Designing, Building and Explaining an Experiment

Jon Silas & Alex Jones

2025-08-04

On this page

Introduction

This is a companion guide to the module **PSY4180 Practical Methods and Topics in Cognitive Neuroscience**, and is aimed specifically at helping you with the component of the portfolio linked to **Designing, Building and Explaining an Experiment**. It assumes you are using the *PsychoPy Builder* with the standalone installation, and have limited experience with coding.

The pages that follow will guide you through setting up, testing, and troubleshooting your experiment.

> **!** Portfolio Deadline
>
> You should submit your final functioning PsychoPy experiment as a zipped folder containing all relevant documents and programs. Your explanation of your experiment should be submitted as part of your portfolio.
> **Portfolio submission deadline - 12.12.25 10am**
> See instructions on how to submit your work here

For this part of your assessment you will have to design and build a functioning behavioural experiment in a program called **PsychoPy**. The section Using PsychoPy is designed to get you using PsychoPy and help you get comfortable enough to build your own experiment. Once you've gone through the introductory tutorials, check out the Pick an Experiment section - you will need to choose one of these experiments to build in PsychoPy as part of your assessment.

As well as building your experiment, you will also need to explain your experiment. You will need to write 1500 words, following the guidelines outlined in the Explain Your Experiment section.

The resources and support in this online guide should be used in conjunction with the other forms of support available to you; in-class, via email and in one-to-ones with the module leaders. This whole guide can be downloaded as a PDF or word document by clicking the download button in the menu and selecting your preferred format.

How to use this book/website

For the most part you should go through this book in a linear fashion; page-by-page. You can skip forward one page at a time by using the arrow button at the bottom right of each page. If you need to come back to some of the information here, use this as you would any website; the search bar - on the left - should help you find anything you are looking for. This website works best on a computer not on a phone or tablet.

There are three parts to this book:

- Using PsychoPy - Each chapter in this part is an exercise with activities and mini-quizzes for you to work through, aimed at getting you using and exploring PsychoPy.
- Pick an Experiment - Each chapter is one possible experiment for you to choose to design, build and explain for your assessment; you don't need to read all of these but you should skim them to get a feel of which one you want to build then, read that one in more detail.
- Explain Your Experiment - This section outlines what is required from you when you write up an explanation for your experiment that you've built.

You might also find the icons in the navigation bar helpful:

Link to github code for this book.

Click to download this book as a .pdf, .docx, or .epub file.

Click to share to the link to the book with others.

Click to switch between light and dark mode.

Using PsychoPy

Introducion

PsychoPy® is a free cross-platform package developed by psychologists at the University of Nottingham. It allows you to build and run a wide range of psychological computer based experiments. The program is stand alone - you can download and run it on Windows, Mac OS or linux, without anything else - and it can be used almost entirely without having to learn how to code (however, it does help). PsychoPy is based on the programming language Python - whilst there are many different programming languages, all with different specific uses, issues and benefits, Python is increasingly the most commonly used language in the scientific community, world wide.

Teaching and Learning

PSY4180 as a module, and the Cognitive & Clinical Neuroscience MSc at Middlesex, doesn't explicitly teach programming and coding - we think there's enough neuroscience to get through! But, in this module we would be remiss if we didn't provide some introduction to the methods cognitive neuroscientists use to build and deploy experiments. If you are keen in continuing in a career within cognitive neuroscience, specifically, or in the research sciences more broadly, we strongly encourage you to develop your Python skills further (and coding skills more generally) - we provide some links to freely available resources you can use to do this (see, Resources section) . If you are more interested in clinical neuroscience or working in the commercial sector - we feel the skills you will learn here will certainly benefit you long term, but you may not want to peruse coding and Python any further.

Beyond the Code

Whilst we think there is value in learning a little bit of PsychoPy programming in and among itself, we also think there is broader value to be gained from understanding how cognitive neuroscience implements the test of a theory via experimental research. In line with the ethos of the module - understanding the methods of cognitive neuroscience provides you with greater insight and a more critical view on the claims being made by cognitive neuroscientists in the papers you read. This is why a functioning experiment is not the only part of this assessment,

you will also need to explain the underlying principles and theory of your experiment (see Explain your experiment section).

Underlying Method

There are many different approaches to teaching and learning how to use new software. There are dozens of online tutorials that will take you through, step-by-step how to perform specific functions in any given software. While this can be useful for learning how to execute specific operations in a specific software - we don't think this approach teaches what we want you to learn - independently and creatively navigating a software and an understanding of the underlying principles as to why you are using and manipulating the software.

