Building experiments in

PsychoPy

By Johnny Lau

Modified on 26 Mar, 2019

PsychoPy – Exercise
Stroop task

Task1 (~15mins)

- Familiarising with the software
- Basic control

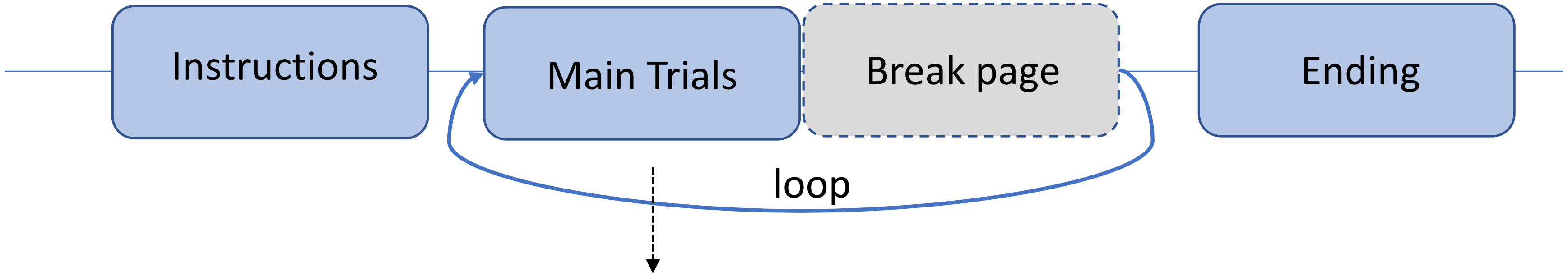
Task 2 (~25mins)

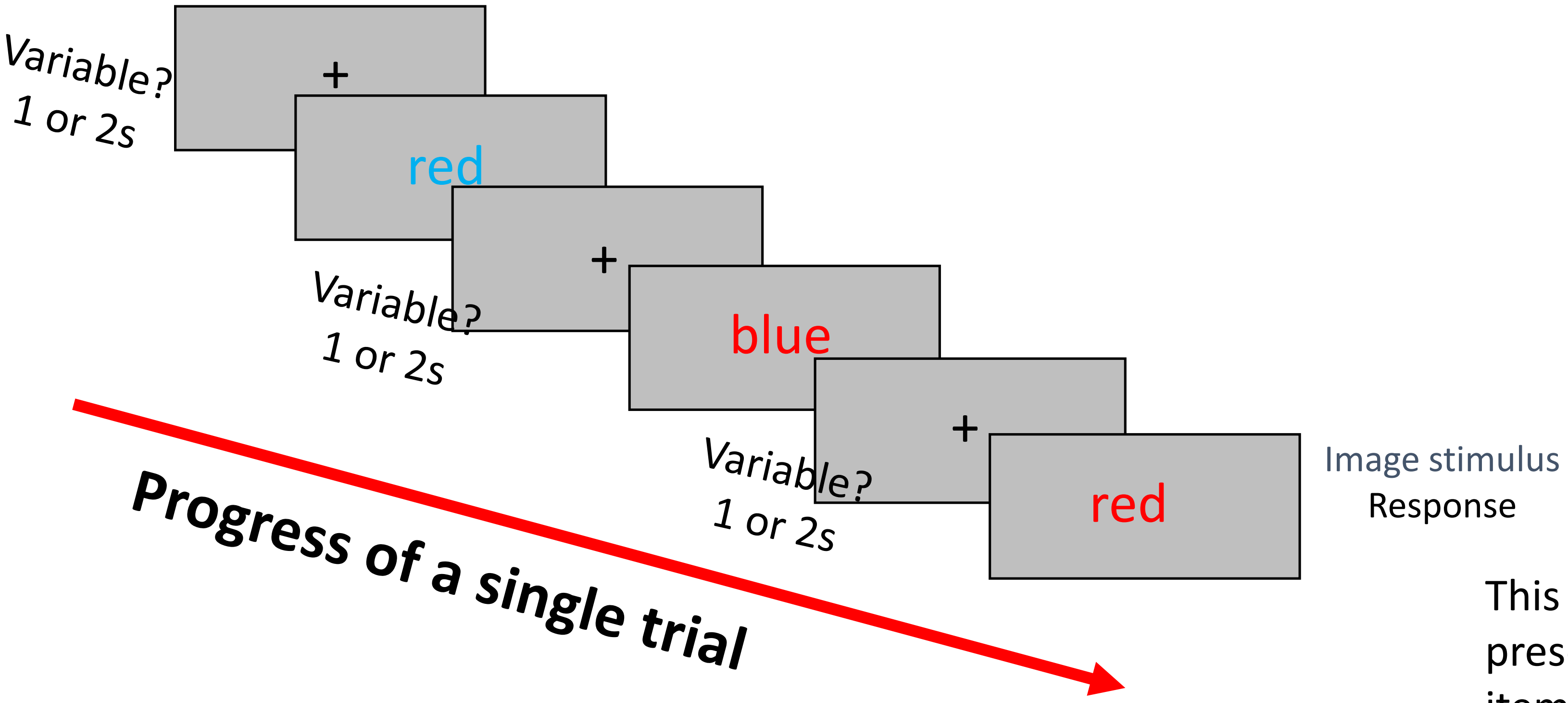
- Exploring other features
- Coding

Task 2

- Exploring other components (Image)
- Simple coding
- Introduce breaks in the experiment

