

# Pyhab User Manual

v0.3

Spring 2018

Dr. Jonathan F. Kominsky

1. What is PyHab?	2
2. Getting PyHab	3
3. How to build your study in PyHab	6
a. Relevant concepts	6
b. Creating your first experiment	8
c. Saving and sharing your experiment	30
4. Running a study in PyHab	31
a. Starting a PyHab study	32
b. Running a PyHab study	35
c. Data and computing reliability	39
d. Preferential looking studies	42
5. Troubleshooting	43

# 1. What is PyHab?

Infant and toddler looking-time studies primarily rely on manual looking-time coding software like jHab and xHab, which are old, opaque, and not open-source. PyHab is a script written for PsychoPy, a free, open-source python-based stimulus presentation software (with which the author is not affiliated), that fully replicates the capabilities of xHab and jHab while adding several features.

The biggest advantages of PyHab over its predecessors are that it can be used to directly control stimulus presentation, and it's free and open-source. Previously, you typically needed one person coding looking times and a second person controlling the stimuli, and communicating when to end one trial and advance to the next, and when to advance from habituation trials to test trials. This introduces logistical hassle and imprecise timing. PyHab can present movie files or animated stimuli and advance trials and trial types appropriately based on live looking time coding. One person can run an entire experiment. A few other programs (notably Habit2) also allow for this, but can be inflexible and are not open-source, so you can't easily modify the program or see exactly what it's doing.

You might worry that this means you can't run these studies with a blind coder. Not at all! Provided that the coder can't determine what the participant is seeing or hearing (which will just depend on your setup), PyHab can be configured to tell a coder only when a trial starts and ends, and use a masked condition file so that the coder only needs to input a code and not know what condition is being presented. In habituation designs when the test trial can come after a variable number of habituation trials, PyHab doesn't even need to tell the coder when the test trial happens, in some ways making it *more* blind than other techniques!

## 2. Getting PyHab

- You need PsychoPy, first of all (available free from [psychopy.org](http://psychopy.org)). PyHab is a script that runs in a PsychoPy environment. You will want at least version 1.90.1 (not the Py3 version, I have not tested it in Py3).
- You DON'T need to know python! Python experience will help you with troubleshooting and allow you to do more ambitious things with PyHab, but there is a rudimentary builder interface that should allow you to build and conduct studies with no programming at all.
- To use PyHab for stimulus presentation, you will also need a computer with a two-screen setup. One screen will be your stimulus presentation screen, which will display stimuli to your participant. The other will be the coder's screen, which will display a small graphical interface telling you when a trial begins and ends. The displays cannot be mirrored, you need to have an extended desktop (like you would for a slide show with presenter view). Note: As of 1.90.1 you may run into issues with the window opening halfway between the two desktops if your primary display is a Mac with Retina display. A fix is in progress.
- If you want to present movie files, you will want your stimuli in .mov, .mp4, or .m4v format for the most reliable performance. PsychoPy's MovieStim works with other formats, but I have not tested all of them with PyHab and I'm not sure how well they work. Some formats work better on mac than windows.
- In addition, **you will want to design movies that have about two to four frames of buffer at the end.** This is because of the way that PsychoPy plays movies. If a movie file gets to the absolute end (i.e. past the last frame), the movie is unloaded from memory. In order to loop movies without having to reload the movie every time, I had to basically force PyHab to stop about 50ms before the end of the movie (roughly 1.5 frames at 30fps, or 3 frames at 60) to guarantee that the movie doesn't unload itself.

- You will need to install VLC media player on any computer that you plan to use to present stimuli. VLC ensures that the right codecs are available on your system for playing movie files. Fortunately, VLC is completely free. Just google it, download and install.
- **On Windows**, you will need ffmpeg installed to be able to use the stimulus presentation features. If you attempt to run a study with stimuli and get an imageio crash, follow these directions: In PsychoPy, open a coder window, then at the bottom select the tab labeled “shell”. At the prompt in this tab, type “import imageio” and hit return, then type “imageio.plugins.ffmpeg.download()” and hit return. This should download and install ffmpeg for PsychoPy. If it fails, close PsychoPy and then run PsychoPy as an administrator (right click > run as administrator).
- **Also on Windows**, you will need your movie files to be relatively small. For unknown reasons, MoviePy will use a prohibitive amount of memory to load larger movie files. Your movie files may need to be only a few hundred KB and have a low bitrate. You may find that you need to export them using VLC in windows to get access to these settings. If your experiment hangs on “loading movies”, this is why. Macs, for some reason, do not have this problem.

# Installation instructions

Assuming you have PsychoPy already installed, all you need to do is go to <https://github.com/jfkominsky/PyHab/releases> and click “Source code (zip)”. This will download a .zip file called PyHab-master, which you can then unzip into a folder. This folder has everything you need to get started. Put it wherever you want, but don’t modify anything inside the folder or move anything into or out of it.

Once you’ve got your PyHab folder where you want it, open PsychoPy in coder view (hit command-L or ctrl-L).<sup>1</sup> Then go to file > open and find the script called “NewPyHabProject.py” in the PyHab-master folder. This will open a very, very short script in the coder window. All you need to do is hit “Run” (the green running man icon) to launch the PyHab builder (see Chapter 3).

If you want to see a PyHab study in action, from the coder, go to file > open, find the PyHab-master folder, then open the “PyHabDemo” folder within it, and finally select “PyHabDemoLauncher.py”. This will open a somewhat longer script. Again, just hit the “Run” button, and this will open a dialog that will allow you to either run the demo experiment (see Chapter 4) or open its builder view to see how it is put together.

---

<sup>1</sup> On windows, you should run PsychoPy as an administrator. When opening the program, just right-click on the icon and select “run as administrator.” This is necessary to allow PyHab to make new experiments, access movie codecs, that kind of thing.

## 3. How to build your study in PyHab

### a. Relevant concepts

Let's define a few important terms:

- **Launcher:** The script you run initially, which gives you access to both running studies and editing them in the builder.
- **PyHab Builder:** A rudimentary GUI that allows you to construct your experiment. This is not to be confused with the (much, much superior) PsychoPy builder. Rather, this is something I built for PyHab specifically that is a little less refined but perfectly functional. Future updates may better integrate it into the PsychoPy builder.
- **Experimenter window:** The window presented to the experimenter running the study, which includes information about the current trial and (optionally) the live status of the coding keys.
- **Stimulus window:** The window on which the stimuli appear, if PyHab is being used for stimulus presentation.
- **Trial type:** PyHab builds studies in terms of trial types, each with its own label. Each label is enclosed in single quotes and is usually text (e.g., 'A'). Avoid symbols like \ or / or %, which can confuse Python in various ways (especially \). Also, I recommend that you not name trial types with just numbers (mixes of numbers and letters are fine). It should work, in theory, but Python sometimes gets weird about text that it can interpret as numbers.
  - There is exactly one special trial type that changes the behavior of the program: 'Hab'. The 'Hab' trial type will create a series of habituation trials that will go until a habituation criterion is met (see Habituation Settings). If you aren't running a habituation study, don't call any of your trial types 'Hab', but literally any other series of letters should be fine.
- **Stimulus file/stimuli:** PyHab expects all stimuli to be in the form of movie files, which

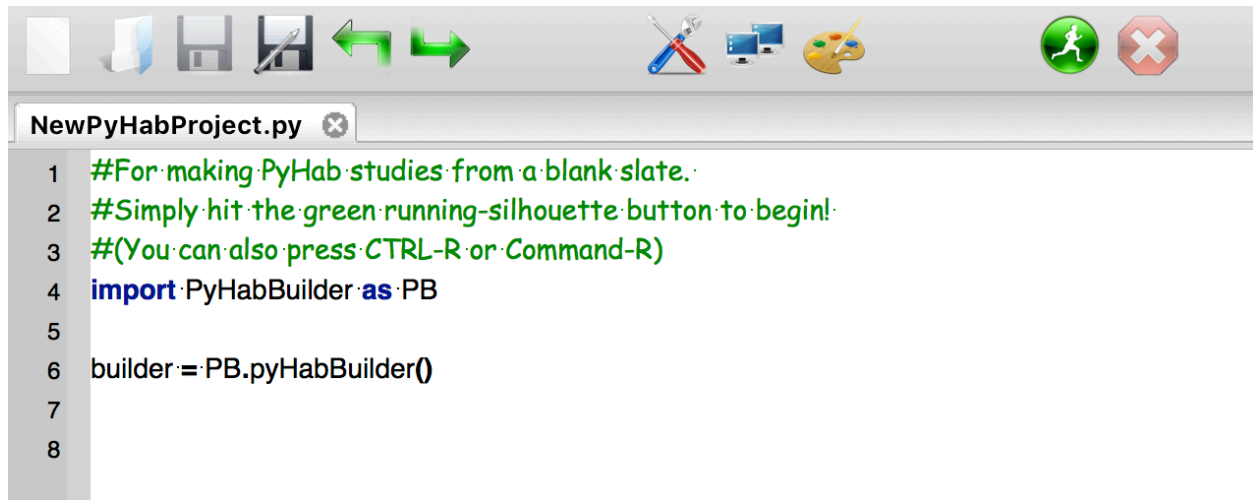
you tell PyHab where to find in the settings. You can also run PyHab just as a looking-time coding software with no stimulus presentation, in which case no stimulus files are needed.

- **Attention-getter:** A short sound and animation designed to attract an infant's attention to the stimulus screen.
- **Primary coder and Secondary coder:** When to advance through trials, PyHab *only* pays attention to the primary coder. The secondary coder is someone coding at the same time for determining reliability, but is completely optional.
- **Verbose data vs. Summary data:** Summary data is a single file that gives one line per trial, and some summary statistics for that trial. Verbose data is a data file where each line is one gaze on or gaze off. Verbose data is much more granular and is necessary for calculating certain forms of inter-rater reliability. Each coder gets their own verbose data file.
- **Preferential looking version:** PyHab comes in two versions, and the main difference between them is that the one called "PrefLook" is designed to be used with preferential looking designs, i.e. studies in which there are two targets that the infant can look at, and you want to code which one they are looking at. The "normal" PyHab only allows you to code whether they are looking at the screen or not. The preferential looking version does not allow for multiple coders and does not compute reliability (but reliability can be computed after the fact).