What to do

In these pages, and in the materials we have created to help you, we haven't provided any click-by-click guide on how to use PsychoPy, or build your experiment. Rather, in what follows, there are some ready built simple PsychoPy programs. You need to download them and amend them - the idea is that in amending, editing and playing with them creatively, you will become accustomed to manipulating the software and develop the knowledge and skills you need to build your own experiment! Once you have completed the Getteing Started, in the coming sections you should:

- Download the *.zip folder.
- Extract the folder locally to your computer.
- Open the PsychoPy experiment (*.psypexp).
- The read-me file will open - follow the instructions to make changes to the PsychoPy program.
- Be creative - you can't do anything wrong - break and fix and change the basic program as much as you like.

Where to Get Help?

- You will be given time in class to explore PsychoPy - speak to tutors and get in person support.
- Use the Resources section for online help.
- Book an online or in-person one-to-one session with module leaders.

0 | Getting Started

- Download and install standalone PsychoPy program.
- Install PsychoPy on your machine.
- Explore some of the demos automatically installed as part of the package or work your way through the tutorials in the coming chapters.

1 | Hello World

First download the HelloWorld.zip folder and unpack it, open the folder and open HelloWorld_experiment.psyexp.

This is a very simple 'experiment' (it's not really an experiment). Run the experiment before you start editing it by pressing the play button, entering any participant number and pressing 'ok'. You should see a screen that displays the numbers 3, 2 then 1 - each number is on the screen for one second. Finally, a screen saying 'Hello World (press space to end)' will be shown - press space to finish. Now you know what the experimental file does, it's time to start playing around with it. Feel free to try whatever you want, but below are some things to guide your exploration.

What you learn

The idea is that making changes to the PsychoPy experiment will help you get a feel of opening, running and modifying PsychoPy experiments. This introductory experiment is aimed at helping you:

- Modifying basic components of loops, routines and components.
- Getting used to the idea of linking information in the excel file to the PsychoPy experiment.
- Changing the Excel file.
- Adding new components.

Things to change

1. Change the order of the numbers so they appear in random.

Hint

Click on the `trials` loop - see what options are available to you when you click on it, what happens when you change these?

2. Make the numbers appear on the screen for only 0.5 seconds.

Hint

Click on the `count` routine, within there is a component called `text` - see what options are available to you when you click on it. You can try changing any of these options and see what happens.

3. Add more numbers - count down from 10.

Hint

This is a little less obvious, so don't worry if you are having trouble. In the folder where the experiment file was located there is an Excel file called - 'Trials' - open this up, make some changes, save it and go back to the PsychoPy experiment. You may need to reload the Excel file into PsychoPy this is done from within the `Trials` loop.

4. Make the numbers stay on the screen until a response from the participant is given.

Hint

To do this you will have to modify the duration of the component `text` within the `count` routine, and you will also need to add a new response component to the same routine.

5. Change the numbers so that they display traffic light images - red, amber and green.

Hint

This is a bit more involved and really there are a few ways you can do this. You should remove the `text` component from the `count` routine. Then you can add an image, or a polygon. If you add an image you will need to make (or find online) three seperate files - red, amber and green - the name of the image should be linked to the Excel file. Alternatively, you can link the colour of a polygon to the Excel file. Play about - find the most efficient way for you to make it work.

Test yourself

Question 1 | On the last screen of the experiment you have to press space for the experiment to exit. What column in the data generated for any run of the experiment gives you the time taken for a person to press space (the reaction time)?_____

Answer

key_resp.rt

Explain

All data generated for an experiment is saved the data folder automatically generated when you run a PsychoPy Experiment. Within that folder - each participant has a spreadsheet, text document and PSYDAT file. Opening the spreadsheet is the easiest way to get a quick overview of the data collected. Within that spreadsheet, components from the experiment have their own column, and names are appended to indicate what they reflect. In this case the

component is called `key_resp` and is appended with `.rt` to reflect that data in that column is reaction time.

Question 2 | What value should you enter if you want a stimuli to remain on the screen indefinately?

- (A) 0
- (B) 9999
- (C) 'blank'
- (D) 0000

Explain

In the `duration (s)` property of any component you can specify how long that component should appear on the screen for. If you want, as we often do in an experiment, to wait for a response you can specify that the component is kept on the screen indefinitely by leaving this value blank.

Question 3 | If you wanted to accept multiple different keys in a keyboard response what would you use to deliminate keys?

- (A) comma
- (B) semi-colon
- (C) colon
- (D) dash

Explain

Sometimes you will need a participant to choose from more than one response, in this case make sure the 'Allowed keys' property of a response component has the keys separated by a comma.

2 | Simon Experiment

First, download the SimonEffect.zip folder and unpack it, then open the folder and open the 'SimonEffect_experiment.psyexp' PsychoPy file.

We are already starting to get a little more complicated with our PsychoPy experiments. Run the experiment before you start editing it by pressing the play button, entering any participant number, and pressing 'ok'. In this experiment you are first given some instructions and then asked to respond to 'green' and 'red' circle stimuli. The Simon effect is an effect of reaction times - people are faster at responding with their left hand, on the left, when a stimulus is on the left – and this is true for the right side too. This is compared to when you make a response using your hand on the opposite side of a stimulus. This little experiment will actually generate some data!

