DA1 - Getting Started

Live demo

In [ ]:
```
# checking that our working directory is set correctly
import os
print(os.getcwd())
```

In [ ]:
```
# Another way of doing this from within a notebook using ! to access commandline utilities
!pwd
```

In [ ]:
```
# Another way of creating our directory structure
!mkdir data
!mkdir scripts
!mkdir images
# a way to show directory structure is the 'tree utility
!tree
```

In [ ]:
```
# we need to make sure that the requisite packages are installed on our machine
# one way is to use the 'pip' command which will fetch them into our current environment.

!pip install pandas
!pip install matplotlib
!pip install plotnine
!pip install seaborn
```

In [ ]:
```
# Our first step is usually to build our compute environment by importing/loading the
# packages for our analysis
# e.g. pandas and give it a 'stem' name pd
import pandas as pd
# an alternate was is to use * (then the 'stem' is the package name
from plotnine import *
# If you want to use the seaborn alternative to plotnine examples
import seaborn as sns
import matplotlib.pyplot as plt
```

Data objects

Lets create and use some objects.

We can convert fahrenheit to celcius with the following formula.
Celsius = (Fahrenheit - 32) / 1.8

In [ ]:
```
# Start by creating a variable for Fahrenheit
fahrenheit = 46
print(fahrenheit)
```

In [ ]:
```
print((fahrenheit -32) /1.8)
```

In [ ]:
```
# Let's store the result of a conversion in another object
celcius = (fahrenheit -32) /1.8
print(celcius)
type(celcius)
```

Collections:

In [ ]:
```
# We need the 'numpy' package
import numpy as np
```

In [ ]:
```
# Createa a 1D collection
heights = np.array([1.8,1.95,1.7,1.6,2])

print(heights) # Note how last item is displayed
```

In [ ]:
```
# We can use type function again to check heights
type(heights)
```

Lists

In [ ]:
```
# similarly for character data (strings)
climate = (["cold","warm","hot"])

print(climate)
type(climate)
```

In [ ]:
```
# numbers can be in a list but note they are not so fast as numpy arrays
height_list = [1.8,1.95,1.7,1.6,2]

print(height_list)
type(height_list)
```

Tuples The immutable object often used to return multiple vales from a function.
They are more memory efficient and faster to access than lists.

In [ ]:
```
example_tup = (1.41, 2.718, 3.142)

print(example_tup)
type(example_tup)
```

In [ ]:
```
# When doing indexing, which value is returned by code below?
print(example_tup[2])
```

In [ ]:
```

```