PsychoPy: a crash course

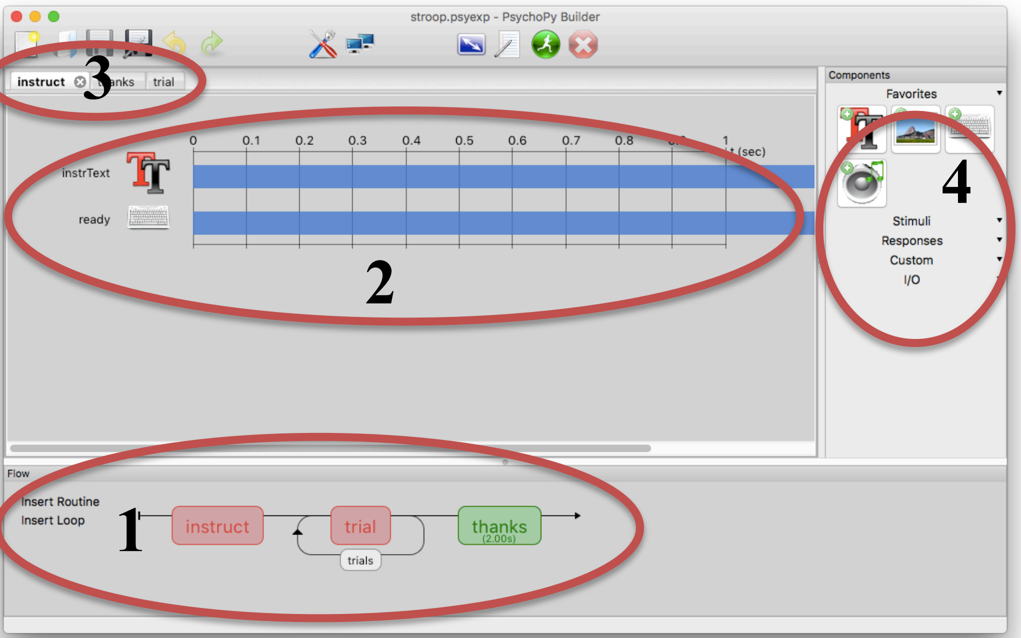
Florent Perek – Tutorial at ICCG-10, Paris, 19 July 2018

In this session, we will learn to use PsychoPy, a software package that can be used to design computerized experiments. PsychoPy is free and multi-platform (available on Windows, Mac and Linux). It is easy to use through a graphical interface. To follow this tutorial you will first need to download the latest version of PsychoPy from http://www.psychopy.org/ and install it on your machine.

PsychoPy is primarily a Python library, i.e., a set of routines written in Python to serve as building blocks for designing experiments. Hence, an experiment created in PsychoPy is in fact a Python script using the PsychoPy library to control screen display, computer inputs, file outputs, etc. The graphical interface, PsychoPy Builder, is merely a frontend to the functions of the Python library: from a WYSIWYG, "point-and-click" representation of the experiment (saved as a .psyexp file), it generates a Python script that can be executed by the interpreter.

1. A first example: the stroop experiment

Launch PsychoPy. Go to the "Demos" menu and select "stroop". Click on "Run" (also in the "Tools" menu) to see what the experiment does:



1. **Flow**: this shows the general structure of the experiment. A PsychoPy experiment consists of a succession of routines, each corresponding to a

different screen. Routines can be within a loop, which causes them to be repeated. Routines can be opened by clicking on them.
2. **Routine panel**: this shows the internal structure of a routine. Routines consist of components, listed on the left, which specified what should be displayed and the different kinds of responses that can be provided by participants. The blue bars represent the temporal extent of each component.
3. **Routine tabs**: list all the routines currently opened and allow to switch between them.
4. **Components toolbox**: shows components that can be added to the current routine. Stimuli components specify what is displayed on the screen. Responses components record participants' feedback using a range of input devices (keyboard, mouse, microphone, button box, etc.).