As it stands, the experiment will first, give you some instructions then run you through two 'congruent' trials - where the side of the stimuli and the side of the response match. After that it will end.

What you learn

The idea is that making changes to the PsychoPy experiment will help you get a feel of changing parameters within components, routines and loops. This introductory experiment is aimed at helping you to get better at:

- Modifying components of loops, routines and components.
- Modifying Excel file to result in changes in the experiment.
- Modifying routine's length.
- Adding correct answers.

Things to change

Let's see if you can make a few changes and perhaps even get some simple data to look at.

1. Can you add two trials in the 'incongruent' condition - so that a red circle appears on the right in one trial and a green circle on the left in another trial?

Hint

You will need to modify the Excel file, save it, and upload it again to the `trials` loop. You want two more trials that are in effect the opposite of what you see for the congruent trials - label them as `incongruent`.

2. Once you've added more conditions, make sure they appear in a random order and add more repetitions of trials; so that the number of trials per condition repeat - set it for 5 repetitions per condition.

Hint

You can change all of this by manipulating the properties of the `trials` loop.

Answer

Within the properties of the 'trials' loop, change `loopType` to `random` and change `nReps` to `5`.

3. At the moment the stimuli stay on the screen until a response. To increase errors and speed up responses can you set a time limit so that the participant only has 2 seconds to respond?

Hint

You can change all of this by manipulating the properties of the components within the `trial` routine.

Answer

Click on the `key_resp` component within the `trial` routine.Change `duation (s)` to `2`.

4. Can you manipulate the `key_resp` component in the `trial` routine so that it stores the correct answer?

Hint

As well as modifying the properties in `key_resp` Data tab, you will also have to make some changes to the spreadsheet so that you can identify what response is correct in any given condition.

Answer

In the excel spreadsheet add a new column called `correct` include in there the correct key response for each trial. In the Data tab of the `key_resp` component, click `Store correct` and enter `$` followed by the name of your column in the `Correct answer` field.

Test yourself

Question 1 | If you wanted to allow the letters p and q as a response (instead of a and k) what would you type in the 'Allowed keys' section of the response properties?

- (A) 'q', 'p'
- (B) 'q,p'
- (C) 'q-p'
- (D) Q,P

Question 2 | In the `trial` component, the `left_stim` and `right_stim` components get their colour from the spreadsheet. What symbol tell PsychoPy that the colour is listed in the spreadsheet?

- (A)

- (B) £
- (C) &
- (D) $

Question 3 | If you want any property of a component to change on a trial-by-trial basis which of the following should you select?

- (A) Constant
- (B) Set every frame
- (C) Set every repeat

Question 4 | What is the name of the column in your data that tells you if a response was correct or not?_____

Answer

key_resp.corr

Explain

Once you've correctly allowed PsychoPy to know what a correct reponse is (see 'what to change' point 4)

3 | Simon Feedback

First, download the Feedback.zip folder and unpack it, then open the folder and open the 'SimonEffect_feedback.psyexp' PsychoPy file.

In this experiment we are again increasing complexity but this time in an incremental fashion - we'll be adding to the Simon Effect from the last experiment. If you run the experiment from scratch, you should see that it doesn't work properly; it will always tell you that your answer is 'incorrect'. You will need to fix the experiment as you make changes below. The idea is that after each trial you should be told whether or not you responded correctly. In this experiment we are starting to introduce some very basic code - if you're not familiar with code **don't panic**. We'll introduce some simple snippets of code, and you won't need to be an expert to make your own experiment do what you want.

What you learn

Things are getting a little more complicated but we hope playing around here will teach you the following:

- How to insert code components.
- How to modify basic code and even write some of your own.
- How to implement basic feedback.
- Introduction into troubleshooting.

Things to change

As we said the experiment is again looking at the Simon effect - there are 4 trials 2 congruent and 2 incongruent. But if you run the experiment you don't get the appropriate feedback! Let's make some changes so that it works and then experiment a bit further:

1. Try to get the feedback working!

Hint 1

You will need to tell PsychoPy what a correct answer is! You can do this in the `trial` routine, in the `key_resp` component.

Hint 2

The correct response can't be static - rather you will have to reference to a column in the excel spreadsheet.

Answer

In the `trial` routine in the `key_resp` component, go to the `Data` tab and click `Store correct`. In the 'Correct answer' box type `$answer`. Then open the Excel file called `trials` add a new column with the heading `answer` in that column add the correct response expected for each trial. Save and close the excel document - reload it in the `trials` loop and run the experiment again.

2. Change the colour and text of the feedback to anything you want.

Hint 1

Locate the code component in the `Feedback` routine - open it up and see what you can understand.

Hint 2

In the code just try changing `fb_text =` and the `fb_col=` what colors does it recognise?