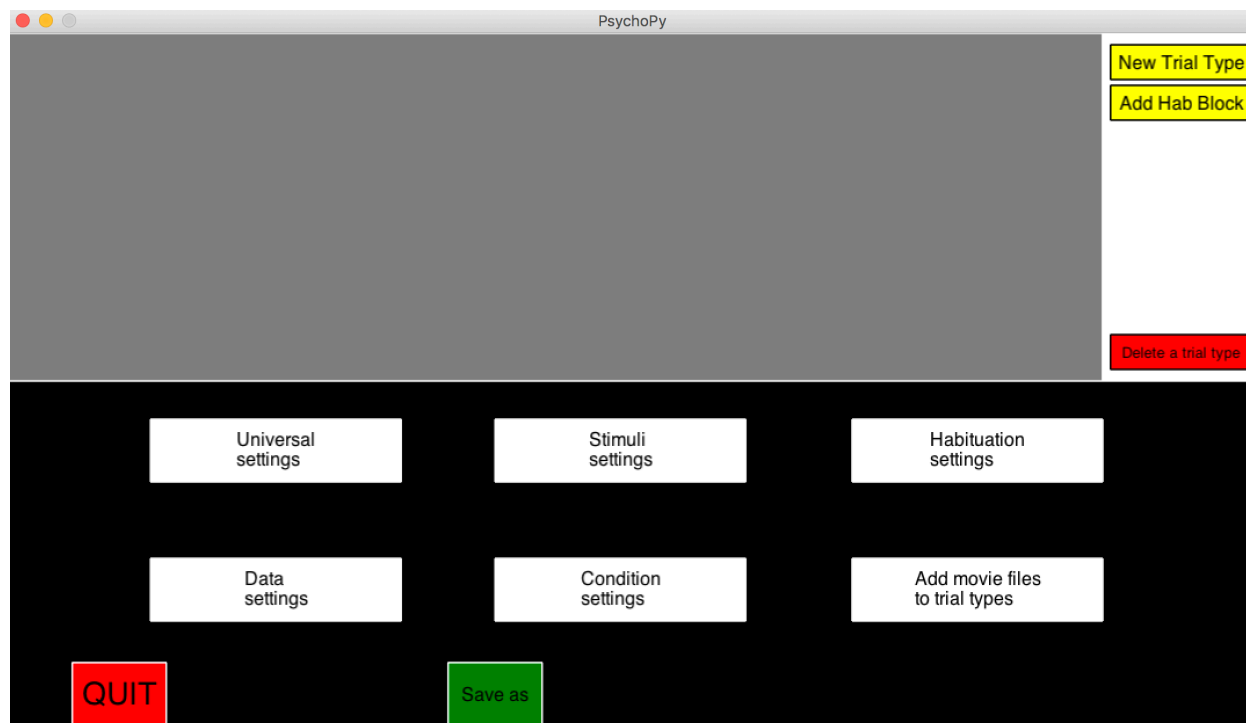


## b. Creating your first experiment

Open the “NewPyHabProject.py” script in the PsychoPy coder window (see Chapter 2). You will see something like this. Hit the green running man icon on the right to launch the builder window.



This may take a few seconds to start up, but a new window should open that looks like this.



This is your main builder view, and what you will be dealing with most of the time. The top area with the grey background is the study flow, which will ultimately show you the order in which your trials will be presented. The area on the right is the trial types palette, where you can define new trial types for your study, edit existing trial types, or delete trial types. All of the other settings can be accessed through the six buttons in the lower half of the screen.

Each of the buttons here will open up a separate dialog box to let you change the settings. Note that, on Windows, the builder window will close while the dialog box is present. This is intended behavior. It will reopen when you click “OK” or “Cancel” in the dialog. On Mac, the builder window will remain open, just in the background.

## Trial types

The first thing to do is create a trial type. In the white area in the top-right corner of the builder window, there are three buttons: “New Trial Type”, “Add Hab Block” and “Delete Trial Type.” The “Add Hab Block” button will create a new “Hab” trial type, which is a block of habituation trials. Habituation blocks have certain special properties, and the name of the trial type MUST be “Hab”. You cannot name any other trial type “Hab”, it is the only reserved trial type name.

When you click “New Trial Type”, a window will appear that looks like this<sup>2</sup>:

The top line is the name of your trial type. “TrialTypeNew” is a default, you can replace it with whatever you want (except Hab). The second line sets the maximum duration of the trial, in seconds. By default this is set to 60 seconds. Whatever you put here must be a number.

The next line is a drop-down menu that lets you set whether the trial type is gaze-contingent, which has three settings. If this is set to “no”, the trial will run until its max duration, every time, regardless of infant’s looking behavior. If it is set to “OnOnly”, the trial will run until the maximum duration or the “Minimum on-time” criterion has been met (see “Universal Settings”). In other words, this is for a trial you want to end after the infant has looked at it, but regardless of whether they look away. If it is set to “Yes”, it will end either at the max duration or when the minimum-on AND look-away criterion are met.

After that is a check box determining whether you want the experiment to auto-advance

---

<sup>2</sup> A different version of this window will appear when you add a habituation block.

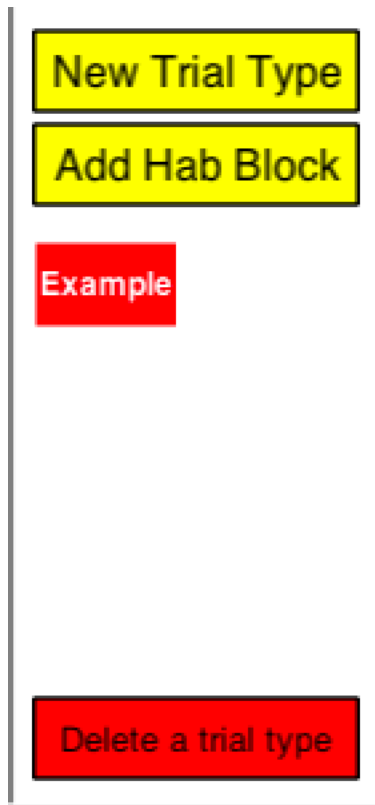
into that trial. PyHab’s default behavior is that each trial must be started manually. In stimulus presentation studies this typically means pressing a key to present the attention-getter and then starting the next trial. If this box is checked, PyHab will instead automatically begin trials of this type after ending the previous trial, as soon as the ITI in the universal settings has passed. If you also turn off the built-in attention-getter (see stimulus settings), then this will automatically start playing the next movie as well.

Following that is a check-box to determine whether to play the built-in attention-getter for that trial. The built-in attention-getter is a looming, spinning yellow rectangle accompanied by a rising musical scale. I’ve found it to be pretty effective. If your stimulus files have built-in attention-getters, just uncheck this box. Note that if you do, you won’t be able to pause between the attention-getter and the rest of the stimulus, it will just start playing the stimulus as soon as you start the trial. This also affects coding when you are not using PyHab for stimulus presentation: If this is set to “yes”, each trial will start at the first gaze-on (if it is not set to auto-advance). If not, it will start as soon as the attention-getter key is pressed, or if auto-advance is turned on, as soon as the previous trial ends.

The final check-box is a little dangerous. Say your stimulus video is a movie, and you don’t want the movie to end anywhere in the middle, but once some criteria have been met you want the trial to end at the end of the video’s current loop. In that case, you would check this box. The dangerous part is, this can make the timing of your trials different depending on whether you are running in stimulus presentation mode or not, i.e. if you are doing secondary coding. This is the only setting that can really strongly affect that. If you aren’t using PyHab to control your stimulus presentation, this check-box does nothing.

Once you have named your trial type and set the options, hit “OK”. This will add the trial type to your trial type palette on the right. At current, PyHab supports a maximum of eight distinct trial types. Note that these are not the number of trials in the experiment, just the number of different *kinds* of trials. There is no hard limit on the number of trials in your experiment.

Now your trial type palette will look something like this:



From here, there are three things you can do:

- Left-click on the name of the trial type to add an instance of that trial type to the study flow.
- Right-click on the name to edit information about the trial type, like changing its maximum duration or whether it is gaze-contingent, or to remove movie stimuli from it (adding movie stimuli is different).
- Click the “Delete a trial type” button to remove the trial type from the palette (and all instances of the trial type from the study flow).

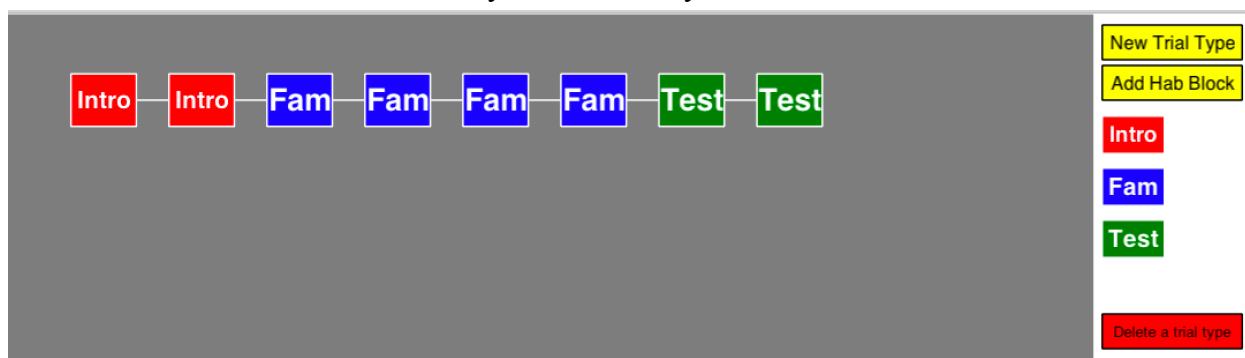
We’ll go through each of these in turn.