Illustration of the experimental task



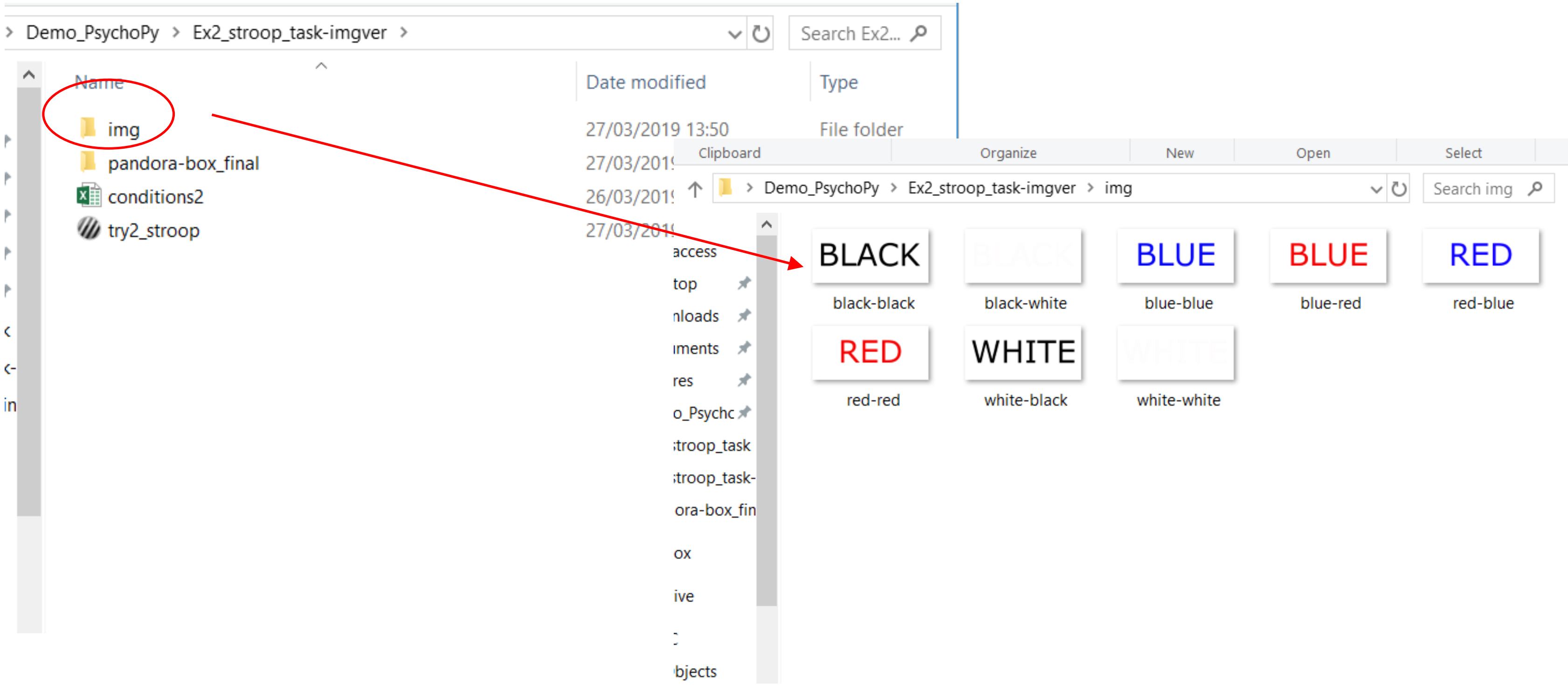


This time we present the stimulus items as images

Preparation

To complete this task, you can either modify your programme from Task1 or start fresh by using the folder ‘**Ex2_stroop_task-imgver**’

In the folder ‘**Ex2_stroop_task-imgver**’, there is a subfolder ‘img’ which contains all the image files you need to complete Task 2. There are 8 images altogether.



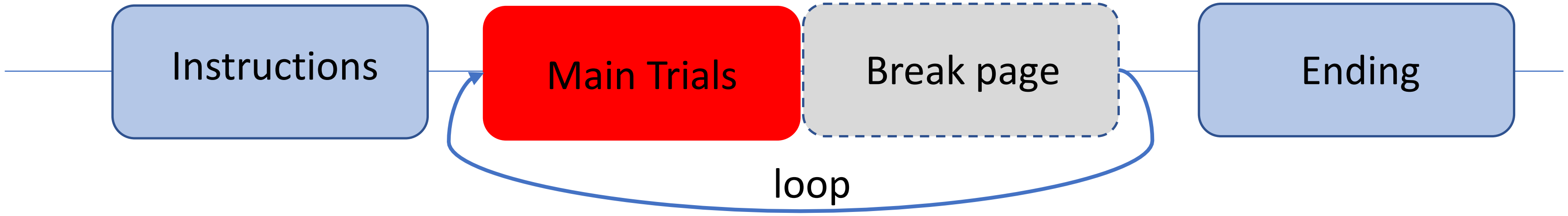
Demo_PsychoPy > Ex2_stroop_task-imgver >

Name	Date modified	Type
img	27/03/2019 13:50	File folder
pandora-box_final	27/03/201	
conditions2	26/03/201	
try2_stroop	27/03/201	

Demo_PsychoPy > Ex2_stroop_task-imgver > img

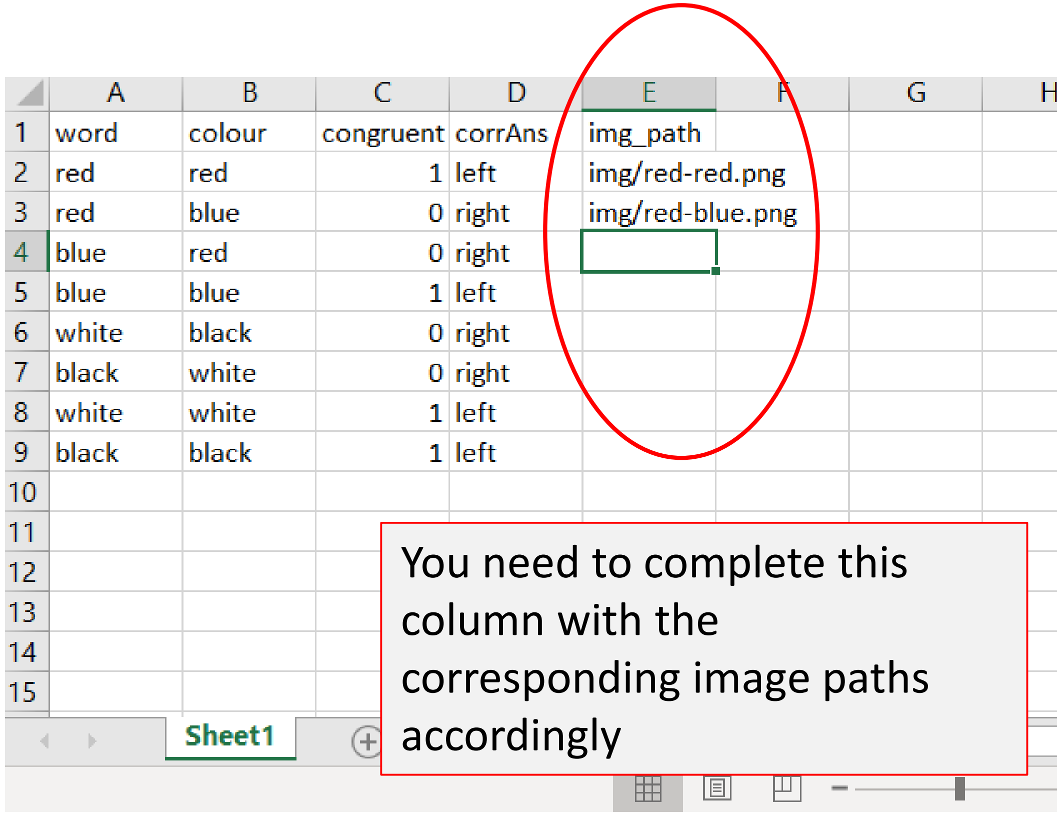
BLACK black-black	BLACK black-white	BLUE blue-blue	BLUE blue-red	RED red-blue
RED red-red	WHITE white-black	WHITE white-white		

Conditions file



Similar to task 1, you need a conditions file so that the programme would know what stimulus items you are actually presenting

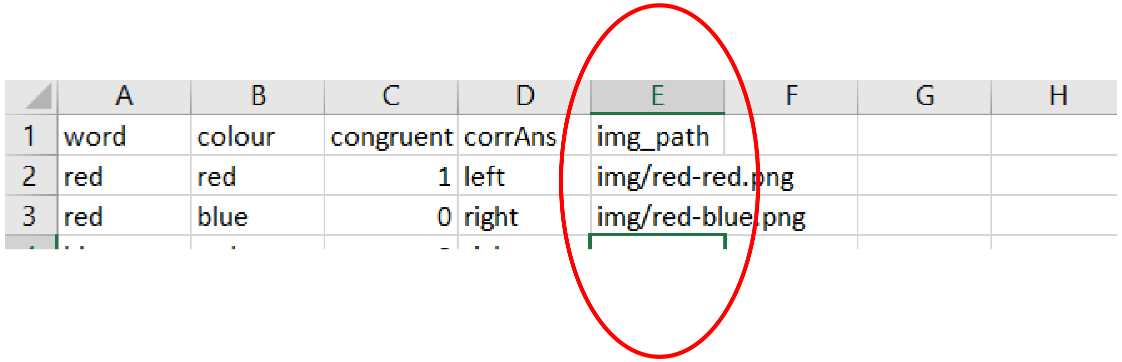
- Now, open the ‘**conditions2**’ excel file
- You probably notice the first 4 columns contain exactly the same information as in the conditions file in Task 1 (i.e. **word, colour, congruent, corrAns**)
- In addition, in this ‘**conditions2**’ file, we need an extra column ‘**img_path**’ to record where the image files are located