3. Modify the experiment so that you have a time limit to respond in - can you then modify the feedback so it says 'Too slow!' if no response was given? (this can be tricky).

Hint 1

This is a hard one - don't give up yet; try googling... seriously, learning how to find the solution to a coding problem yourself by using the internet is a useful skill and can be really satisfying.

Hint 2

Ok, ok - I'll help. Once you've set a timelimit on all components on the `trial` routine you'll need to change some code. Instead of `if key_resp.corr:` try `if key_resp.corr == 1`. Then think about trying an `elseif` code line.

Hint 3

Still a bit tricky, I get it. You can use `and` in `if` functions, you can also use not but that's expressed like `!=` which effectively means 'is not'.

Answer

Don't give up! Try the PsychoPy forums first!

Real answer

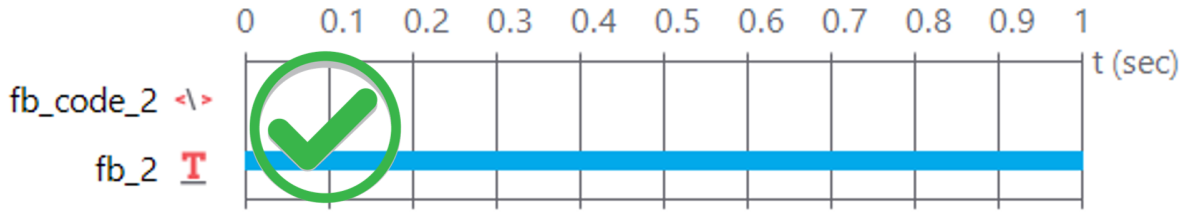
It's a tricky one and if you haven't coded before don't feel to disheartened if you didn't get it. You'll need the following code in the `Feedback` routine in the `fb_code_2` component in the

`Begin Routine` tab, you can copy and paste from the code below to replace what's currently there:

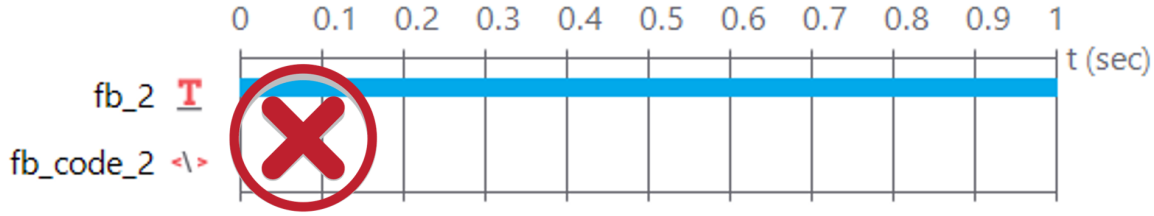
```
if key_resp.corr == 1:
  fb_text = "Correct!"
  fb_col ="green"
elif key_resp.corr == 0 and key_resp.keys != None:
  fb_text = "Wrong!"
  fb_col ="red"
elif key_resp.corr == 0 and key_resp.keys == None:
  fb_text = "Too slow!"
  fb_col ="black"
```

Caution

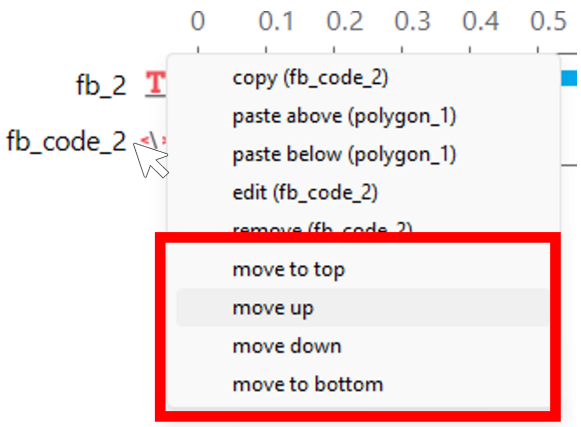
When you are adding code that will define the content of text stimuli, as in this experiment, it's important that the code come before text component like this:



Not after, like this:



You can move any component up or down by right clicking on the component and selecting any of the 'move' options, like this:



Test yourself

Question 1 | In a code chunk, what is the name of the coding language used to programme PsychoPy?_____

Question 2 | What charachter is used to 'comment out' text?

- (A) &
- (B)

- (C) –
- (D) %

Explain

When coding it's good practice to annotate your code so other people find it easier to understand and you will remember what you did better. Leave comments in code chunks within PsychoPy by using '#'

Question 3 | What's the name of the property that you need to change to increase the number of repetitions that a loop will go through?_____

If you need help - speak to instructors in class, check the online guide or contact a member of staff via email.

4 | Branching

First, download the Branching.zip folder and unpack it, then open the folder and open the 'Branching.psyexp' PsychoPy file.