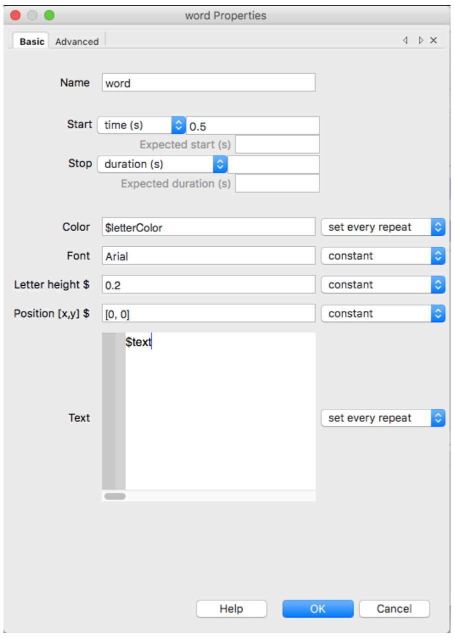
Take some time to poke around and look at the routines and their components. What is the function of each routine? What does each component do within its routine?

NB: You can quit an experiment at any time by pressing the Esc key.

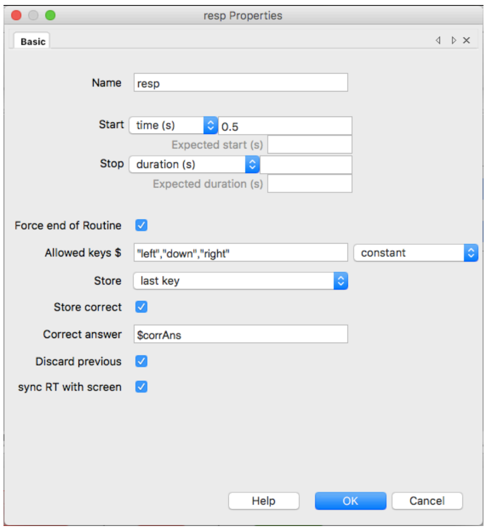
2. Basic features

Properties of components

By clicking on a component, you can display a form listing its properties and allowing you to change them. For instance, you can edit the text shown by a Text component. You can also change various aspects of the behavior of components.


word Properties

Name	word	
Start	time (s)	0.5
Expected start (s)		
Stop	duration (s)	
Expected duration (s)		
Color	$letterColor	set every repeat
Font	Arial	constant
Letter height $	0.2	constant
Position [x,y] $	[0, 0]	constant
Text	$text	set every repeat


resp Properties

Name	resp	
Start	time (s)	0.5
Expected start (s)		
Stop	duration (s)	
Expected duration (s)		
Force end of Routine	✓	
Allowed keys $	"left","down","right"	constant
Store	last key	
Store correct	✓	
Correct answer	$corrAns	
Discard previous	✓	
sync RT with screen	✓	

Start/stop

All components have a "Start" and a "Stop" property. This controls when the component is active. If the "Stop" property is left blank, the component stays active until the end of the routine.

Ending a routine

There are two ways to end a routine. First, a routine automatically ends when all of its components have become inactive, i.e., their "Stop" property has been reached (timeout). Second, a response component can force a routine to end if its property "Force end of Routine" is checked. In that case, the routine ends once an appropriate response has been received. Either way, when a routine ends, the next routine in the flow is executed.

Loops

Placing loops causes one or more routine(s) to be repeated a certain number of times. A specific number can be entered in the field "nReps", but it is more typical (and useful) to control a loop with a Conditions file. In that case, the loop will iterate over the rows of the file (and if the value of nReps if greater than 1, this will be repeated that many times). Conditions files must be tabulated data in XLSX (Microsoft Excel) or CSV (comma-separated values) format; the first row of the table must contain the column names. Interestingly, the data in each row will be available as variables (using the names in the first row), most importantly to populate the routines (see below).

Component property values and variables

Most properties of the components have a special setting, located to the right, that controls how the value of the property is set. There are three possible settings:

- Constant: the value is set for the whole experiment.
- Set every repeat: this setting only makes sense if the routine is within a loop. In that case, the value will be set at every iteration of the loop. This is used in particular in order to update the components according to the content of the Conditions file.
- Set every frame: the value is constantly updated. This is useful if the property refers to some variable and should be updated as soon as the variable changes.

