Exercise 4, Python and Fortran

1. Write a function that converts temperature scales. The function should take as input a temperature value, the scale on which it was measured, and the scale to which it should be converted. The scales can be Fahrenheit, Celsius, or Kelvin. You will need other functions to convert Fahrenheit to Celsius, Celsius to Fahrenheit, Celsius to Kelvin, and Kelvin to Celsius. The conversion from K to C is

C=K-273.15

You do not need Fahrenheit to Kelvin (why not?). You can look up the C/F conversions.

2. Write a function or subroutine that takes an array of numbers and computes and returns the mean and standard deviation. Do not use any built-in functions other than sqrt.

Python: we have not discussed tuples. They are very simple and have many applications. Tuples are like lists except they are immutable. This means that they must be created with the size and contents they should have because you cannot append, delete, insert, etc. elements. Tuples are surrounded by parentheses () and, like lists and arrays, the elements are separated by commas. You also refer to individual elements by square brackets as for lists and arrays, and you can also specify slices with colons as for lists and arrays.

Example: (x,y)

```
(‘First’,1,80.)
```

You can use tuples to return multiple items from a function. Tuples are often better for this purpose than lists because of their immutability.

If you return a tuple then if you store the variables upon return, you’ll need a tuple. Example:

```
(mean, std)=stats(A)
```

You can also refer to a single element of the tuple

```
mean=stats(A)[0]
```

3. Write a function or subroutine that takes an array of numbers and returns it sorted in the same variable. Python: it will need to be a numpy array.

You may use any sorting algorithm you wish but most of you will probably want to use bubble sort. It is straightforward to implement but it is slow. Horribly, horribly slow. Don’t use it for real code. It is fine for this exercise, however. You can find a pseudocode implementation on Wikipedia. You don’t need a repeat-until structure but you do need some form of double loop.

Extra for the ambitious: write a function/subroutine that takes a rank-1 array and returns a sorted array plus the *permutation vector*. The permutation vector gives the original location of the element in the sorted array. It is obtained by applying the same operations to an index vector as to the values vector.

For Exercises 2 and 3, use the data file windspeed.txt that is in Resources/Data/FP_HW on Collab.