So far, we’ve processed data that was either directly written into the program itself *and/or* read a structured text file into arrays using numpy.loadtxt(). What should we do if we want to read/interpret/process more complex information that is stored in a text file on the hard drive of our computer?

**Approach**:

1. Read a text file.
2. Decode - Parse and interpret the lines of the text file.
3. Act on the data from step 2 by doing some calculations.
4. Encode - Output results to screen and/or a text file.

So far, we have mostly focused on step 3. We will first take steps 1, 3, and 4 for granted and focus some attention on step 2.

Let’s think about the human things we do to process a document (text file). First some definitions of the mechanics: taking you back to grade school English class (typically painful for engineers ☺ ):

***parse*** – *verb*, analyze (a sentence) into its parts and describe their syntactic roles.

computing – analyze (a string or text) into logical syntactic components, typically in order to test conformability to a logical grammar.

***string*** – *noun*, computing – a linear sequence of characters, words, or other data.

***character*** – *noun*, computing – a symbol representing a letter or number.

***syntax*** – *noun*, the arrangement of words and phrases to create well-formed sentences in a language.

***phrase*****–** *noun*, a small group standing together as a conceptual unit, typically forming a component of a clause.

***clause*** – *noun*, a unit of grammatical organization next below the sentence in rank and in traditional grammar said to consist of a subject and a predicate.

***predicate*** – *noun*, the part of a sentence or clause containing a verb and stating something about the subject.

***subject*** – *noun*, a person or thing that is being discussed, described, or dealt with.

***delimiter*** – *noun*, a sequence of one or more characters for specifying the boundary between separate, independent regions in plain text or other data streams.

**Examples** of mathematical sentences:

 : ***subject*** – “*y”*, ***predicate*** – “= *mx + b”*

*flow rate* 30 gpm : ***subject*** – “*flow rate”*, ***predicate*** – (implied) “(is or equals) 30 gpm”

As humans trained to read English, ***parsing*** these mathematical sentences is very easy. We see the important left hand side (***clause***) of an equation, identify this ***subject*** as the *dependent variable*, recognize the *relational operator* (explicit or implied) as a ***delimiter*** identifying theright hand side (***clause***), which is interpretedas the *independent variable* or *function* *of the independent variable*. We intuitively see that the ***clauses*** may be further ***parsed*.**

**BUT** computers only see a list of characters and are **INCAPABLE** of understanding what information is being conveyed in the sentences. It is up to us, the programmer, to assign meaning to strings contained in a text file; the process of ***parsing***.

**For example**, in the sentence “*flow rate* 30 gpm”, we can easily see that what is intended is that *“flow rate*” is a ***phrase*** (let’s call it a key word) and 30 is a number and “gpm” is the units (we understand the shorthand to mean gallons per minute). But, what does a computer see? If it is using the space between words as a ***delimiter***, it would logically see that “*flow”* and “*rate”* are two different words rather than our intention that they should be grouped together into the key word “*flow rate*”. Perhaps we should tell the computer to look at the first 9 characters for a key word? But, if we throw in “*volumetric flow rate*”, our human brain has no trouble interpreting what this means while the program would see the unintelligible “*volumetri*” as the keyword. Also, the computer sees *“Flow”* and “*flow*” as completely different words because of capitalization. Finally, is 30 seen as an integer, a floating point number, or a string? We know that 30 is probably a floating point number to be interpreted as 30.0, but the computer has no way to intuit this unless we are very specific about telling it to do so.

**Revised example:** If the author of the data file was kind enough the write the sentence as: “*flow rate*, 30 gpm”, we could easily tell the computer that the comma “,” is the delimiter and we should split the string into left and right sides. On the left side, we would expect the ***subject***/key word. On the right side we would expect the value and (perhaps) units.

**Python string methods** (<https://docs.python.org/3/library/stdtypes.html#string-methods>)

A string in Python has many methods associated with it for ***parsing***. The usual format is:

MyString = “This is an example of a string.”

MyString = MyString.lower()

Print(MyString)

≫”this is an example of a string.”

That is to say, the method of the string returns a new string. In this example, the .lower() method makes all letters lower case and returns a new string.

We will go through a detailed example of ***parsing*** a data file in the video. This will also handle steps 1, 3 and 4.