**Word(s) of the day**:

**per·snick·et·y** - requiring a particularly precise or careful approach.

**par·si·mo·ni·ous**  - stingy or frugal. Unwilling to spend resources.

**Review**

*jargon*

*import* – *used to access functions stored in other files* (**why?**)

(**Occum’s razor** - *Entia non sunt multiplicanda praeter necessitate*: entities should not be multiplied beyond necessity; the principle of parsimony)

*module* – *python file containing functions that can be called*

*function* – *a bit of python code that does a calculation and returns a value (usually)* (**why?**)

* *functions should be as short as possible, well-defined with well-defined arguments and return a value (even none)*

*alias* – *a short name for an imported module or function* (**why?**)

*scope* – *the context in which a variable is valid (global vs. local)* (**why?**)

*docstring* – *use doc strings to describe your functions, arguments and returns for you and other users* (**why?**)

*folding* – *the pycharm utility for collapsing/expanding functions for readability and debugging*

*debug* – *use IDE to test/repair programs* (**why?**)

*IDE* – *integrated development environment (pycharm)* (**why?**)

*regions* – *tags used to mark logical sections of your code (e.g., imports and functions)* (**why?**)

*lists* – *defined by [] and contains a number of elements (can be mixed type, but not recommended).*

*tuple* – *defined by () and contains an immutable set of elements*

*CLI* – *command line interface*

*GUI* – *graphical user interface*

*formatted output*

* print(“Format number as floating point with 3 digits {num1:0.3f}”.format(5.0))
  + *Format number as floating point with 3 digits 5.000*
* print(“{num1:0.4f} and {num2:0.3f}”.format(num2=3, num1=5) )
  + *5.0000 and 3.000*
* print(“{:0.4f} and {:0.3f}”.format(3, 5) )
  + *3.0000 and 5.000*

*positional arguments* (**why?**)

* **MUST** be passed to function from calling function.
* **MUST** be passed in the proper order (if just passing values).
* **MAY** initialize by name (and out of order, but not advised).
* What is the **scope** of the argument(s)?

*keyword arguments* (**why?**)

* **MUST** have an initial (default) value in function define.
* **OPTIONAL** to pass keyword arguments.
* **IF** passing keyword argument, **USUALLY** specify by name, works by position too.

*Importing from modules*

* *module*: A file containing a set of functions you want to use in your application.
  + *import* ***mmm*** *as* ***xxx***: access functions by: xxx.fff
  + *import* ***mmm***: access functions by: mmm.fff
  + *from* ***mmm*** *import* ***fff***: access function directly by: fff
  + *from* ***mmm*** *import* ***\****: access all functions directly. (frowned upon)

*Functional Programming* (**why?**)

<https://en.wikipedia.org/wiki/Programming_paradigm>

**New**

*main guard*:

* if \_\_name\_\_ == ‘main’:

*function overloading:*

* same function name, different behavior because different arguments

*some PyCharm tricks* (Lists and Loops.py)

* *Folding: Ctrl+-, Unfolding: Ctrl++, Folding All: Shift+Ctrl+-, Unfolding all: Shift+Ctrl++*
* *Go to function: Ctrl+click*
* Commenting blocks: *Ctrl+/*
* *Uncommenting blocks: Ctrl+Shift+/*

*Pop and insert* (Determinant.py, Gauss\_Elim.py)

* *pop*: pulls a row (element) out of a list and all elements “below” that element decrease their index by 1.
* *Insert*: pushes a row (element) into a list at a given index and causes all elements “below” that index to increase their index by 1.

*copy*:

* deep copy
* shallow copy

*lambda functions (*[*https://en.wikipedia.org/wiki/Lambda\_calculus*](https://en.wikipedia.org/wiki/Lambda_calculus)*)* (Functions passed to Functions and Lambda.py)

* A small, anonymous function.

*callbacks - (*[*https://en.wikipedia.org/wiki/Callback\_(computer\_programming)*](https://en.wikipedia.org/wiki/Callback_(computer_programming)) *)*

* A function can be passed as an argument (a.k.a., a *callback function*) to another function (a.k.a., the *called function*)
* The *callback function* is executed from within the *called function*.

**Numerical method for the Day:**

Gaussian elimination to solve **Ax**=**b → A-1Ax=A-1b → x=A-1b**

To reduce a matrix **A** to echelon form, we perform the basic row operations of:

1. exchange rows
2. multiply row by a scalar
3. add scalar multiple of another row to replace a row

Goal is to get 1’s along the matrix diagonal

Observation: **A-1A = I**

If I use Gauss-Jordan method on matrix [**A**|**I**], I will find the result to be [**I**|**A**-1]

Example:  Find x1, x2, x3