## Making your study flow

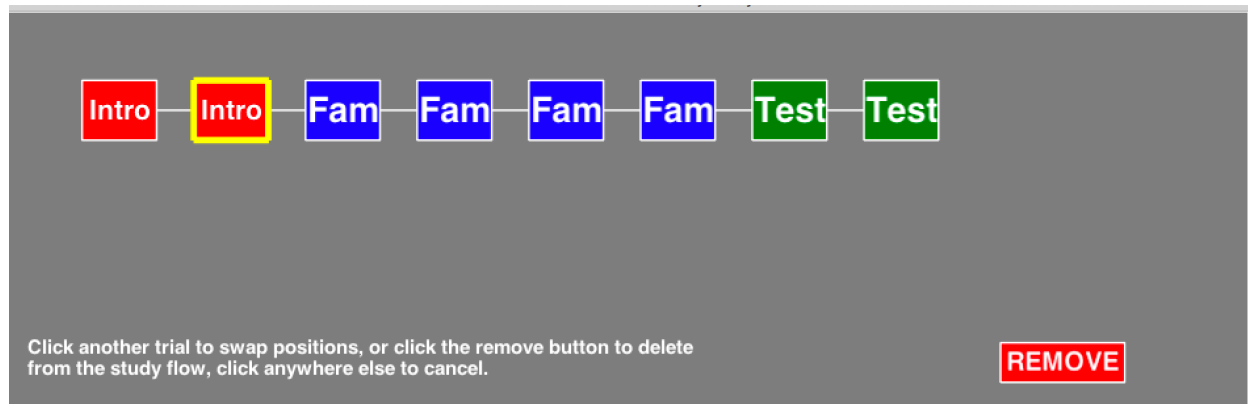
The “study flow” is the order in which different trial types will be presented. Say you have three trial types, intro, familiarization, and test. Your palette might look something like this.



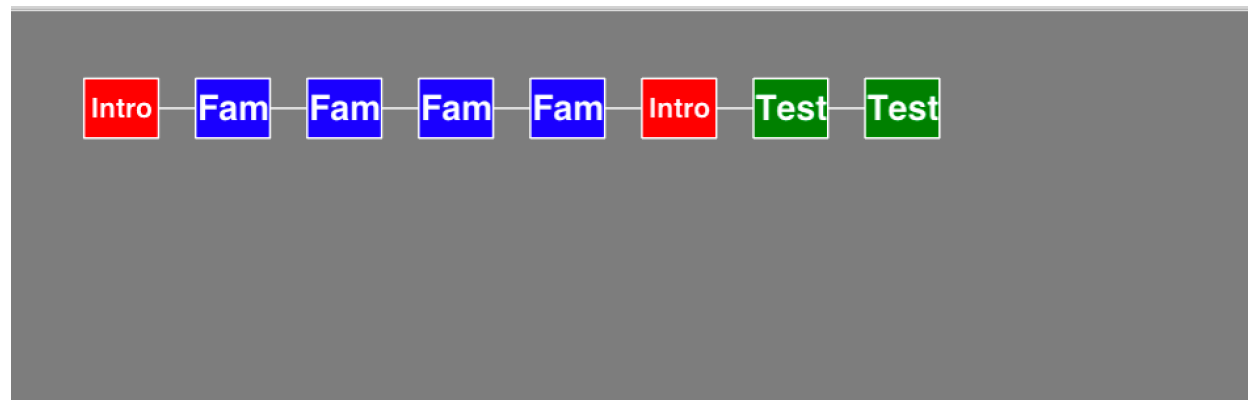
Now say you wanted participants to go through two intro trials, four familiarization trials, and two test trials. You would click the intro button twice, the “fam” button four times, and the “test” button twice. This would leave you with a study flow that looked like this.



To move trials around in the study flow, or to remove them altogether, simply click on the trial in the flow. Say you wanted to move the second intro trial so that it was between the last familiarization trial and the first test trial. First, click on the second intro trial, which will highlight it, like so.



The instructions at the bottom are pretty self-explanatory. To move the intro trial to after the last Fam trial, simply click on the last Fam trial and they will switch places.



Note that this is ALL you can do with the study flow directly. If you want to change the order of stimuli within a trial type, or change the settings of a trial type, all of that has to be done in their respective menus.

Habituation trials are the one special exception, here and everywhere else. You do not insert each habituation trial manually. Rather, you insert a habituation *block*, which includes a number of trials or meta-trials (see “Habituation Blocks”) determined by the habituation settings. To indicate this, “Hab” blocks are double the width of trials in the study flow.



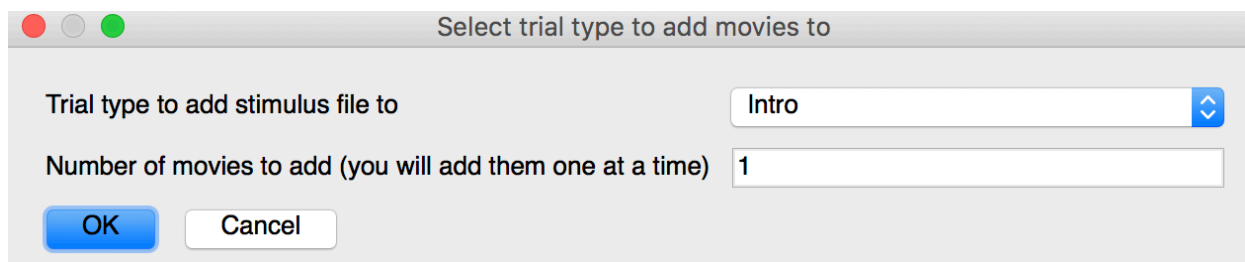
This study flow could have any number of trials, based on the settings for the Hab block. All the flow will show you is the trials before and after the habituation block, not those during.

## Stimuli and trial types

You don't need to use PyHab to control your stimulus presentation, but you can. By default, PyHab is designed to present video files as stimuli. We will delve into the stimulus settings later, here I will focus on the interaction between stimuli and trial types.

The logic of PyHab is that for every trial type (except Hab) there will be some number of stimuli files associated with it. Every time an instance of that trial type appears in the study flow, PyHab will present one stimulus file. The next time that trial type appears in the flow, it will present the next stimulus file on the list, and so on for however many times that trial type appears in the flow. If there are more instances of a trial type than stimulus files, PyHab will simply loop around and present the first stimulus file again. The order (or even inclusion) of stimulus files can be controlled with the Conditions settings, but here we will start with the basics of adding and removing stimulus files from trial types.

To add a stimulus file or multiple files to a trial type, simply click the button that says “Add movie files to trial types.” This will bring up a dialog that looks like this.



The top field is a drop-down list of all of the trial types. Pick the one you want to add the stimuli files to. The second line is the number of stimuli files you will be adding. You can set this to whatever number you want, and PyHab will then allow you to add that many files, selecting them one at a time.

When you hit “OK”, PyHab will open your operating system’s “open file” dialog. Simply navigate to the movie file you wish to add, select it, and click “open”. This process will repeat as



many times as you specified. The order in which movies are added is the default order in which they will be presented for that trial type<sup>3</sup>.

Say you want to remove a stimulus file from a trial type. Simply right-click the trial type in the trial type palette in the top right. This will open up a dialog that is very similar to the one you used to create the trial in the first place, but with some new additions.

You can modify the name of the trial type and all of its other settings just as you did before. However, if you want to remove a movie file from a trial type, simply uncheck the box next to the movie file's name and hit "OK", and that movie will be removed. If you are working with a PyHab experiment that has been previously saved, it will remain in the stimuli folder (see "Saving and sharing your experiment"), but it will no longer be part of the trial type.<sup>4</sup>

## Deleting trial types

The "delete a trial type" button at the bottom of the trial palette is straightforward. Click it, and a dialog will open with a drop-down list of the trial types you have created. Select the one you wish to delete and hit "OK". This will remove *everything* associated with that trial type. Any

<sup>3</sup> Habituation trial types are, as ever, an exception: They only ever show a single stimulus file, and if you add multiple stimuli files to the habituation type, only the first one will be shown. However, which one is the first one can be changed in the conditions, so if you want to have different habituation conditions, add all the movies to the tab trial type and then set your conditions accordingly.

<sup>4</sup> If you are using conditions as well, you will need to change or remake your conditions file to account for these changes or the program may crash.

instances of that trial type will be erased from the study flow, all stimuli associated with it will be disassociated from the experiment and need to be re-added to other trial types, and any settings that were specific to that trial type (e.g., maximum duration) will be forgotten.

## Habituation blocks

Habituation blocks are not like normal trial types. Clicking the “Add Hab Block” button will do two things. First, it will create the ‘Hab’ trial type, which is the trial that will be used to compute habituation criteria and when those criteria have been met. It’s a trial that is in most ways like any other trial type, but with certain restrictions. If you just have a simple habituation design where you present a single habituation trial over and over again, that’s all you need. However, it also gives you the option to create Habituation meta-trials that actually consist of multiple distinct trial types. These are a little complicated, but allow you to make very sophisticated designs.

For example, say you wanted a design where, on every habituation trial, the habituation criterion was computed on the basis of looking at a still frame at the end of a video, but not counting the infants’ looking to the video itself. To accomplish this in PyHab, you would first create a trial type for the video alone (call it ‘A’ for convenience), and then create a Habituation meta-trial that consisted of two sub-trials, one of type ‘A’ and one of type ‘Hab’, where the content of the ‘Hab’ trial type was the still frame of the video.<sup>5</sup>

The key features of these meta-trials are:

- Habituation criteria, both setting the criteria and determining when they have been met, will only pay attention to looking behavior during ‘Hab’ trials within a meta-trial, but data for all sub-trials will be recorded.
- You must create the trial types *other than* ‘Hab’ before creating the meta-trial. The ‘Hab’ trial is created when you create the meta-trial.