	A	B	C	D	E	F	G	H
1	word	colour	congruent	corrAns	img_path			
2	red	red	1	left	img/red-red.png			
3	red	blue	0	right	img/red-blue.png			
4	blue	red	0	right				
5	blue	blue	1	left				
6	white	black	0	right				
7	black	white	0	right				
8	white	white	1	left				
9	black	black	1	left				
10								
11								
12								
13								
14								
15								

You need to complete this column with the corresponding image paths accordingly

A bit more info. about specifying the path of an image



	A	B	C	D	E	F	G	H
1	word	colour	congruent	corrAns	img_path			
2	red	red	1	left	img/red-red.png			
3	red	blue	0	right	img/red-blue.png			

For example, for the image **red-blue.jpg** (which is the word 'RED' coloured in blue), its path should be written like this in the conditions file:

img/red-blue.png

('img' folder is on the same level as your programme file and you will need to specify where your image is located ONLY from this level)

*You don't need to write out its full path (e.g. C:\Username\Desktop\WS3_sempriming\...) because PsychoPy will automatically fill this out for you

Now, open the ‘**try2_stroop.psyexp**’ programme file (builder view) in the folder ‘**Ex2_stroop_task-imgver**’.

The main trials

Creating the main trials:

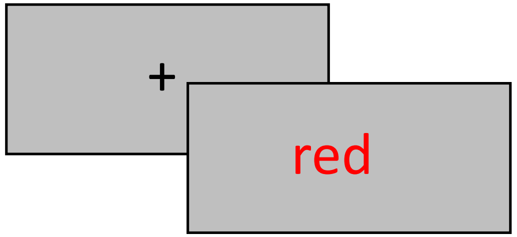
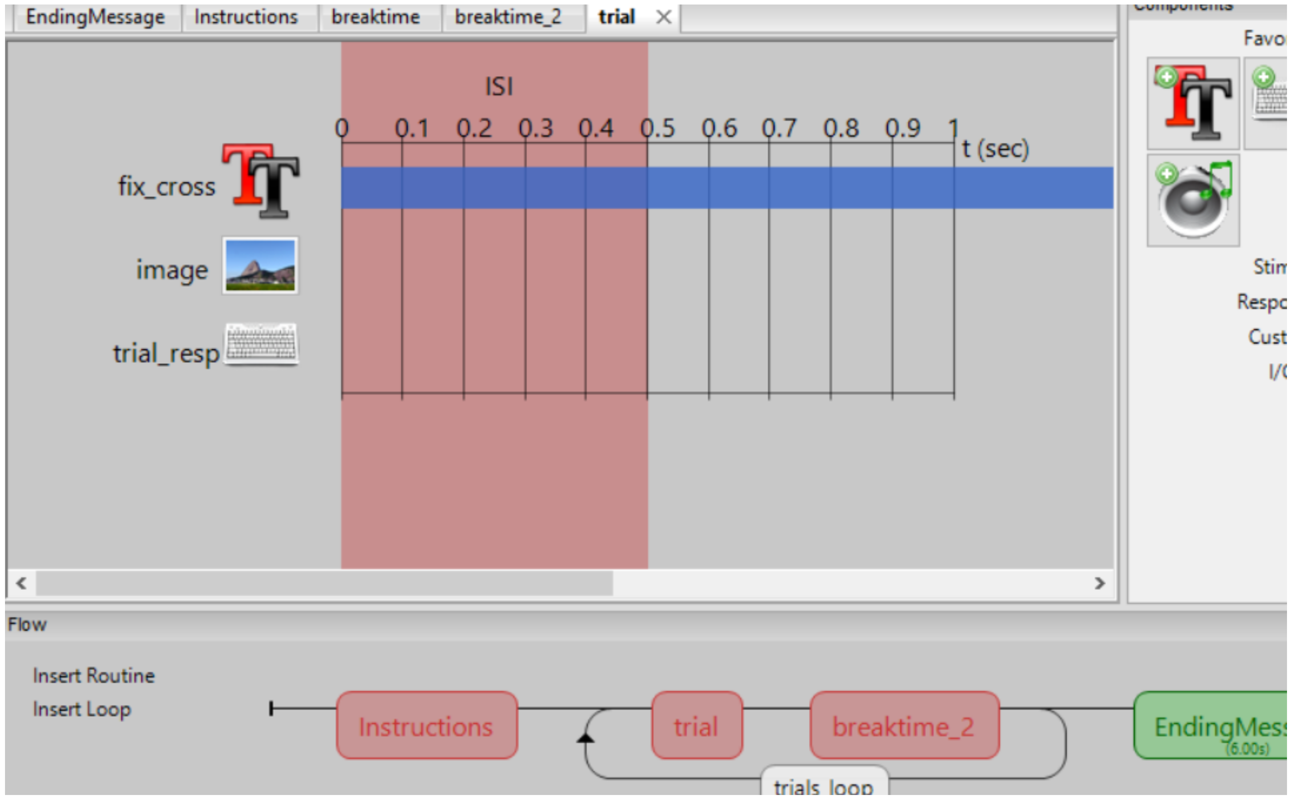