You've been introduced to a small amount of coding and hopefully you can see how powerful it is - based on responses, and their relation to the components and properties of the experiment, you can modify what happens. Another useful aspect of this is 'branching' - based on a response from the participant you may want to re-run a practice set of trials or exit an experiment early if the participant doesn't consent (for example).

Although this experiment is simpler than the previous one - more is missing and there's more for you to do to make it function as it is supposed to - you may have to type some code! Remember **don't panic** we'll help you through it. At the moment, you see some instructions, and then irrespective of what key you press you then see a picture of a cat and then a dog. The idea is that you should be able to choose what you see next.

You'll need to get the experiment working appropriately, check the hints below for support and if you get stuck ask an instructor in class or send them an email.

What you'll learn

We want to increase your comfort with code without having to learn all the specific details. Going through this guide aims to teach you to:

- Become a little more comfortable with code components in PsychoPy.
- Learn how to 'branch' experiments based on responses.
- Increase your comfort with creating and using an Excel document to reference content in the experiment.

Things to change

1. Make the appropriate changes to the code and the routines so that when you press C you will see an image of a cat, when you press D you see a picture of a dog and when you press R the instructions repeat.

Hint 1

Think about modifying the `nReps` of the loops, make them 0 if you want to skip a routine and 1 if you want to include it in the procedure.

Hint 2

Once you've made sure that the `nReps` in cat and dog can be set by code you need to write an if statement that runs in the `start` routine.

Hint 3

Your if statement should be in a code component in the `start` routine. It should ensure that if a certain response is made - stored in `key_resp.keys =` - then whatever you called the prperty in the nReps field should change. You should also add this in the tab of the `code` component called `End routine`

Answer

Ok first, you need to name nReps for the `cat_loop` and the `dog_loop`, let's call them `CatReps` and `DogReps` respectively, then add a `Code` component to the `start` routine, in the `End routine` tab add the following code:

```
if key_resp.keys == "d":
        CatReps = 0
        DogReps = 1

if key_resp.keys == "c":
        CatReps = 1
        DogReps = 0
```

2. Change the images, draw from a list in an excel document and do so randomly.

Hint 1

The first part of this hopefully shouildn't be too tricky, get some new images from the internet, change the image path in the routine to link to the excel file that will have the name of the image file. If you're getting stuck make sure you've instructed the image to set every repeat, insure you've included the file type extension correctly.

Hint 2

Once you've managed to read an image from a list and do so randomly, you'll probably notice that the loop won't exit till it's gone through all images. To exit after one loop you'll need to add some code to both the `cat_pic` routine and the `dog_pic` routine. Before you check the solution below try seeing if you can figure it out by using online resources.

Answer

Ok, first find at least 3 images on the internet of dogs and three of cats - save these to the `4.Branching` folder. In the same folder, create a new Excel document for dogs and one for cats, you can call the Excel documents `Dogs` and `Cats` respectively. Each Excel document should have one column that should be called `DogImages` and `CatImages` in that column list you image file names and include the file type for example `catpic1.jpeg`. In the PsychoPy experiment load up each Excel document to the appropriate loop. in the `dog_pic` routine and the `cat_pic` routine select the image and change the `Image` property to `$DogImages` and `$CatImages` and set it to `set every repeat`. Finally, in order to exit the loop after one image is shown, add a code component to the routine. In the code component add some code to the `End Routine` tab this should be `dog_loop.finished=True` for the dog loop and, `cat_loop.finished=True` for the cat loop.

3. Add responses to the images can you add new routines to jump to? Maybe add a goodbye message before the experiment ends?

Hint

This is the same logic to the code you've already written - just make some minor amendments so it can jump to a final screen - you're welcome to be creative here.

4. Can you set up something useful - could you modify the experiment to have a 'press y to consent and continue' or 'press n to refuse consent and exit?

Hint

Again - use the same logic that you used to get the experiment working - simpply change the letters in the argument and the content of the routines.

Test yourself

Question 1 | What is the name of the routine you need to put the code in for the branching to work?_____

Question 2 | What is the name of the tab the code should be in for your branching to work?_____

Explain answer

The code needs to be able to effect what happens straight after input is made so it goes in the `start` routine where a response is inputted. However, it makes changes once the response has been given and so must go in the `end routine` tab.

5 | Blocks

First, download the Blocks.zip folder and unpack it, then open the folder and open the 'Blocks.psyexp' PsychoPy file.

Up till now we have had a strong focus on 'trials' - that is, one instance of one condition that a participant typically generates one response for (complicated experiments may vary slightly). A 'block' of trials is usually the repetition of multiple trials. Within a block each trial may vary based on its condition and the only reason you would 'block' trials would be to give participants a break. Sometimes, when cognition might be affected by adjacent trials, it might be better to separate conditions across blocks - this is typically referred to as a 'blocked' condition or paradigm.

In this experiment we will have a simple trial with two conditions - we will then try to set up the experiment so that we vary those conditions randomly, within a block, and take breaks. Then we will try to set up the experiment so that there is one condition in each block.

