AOData Walkthrough
The purpose of this tutorial is to demonstrate, step-by-step, how to get started with AOData. The walkthrough will take you through the process of setting up a custom AOData package (the `+tutorial` package within this folder).

This document is meant to be a companion to the Documentation PDF as well as the documentation set up within the code (i.e. what you get from `doc aod.core.Experiment` or `help aod.core.Experiment/setHomeDirectory`. To get a full sense of AOData, check out the accompanying documentation at each step. Opening each of the variables (particularly the entities in MATLAB's variable viewer and exploring the contents will also help you make the most of the walkthrough. If you aren't comfortable with MATLAB or object oriented programming, opening up the underlying code and checking out how different steps are implemented can be helpful too.

Before beginning, the walkthrough you should follow the instructions in the documentation for **Installation** and for adding AOData to your search path.

AOData is a framework for managing experimental data, metadata and code. In other words, AOData is meant to provide a strong, standardized foundation geared towards maximizing reproducibility, accessibility and collaboration. It's a platform for end-users to customize for their individual experiments and workflows. At this point, make sure you've read the documentation sections on **"AOData Object Model"** and how it maps to an HDF5 file. Opening up an example file in **AODataViewer** and clicking around will be helpful too.

You will need to ask a few questions about your data:

1) Conceptually, how does your experiment map onto the AOData object model?

- What are the different types of Calibrations you perform (i.e. power measurements, AO calibrations, PMT optimizations)?
- Is there any information you need to log about the status of your System (i.e. ad hoc additions like NDFs and filters not recorded in the system diagram, serial numbers of devices like PMTs, etc)?
- How many different types of imaging do you do and are they performed in different videos (a.k.a. Epochs)?
- Are there important relationships between entities that cut across the AOData object model? For example,

2) Logistically, where and how is the information stored?

- Is it hand-written in an imaging log or saved as a file?
- What are the relevant files, where are they located in an experiment folder, what file formats, and how are they tied to the object they describe (e.g. saving the video number in the file name ties it to a specific Epoch)?
- Does any of this information rely on custom code (either to be generated or interpreted)? For example, did you write any code to generate a Stimulus or process a Calibration?

3) Programmatically, what code do you need to write to get that information into AOData's object model?

- What custom classes do you need to create?

Figuring out #1 and #2 is ultimately up to you. This walkthrough is focused on helping you with #3 and providing the familiarity with AOData's functionality likely will help with #1 as well. As you work through the walkthrough, keep an eye towards how the concepts introduced could apply to your own experiments.

Initialization
The first step is initializing AOData. If you haven't already, simply run the line below:

```
initializeAOData()
```

A GUI called **AODataManagerApp** should have opened up. There are 3 tabs which reflect the 3 settings AOData adds to your MATLAB user preferences:

1. BasePackage: This is the location on your computer of the main AOData folder. If you move the folder for some reason, you'll want to re-run `initializeAOData()`
2. SearchPaths: This a list of the folders containing packages (i.e. the folder containing the first +folder. All subfolders that are packages will be added too). Some parts of AOData require knowledge of all the available classes and custom subclasses. For now, AOData's src folder should be added as this contains the +aod package. The subfolders that are packages do not need to be added here (e.g. "+core", "+persistent").
3. GitRepos: AOData tracks the commit IDs of any folders added here that are git repositories. When you create your own package, if you choose to track it with git (strongly recommended!), then you will add your git repository folder here.

For now, you can close out of it as `initializeAOData()` will have set the necessary initial values.

Creating a new package
Create a git respository

Create a new git repository (see the Git Tutorial if you aren't familiar with this step). Name it something that identifies it as your personal AOData package (e.g. "yourname-aodata-package", "yourname-package", etc).

Create a namespace

Read the documentation section on **"Namespaces"** and refer to the linked MATLAB documentation for any additional questions that arise. As with all other concepts introduced here, looking at how AOData is organized can be helpful too.

An example of a class is `aod.core.Device`. The class is named `Device` and `aod` and `core` are the namespaces (`Device` is located in a folder called "+core" which is located in a folder called "+aod"). You will want to make your own namespace in your new repository folder. When in doubt, try the following template: "labname.yourname" (e.g. "williamslab.sara"). You want the namespace to be unique to you.

It can be useful to have packages contained within another folder so I'd recommend creating a folder named "src" within your main folder. If your chosen namespace is "labname.yourname", then inside "src", create a folder named "+labname". Inside that, create "+yourname".

Example
To familiarize you with process of mapping an experiment to AOData, we'll walk through an example experiment from the Rochester one-photon AOSLO. Data is acquired from two channels: a reflectance channel focused on the cone mosaic and a fluorescence channel focused on the ganglion cell layer. In addition, there's a wavefront sensing channel and a channel for providing visual stimuli to the cones.

A small experiment folder is available at: link. Download this and save it somewhere to your computer. The raw, acquired data

Source: Each experiment involves just 1 primate. Within that primate, there are two eyes, both of which may be imaged during an experiment and which have eye-specific metadata like axial length. Finally, within each eye, there are multiple locations imaged. These may not have metadata but still need to be recorded so the data can be sorted by location later.

This situation demonstrates why Source offers a nested heirarchy of Sources. The animal should be a Source within experiment. The eye or eyes imaged should be a Source within the animal's Source. Finally the locations imaged should be a Source within the eye. The `aod.builtin.locations` contains classes for this exact situation: `Subject`, `Eye`, and `Location`.