ENGG1003 - Thursday Week 11

MATLAB vs. Python

Sarah Johnson and Steve Weller

University of Newcastle

20 May 2021

Last compiled: May 20, 2021 12:22pm +10:00

Lecture overview

1. blah
2. blah
3. blah

MATLAB vs. Python

- xxx

https://realpython.com/matlab-vs-python/

MATLAB

- MATLAB is an abbreviation for Matrix Laboratory. It is a programming language perfectly suited for matrix manipulation and linear algebra.
- MATLAB offers many additional Toolboxes such as control design, image processing, digital signal processing, fluid dynamics etc. They are all very well documented.
- MATLAB includes Simulink which allows visualisation of systems begin simulated which can be very useful in many engineering applications.

MATLAB

- For this reason you may use MATLAB in some courses later on. A non exhaustive list of courses currently using MATLAB includes: CHEE4945, CHEE4975, ENGG2440, AERO3600, MCHA3400, MCHA3500, MCHA4100 ELEC2132, ELEC2430, ELEC3400, ELEC3410, ELEC3540, ELEC4100, many FYPs

Why are we teaching Python then?

- Unlike Python which is free, MATLAB licences are very expensive. You will be able to keep programming in Python once you graduate.
- Matplotlib and Numpy have functionality very similar to MATLAB (making it easier to move across).
- Python libraries offer much of the functionality as the MATLAB toolboxes and are growing much faster.
- Python has become a very popular, in demand language. Many students are finding that employers are requesting Python coding skills for both positions and internships

Syntax

A few general differences between MATLAB and Python:

- In MATLAB comments start with %. In Python, comments start with #.
- White-space and indenting are very important in Python. MATLAB does not require the same (but it is highly recommended anyway for readability)
- MATLAB function `disp( )` replaces `print( )`
- help for a function via `help fname` rather than `help(fname)`

Maths

- Addition, subtraction, multiplication and division are the same. A difference is that
  - MATLAB uses ^ not ** for exponential.
  - I.e. `x**2` becomes `x^ 2`
- Relational operators ==, >, <, >=, <= are the same, except
  - MATLAB uses ~= instead of != for 'not equal to'
- In MATLAB the value of a variable is automatically printed to the terminal. You use ';' to suppress
  - I.e. `a = 3` prints value of a, whereas `a = 3;` does not

Flow control

- If Else statements and For / While loops work exactly the same way in both languages with some small differences in syntax
  - MATLAB does not need the ':' used in Python at the end of the loop definition / if condition
  - MATLAB designates the end of an If statement or loop by 'end' instead of by indenting
  - Python shortens `elseif` to `elif`. MATLAB does not.

Flow control

E.g. Nested If Else

```python
num = 10                                    # Python
if num == 10:
    print("num is equal to 10")
elif num == 20:
    print("num is equal to 20")
else:
    print("num is neither 10 nor 20")
```

```matlab
num = 10                                    % MATLAB
if num == 10
    disp("num is equal to 10")
elseif num == 20
    disp("num is equal to 20")
else
    disp("num is neither 10 nor 20")
end
```

Flow control

E.g. Use a For loop to add the integers from 1 to 10

```python
sum = 0                          # Python
for i in range(1,11):
    sum = sum+i
```

```matlab
sum = 0                          % MATLAB
for i = 1:10
    sum = sum+i
end
```

Flow control

E.g. Use a While loop to find the first (smallest) integer which when squared is greater than 100

```python
n = 1                                   # Python
n_squared = 1
while n_squared < 100:
    n = n+1
    n_squared = n**2
```

```matlab
n = 1                                   % MATLAB
n_squared = 1
while n_squared < 100
    n = n+1
    n_squared = n^2
end
```

Arrays

- MATLAB arrays work a lot like `numpy` arrays. If you can work with Numpy arrays you will find MATLAB arrays are easy. Both do vectorisation in the same way
- There are a couple of syntax differences to note:
  - In MATLAB, when you want to index an array, you use round brackets `'()'`. Square brackets `'[]'` are used to create arrays

```python
arr = np.array([10, 20, 30])    # Python
s = arr[i]
```

```
arr = [10, 20, 30]              % MATLAB
s = arr(i)
```

Arrays

- 2D arrays use ';' to designate the next row

```python
a = np.array([[2,3],[4,5]])     # Python
s = arr[i,j]
```

```
a = [2 3; 4 5]                  % MATLAB
s = a(i,j)
```

- All zero / all one matrices are very similar

```python
m = np.zeros([5,10,3])  # Python 5x10x3 array
```

```
m = zeros(5,10,3)               % MATLAB 5x10x3 array
```

- In MATLAB a new copy of an array is created by default

```python
b = a.copy()  # Python
```

```
b = a                   % MATLAB
```

Arrays

- Many Numpy functions exist in MATLAB including `linspace()` `sqrt()` `log()` `exp()` `round()` `ceil()` `floor()` `max()` `min()` `mean()` and many more
- A few tips to keep in mind:
  - In Python, the index of the first element in an array is '0', in MATLAB it is '1'
  - In Python, the index of the last element in an array is '-1', in MATLAB it is '`end`'
  - The length of an array is `length` rather than `len`
  - Array slicing works similarly in MATLAB to Python
  - You don't need to call `linspace()` to create an array `start:step:end` will do

```
1 a = 5:1:100                    % MATLAB
```

Plotting

- Plotting in MATLAB is very similar to `Matplotlib.pyplot`
- Including the functions `plot()` `imshow()` `subplot()` `scatter()` `title()` `axis()` `xlabel()` `tick()` `figure()` and many more
- Two tips tips be aware of:
  - In Python plotting a new curve will add it to the figure. In MATLAB it will replace the first curve unless you use a `hold on` command first
  - MATLAB does not need a `show` command, the figure is shown automatically and it does not need to be closed for the remainder of the program to be run.

```python
import numpy as np                                  # Python
import matplotlib.pyplot as plt
v0 = 5
g = 9.81
t = np.linspace(0,1,1001)
y = v0*t - 0.5*g*t**2
plt.plot(t, y)
plt.xlabel('t (s)')
plt.title('Velocity over time')
plt.show()
```

```
v0 = 5                                              % MATLAB
g = 9.81
t = linspace(0,1,1001)
y = v0*t - 0.5*g*t.^2
plot(t, y)
xlabel('t (s)')
title('Velocity over time')
```

Functions

- Functions work exactly the same way as in Python.
  - You declare them by defining the function name, inputs, outputs and operation in the function declaration and then call the function by name as needed.
  - Functions start with the keyword `function` not `def`
  - The function output is defined at the start of the function not the end
  - The end of the function is indicated with the keyword `end`

```python
def addition(num_1, num_2):                    # Python
    total = num_1 + num_2
    return total
```

```
function [total] = addition(num_1, num_2)  % MATLAB
    total = num_1 + num_2;
end
```