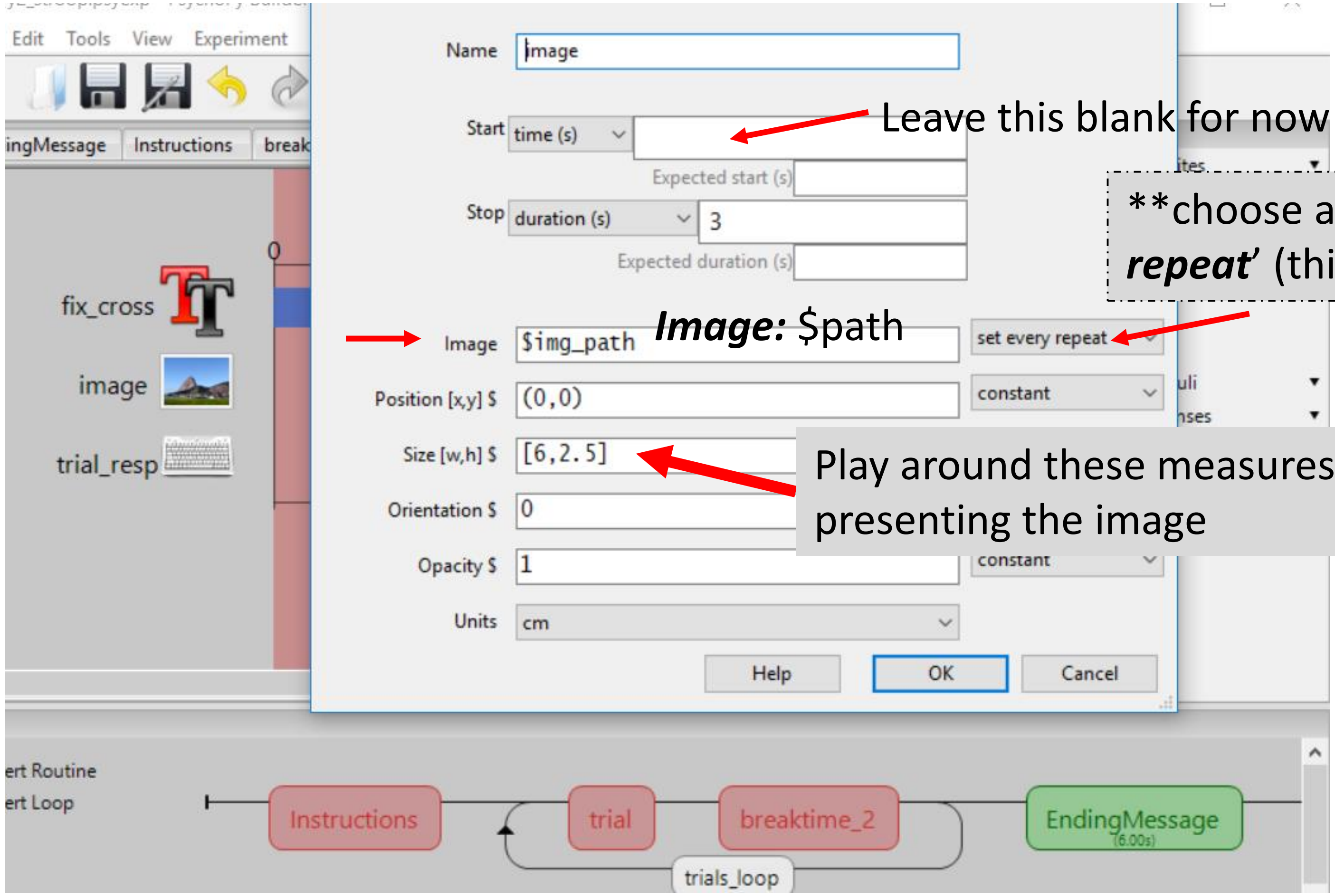
Image stimulus
Response

How this would be represented in PsychoPy:



The main trials: Target image

Update the fields in Image Properties as follows



Name: image

Start: time (s)

Leave this blank for now

Expected start (s)

Stop: duration (s) 3

Expected duration (s)

Image: $img_path — set every repeat

Image: $path

choose among the drop-down options: '*set every repeat***' (this changes trial after trial)

Position [x,y] $: (0,0) — constant

Size [w,h] $: [6,2.5]

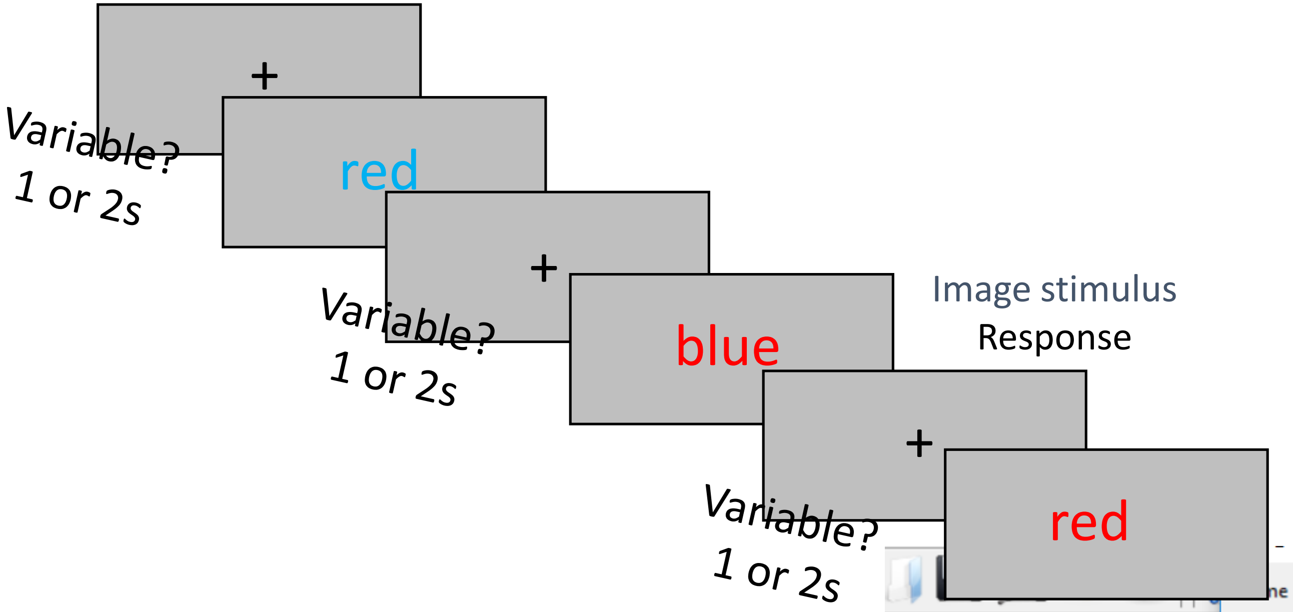
Play around these measures to achieve the best dimension for presenting the image

Orientation $: 0

Opacity $: 1 — constant

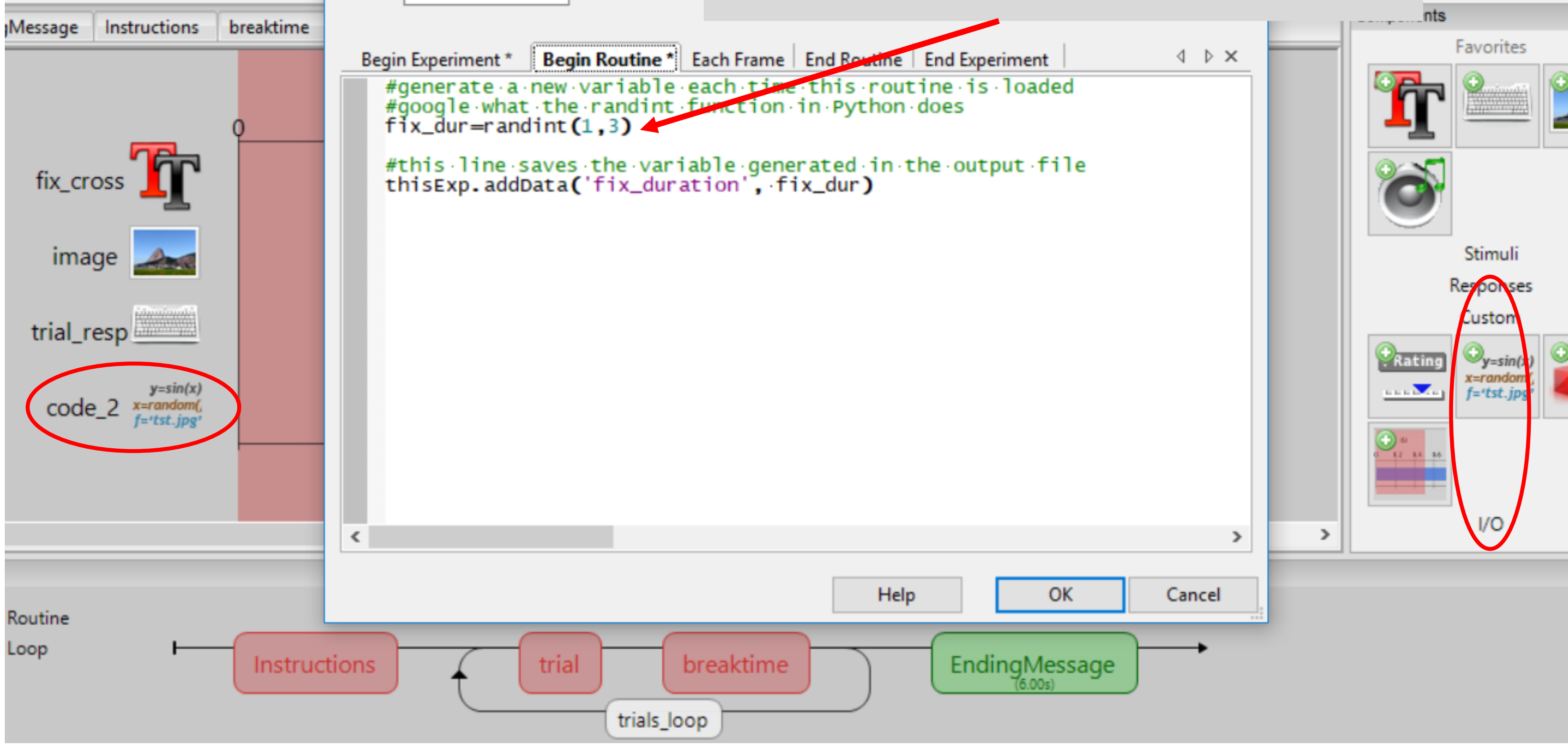
Units: cm

This time we want the fixation cross to be presented with a variable length of duration each time (i.e. different duration in a different trial)