---

<sup>5</sup> Under current implementation this would actually be a looping video of the final frame. Upcoming versions will allow still images as stimuli.

- The habituation **block** will consist of repetitions of these meta-trials. You cannot change the content of the meta-trials within a given run of the study. If you set it to be ‘A’, ‘B’, ‘Hab’, it will always be ‘A’, ‘B’, ‘Hab’.
- If you want some videos to only be played before the first habituation trial, but not before all subsequent habituation trials, you would **not** make a meta-trial, you would just put the other trial before the habituation block in the study flow.
- You can have trial types that only appear as part of habituation blocks and not in any other part of the study.

When you click “Add Hab block”, this window will open:

The screenshot shows a window titled "Hab block creator" with the following settings:

Hab trial settings	
Maximum duration	60
Auto-advance INTO trial without waiting for experimenter?	<input type="checkbox"/>
Play attention-getter on this trial type? (Stim presentation mode only)	<input checked="" type="checkbox"/>
Hab block sub-trials	
Use sub-trials? (If checked, new dialog will open)	<input type="checkbox"/>
Number of sub-trials (INCLUDING Hab trial)	0

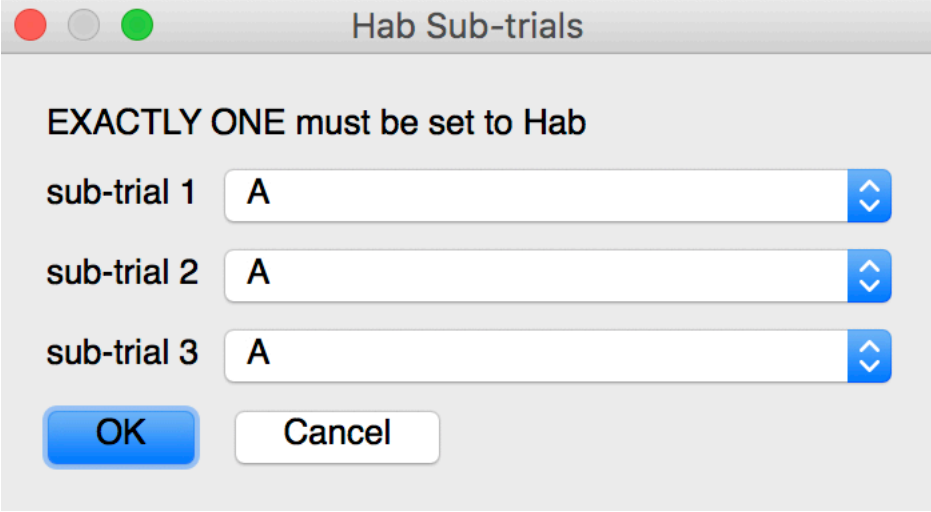
At the bottom are "OK" and "Cancel" buttons.

The first three settings define the features of the habituation trial type itself, which is a trial type much like other trial types you have created, except it will always be named ‘Hab’ and it will always be gaze-contingent. Again, it is this ‘Hab’ trial type ALONE that will be used when computing looking times for habituation criteria. If you just want a simple habituation trial, you can stop there and ignore the last two lines.

The last two lines allow you to create habituation meta-trials, that consist of multiple sub-trials. You check the box, and enter the number of trials in the habituation meta-trial in the next box. Note that this number must include the ‘Hab’ trial itself. So, for example, if you wanted each habituation meta-trial to consist of one trial of type A, one of type B, and the actual ‘Hab’ trial, you would set the number in the final box to 3.

If you have checked the sub-trials box and entered a number greater than 1 in the last box,

a new dialog will open that looks like this. In this example, we put 3 in the final box.



**Hab Sub-trials**

EXACTLY ONE must be set to Hab

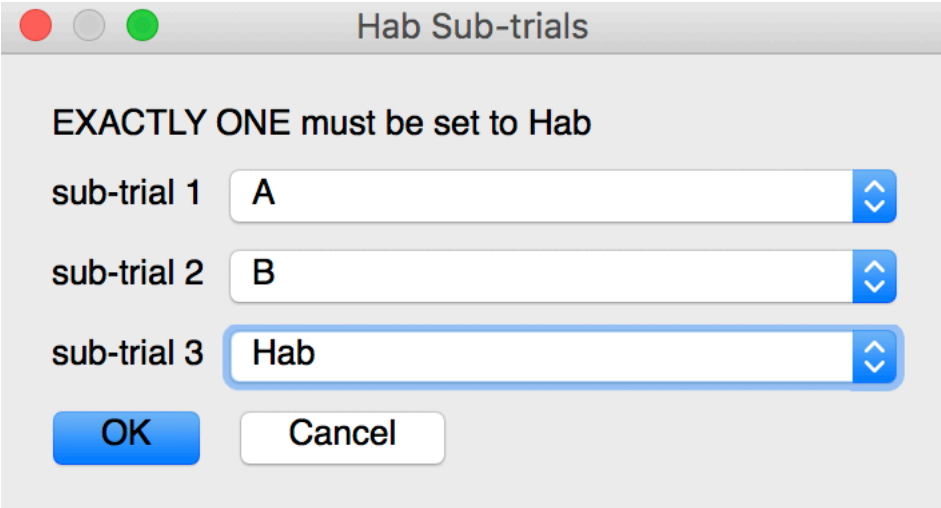
sub-trial 1 A

sub-trial 2 A

sub-trial 3 A

OK Cancel

This is like a miniature study-flow for each of your habituation meta-trials. Each drop-down has a list of all the trial types, including the ‘Hab’ type you have just created. Exactly one of these drop-downs must be set to ‘Hab’, not more, not less. So, for example, we might set up a meta-trial like this:



**Hab Sub-trials**

EXACTLY ONE must be set to Hab

sub-trial 1 A

sub-trial 2 B

sub-trial 3 Hab

OK Cancel

This says that during your habituation block, participants will see an instance of trial type A, an instance of trial type B, and an instance of the ‘Hab’ trial type, and this sequence of three will repeat until the habituation criteria have been met.

When we put that into our study flow, it looks like this:



**Note that trial types A and B do not actually appear in the study flow**, but they are included in the “Hab” block. This flow would present ‘A’, ‘B’, ‘Hab’, looping in that order, until the habituation criteria were met, and then present a single trial of type ‘C’.

To change the content of a meta-trial, simply click the “Mod Hab Block” button that replaces the “Add Hab Block” button.

# Settings windows

## Universal settings

Universal settings

Experiment name: PyHab

Number of continuous seconds looking away to end trial: 2

Minimum time looking at screen before stimuli can be ended (not consecutive): 1

Minimum ISI between loops, in seconds: 0

Experimenter blinding: 0 = no blinding, 1 = Trial number and gaze on/off only, 2 = Trial active/not active only: 0

Single-target or preferential looking?: Single-target

OK Cancel

The “Universal Settings” button will bring up the universal settings dialog, which allows you to control various aspects of the experiment that apply regardless of anything else. Whether or not you are using PyHab for stimulus presentation, no matter what kinds of trials you have, these settings apply to everything you do!

The “experiment name” is the prefix that will be associated with the launcher script for your experiment (see “saving and sharing your experiment”) as well as the prefix for all data files.

The next two lines control when gaze-contingent trials end. With the settings in this image, fully gaze-contingent trials will end when the infant has looked away for two continuous seconds after looking at the display for a total of one second, or when the maximum duration of that trial type has been reached, whichever comes first. For “OnOnly” gaze-contingent trials, trials will end after the infant has looked at the display for a total of one second (not consecutively).

The ISI between loops (minimum) is for when you want to enforce a pause between loops of your stimuli during a trial. Decimals are allowed. Even if you are not using PyHab for stimulus presentation, this is used to determine the timing of a trial.

There is a separate setting for short delays at the start of a trial, after the attention-getter, but this is “how long after the end of a trial do you want to wait before the experimenter can start

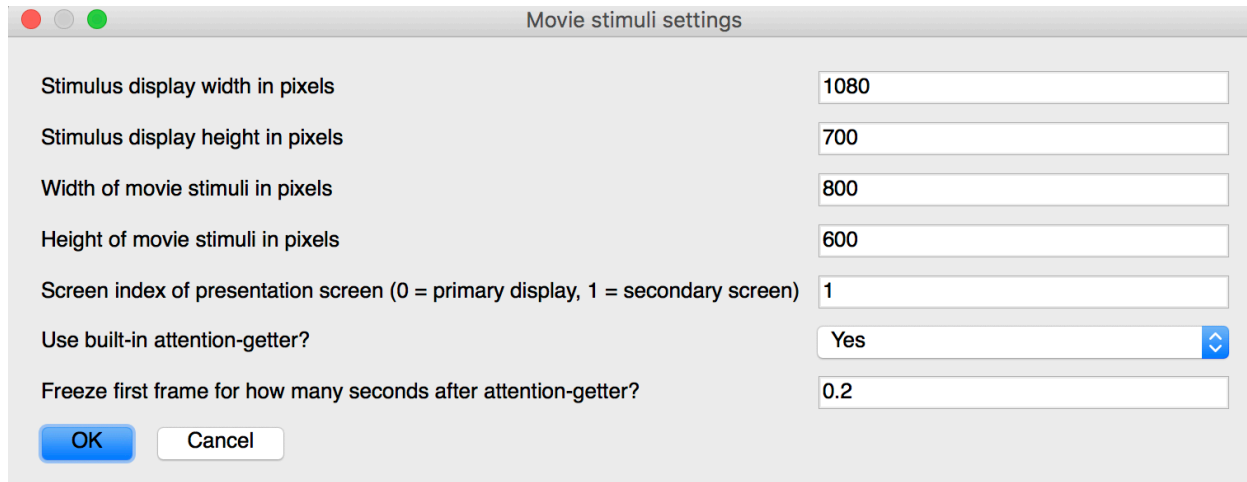
the attention-getter for the next trial?” The number is in seconds. Decimals are allowed. That can be found in the Stimuli Settings.

