GRADE Workshop Frankfurt, May 11-12th 2021

Aylin Kallmayer, Melvin Kallmayer, Leah Kumle

Helper Document

Consult this document if you need some inspiration for the psychopy tasks. Navigate to the section you are working on right now. Start with the first tipp. If you can't solve the task with this tipp, go on and read the next one. You can also always ask the people in your group for help or discuss the steps together. In case something is still unclear, ask for help from Melvin, Aylin or Leah.

PART 1

1. **Fixation Cross:**
    a. You can just draw the letter "X" or "+" on the screen using a TextStim. Check the "Intro to psychopy" presentation again or look up the documentation here https://www.psychopy.org/api/visual/textstim.html#psychopy.visual.TextStim For more information on how to define a TextStim.

    b. The position of the middle of the screen is specified by using (0,0) in the position argument.

    c. Use core.wait() and win.flip() to specify the timing of your fixation cross.

    d. Check out different arguments in TextStim to play around with the presentation of the fixation cross. For example: Use "height" to change its size (unit = pixel).

2. **Neutral Gaze:**

a. Use the function ImageStim(). Find out more in the “Intro to psychopy” presentation or here: www.psychopy.org/api/visual/imagestim.html#psychopy.visual.ImageStim
b. The timing and position is exactly like for the fixation cross.

3. Gaze Cue:

a. In order for the ImageStim function to find the correct image, you first need to determine the path + name of the image (“stimuli/gaze_left.png”). We want to keep this general so that later we can use this function to draw cues that look to the right *and* left. The function has one input argument: **gaze** which will be “left” or “right” (see conditions function). We can use this to piece together the path.

b. Remember: You can put strings together by using the “+” operator.

```
“This” + “_works” = “This_works”
```

c. Try to combine **“stimuli/”**, **“_gaze.png”** and the argument ***gaze*** (can be either “left” or “right”) to create “stimuli/gaze_left.png” or “stimuli/gaze_right.png”.

d. Assign the path to your gaze cue image to a variable. Hand this variable (e.g. called *path*) to the ImageStim function which will load in the correct image.

4. Response Screen:

A)

a. Create two TextStims. One should present the distractor (always letter “X”). The other one should present the target letter. Use target_tuple and dist_tuple to assign position within the TextStims.

b. Remember: The target letter is stored in the “target” variable.

B)

a. Find and copy the two lines from the present_gaze_cue() function that you need here to create the path to the correct image and draw the stimulus. Paste them them before win.flip()

b. Remember you are drawing all your stimuli first (text and image), then flipping the window once to make them appear.

C)

a. Have a look at the "event.waitKeys()"- function.

b. Remember: We only want to use the keys "y" and "m". Use the argument *keyList* to provide them to the event.waitKeys. The name of the argument gives it aways: you need to wrap them in a list (e.g. keyList = [...]).

PART 2

5. Increasing Number of Trials:

a. Try to find the section in the code where we call the one_trial() function. Look out for #RUN EXPERIMENT.

b. Wrap a loop around the one_trial function.

c. Use the range() function to repeat the trial 3 times. The body of the loop should simply call the one_trial function and the "for … :" part should specify the number of repetitions.

6. Vary the Task Conditions:

A)

a. We already know that we can loop through lists as well (e.g. **for trial in conditions**). The counter (in this case trial) will now take on the value of each

element in the list - that is, trial will equal (“left”,”F”, “left”) in the first iteration of the loop and (“right”, “F”, “left) in the second iteration.

b. “Unpacking” the tuple means assigning each value in e.g. (“left, “F, “left) to its own variable. E.g:

```
gaze = "left"
target = "F"
target_position = "left"
```

c. You can do this in one line of code: gaze, target, target_position = ...

d. Remember: Our tuple in this case is called *trial* inside of the loop.

B)

e. Try to determine the total number of combinations (Hint: Multiply the number of possibilities for each variable).

C)

f. Go to the make_response_screen function. Right now we specify target_tuple = (-200,0) and dist_tuple = (200, 0). This is the correct position if target_position == “left”. Following from this, how should target_tuple and dist_tuple look like if target_position == “right”?

g. Hint: In case target_position == right, target_tuple should be (200, 0).

h. Try wrapping this in an if...elif statement. Check if target_position is “left” or “right” and conditional on this assign target_tuple and dist_tuple.

D)

i. Try googling the function *shuffle()*.

7. Data logging:

a. To return the key pressed you need to do four things:

i. First, you need to tell psychopy that the make_response_screen() function should have an output by returning the variable response at the end of the make_response_screen() function. Returning an output of a function looks something like this:

```
return output
```

ii. In the one_trial() function when we call the make_response_screen() function we need to assign the output of make_response_screen() to a variable (in order to later return this variable).
response = make_response_screen(...)

iii. To make the key available *outside* the one_trial() function, we also need to add it as an output to the one_trial() function. Complicated, I know! Just look at what you did under i). You basically need to do the same thing here for the one_trial() function.

iv. We then also have to "catch" the output (similarly to how we catch it within the one_trial() function by assigning it to response). Otherwise the one_trial functions will just output into nothingness and we will lose the information. We can catch the outputs by assigning them to variables under RUN EXPERIMENT in our loop.

```
response =  one_trial(...)
```