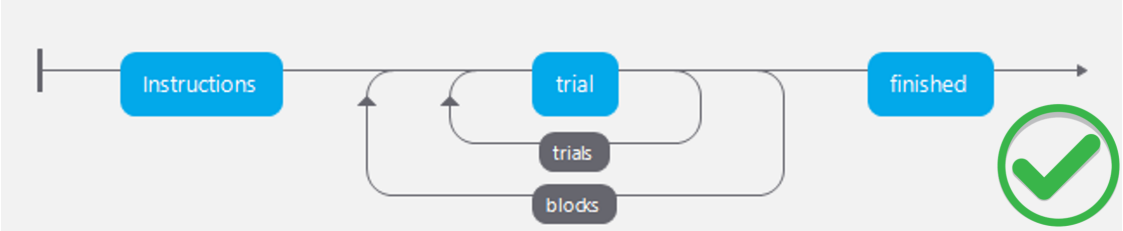
This task is a perceptual discrimination task - whilst fixating your gaze on the fixation cross participants are asked to judge if lines on either side of the cross are the same length or different. Currently the experiment runs through instructions, two trials (one in the 'same' condition, the other in the 'different' condition) a break screen and then ends.

What you'll learn

- How to implement blocks.
- How to implement block designs and blocks for breaks.
- How to change the break message based on blocks.
- Become more comfortable with loops and code.

> Caution
>
> When attempting to make blocks a common mistake is to make separate trials with separate routines in them like the image below:

Instead use loops within loops to cause blocks - this means you can maintain trial consistency. See the below image:



Things to change

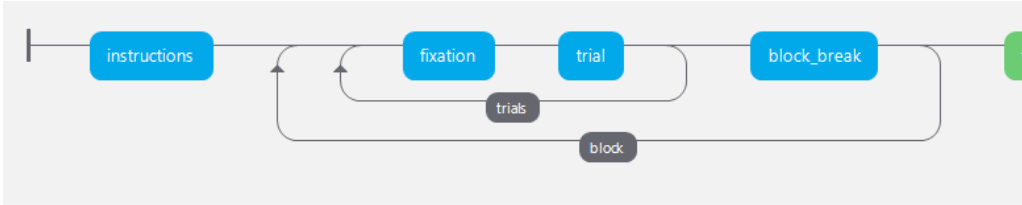
1. Add another block, so that you are presented with one trial from each condition then a break before another block of the same.

Hint

Add another loop around the `trials` loop that ends after the `block_break` routine - call this loop `block`.

Answer

Your flow should look like this:



2. Modify your experiment so that you present two trials from the 'same' condition in one block, and two trials from the 'different' condition in another.

Hint

You will need to modify the Excel documents and create new ones for this to work. You'll need three files in total.

Answer

You will need to create three Excel documents, one for the `same` trials and one for the `different` trials. In a third document - call it `blocks.xlsx`, provide the names of the new

Excel document in a column called conditions (remember to use `.xlsx` in the name). In the `trials` loop refernce the file names by entering `$conditions`. Finally, in the `block` loop load the file `blocks`.

3. For either block, format present a different message in the break for each block, for example 'that was the first block, you have one more to go' for the end of the first block but something different for the last block.

Hint 1

You will need to add some code for this. Add a code component to the `block_break` routine. Check back to the feedback task and look at how you defined a message using code

Hint 2

You'll need to define a `block counter` this should equal to zero when the experiment starts but add one as each block passes.

Answer

Add a code component to the `block_break` routine. In the `Begin Experiment` tab and add the following code.

```
# define a block counter that will start at zero.
block_count = 0
```

Still in the the `block_break` routine, select the `text_3` routine - delete the words in the `Text` field and add a variable, we'll call it `$block_msg`. Also, change the properties of the `Text` field so that it shows `set every repeat`.

Then in the `Begin Routine` tab add the following.

```
#add one to the block counter at the beginning of the routine.
block_count = block_count + 1;

#add an if function that writes a messages based on the block.
if block_count == 1:
  block_msg = 'That is one block complete, you have one more to go. \n \n Press space when yo

elif block_count == 2:
  block_msg = 'Great - the experiment is over. \n \n Press space to end the experiment.'
```

Your block messaging should now work!

Test yourself

Question 1 | When coding a message to be displayed to participants, as in this experiment, what do you type to force a new line in the text?

- (A) \n
- (B)
- (C) <
- (D) p

Question 2 | What's the name of the component where reaction times to same/different jusdgments are stored? _____

Answer

key_resp_2

Question 3 | Would the storage of RTs, accuracy or any participant data change as a function of which block is running?

- (A) Yes
- (B) No

Explain

This is the advantage of this method of creating blocks - because routines and components are identical - it's easier to organise and analyse the data.