The experimenter blinding settings determine how much information is available to the experimenter while running the study (see Ch. 4). There are three settings:

- 0: Maximum information is presented to the experimenter, including trial number, trial type, upcoming trial type, and when each coder is indicating that the infant is looking at the screen or not.
- 1: Experimenter is blind to trial type, but can see trial number and looking coding on/off for each coder. Good for standard blinding to condition/trial type.
- 2: Maximum blinding. The only information the experimenter sees is whether the trial is active or not. If it is, the coder display box(es) will be blue, and when a trial is not active, they will be black. Great for doing reliability coding.

Finally, the very last setting may be the most important of all: Whether your experiment is single-target or preferential looking. A single-target experiment is one in which you only care whether the participant is looking at the display or not. Preferential looking is when you want to know not just *whether* they are looking at the display but *which* part of the display they are looking at. Preferential looking enables PyHab to code gaze-on L and R, whereas single-target only codes gaze-on and gaze-off. If you have *any* preferential-looking trials in your experiment, this must be set to preferential looking.

## Stimuli settings



Setting	Value
Stimulus display width in pixels	1080
Stimulus display height in pixels	700
Width of movie stimuli in pixels	800
Height of movie stimuli in pixels	600
Screen index of presentation screen (0 = primary display, 1 = secondary screen)	1
Use built-in attention-getter?	Yes
Freeze first frame for how many seconds after attention-getter?	0.2

Buttons: OK, Cancel

Clicking the “stimuli settings” button will bring up this dialog. If you are not using PyHab to control your stimulus presentation, the last two lines are relevant, but nothing else is. If *you are* using PyHab to control stimulus presentation, then all of this matters.

The top two lines are the width and height of the stimulus display area. Generally speaking this will be the resolution of your presentation screen. If you’re using a 1024x768 resolution, for example, you would put 1024 in the top box and 768 in the second.

The next two lines are the width and height of the stimuli *within* that display area. If you want your stimuli to fill the entire screen, these numbers should match the first two. If they are not the same, your stimuli will appear on a black background.

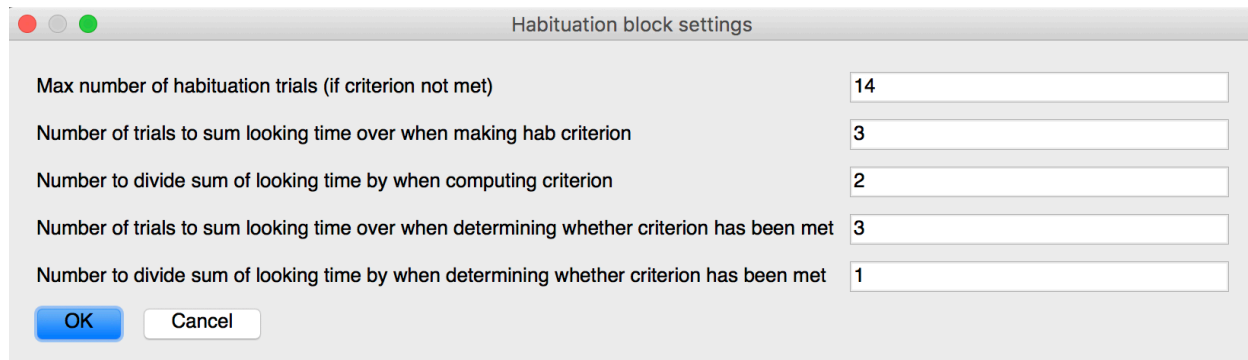
The “Screen index” is basically “which monitor should the stimuli appear on”. By default, on most computers, the “secondary monitor” will be screen index 1, while the primary display will be screen index 0. If you have more than two screens, you can put this up to 2 if you want. Notably, the experimenter interface will typically appear on screen index 0, but if that’s your presentation screen you can manually move it to a different screen.

The final line only matters if the preceding line is set to “yes”. This imposes a minimum delay between the end of the attention-getter and the start of the stimulus. I recommend setting this to at least .1 if you are using PyHab to control stimulus presentation, because if there is no delay the sound from the attention-getter can conflict with any sound in your stimuli. If you aren’t using PyHab for stimulus presentation and auto-advance is NOT checked for a given trial



type, then when you hit the attention-getter key there is a delay of 500ms + whatever is in this box. This is useful for matching the duration of an attention-getter in a recording, for example.

## Habituation settings



Habituation block settings	
Max number of habituation trials (if criterion not met)	14
Number of trials to sum looking time over when making hab criterion	3
Number to divide sum of looking time by when computing criterion	2
Number of trials to sum looking time over when determining whether criterion has been met	3
Number to divide sum of looking time by when determining whether criterion has been met	1
<input type="button" value="OK"/> <input type="button" value="Cancel"/>	

If you click the “Habituation settings” button, you will see this dialog. These settings only matter if your experiment includes a “Hab” trial type, in other words a block of habituation trials (or meta-trials) the length of which is determined by the habituation criterion.

The first line is the maximum number of habituation trials, period. Even if the infant never reaches your habituation criterion, after this many habituation trials, another trial type will be shown. For example, in this study flow:



After *at most* 14 habituation trials, the test trial will be shown, no matter what.

The next two lines determine how the criterion is determined initially. The first number is how many trials you sum the looking time across. The second number is what you divide that total looking time by to set your criterion. So, with the settings pictured, the habituation criterion would be the sum of the looking time across the first three habituation trials, divided by two.

This always counts from the first habituation trial.

The next two numbers determine what looking time is compared to the habituation criterion to determine if the participant has habituated to the stimuli. The first number is again the number of trials to sum looking times across. This picks up from the first trial after the criterion has been set, and then creates a moving window of that many trials. The second number is then what the total looking time in those trials is divided by before comparing it to the criterion. With the settings above, after the 6<sup>th</sup> habituation trial, the program would compare the sum of looking times across trials 4-6 (divided by 1, so, just the sum) to the criterion, and if that value was less than the criterion, the participant would be considered habituated and the test trial would be shown. If the sum of looking times across those three trials was greater than the criterion, after the next habituation trial the program would sum looking times across trials 5-7, and so on and so forth until the sum is less than the criterion or the maximum number of habituation trials has been reached.

## Data settings

The data settings window is just a long list of fields that will appear in your summary data file, with a check-box next to every field. Most field names are self-explanatory, with a few exceptions:

- GNG: Usable/not usable trial. 1 if the trial is usable, 0 if trial was aborted or redone.
- stimName: The name of the movie file (in a stimulus presentation version, blank otherwise)
- condLabel: The label of the condition selected by the experimenter, if condition randomization is used. If you are not using condition randomization, this will simply match the "cond" field.

There are two extremely important things to know about these settings:

1. **Anything not checked will not be recorded in your data in any form.** There will be no

hidden logs or anything of the sort, that information will simply not exist.

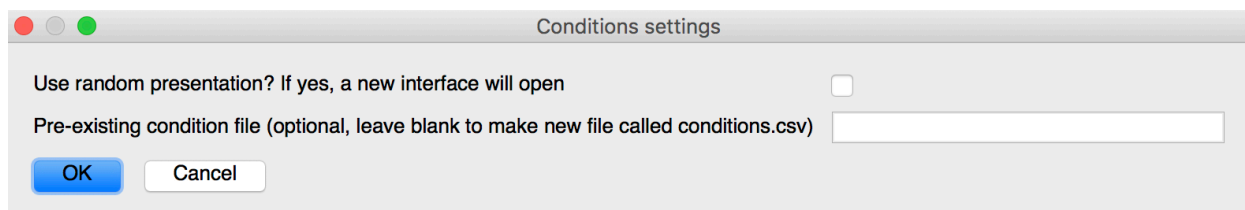
2. **If you change your study from single-target to preferential looking, these options will reset, check every box (so all data is recorded by default), and change the list for the appropriate set of columns.** The data recorded are actually different depending on what kind of study you are running. The program always defaults to including everything, but if for whatever reason you want to omit a column, you will have to revisit these settings if you switch from single-target to preferential looking.

