The Matrix Laboratory: MATLAB

Prof Matthew Fricke

Version 1.0. Send corrections to mfricke@unm.edu.

You have all been using functions already.

3 reasons:

1) You are learning programming principles not Python alone
2) MATLAB is better at some things than Python and vice versa
3) The UNM engineering school uses MATLAB

Origins

Cleve Moler, UNM Professor

Started in 1983 and now a billion dollar/yr company

Logistics

Look for the email Mathworks sent you inviting you to CS151

There are only 4 problems assigned

Office hours if you have a UNM approved deadline extension

Returning exams on Monday

MATLAB

Most programming languages support variables, operators, iteration, and functions.

You have already seen examples of these in Python

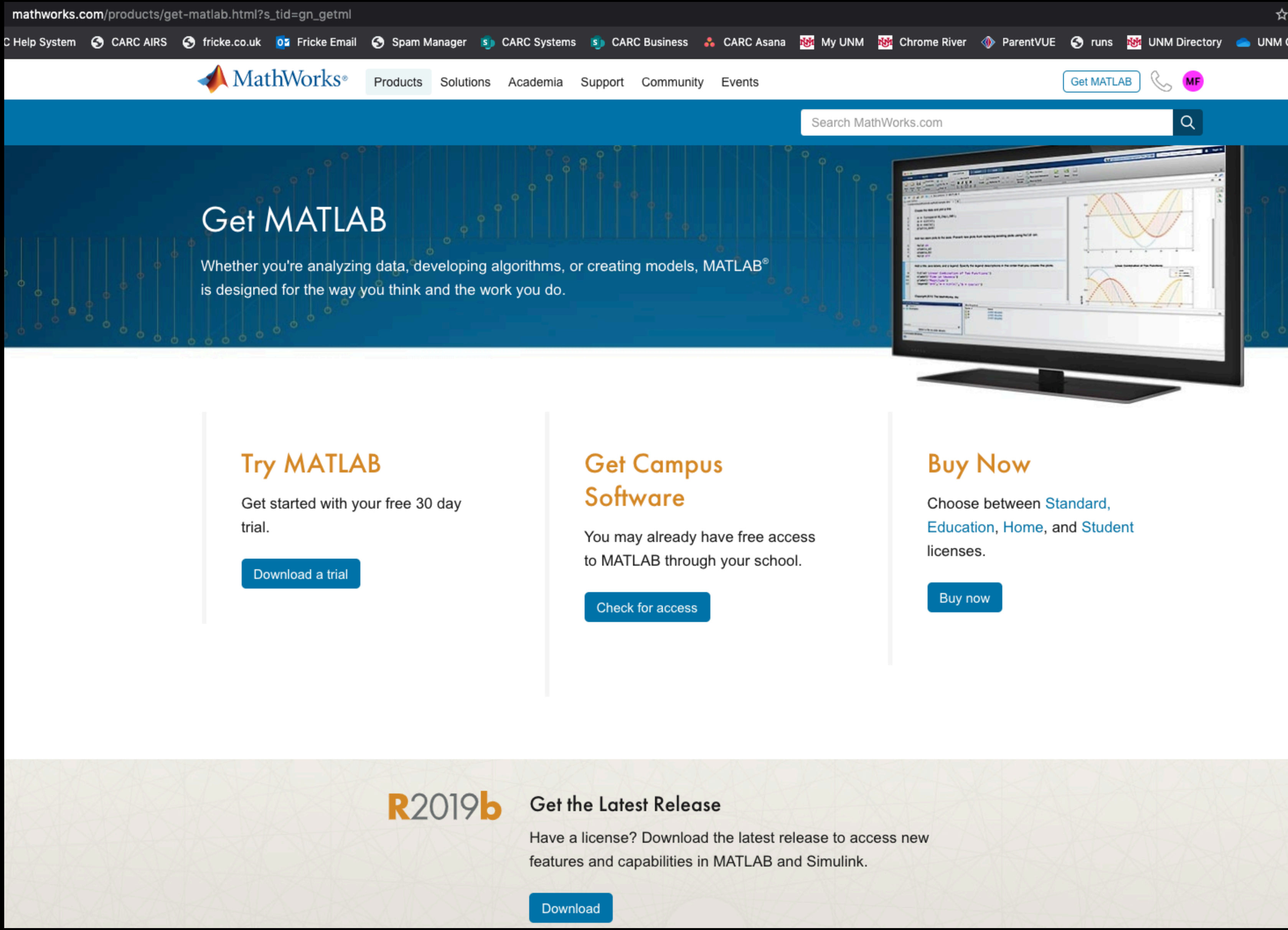
MATLAB supports all these too but the syntax is different

You can get it for free as a UNM student.