- Troubleshooting -

Introduction

This troubleshooting guide is designed to help you get started with troubleshooting common issues in PsychoPy, especially when using the **Builder view**. Most students won't need to write code from scratch — and that's okay. The Builder allows you to design experiments visually, but sometimes things don't work as expected.

This guide highlights common issues that can arise while using Builder (and occasionally small bits of code), along with practical strategies to fix them.

> **i** What this troubleshooting guide is (and isn't)
>
> This is **not a complete manual**, but a practical starting point for common problems. Most of the time, if something goes wrong, you're not the first person it's happened to — and it's usually fixable!

> **!** Why bother with troubleshooting?
>
> Troubleshooting can feel frustrating — but it's also where real learning happens. Every problem you solve helps you understand how your experiment works and makes you more confident next time.

Components Not Appearing

- **Common issues:**
  - Text, images, or stimuli don't appear when the experiment runs.
- **Checklist:**
  - Check that your component's **Start and Stop times** are correct (e.g., start at 0s or at frame 0).

- Make sure the component is **set to draw** (not disabled or hidden).
- Are you calling `win.flip()` with a custom code snippet? If so, that may override Builder's drawing logic — check with your instructor before modifying window display.
- Ensure **order in the Routine** is correct — components lower down in the list might overwrite earlier ones if positioned in the same location.

Loop or Routine Not Running

- **Common issues:**
  - You expect something to repeat but it doesn't — or the experiment skips sections entirely.
- **Checklist:**
  - Check if a **loop** is actually created around the routine in your Flow panel.
  - Ensure your **conditions file** is properly attached and formatted (.csv or .xlsx).
  - Look out for **typos in column headers** in your spreadsheet — these must exactly match the field names in Builder (e.g., `$image`, `$word`).

File Not Found

- **Common issues:**
  - Images, sounds, or videos don't load, or you get a "file not found" error.
- **Checklist:**
  - Are your files in the **same folder** as your experiment file, or in a **relative path** (like `images/cat.png`)?
  - Avoid using **absolute file paths** like `C:/Users/YourName/Desktop/image.png` — these often break on other machines.
  - File names are **case-sensitive** — check for capitalisation mismatches.

Unexpected Behaviour or Errors

- **Common issues:**
  - The experiment crashes when you run it.
  - Unexpected results (e.g., blank screens, skipped stimuli).
- **Checklist:**
  - Use the **Runner panel (bottom section)** to read error messages.
  - If you added **a Code component**, check whether the error message refers to something in that custom code.
  - Try running a simplified version of the experiment with fewer components — this can help narrow down what's causing the issue.

> Don't ignore error messages
>
> Even if error messages look technical, they usually point to the problem. Read the first few lines and look for filenames, line numbers, or words like "NameError" or "IndexError". You can often spot the issue by searching online for the exact message.

Timing or Overlap Issues

- **Common issues:**
  - A stimulus disappears too quickly, or doesn't show up at all.
- **Checklist:**
  - Make sure the **duration settings** for your component are correct.
  - Check whether **another component starts too soon** and overlaps or replaces the one you want.
  - If your stimuli depend on keyboard or mouse input, ensure the response is configured to allow screen updates as expected.

PsychoPy Crashes or Freezes

- **Common issues:**
  - The PsychoPy app freezes or refuses to launch.
- **Checklist:**
  - Try **closing and reopening PsychoPy**.
  - If problems persist, **restart your computer**.
  - Still broken? **Re-download and reinstall** the latest version of the standalone PsychoPy app from the official site: https://psychopy.org/download.html

---

Version Compatibility

- **Common issues:**
  - Something worked last week but no longer does, or behaviour is different on another machine.
- **Checklist:**
  - Check what version of PsychoPy you're using (shown on the app's launch screen).
  - Stick with the recommended version provided by your course team unless told otherwise.
  - If something suddenly stops working, try reinstalling the same version you originally used.

Debugging Tools (Builder-Friendly)

- Use the **"Print to Runner"** box in the Code component to output messages to the Runner.
- You can temporarily add a **simple text component** to show variable values onscreen for testing.
- Use **the "Runner" panel** to read errors and trace problems step-by-step.

Still stuck?

Before asking for help…

Try this first: 1. Double-check your Builder settings. 2. Read the error in the Runner panel. 3. Try removing or simplifying a component to isolate the problem. 4. Search online or check the PsychoPy Forum.

Final thought

You don't need to be a programmer to build experiments — but becoming confident in solving small problems is a huge step forward. Don't panic. Stay curious. You've got this.

Pick an Experiment

Choose one of the following experiments to build in PsychoPy. Along with the PsychoPy experiment forming part of your submission, you should also provide a 1500-word summary. The summary should explain what design decisions you have made and why you made them – specifically what research question are you addressing with your design choices. Also, briefly review the available evidence that suggests that your task is underpinned by a specific neural region or network (see details on explaining your experiment here).