With this line, you create a variable fix_dur which varies in value between 1 and 2 each time

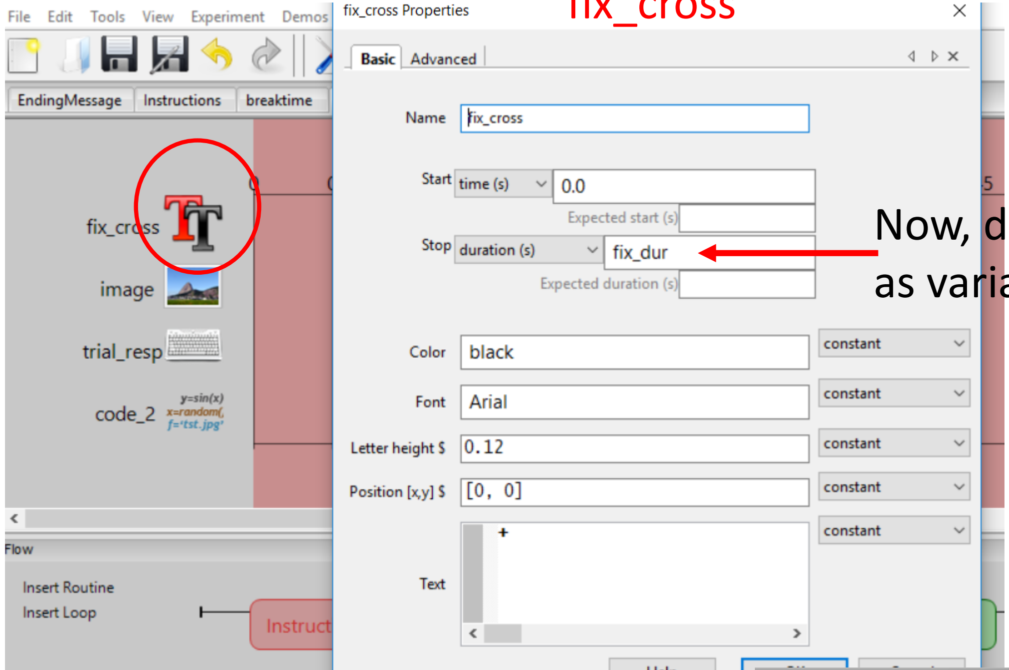
This can be achieved by writing some simple codes in a Code component



```
#generate a new variable each time this routine is loaded
#google what the randint function in Python does
fix_dur=randint(1,3)

#this line saves the variable generated in the output file
thisExp.addData('fix_duration', fix_dur)
```

fix_cross



fix_cross Properties

Name: fix_cross

Start: time (s) 0.0

Expected start (s)

Stop: duration (s) fix_dur

Expected duration (s)

Color: black (constant)

Font: Arial (constant)

Letter height $: 0.12 (constant)

Position [x,y] $: [0, 0] (constant)

Text: (constant)

Now, define the duration as variable 'fix_dur'

image

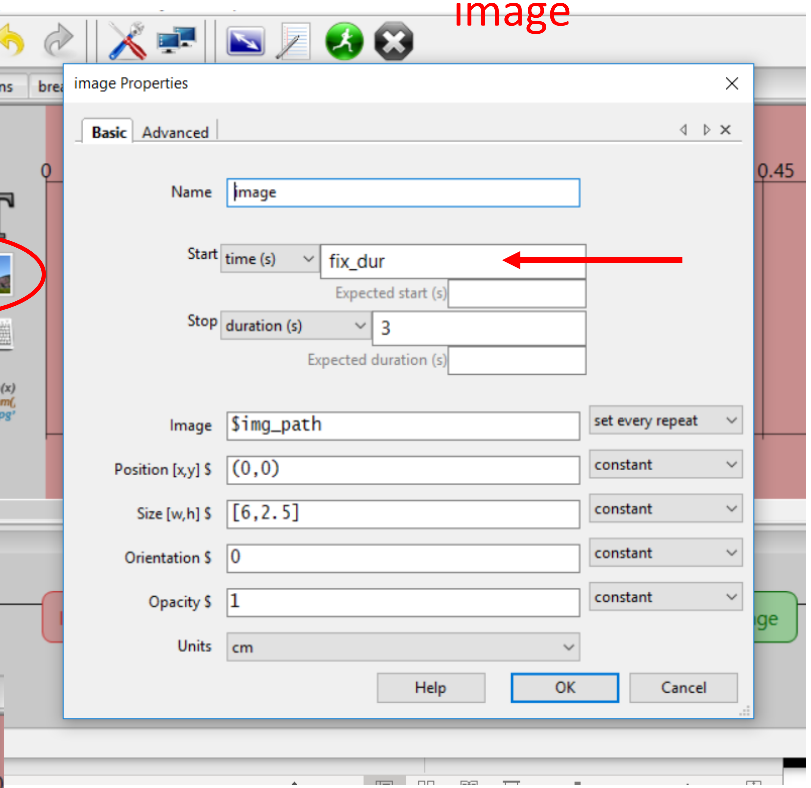


image Properties

Name: image

Start: time (s) fix_dur

Expected start (s)

Stop: duration (s) 3

Expected duration (s)

Image: $img_path (set every repeat)

Position [x,y] $: (0,0) (constant)

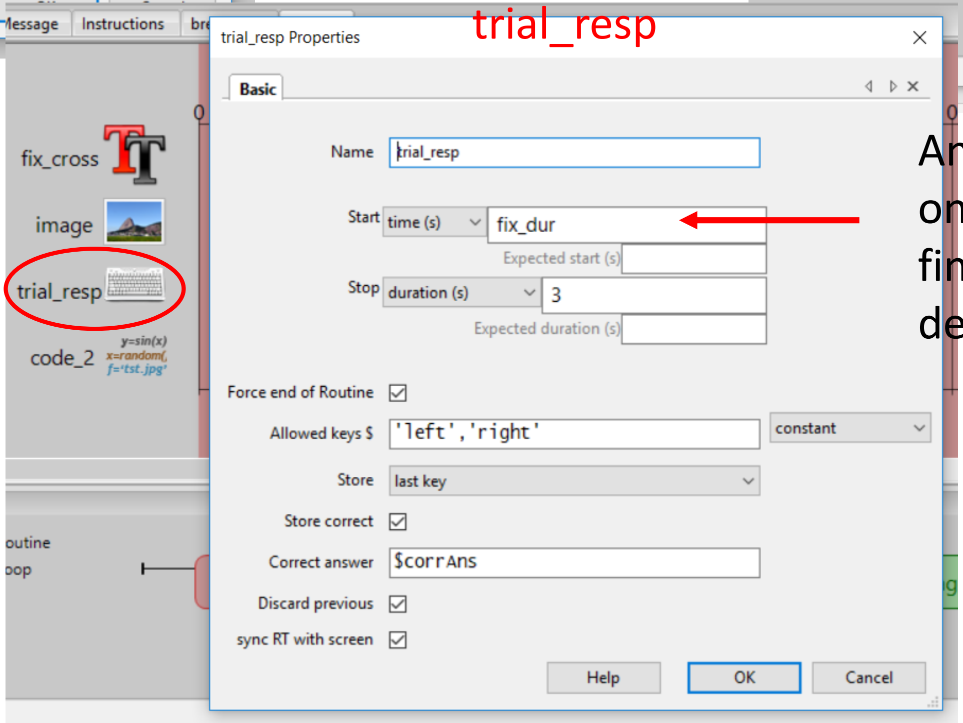
Size [w,h] $: [6,2.5] (constant)

