navigation
- Main page
- Community portal
- Current events
- Recent changes
- Random page
- Help

search

Go Search

toolbox
- What links here
- Related changes
- Upload file
- Special pages
- Printable version
- Permanent link

My Python Coding Conventions

Contents [hide]

Coding Conventions Etc.

[edit]

In reading my code it may be of some use to know what conventions I have (tried) to follow. The code has been developed over quite a period of time so the standards are not uniform. What I write here are the standards that are in quite a bit of the code and the directions that I am trying to move. In all of the coding consistency is an important standard, I have a ways to go. I am now only coding in Python 3.6 or up. Here are some types of conventions. Ultimately if you want to understand the code, read it. I work hard to make it readable, so please try, you can let me know of shortcoming, but note that I am aware of the fact that it still need improvement.

Names

[edit]

I try to be consistent: this is good but have not been very successful in standards: I keep changing my mind. I am avoiding short names and try to make names descriptive enough that they are somewhat self documenting. References are often copied across objects for easy access (lots of parameters for example); when this happens the name of the object is generally (should be always) the same in both objects.

Formatting

[edit]

Nothing special here but I like white space and use a lot. This is not standard Python. But this is what I like.

Docstrings

[edit]

I am working towards using them but have not arrived at a format that I both like to read and which is quick enough to write. Not good as of 2017 Jan

Imports

[edit]

- In most cases use the format "import xyz" so the name space is not polluted and so it is easy to identify just what an imported class is.
- In in objects that are almost all GUI then using "from Tkinter import *" is ok but better is: "import Tkinter as Tk"
- I normally have only one or a few classes in a file so there are a lot of files and a lot of what I call "local imports".
- Almost all imports are at the top of a file, std library imports first then "local imports".

Object Orientation

[edit]

Almost everything is a class. Not much in the way of module functions, not many classes in a module. I think my Java experience has led me to overuse classes and under use functions.... at the module level.

Documentation for Class Instance Methods

[edit]

Look something like this:

```
    def create_class_from_strings( self, module_name, class_name):
        """
        This will load a class from string names
        It makes it easier to specify classes in the parameter file.
        I believe it is used for both the comm drive and the "processor"
        args:  strings
        ret:   instance of the class
        Side effects Class created
        """
```

The comment should give the intent of the method, some hint as to the args (which hopefully have good names), and some info. on the return value. zip means nothing, void....

I am moving toward using __ and _ as prefixes for more private methods, but have not gone too far in this direction.

Application Structure

[edit]

working on it..... so far:

MVC Structure

[edit]

I try to separate the Model, View, and Controller into separate modules and classes. The View is the class GUI. The controller is normally the main routine where the app is started.

GUI Structure

[edit]

coming...

- Construct frame by frame in a subroutine returning the frame reference, the caller then places the frame. Typically constructed and placed from top to bottom from left to right.
- Elements:
    - Buttons for application management: restart the app, edit the parameters, edit the log, help.
    - Buttons.... for the application.
    - A text control for application messages, sort of a mini console for the application.

Application Parameters

[edit]

see: **Configuration Files For Python**

Typical Application Functionality

[edit]

coming...

Global Variables and Application State

[edit]

Globals are often.....

MultiThreading

[edit]

In any GUI code using Tkinter if the GUI kicks off a long running piece of code the GUI becomes unresponsive while the code runs. To avoid this you can use a second thread. This is frequently what I do. I call the two threads the gui thread, and the helper thread. To coordinate the two I have two queues, one from gui thread --> helper thread and from helper thread --> gui thread. I try to isolate the code for each thread into its own modules and classes, and comment the code as to the required thread in cases where I think there might be doubt.

Python Logging

[edit]

I usually implement the standard python logging utility, messages into a file configured by parameters.py, the gui typically has a button to view this file.

Links

[edit]

- Configuration Files For Python
- Configuration Files For Python

Categories: Python SmartPlug | Python