## Condition settings

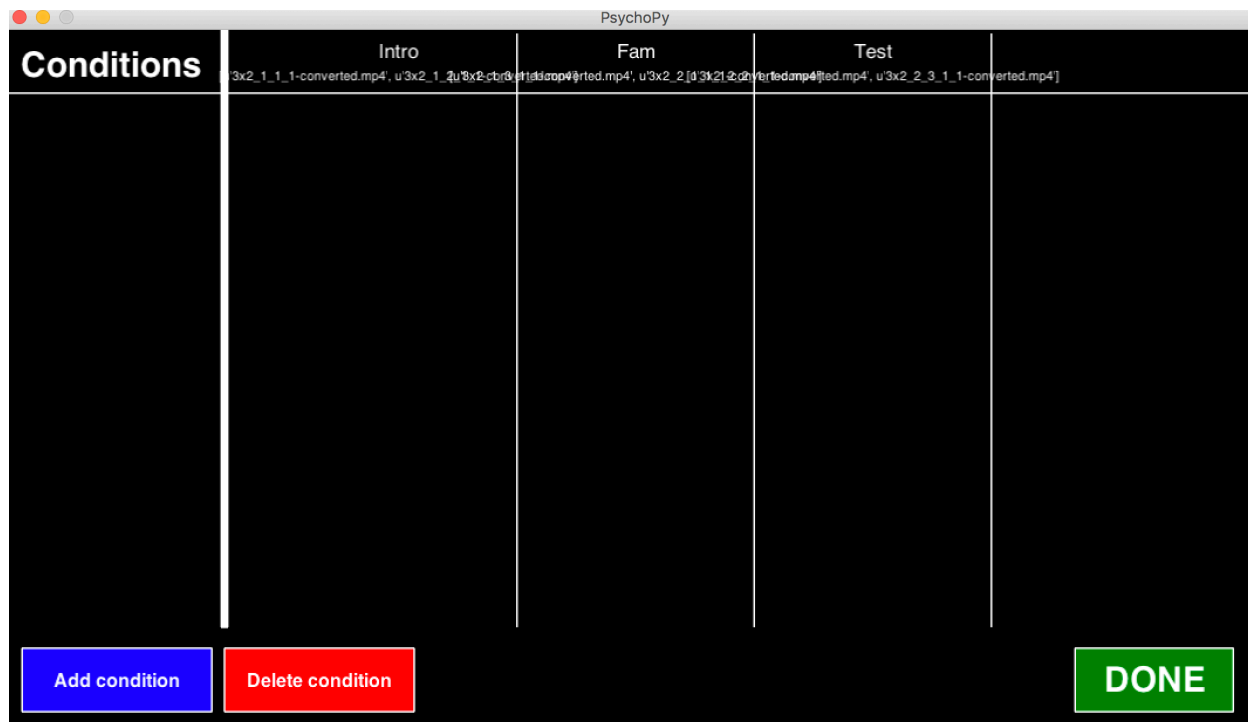
Here is where things get a little complicated. Before you start messing with these settings, here's what you need to know:

1. If you are using PyHab for stimulus presentation, you *must* have all of your movie files attached to all of your trial types before setting up conditions. If you do not, or if you change the movie files later, you will need to redo the condition settings.
2. If you are not using PyHab for stimulus presentation, you will be given the option to manually enter a list of conditions so that you can use a drop-down menu when running the study, instead of entering the condition manually. This list must be enclosed in square brackets, each condition label enclosed in single quotes, separated by commas. For example: ['Condition A', 'Condition B', 'Condition C']

I will focus on condition settings when you are using PyHab for stimulus presentation, because that's when it actually matters. Once you have all of your movies associated with their trial types, click on "condition settings" and you will see this dialog:

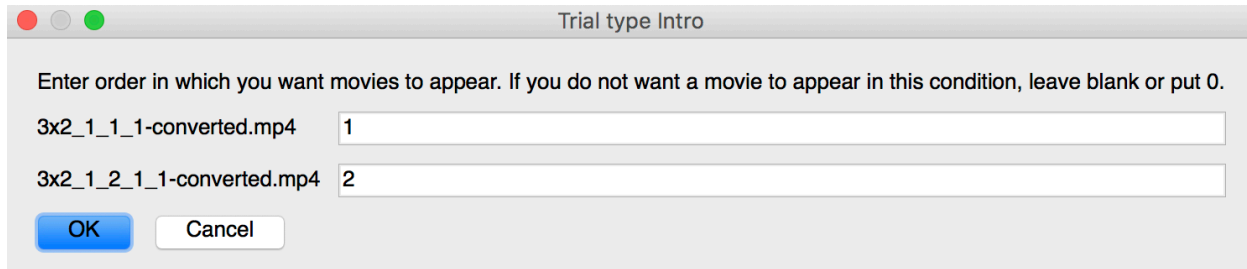


If you want to use condition randomization, check that box. If you already have a condition file (and you can create them manually), you can enter the filename in the second line. If you have movie files associated with trial types, check the box, and hit OK, the whole builder interface will change to the condition interface, which looks like this:



The “Done” button in the bottom-right exits to the regular builder. Across the top you have the list of conditions in the left column, and then each trial type in another column. Under each trial type, with horrible formatting, you have a list of the stimuli files associated with that trial type. That’s mostly there for reference, and it will probably be cleaned up at some point in the near future.

When you click the “Add condition” button, you will go through a series of dialogs. In the first one, you simply set the name of the condition, which will appear in a drop-down menu when you run the study. Then, you will go through a separate dialog for each trial type (according to the order in which the trial types were created). In each of those dialogs, you will see a list of all the movies in that trial type, with numbers next to them. For example, in the demo experiment, if you go to this menu and hit “add condition”, after naming the condition you will see this dialog for the intro trial type:



The numbers represent the order of the movies *in that condition*. So, if in this particular condition you wanted the second movie to appear first, you would simply switch these two numbers. Remember earlier when I mentioned that when you have multiple instances of a trial type in your study flow, it goes through the movies associated with that trial type in order, and loops around as needed? Here is where you determine that order, if you want it to be something other than the default.

In addition, if you want to have a between-subjects manipulation such that different participants see entirely different sets of movies, you can do that with this interface as well. If you do not want a movie to be presented in a given condition, simply leave the field next to it blank or put a 0. For participants in that condition, it will be as if that movie does not exist. This is particularly useful for the habituation trial type, which only ever presents one movie stimulus.

When you hit “OK”, you will then be prompted to do this again for each of the remaining trial types. After the last trial type, you will be returned to the condition screen, and the condition will now be listed.



Conditions	Intro	Fam	Test
A	[1, 2]	[1, 2]	[1, 2]

Add condition
 Delete condition
 DONE

The numbers refer to the default order of movies. So, this condition is identical to the default.

You can modify a condition by clicking on the row it occupies, and delete it using the “delete condition” button, which will bring up a list of conditions much like the “delete trial type” button. You can have as many conditions as you want. When you are finished simply click “Done” to be returned to the regular builder screen.

## c. Saving and sharing your experiment

When you first create an experiment, there will be a “save as” button and a “quit” button in the bottom left of the builder window. The quit button will also prompt you to save before you quit. When you save a study for the first time, a standard save dialog will open. Choose where you want to save the study to, type a name for the folder the study will occupy, and hit save.

This will create a completely self-contained folder that includes everything needed to build and modify your study. In this folder will be a file named “[studyname]Launcher.py”. Whenever you want to run or modify your study, simply open this script in PsychoPy and run it. It will launch a dialog that allows you to select whether to run the study or launch builder, and if you are running the study it will allow you to choose whether to present stimuli.

This folder will also contain a folder with complete copies of the builder and running scripts, as well as a folder containing all of your stimuli (it will be copied from its original location, not moved), and a folder where all of your data will be saved.

Because of this, moving your study between computers is trivial. Depending on the size of your stimuli you could email it, or if not use a flash drive. Best of all, it is trivial to share your experiment on open science platforms like OSF. Simply upload the folder as-is, and anyone should be able to download and run it.

There are potential obstacles to this simplicity when it comes to moving between Windows and Mac, or more accurately when moving from Mac to Windows. For whatever reason, PsychoPy’s movie-playing functionality is much, much better on Mac, and some movie files might not work on Windows machines.

## 4. Running a study in PyHab

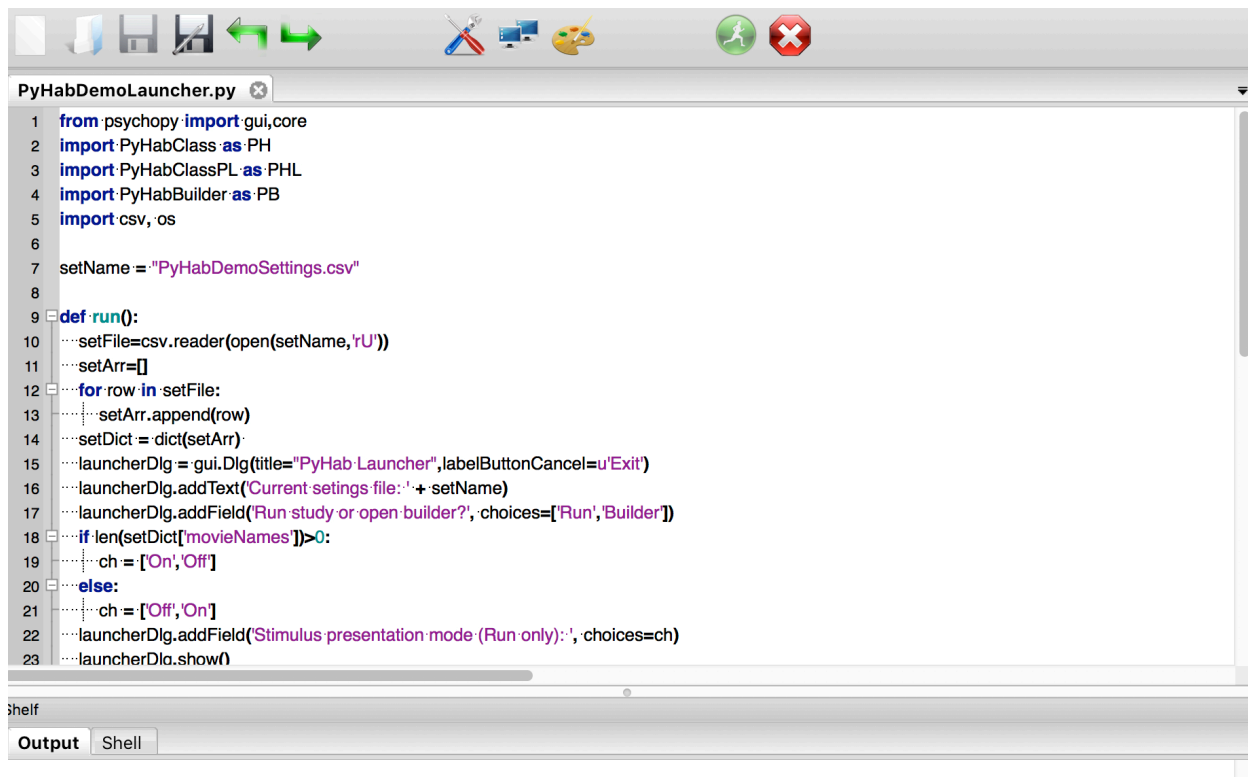
You (or your PI) built a study in PyHab and now you're ready to run some participants! These instructions are meant to be accessible to anyone, whether you're the one who designed the study or an RA helping to run it. Once the study is built, this is everything you need to know.