If the value of the property starts with a dollar sign ($), it is interpreted as a variable name. Variables contain information that might change during the experiment. A common example of variables in PsychoPy are the columns of a Conditions file: at every iteration of a loop, a new row is read off the Conditions file and the data is placed in a set of variables, using the column names. Note that these variables are only accessible from within the loop.

Examine how this feature is used in the stroop experiment demo. Look in particular at the properties of the “word” component in the “trial” routine.

NB: constant character strings and variables can be combined using the following syntax: ‘This is a %s.’ %($name). Any number of variables can be added in %(…). Each variable is matched to a %s symbol in the string, in order of occurrence.

Recording responses

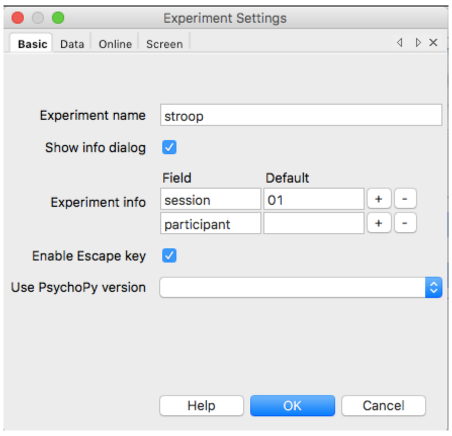
Every time the experiment is run, a different set of output files is created to record responses. The output files are stored in a “data” folder in the same location as the experiment (i.e., the .psyexp file; in the case of the stroop demo, it should be located in a “demos” folder within your home folder). The name of the output files is generated automatically from the name of the experiment, the participant ID, and the date and time of execution. By default, there are three files: a .psydat file, a .log file, and a .csv file. The latter is the most useful for data analysis, as it contains the data for each trial in tabular form.

All response components added to an experiment automatically add data to the output file. The nature of the data depends on the type of component: character or text for the Keyboard component, position of the mouse pointer on the screen (x, y) for the Mouse component, name of the recorded sound file for the Microphone component, etc. In addition, the reaction time is always stored for all types of response components.

Experiment settings

By clicking on this icon, you can change various properties of the experiment.

Notably, you can specify experiment information: variables that are collected every time the experiment is run. This is a convenient way to collect information about the session and the participant (ID number, age, sex, etc.). Experiment information is automatically recorded in the output file.

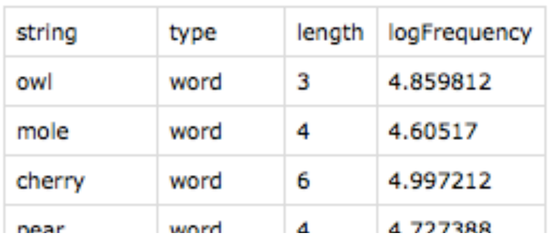


Experiment name	stroop	
Show info dialog	✓	
Experiment info	Field	Default
	session	01
	participant	
Enable Escape key	✓	
Use PsychoPy version		

3. Another example: lexical decision task

Now we will create an experiment in PsychoPy from the grounds up. Namely, we will implement a lexical decision experiment of the kind described in the last session. As a reminder, such an experiment consists in participants deciding, in each trial, if a given string of characters is a word or not.

The starting point is to create a stimulus file; here, it is ready for you:



string	type	length	logFrequency
owl	word	3	4.859812
mole	word	4	4.60517
cherry	word	6	4.997212
pear	word	4	4.727388

This dataset was adapted from the lexical decision dataset ("lexdec") provided by Harald Baayen in the "languageR" package for R. The column "string" contain the string of characters to display (word or non-word), and "type" specifies whether it is a word or a non-word. The other two columns, "length" and "logFrequency", contain the length of the word and non-words in characters, and the log-transformed frequency of the word (from the CELEX database). These two variables are independent variables, but they will not be used in PsychoPy. It is nevertheless good practice to include them in the stimulus file: they will be carried over to the results file, which will be convenient for data analysis.