Orientation $: 0 (constant)

Opacity $: 1 (constant)

Units: cm

trial_resp



trial_resp Properties

Name: trial_resp

Start: time (s) fix_dur

Expected start (s)

Stop: duration (s) 3

Expected duration (s)

Force end of Routine ☑

Allowed keys $: 'left','right' (constant)

Store: last key

Store correct ☑

Correct answer: $corrAns

Discard previous ☑

sync RT with screen ☑

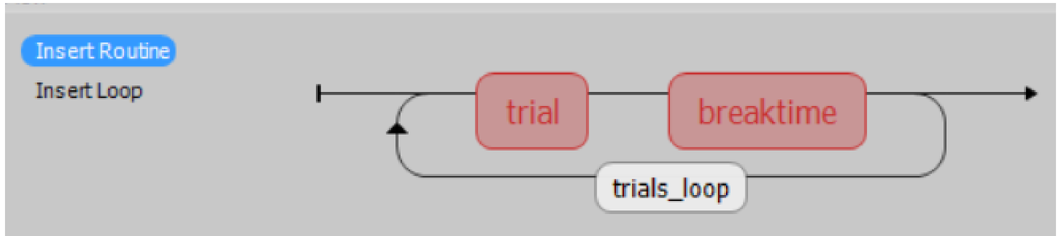
And the response key starts only after the fix_cross finishes i.e. after the time defined by variable 'fix_dur'

The main trials: Loops

Just similar to the previous example, Task 1

Between trials: give me a break!

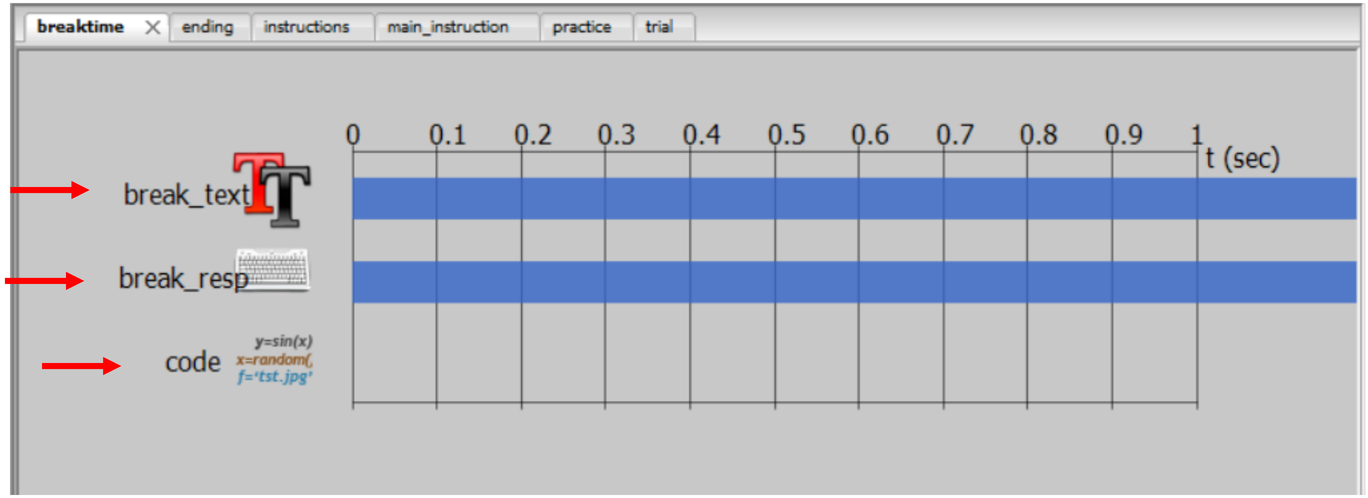
Inserting breaks between trials



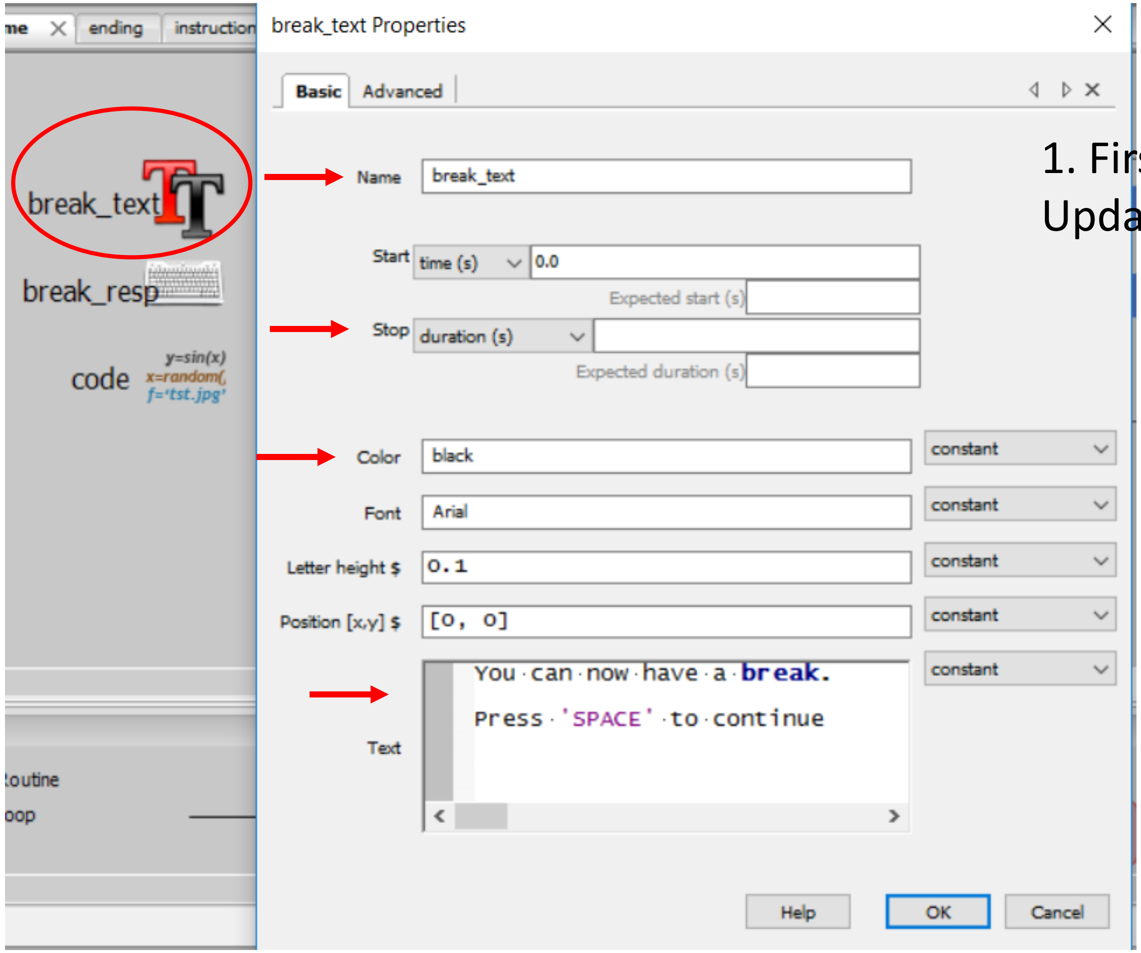
- Click Insert Routine > (new)
- Type ‘**breaktime**’
- Place it after **trial** routine and make sure it is wrapped within the loop

- Introducing breaks among trials = splitting the trials into various parts
- You will need to create a new routine and perhaps name it ‘**breaktime**’. This ‘breaktime’ routine should also be wrapped around inside the trials loop.