## a. Starting a PyHab study

First, open PsychoPy and make sure you are in Coder view. Then, in PsychoPy, open the “[studyname]Launcher.py” script located in the study folder. There will be a bunch of other .py scripts in this folder. You can ignore them.

Once you have opened the script, your window will look something like this.



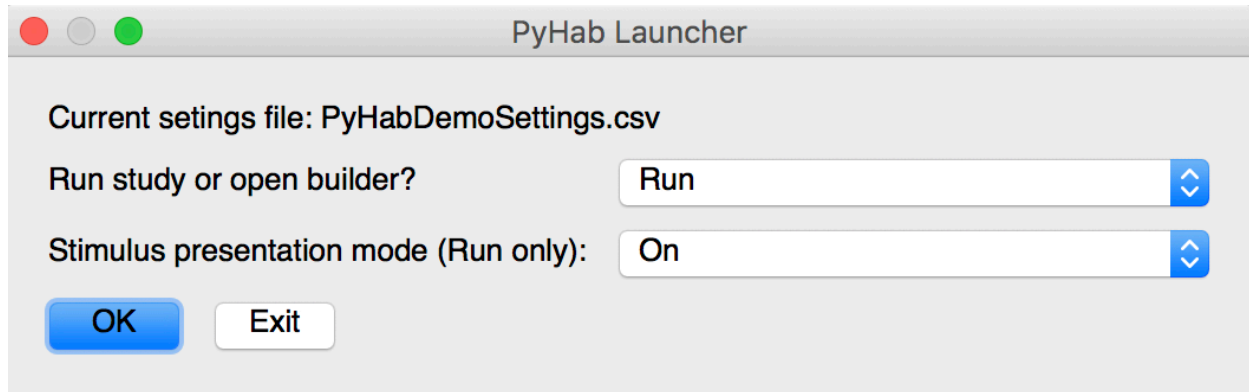
The screenshot shows the PsychoPy Coder window with the file 'PyHabDemoLauncher.py' open. The code is as follows:

```

1 from psychopy import gui, core
2 import PyHabClass as PH
3 import PyHabClassPL as PHL
4 import PyHabBuilder as PB
5 import csv, os
6
7 setName = "PyHabDemoSettings.csv"
8
9 def run():
10     setFile = csv.reader(open(setName, 'rU'))
11     setArr = []
12     for row in setFile:
13         setArr.append(row)
14     setDict = dict(setArr)
15     launcherDlg = gui.Dlg(title="PyHab Launcher", labelButtonCancel=u'Exit')
16     launcherDlg.addText('Current settings file: ' + setName)
17     launcherDlg.addField('Run study or open builder?', choices=['Run', 'Builder'])
18     if len(setDict['movieNames']) > 0:
19         ch = ['On', 'Off']
20     else:
21         ch = ['Off', 'On']
22     launcherDlg.addField('Stimulus presentation mode (Run only):', choices=ch)
23     launcherDlg.show()
  
```

At the bottom of the window, there is a 'shelf' area with 'Output' and 'Shell' tabs.

Simply click the green “Run” button in the top right. This will open a dialog that looks like this.



The first line specifies whether you want to run the study or open up the builder view to modify it. Most of the time you will have it on “Run”. The second line is whether or not you want PyHab to present stimuli. This can be set to “off” for offline coding or for studies in which PyHab is not responsible for the stimulus presentation (for example, if your stimuli are a live puppet show). When you are ready, hit “OK”.

When you hit OK, after a few seconds, a dialog box will pop up. There are a few different ways it can look, but the simplest one looks like this:

 A screenshot of the 'PyHab Movie Presentation Experiment' dialog box. It has a title bar with standard macOS window controls. The form contains several input fields under the heading 'Subject info': 'Subject Number:', 'Subject ID:', 'sex:', 'DOB(month):', 'DOB(day):', 'DOB(year):', and 'Cond:'. Each of these has a corresponding empty text input box. Below these is a dropdown menu for 'Verbose data/multiple coders?' which is currently set to 'Y'. At the bottom are two buttons: 'OK' (highlighted in blue) and 'Cancel'.

Here’s what each of those fields does:

- The subject number and subject ID go into how the data files are named. Importantly, if you have already run a participant with the same subject number and ID, it will overwrite the existing file! You can’t leave these fields blank or the program will

crash.

- Sex is just for the data file. It can be left blank.
- The DOB fields are for the participant's Date of Birth, which is used to compute their age at the time of the study run. You need to input the month, day, and year separately, each one in **two-digit** form. So, for a birthday of January 15, 2015, you would put 1 in the DOB(month) field, 15 in the DOB(day) field, and 15 in the DOB(year) field. DO NOT put the four-digit year, it will give you a very strange age calculation.<sup>6</sup>
- Cond is condition the participant is assigned to. It can either be a fill-in-the-blank, as you see in this image, or a drop-down list of different conditions.
- Verbose data/multiple coders is a drop-down list of Y or N that defaults to Y. If it's set to Y, it will produce a "verbose" data file that includes every single look and look away. If set to Y it also allows a secondary coder, and if one is detected it will compute inter-rater reliability. If it is set to N you will **only** get the summary data file.
- If you are not using PyHab for stimulus presentation, there will be three further fields not shown on this screenshot, that say DOT(month), DOT(day), and DOT(year). These are for recording the Date of Test, i.e. the day on which the study was run. This allows you to re-code data after the fact, and still get an accurate age calculation in the output file.

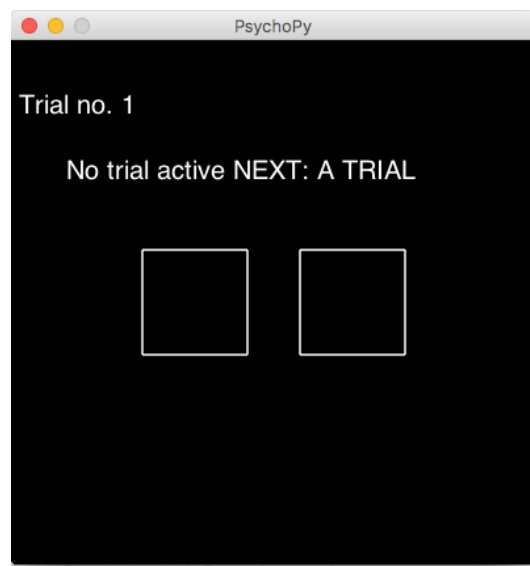
When you've entered all this information, hit OK. This will start the study run.

---

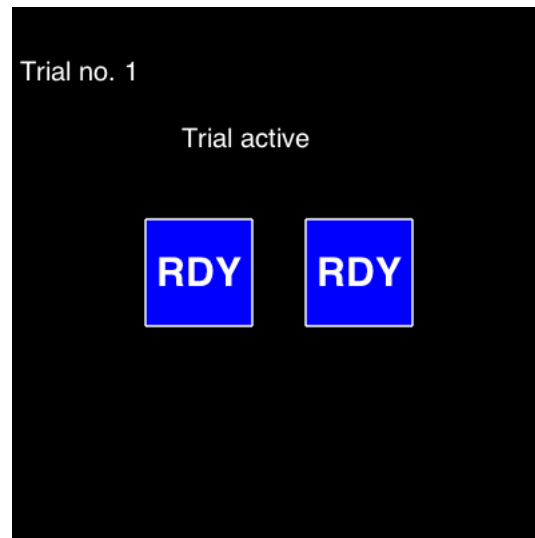
<sup>6</sup> If you are running this program in the year 2100 or later (over 80 years after its initial creation), the age-at-run computation code may need to be modified for people born in the 2090s. Also holy sh\*t.

## b. Running a PyHab study

Once you have hit the “OK” button, if PyHab is being used to present stimuli, the stimulus window will appear on the stimulus presentation screen. Whether or not it is being used to present stimuli, a window will pop up on the experimenter screen that looks something like this:



How much information is present depends on the blinding settings, but this is an example of the most informative mode. The trial number is displayed at the top. The trial status is the next line of text, along with the trial type of the next trial. The two boxes are the coding status for two coders (if verbose is set to N, only one box will be displayed). The left box corresponds to the primary coder, the right box is the secondary coder. If you hit the “A” key, it will prepare or begin the first trial, and the display will look like this:



The blue color tells you it is now ready to begin the trial. If the trial does not start automatically (which is an option that can be configured with `autoAdvance`, see universal settings), the primary coder can start the trial by initiating the first gaze at the screen by pressing the 'B' key.

Depending on the blinding settings, the boxes may show whether each coder is indicating that the participant is looking at the screen or not. When a coder's key is held down, their box will be green and say "ON". When they are not holding down a key, it will be red and say "OFF". When the trial ends, both boxes will go black. With strict experimenter blinding, the boxes will not indicate ON or OFF, but remain blue while a trial is active and turn black when the trial ends.

When the final trial is finished or the "end experiment" key is pressed, PyHab will save the data and (attempt to) close. For various reasons having to do with PsychoPy, Python, and certain operating systems, it will sometimes get stuck while closing and you will have to hit the red "stop" button to close it down completely. The data will still be saved, however.

If PyHab ends the study run normally, it will return to the launcher window it started with, allowing you to either run it again, go into builder view, or exit.

