*compiled for MECH3750 by Tom Reddell*

# Python3 quick reference guide

## Important Libraries

**math:** standard math functions and constants
**numpy:**arrays, linear algebra, data processing
**scipy:** scientific algorithms and routines
**matplotlib:** plotting and visualization tools

## Importing Modules

```
# import <package> as <reference>
import numpy as np, matplotlib.pyplot as plt
# import <package> as <package>
import math
# import <object> from <package>
from scipy.optimize import fsolve
```

## Inbuilt Help

```
dir(math) # see what's in the math module
help(math.atan2) # get help on atan2 function
```

## Printing and Formatting

```
# standard print using string.format
# f: float ; e:scientific ; g:automatic
# .2 indicates use two significant figures
x = math.pi
print("x = {0:.2f}".format(x))
# f-string print (python3.6+)
print(f"x = {x:.2f}")
```

## Basic Mathematics

```
x = 2.5 ; y = 3.5
z = x + y ; z = x - y # addition/subtraction
z = x * y ; z = x / y # multiplication/division
z = x**y              # exponentiation
z = x//y ; z = y % x  # integer division/modulo
x+= y ; x-=y ; x*=y   # in place operations
x/=y ; x**=y ; x//=y ; x%=y
z = 3 +5.j            # complex numbers
u = 1_000_000         # underscoring (python3.6+)
x == y ; x!=y         # equality boolean checks
x < y ; y>= x;        # inequality boolean checks
```

## Loops and Conditionals

```
# loops from i = 1 to i = 10
for i in range(1,10):
    print(i)
# loops over each value in the list
for x in [1.5, 2.5, 3.5, 4.5]:
    print(x)
# will loop while boolean <condition> is true
while condition:
    do_something()
# will loop over every index, value in the list
for i, x in enumerate([1.5, 2.5, 3.5, 4.5]):
    print(i,x)
# will loop over each tuple pair (-1,3), (-2,4)
for x,y in zip([-1,-2],[3,4]):
    print(x,y)
# inline list of first 10 even numbers
evens = [2*n for n in range(10)]
# inline summations using generators
basel = sum(1/(i**2) for i in range(1,100))
```

## Basic Python Structures

```
# lists, ordered arrays, can be mixed data
a = [1, 2, 3.5, '4', [5, 6], 7+8.j]
# dictionaries, key/value mappings
b = {'course':'MECH3750', 'grade':7}
# sets, unordered collections of unique data
c = {1,2,4,4,5} # duplicate 4 will be removed
# functions, input --> output maps
def add(x,y):
    return x + y
z = add(x,y) # calling function 'add'
```

## numpy and Linear Algebra

```
# define a vector of double precision floats
b = np.array([1, 2, 3], float)
# define a two dimensional array
A = np.array([[3, -4, 2],
              [-1, 4, 3],
              [0, -7, 6]], float)
# generate an array of linearly spaced values
a = np.linspace(-np.pi, np.pi, 100)
# generate an array of log-space values
m = np.logspace(-8,-1,8)
# dot product of two vectors
u = b @ b
# matrix/matrix multiplication
C = A @ A
# matrix/vector multiplication
v = A @ b
# element wise basic math on arrays
w = v * u ; w = v + u ; w = v/u #etc
# scalar multiplication by array
r = 2.5*A
# matrix transpose
D = A.T
# array operations in place
A += C ; A -=C # etc, same as basic math
# get element of array
s = D[0,1]
# copy array safely
A2 = np.copy(A)
# get row/column of array as new array
e = np.copy(A[:,2])
# solve linear system A*x = b
x = np.linalg.solve(A,b)
# update element of array
A2[2,2] = np.pi
#update slice of array
A2[1,1:2] = [-1, 0]
```

## Solving Non-Linear Equations

```
from scipy.optimize import fsolve
# solve x^2 - y^2 = 0 ; x*y = 2
def zero_function(X):
    x,y = X
    return [x**2 - y**2,
            x*y - 2]
# input function to solve/initial guess
root = fsolve(zero_function,[1,1])
```

## Plotting

```
import matplotlib.pyplot as plt
xdata = [1,2,3] ; ydata = [2,-4,6]
plt.figure() # figure 1
plt.plot(xdata, ydata, '-r')
# can include multiple plot statements
plt.show()
```

## Loading Data from File

```
# load a text file of numeric data
data = np.loadtxt('data_file.txt')
```