PYTHON BASIC SYNTAX

```
>>> x = 1
>>> y = [1,2,3,4,5] # list
>>>
>>> range(1,6,2)
[1,3,5]
>>>
>>> y[0]    # zero-indexed
1
```

MATLAB BASIC SYNTAX

```
>> x = 1;
>> y = [1 2 3 4 5];  % row vector
>> z = [1 2 3 4 5]'; % column vector
>> 1:2:6
ans =
     1    3    5
>> y(1)    % one-indexed
ans =
     1
```

OPERATORS & NUMBERS

BOOLEAN OPERATORS

	+	-	*	/	**		:	%	#	=	1j	np.nan	np.inf	==	<	>	<=	>=	!=	and	or	not
	+	-	.*	./	.^	.\	:	mod(*x*,*n*)	%	=	i	NaN	Inf	==	<	>	<=	>=	~=	&&	||	~

SPECIAL FUNCTIONS

from numpy import *   # many of these functions also available in math

	sin	cos	tan	sinh	cosh	tanh	arcsin	arcsinh	exp	expm1	deg2rad	rad2deg	log	log10	sqrt
	sin	cos	tan	sinh	cosh	tanh	asin	asinh	exp	expm1	deg2rad	rad2deg	log	log10	sqrt
	sind	cosd	tand				asind								

from scipy.special import *

	j0	jn	jv	y0	yn	yv	gamma	erf	erfc	hyp2f1	binom	poch	airy		
	besselj			bessely			gamma	erf	erfc	hypergeom	nchoosek		airy	isprime	nthroot

ARRAYS & OPERATIONS

from numpy import *

	array(((a,b,c),(d,e,f)))	eye(*n*)	zeros((*m*,*n*))	ones((*m*,*n*))	rand	empty((*m*,*n*))	None
	[a b c; d e f]	eye(*n*)	zeros(*m*,*n*)	ones(*m*,*n*)	rand		[]

	linspace	arange(*a*,*b*,*d*)	diag	vstack	hstack	meshgrid	reshape	ravel	tile
	linspace	*a*:*d*:*b*	diag	vertcat	horzcat	meshgrid	reshape		repmat

NumPy also provides *universal functions*.

PLOTTING

```
import numpy as np
import scipy.special
x = np.linspace(0,6,201)
y = scipy.special.j0(x)
fig = plt.figure()
ax  = fig.add_subplot(111)
ax.plot(x,y,'r-',lw=2,label='J_0(x)')

ax.set_title('Zeroth-Order Bessel \
  Function', fontsize=24, family='serif')
ax.set_ylabel('f(x)', fontsize=18)
ax.set_xlabel('x', fontsize=18)
ax.set_ylim((-1, 2))
ax.legend()
plt.show()
```

```
x = linspace(0,6,201);
y = besselj(0,x);
figure1 = figure;
axes1 = axes('Parent',figure1);
plot(x,y,'r-','DisplayName','J_0(x)',
  'LineWidth',2);
title({'Zeroth-Order Bessel Function'});

xlabel('x');
ylabel('f(x)');
ylim([-1 2])
legend();
```

from numpy import *
from matplotlib.pyplot import *

	plot			plot_surface			contour	legend			
	plot	fplot	ezplot	surf	ezsurf	plot3	contour	legend	imread	imshow	imwrite

LINEAR ALGEBRA

from numpy import *
from scipy.linalg import *

	dot	cross	*A*.T	inv	det	trace	inner	outer	matmul	eig	solve	qr	svd	lu	expm	logm	cholesky
	dot	cross	*A*'	inv	det	trace	*		*	eig	\	qr	svd	lu	expm	logm	chol

POLYNOMIALS & CURVE FITTING

```
from numpy import *
```

poly(*v*)	roots(*p*)	polyval(*p*,*x*)	polyder(*p*,*m*)	polyint(*p*,*m*)	polyfit(*x*,*y*,*n*)
poly(*v*)	roots(*p*)	polyval(*p*,*x*)	polyder(*p*,*m*)	polyint(*p*,*m*)	polyfit(*x*,*y*,*n*)

Polynomials are ordered in *ascending* manner ($x^0+x^1+x^2$) in Python and in *descending* order in MATLAB ($x^2+x^1+x^0$).

```
from scipy.interpolate import *
```

interp1d(*xd*,*yd*,*mt*)	griddata(*xd*,*yd*,(*gx*,*gy*),*mt*)	splrep(*xd*,*yd*)	bisplrep	splprep
interp1(*xd*,*yd*,*x*,*mt*)	interp2(*xd*,*yd*,*x*,*gx*,*gy*)	spline(*xd*,*yd*,*x*)		

STRING OPERATIONS

print	'%f'%np.pi	.find	in	.join	.split	+		.upper
disp	sprintf('%f',pi)	strfind	strcmp	strjoin	strsplit	strcat	strtok	

ADVANCED SYNTAX

CONTROL BLOCKS

```
if expr1:
    code1
elif expr2:
    code2
else:
    code3
```

```
if expr1
    code1
elseif expr2
    code2
else
    code3
end
```

EXCEPTION HANDLING

```
A = 1
try:
    file = open('file.txt')
except IOError, exc:
    print 'file cannot be opened'
except:
    print 'non-IOError'
else:
    print file.readlines()
finally:
    file.close()
```

```
A = rand(3);
B = ones(5);
try
    C = [A; B];
catch err
    error('Dimension mismatch');
end
```

LOOPS

```
for v in arange(1.0,0.0,-0.2):
    print v
n = 0
while n < 10:
    print n
    n -= 1
```

```
for v = 1.0:-0.2:0.0
    disp(v)
end
n = 0;
while n < 10
    disp(n);
    n = n - 1;
end
```

FUNCTION DEFINITIONS

```
def foo(x):
    y = x ** 2
    return y
```

may be defined in any block (including in nested blocks)

```
function [y] = foo(x)
    y = x .^ 2
end
```

must be in file named `foo.m`

ANONYMOUS FUNCTIONS

```
lambda x: x ** 2
```

```
@(x) x .^ 2;
```

USING CODE & SCRIPTING

FILE INPUT & OUTPUT

import	eval	execfile	open	.read	.readline	.write	.close	np.loadtxt	np.savetxt
import	eval	run	open	fileread	fgetl	fprintf	fclose	load	save

DISTINCTIONS

```
import this
from __future__ import division
@decorator
with x as y:
    pass
```

IPython makes an excellent default interpreter, as does the Jupyter notebook.

```
tic; expr; toc % stopwatch timer
clc            % clear display
format short   % or long (change # disp)
echo on        % or off (echo script cmds)
!cd dir        % run shell commands
```

MATLAB has an extensive collection of Toolboxes.