## Key commands

Key	Function
A	Play attention-getter (if PyHab's attention-getter is enabled) and ready trial, or start it if auto-advance is enabled.
B	Primary coder gaze-on (or gaze-on Left for pref. looking). Hold down as long as the infant is looking at the screen, release when they are looking away. If auto-advance is not enabled, the trial will start when this key is pressed for the first time.
L	Secondary coder gaze-on. Just like primary coder, but does not control the flow of the study.
Y	End experiment/quit. For fuss-outs or other situations where you need to end the run early. Immediately saves data and quits the program.
R	Abort/redo trial. If pressed during a trial, ends the trial and marks it as no-good, and then allows the experimenter to restart that trial from the beginning. If pressed between trials, allows the experimenter to redo the last trial, marking the previously recorded trial as no-good.
J	If you are using a habituation-dishabituation design, you can press this between trials during the habituation block to immediately jump to the test trials. In other words, it behaves as though the habituation criterion was met even when it wasn't.

I	If you are using a habituation-dishabituation design and are right before the test trial (because the criterion has been met or because of the number of habituation trials presented), this key will insert another habituation trial.
P	Print trials so far to the output window in PsychoPy. Only works when not presenting stimuli, and only works in-between trials. Will not work if auto-advancing.
M	Preferential looking version only. Gaze-on Right. Preferential looking only supports one coder, so this key can also mark the beginning of the first gaze-on to start a trial. In every way identical to the B key, except indicating that the participant is looking at the other side of the screen.

## c. Data and computing reliability

### Data

PyHab saves up to four different .csv files at the conclusion of a study:

- Summary data file: Each line is one trial. Columns are determined by the data settings and can be in whatever order the settings specify. The list of all possible settings is:
  - sNum: Subject number (as entered when you start the study)
  - months: Participant age in months; days, days is in the next column.
  - days: Participant age in days above and beyond the age in months.
  - sex: Participant sex
  - cond: Condition. If a condition file is used, this will correspond to the right column of the condition file for that participant's condition. Otherwise it's whatever is entered in the "cond" field.
  - condLabel: If a condition file is used, this will correspond to the left column, i.e. the condition selected. If a condition file is not used, it will be the same as cond.
  - trial: Trial #
  - GNG: Good/No-Good. 1 if the trial is usable, 0 if the trial was aborted or redone. Usually 1.
  - trialType: The trial type
  - stimName: If stimuli are being presented, this will correspond to the name of the movie file for that particular trial.
  - habCrit: Habituation criterion, if applicable. If the study does not involve a habituation sequence, this will be 0. If it does, it will be 0 until the trial on which the criterion is calculated, at which point it will be the habituation



criterion.

- sumOnA: Total gaze-on time in the trial, as coded by the primary coder.
  - numOnA: Number of gaze onsets in the trial, as coded by the primary coder.
  - sumOffA: Total time looking away in the trial, as coded by the primary coder.
  - numOffA: Number of gaze offsets in the trial, as coded by the primary coder
  - sumOnB, numOnB, sumOffB, numOffB: As above, but for the secondary coder.
- Verbose data file A: The primary coder's "verbose" data file is saved if verbose is set to Y when the study is run. The verbose data file is very similar to the summary file, except that instead of recording summary statistics for each trial, each row is a single gaze-on or gaze-away. Some of the columns are identical to the ones in the summary data file, but there are four new ones:
    - GazeOnOff: Indicates whether the row is showing a time when the gaze-on button was held down, or a time when the participant was looking away. 1 if gaze-on, 0 if gaze-off.
    - StartTime: The time at which the gaze-on or gaze-off began, relative to the start of the trial.
    - EndTime: The time at which the gaze-on or gaze-off ended, relative to the start of the trial.
    - Duration: The difference between EndTime and StartTime.
  - Verbose data file B: If there are two coders, and the secondary coder inputs at least one gaze, a second verbose file will be created for the secondary coder. It is identical to the verbose data file for the primary coder.
  - Reliability: If a verbose data file B is created, then PyHab will also attempt to compute reliability statistics comparing the two coders, which will be output to this file. There are four reliability statistics:
    - Weighted percentage agreement: Percentage of frames on which the two coders agreed, weighted by the length of the trial.

- Cohen's Kappa: A reliability statistic between 0 and 1, preferred by some journals and researchers.
- Average Observer Agreement: The raw percentage of frames on which the two coders agreed.
- Pearsons R: Standard correlation coefficient.

## **Standalone reliability**

There is also a stand-alone reliability script which allows you to take any two verbose data files and get the afore-mentioned reliability statistics. When run, the script asks for information about the subject, and when you hit OK, it will open two file-open dialogs, one after the other. Select the two verbose data files you want to compute reliability for, and it should save a reliability .csv in whatever folder the script is found in. It will also print the results to the PsychoPy output window.

## d. Preferential looking studies

PyHab has a variant for preferential looking designs, in which infants have two possible targets they can look at and you want to code which one they are looking at, not just whether they are looking at the screen. Running a preferential looking study is *almost* identical to running any other study in PyHab, with a few important differences:

- The preferential looking version only supports one coder. You cannot have two people simultaneously coding a preferential looking design (just not enough keys on the keyboard). However, you can still have one person do live coding and another person do off-line coding and then get reliability using the standalone reliability script.
- The ‘B’ key is now “gaze-on Left”. The ‘M’ key is “gaze-on Right”. They work completely identically except that one indicates a gaze to the left and the other indicates a gaze to the right. Either can be used to start a trial, time spent looking away (for determining when to end a trial) is only computed when neither button is being held down.
  - If your study has some single-target trials and some preferential looking trials, just use one of the keys for the single target trial.
- In the data, the looking-time columns will now be “sumOnL”, “numOnL”, “sumOnR”, “numOnR”, “sumOff”, and “numOff”. No more A and B (because only one coder). As you might expect, if it says L it refers to gazes to the left, if it says R it refers to gazes to the right.
- The verbose data file now has three codes in the gazeOnOff column: 0 = “off”, 1 = “gaze-on Left”, 2 = “gaze-on Right”.

Everything else is the same between the two versions.

## 5. Troubleshooting

PyHab is far from perfect. Development will continue for some time. Let me start by pointing out a few things you don't need to worry about: Every time you run PyHab, there will be some number of messages in PsychoPy's "Output" area at the bottom of the coder interface. If PyHab crashes unexpectedly, there will probably be some blue text here telling you the nature of the crash. Even when it runs successfully, there will be some green text that says WARNING or brown text that says ERROR. Don't worry about those. They don't affect the functionality of PyHab, they're just quirks of PsychoPy and some of the libraries PsychoPy uses.

As for everything else, here are some known issues you might encounter:

- **Program crashes right after pressing the green button, says something involving "segmentation fault":** This is an issue that occurs mostly on Macs, and it's not a PyHab problem. It's just a known issue with Python and MacOS, and it's relatively harmless. Just hit "ignore" on the the error dialog that pops up and try running it again. It sometimes takes two or three tries but it does eventually work.
- **Immediately after entering participant info and hitting OK, the program crashes:** If it doesn't open a window at all, and just crashes as soon as you hit OK, it's almost always due to the DOB or DOT being entered correctly. Leaving other fields blank won't make the program crash outright, but the first thing it does, before even opening the experimenter window and stimulus window, is try to compute the participant's age. If that calculation breaks down for whatever reason, it will crash then and there. Make sure you are entering TWO-DIGIT values for each line of the DOB and DOT, and make sure you're not getting month/day/year mixed up and putting in an impossible value for one or the other. No letters, no symbols, don't leave any DOB or DOT line blank.
- **The stimulus presentation window opens halfway between the two screens:** This is a problem that seems to mostly occur on Macs with Retina displays running PsychoPy 1.90 or later. Unfortunately there isn't really a good fix for it yet, but it should not affect other operating systems or Macs that do not have retina displays.

- **When running PyHab, the program freezes on “Loading Movies”:** This is most likely to occur on Windows, and it basically means your movie files are too big or have too high a bitrate for PsychoPy’s movie system to handle. The only fix for this is to modify the movie files themselves. Since you need VLC anyways, you can open the movies in VLC and use it’s “save/convert” function. Try converting your movies to MP4 and, if possible, reducing the resolution or framerate slightly. Hopefully future versions of PsychoPy or the underlying moviepy code will make this less of a problem.
- **PyHab runs the experiment successfully, but hangs at the very end, either when closing the windows, or the windows close but the red stop button remains highlighted and the green button is grayed out:** If it fails to close the windows, and you had two coders, that’s probably an issue with the reliability calculation. There is one very rare infinite loop in the reliability calculation that I haven’t been able to track down, but it seems to be most common if you only have one trial with two coders on it. If it did close the windows, this is another PsychoPy issue that can’t really be fixed. Again, more common on MacOS than Windows. In either case, just hit the red stop button. In either case, your data will be saved, so there’s no risk in doing this (in the first case the reliability file might not save, but everything else will).
- **At any point, it crashes and displays a “permission error”:** This is usually a Windows problem. Make sure you run PsychoPy as an administrator (right click on the icon and select “run as administrator” or right click and go to properties and change the settings to always run as administrator). PsychoPy needs to control a lot of very low-level features of your computer, and your operating system won’t allow that unless you make it. If something like this happens on Mac, go to your security settings, “Accessibility”, and add PsychoPy to the list of programs that are allowed to control your computer.
- **The program crashes on load or on a specific trial type while using condition randomization, or the builder crashes when loading the condition settings screen:** Basically if you change something about the way movies are assigned to trial types after changing your condition file, and don’t go back and change your condition file as well, PyHab can get confused. If going into the builder and changing the conditions there doesn’t work, close PyHab and do the following:

- Delete the “conditions.csv” file in your study folder.
- Open “[studyname]settings.csv” in something OTHER THAN excel, like Notepad or TextEdit (Excel will reformat some things in a way that breaks PyHab).
- In the settings file, look for three lines: “condList”, “condPath”, and “condFile”. Erase anything after the comma. For “condList”, put “” after the comma. Save the settings file and re-open PyHab.