The lexical decision task experiment should consist of the following:

- A welcome routine.
- An instructions routine.
- A fixation routine which displays a fixation mark (e.g., '+') in the middle of the screen for 1 sec.
- A trial routine, with the lexical decision task proper. Participants will have to use the right arrow of the keyboard if they classify the string as a word, and the left arrow if they classify it as a non-word.
- Both fixation and trial are within a loop.
- A "thank you" routine, ending after three seconds.

Bonus question

Once you have created the regular lexical decision task, you can try to modify it (and the stimuli file) into the following two variants:

- One in which words and non-words would be presented in blocks of two (in a way that is not noticeable to participants); in some of these blocks the two words would be semantically related. This variant could be used to test if the first word primes the second one.
- One in which two words, or two non-words, or a word and a non-word, are presented on the screen at the same time, and participants have to decide if *both* strings are words or not.

These variants could be used to investigate lexical decisions involving, for instance, semantically related words (e.g., synonyms, co-hyponyms, meronyms, etc.) or words involved in collocations.

4. Advanced features

The Code component

As mentioned earlier, PsychoPy experiments are actually Python scripts. When you run an experiment from PsychoPy Builder, it is first translated into Python code which is then executed. You can see this code in the Coder interface by running the "Compile" command in the "Tools" menu (or pressing F5).

This architecture makes PsychoPy very flexible and powerful: features that are not covered by the components' regular behavior can be programmed by inserting Python code. This is done via the Code component. Code components can be inserted anywhere, but the destination of the code itself in the experiment script depends on which field(s) is used:

- Begin experiment: the code is run before any routine is called.
- Begin routine: the code is run at the beginning of the routine where the Code component is placed.
- Each frame: the code is run constantly, every time the screen is refreshed (typically 60 frames per sec for LCD). This is useful for event-based programming, i.e., to trigger some code once particular conditions are met.
- End routine: at the end of the current routine.
- End experiment: at the very end of the experiment.

Code components are a powerful tool, but they do require some programming skills and knowledge of Python syntax. They are several ways of figuring out how to write code to customize PsychoPy's behavior: (i) look at and study the code generated by Builder in the Coder interface (this is how you can find existing variables and their name, for instance), (ii) read the Coder documentation at http://www.psychopy.org/coder/coder.html, (iii) look for code examples on search engines, on PsychoPy’s forum (https://discourse.psychopy.org/), and on the psychopy-users Google group.

In the case of our lexical decision experiment, Code components could be used to add the following features:

- Add a feedback screen after each trial, which checks the keyboard input against the position of the word, and tells the participant whether it was the correct answer (see the “sternberg” demo for an example of this).
- In the second variant, randomize the position of the two strings at runtime for each trial.

Other Stimuli and Responses components

- Image (stimuli): displays pictures (jpeg, png, etc.). The access path can be defined relative to the experiment’s location.
- Sound and Video (stimuli): plays sound and video files.
- Mouse (responses): records input information from the computer mouse. By default, mouse press can only end a routine, but with some code, more complex behavior can be implemented.
- Rating scale (responses): a Likert-style scale with adjustable length, which can also be used to submit a categorical response instead of a numerical rating. Responses can be submitted with the mouse or the keyboard.
- Microphone (responses): records sound from the computer microphone, which allows participants to provide spoken responses. Recorded sound files are saved in a separate subfolder of the “data” folder for every run. One sound file is recorded for every execution of the component, in .wav format by default.

PsychoPy also supports Cedrus and ioLabs button box with bespoke Responses components. These devices allows to record reaction times with millisecond precision. Note that USB keyboards typically have a response latency of 20-30 msec, making them unsuitable to detect differences in reaction time of that order of magnitude. Built-in keyboards (e.g., on a laptop) might have better precision. See http://www.psychopy.org/general/timing/millisecondPrecision.html for more details.