  
    global i;  
    i = i + v;  
end
```

MATLAB

- Workspace shows scope
- Variables don't get deleted after for-loop
- Each function has it's own workspace




The screenshot shows the MATLAB Workspace window with a table of variables. The table has two columns: 'Name' and 'Value'. The variables listed are 'folder', 'loc', and 'traces'. The 'loc' variable is highlighted in blue, and a purple arrow points from the text 'Variables initiated, so available in scope' to it.

Name ▲	Value
folder	'C:\Users\User\Dropbox (Brain ...
loc	20000x1 double
traces	214x20000 double

Variables initiated, so available in scope

A word on preallocation

- You might have gotten this error (MATLAB):

 The variable 'allFspcell' appears to change size on every loop iteration (within a script). Consider preallocating for speed. [Details ▲](#)

Explanation

The size of the indicated variable or array appears to be changing with each loop iteration. Commonly, this message appears because an array is growing by assignment or concatenation. Growing an array by assignment or concatenation can be expensive. For large arrays, MATLAB must allocate a new block of memory and copy the older array contents to the new array as it makes each assignment.

- What's going on here?

Allocation example

Allocate space for
empty list (say 8 bit)



```
newArray = [];  
for i = 1:10  
    newArray = [newArray, i];  
end
```

- Add int to the allocated space, so need 8 bits more
- Memory not always linear!



What to do?

- No problem if small array
- Preallocate the largest possible number of values

```
newArray = nan(10,1);
```

What to do?

- No problem if small array
- Preallocate the largest possible number of values
- Preallocate in chunks (i.e. every 1000 values add an extra 1000 values)

Summary

- **What is a variable?**
 - **Location** where a value is stored
- **Data types**
 - Statically vs dynamically
 - Type casting
- **Mutability (constant vs variable)**
 - Single value for a variable
- **Scope and visibility**
 - Memory allocation
 - MATLAB: use functions!
- **Preallocation**
 - May speed up your code