Go to http://mathworks.com

And click Get Matlab

Then click Check for Access



Get MATLAB

Whether you're analyzing data, developing algorithms, or creating models, MATLAB® is designed for the way you think and the work you do.

Try MATLAB

Get started with your free 30 day trial.

Download a trial

Get Campus Software

You may already have free access to MATLAB through your school.

Check for access

Buy Now

Choose between Standard, Education, Home, and Student licenses.

Buy now

R2019b

Get the Latest Release

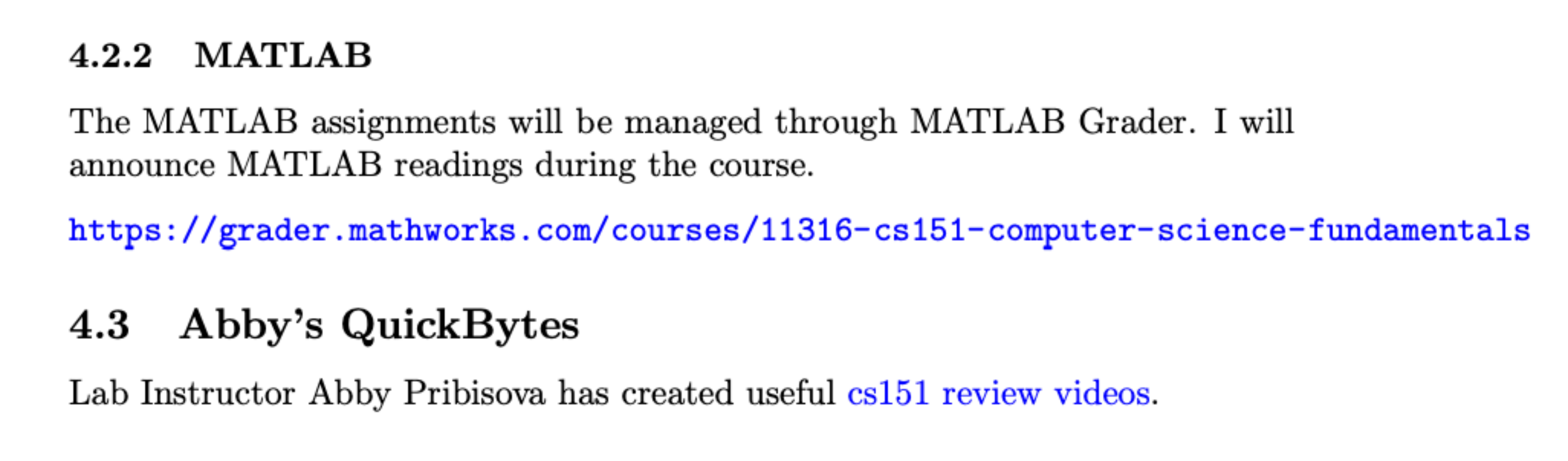
Have a license? Download the latest release to access new features and capabilities in MATLAB and Simulink.

Download

MATLAB Syntax – Homework

Link to the MATLAB homework is on the course syllabus

http://fricke.co.uk



4.2.2 MATLAB

The MATLAB assignments will be managed through MATLAB Grader. I will announce MATLAB readings during the course.

https://grader.mathworks.com/courses/11316-cs151-computer-science-fundamentals

4.3 Abby's QuickBytes

Lab Instructor Abby Pribisova has created useful cs151 review videos.