Inside the breaktime routine, you will need these components



Breaks



break_text Properties

Name: break_text

Start: time (s) 0.0

Expected start (s)

Stop: duration (s)

Expected duration (s)

Color: black — constant

Font: Arial — constant

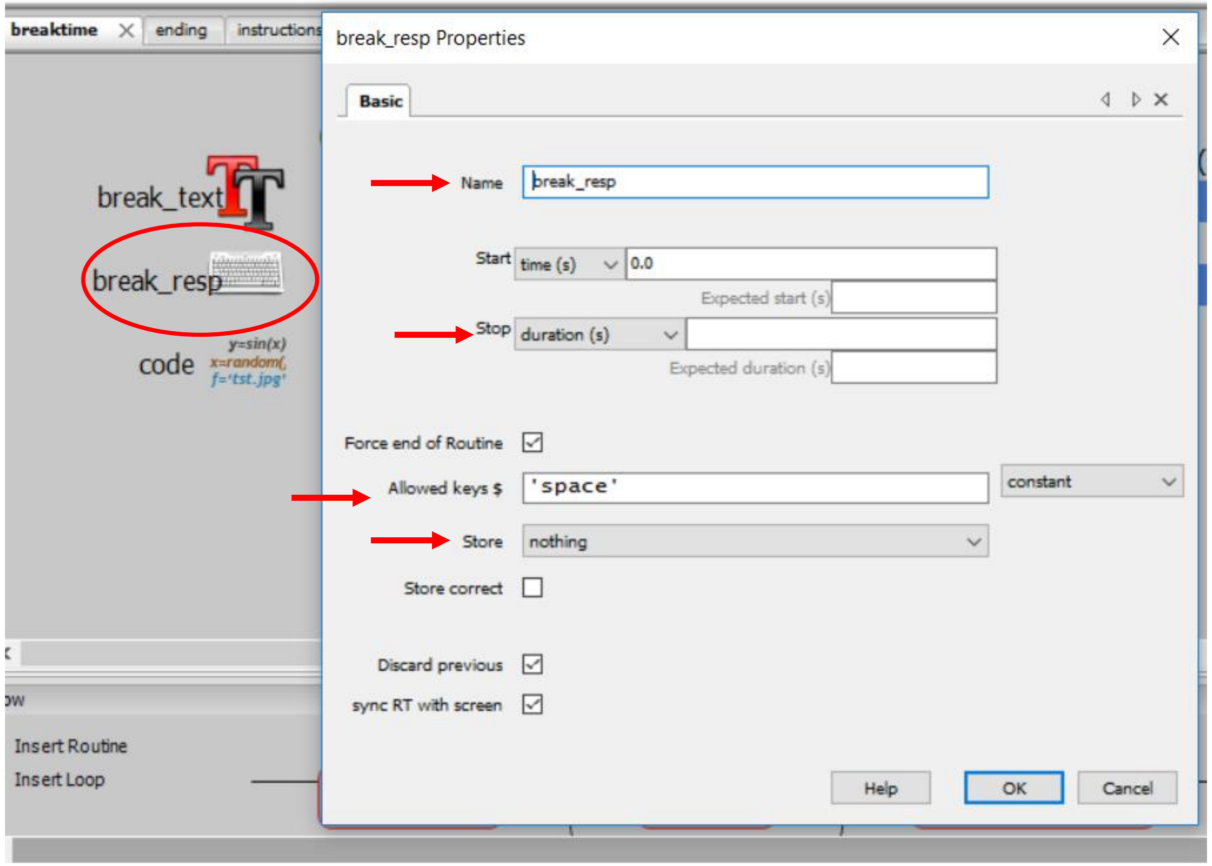
Letter height $: 0.1 — constant

Position [x,y] $: [0, 0] — constant

Text: — constant

```
You can now have a break.
Press 'SPACE' to continue
```

1. First, insert a **Text** component
Update the Properties as shown on the left



break_resp Properties

Name: break_resp

Start: time (s) 0.0

Expected start (s)

Stop: duration (s)

Expected duration (s)

Force end of Routine ☑

Allowed keys $: 'space' — constant

Store: nothing

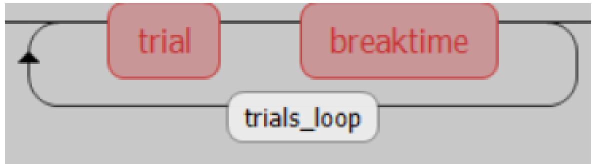
Store correct □

Discard previous ☑

sync RT with screen ☑

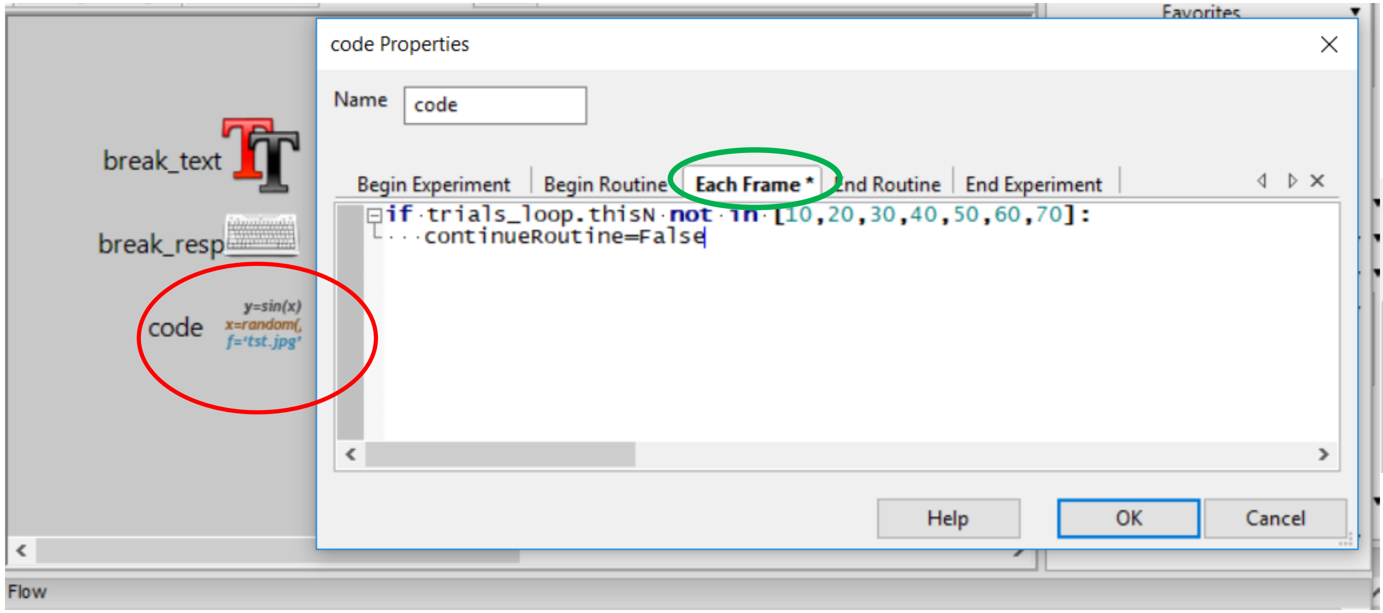
2. Then, use a Keyboard component so that the participant can respond to restart the experiment at their own pace
Update the Properties as shown on the right