MATLAB Numeric Types

double	Double-precision Floating point
single	Single-precision Floating point
int8	8-bit signed integer -- minus 2^7 to 2^7-1
int16	16-bit signed integer -- minus 2^{15} to 2^{15}-1
int32	32-bit signed integer -- minus 2^{32} to 2^{31}-1
int64	64-bit signed integer -- minus 2^{63} to 2^{63}-1
uint8	8-bit unsigned integer -- zero to 2^8-1
uint16	16-bit unsigned integer -- zero to 2^{16}-1
uint32	32-bit unsigned integer -- zero to 2^{32}-1
uint64	64-bit unsigned integer -- zero to 2^{64}-1

MATLAB Syntax – Flow Control: Types

MATLAB Syntax in Yellow

```
x = 3; y = 3.4,
z = "some text; a= 'some text'
whos x
whos y
whos z
whos a

Name   Size   Bytes  Class
x       1x1     8  double
Name   Size   Bytes  Class
y       1x1     8      double
Name   Size   Bytes  Class
z       1x1    156  string
Name   Size   Bytes  Class
a       1x9    18        char
```

Python Syntax in Green

```
>>> type(3)
<class 'int'>
>>> type(3.4)
<class 'float'>
>>> type("some text")
<class 'str'>
>>> type('some text')
<class 'str'>
```

MATLAB Syntax – Array/Vector Indexing

MATLAB Syntax in Yellow

Array

```
x = [1,2,3,4]
x(1)
ans =

     1
```

Python Syntax in Green

Array

```
x = [1,2,3,4]
x[1]
>>> 2
```

MATLAB Syntax – Slicing Arrays

MATLAB Syntax in Yellow

Slicing

```
x = [5,6,7,8,9]
>> x(2:4)

ans =

    6    7    8
```

Python Syntax in Green

Array

```
x = [1,2,3,4]
>>> x[1:4]
[6, 7, 8]
```

MATLAB Syntax – Flow Control: IF … Else…

MATLAB Syntax in Yellow

```
if <Boolean expression>
statements
elseif < Boolean expression>
statements
else
        statements
end
```

Python Syntax in Green

```
if <Boolean expression>:
        statements
elif < Boolean expression>
        statements
else
        statements
```

MATLAB Syntax – Flow Control: IF … Else…

MATLAB Syntax in Yellow

```
if x < 5
        display(“Yes”)
elseif x > 10
display(“No”)
else
        display(“Maybe”)
end
```

Python Syntax in Green

```
if x < 5:
        print(“Yes”)
elif x > 10 :
        print(“No”)
else:
        print(“Maybe”)
```

MATLAB Syntax – Flow Control: For Loops

MATLAB Syntax in Yellow

```
total = 0
for i = 1:10
   total = total+i
end
```

Python Syntax in Green

```
total = 0
for i in
range(1,10):
   total = total+i
```

MATLAB Syntax – Flow Control: For Loops

MATLAB Syntax in Yellow

```
total = 0
for i = 1:10
   total = total+i
end

display(total)
```

```
55
```

Python Syntax in Green

```
total = 0
for i in
range(1,10):
   total = total+i

print(total)
```

```
45
```

MATLAB Syntax – Flow Control: While Loops

MATLAB Syntax in Yellow

```
n = 10;
f = n;
while n > 1
n = n-1;
 f = f*n;
end
```

Python Syntax in Green

```
n = 10
f = n
while n > 1:
    n = n-1
    f = f*n
end
```

MATLAB Syntax – Flow Control: While Loops

MATLAB Syntax in Yellow

```
n = 10;
f = n;
while n > 1
n = n-1;
 f = f*n;
end

display(['n! = ‘,num2str(f)])
```

```
n! = 3628800
```

Python Syntax in Green

```
n = 10
f = n
while n > 1:
    n = n-1
    f = f*n
end

print('n! = ‘ + str(f))
```

```
n! = 3628800
```

MATLAB Syntax – Functions

MATLAB Syntax in Yellow

```
function value = function_name( args )
statement;
Statement;
end
```

Python Syntax in Green

```
def function_name( args ):
    statement
    statement
    return value
```

MATLAB Syntax – Functions

MATLAB Syntax in Yellow

```
function f=myfact( n )
  n = 10;
 f = n;
while n > 1
   n = n-1;
    f = f*n;
end

end
```

Saved in a file called myfact.m

Python Syntax in Green

```
def myfact( n ):
    f = n
    while n > 1:
        n = n-1
        f = f*n
    return f
```

Saved in a file called myfact.py

MATLAB Syntax – Functions

MATLAB Syntax in Yellow

```
function f=fact( n )
  n = 10;
 f = n;
while n > 1
   n = n-1;
    f = f*n;
end

end
```

```
>> myfact(10)

ans = 3628800
```

Python Syntax in Green

```
def myfact( n ):
    f = n
    while n > 1:
        n = n-1
        f = f*n
    return f
```

```
>> import myfact
>> myfact(10)
3628800
```