Breaks



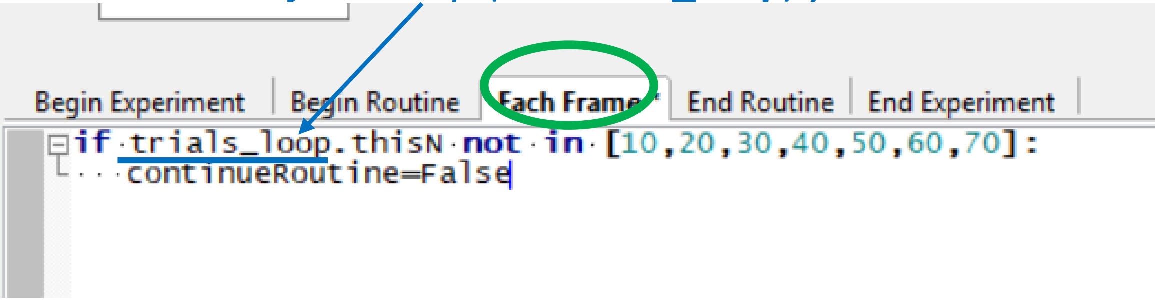
3. Finally, you need to use a ‘**code**’ component

On ‘**Each Frame**’ tab, type these codes:



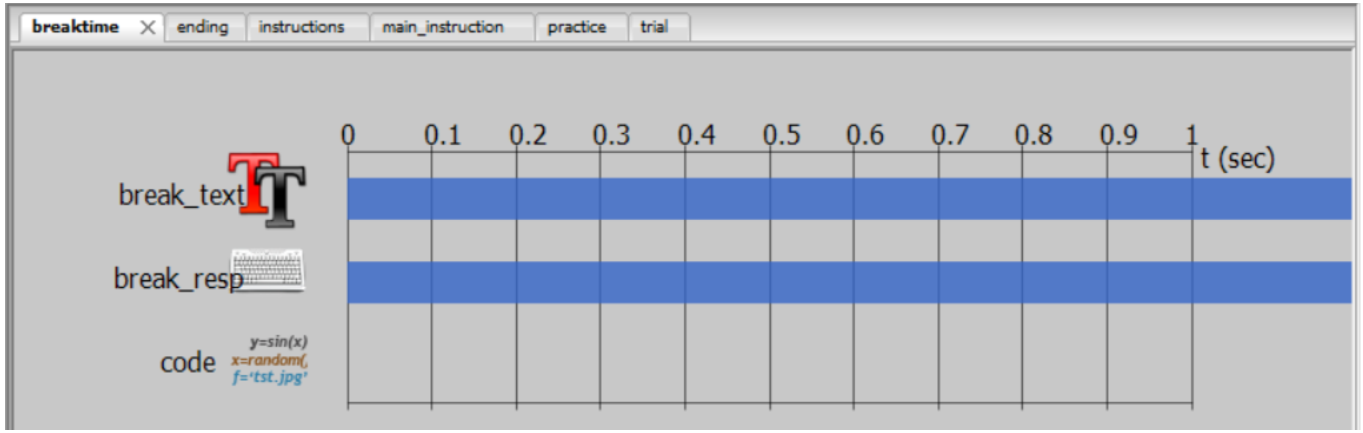
```
if trials_loop.thisN not in [10,20,30,40,50,60,70]:
    continueRoutine=False
```

Make sure this part here is exactly the same as the name of the loop (i.e. ***trials_loop****) you’ve created*



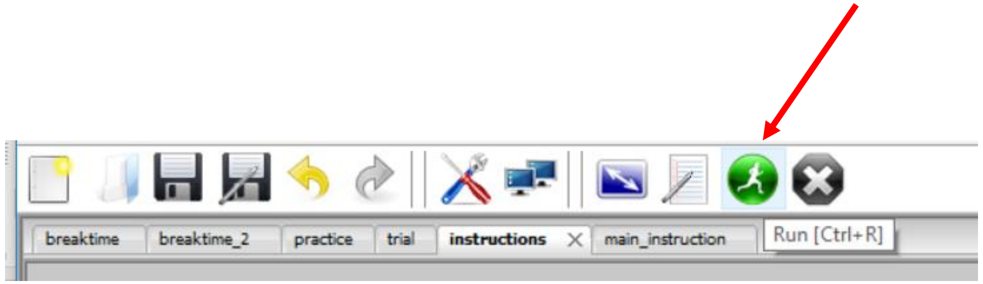
```
if trials_loop.thisN not in [10,20,30,40,50,60,70]:
    continueRoutine=False
```

We will have in total 80 trials (i.e. 8 items x 10 repetitions).
Here, we want to introduce seven breaks (after each 10 trials)



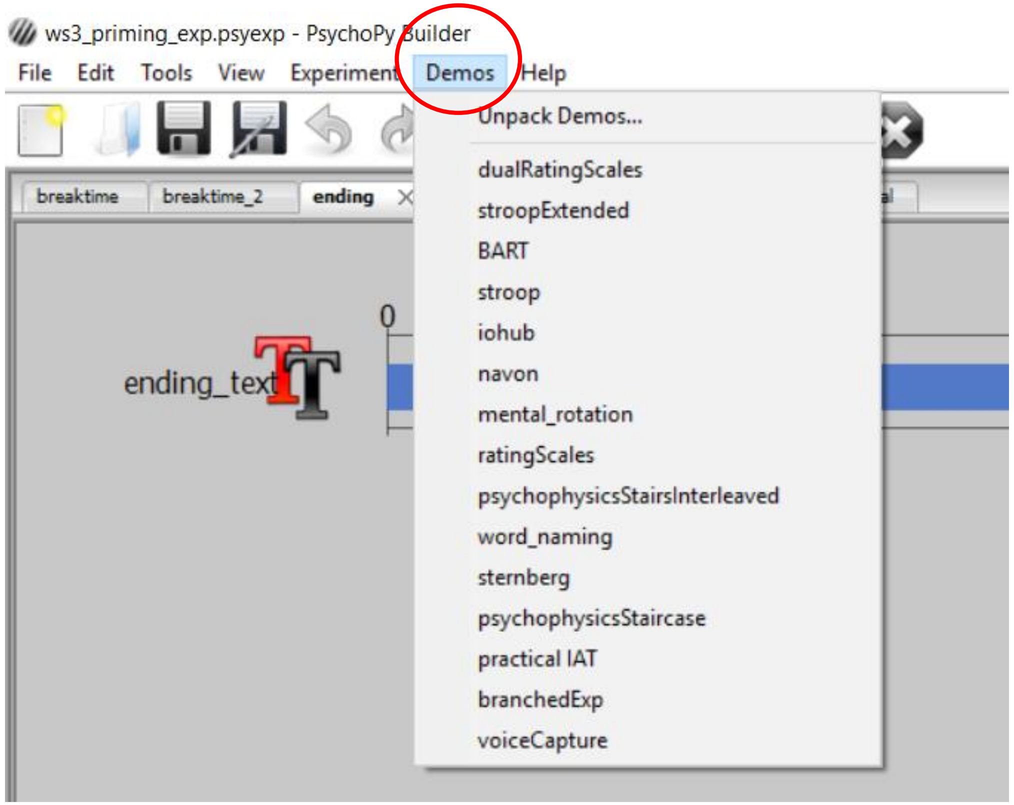
At the end, your ‘**breaktime**’ routine panel should look like this.
Also, it’s important to place the **code** component at the very bottom among all

Now test your programme!



Other resources

1.



2. Official website:

- http://www.psychopy.org/

3. Forum:

- https://discourse.psychopy.org/

4. Google!

PsychoPy Online

https://www.psychopy.org/online/fromBuilder.html

- **Sync your files with Pavlovia.org**

https://www.pavlovia.org/