- Erikson Flanker Task
- Spatial Cuing Task
- Levels of Processing
- N-back Task
- Mental Rotation

It's important that you follow the requirements laid out for the experiment that you pick. Make sure you make changes to the experimental design to make it your own. These changes should:

1. Have a clear rationale based on previous research
2. Be within subject manipulations

Erikson Flanker Task

This task was initially developed by Eriksen & Eriksen (1974) and is aimed at quantifying response inhibition. Reaction time and accuracy are said to be dependent upon a participant's ability to inhibit irrelevant information.
You will need to build a behavioural flanker task using PsychoPy that conform to the requirements outlined below and allow you to measure reaction and accuracy for each condition.

Design

This is a simple 'flat' design with one factor – Congruency, and three levels:

- Congruent – where the central stimuli are the same as the distractor stimuli.
- Incongruent – where the central stimuli are different from the distractor stimuli.
- Neutral - where the central stimuli are different from the distractor stimuli but are not expected to interfere with processing.

* You should include more factors to change the design to test your hypothesis; you must explain your choices in the summary of the experiment.

Experimental requirements

Your experiment must include the following:

- A brief information and consent sheet.
- At least 5 practice trials where feedback is given to the participants based on their performance.
- Feedback in practice trials should be – correct/incorrect/no-response, too slow.
- at least 10 trials in each condition. * Trials, across the three conditions, should be presented in a random order

Trial requirements:

- The interval between trials (also known as the inter-trial-interval: ITI) should be filled with a fixation cross – the length of time the fixation trial is on the screen should vary such that it's offset is unpredictable.
- The target stimuli should be presented centrally.
- The distractor stimuli should be presented peripherally.

Use your imagination

There are several aspects of the experiment that you can change freely; this means that you will be able to test your own specific hypothesis. Here is a list of somethings you can change but it isn't comprehensive; if it isn't specified above – you can change it.

- Nature of the stimuli: words, pictures, letters, symbols etc.
- Location of the stimuli: where peripheral distractors appear.
- Nature of the task: what aspect of the target stimuli are participants asked to respond to.

Spatial Cuing Task

Allocation of attention to a location in space has classically been explored using a Posner et al. (1980) cueing paradigm. This task can, with small modifications, be used to explore either endogenous (voluntary/top-down) or exogenous (automatic/bottom-up) attentional mechanisms. The key dependant variable for this experiment is reaction time. Using different types of stimuli allows the research to explore questions about attentional resources for specific categories.

You will need to build a behavioural spatial cueing task using PsychoPy that conform to the requirements outlined below and allow you to measure reaction and accuracy for each condition.

Design

Outlined here is a simple 'flat' design with one factor – Validity, that has two levels:

- Valid – where the target location is predicted by the cueing stimuli.
- Invalid – where the target location is predicted by the cueing stimuli.

You should include more factors to change the design to test your hypothesis; you must explain your choices in the summary of the experiment.

Experimental requirements

Your experiment must include the following:

- An information and consent sheet.
- At least 5 practice trials where feedback is given to the participants based on their performance.
- Feedback in practice trials should be – correct/incorrect/no-response, too slow.
- 20 trials in each condition.
- Trials, across the three conditions, should be presented in a random order

Trial requirements

- The interval between trials (also known as the inter-trial-interval: ITI) should be filled with a fixation cross – the length of time the fixation trial is on the screen should vary such that it's offset is unpredictable.
- A fixation cross should remain on the screen throughout the trial so participants can focus their gaze.
- The interval between cue and target should vary such that the onset of the target is unpredictable.

Use your imagination

There are several aspects of the experiment that you can change freely; this means that you will be able to test your own specific hypothesis. Here is a list of somethings you can change but it isn't comprehensive; if it isn't specified above – you can change it.

- Nature of the stimuli -> words, pictures, letters, symbols etc.
- Testing endogenous, exogenous attention or both.
- Using central or peripheral cues.

Levels of Processing

The Levels of Processing Framework (Craik & Tulving, 1975) suggests that words are better encoded and hence remembered when they are processed more 'deeply'. For the most part, 'deep' processing is associated with semantic processing whereas 'shallow' processing is associated with perceptual encoding of the stimuli. The key manipulation in this experiment is the instructions given to participants when they are looking at the to-be-remembered words (i.e., during encoding).

You will need to build a memory task using PsychoPy that conform to the requirements outlined below and allow you to measure correctly remembered items, correctly rejected items, incorrectly rejected items and falsely remembered items (from which you can calculate d' see here for information).

Design

This again is a simple 'flat' design with two experimental conditions:

- Shallow encoding – the task given to the participant requires them to attend to perceptual features of the stimuli.
- Deep encoding – the task given to the participant requires them to attend to conceptual features of the stimuli.

Experimental requirements

Your experiment must include the following:

- An information and consent sheet.
- 20 trials in each condition.
- Trials, across the conditions, should be presented in a random order.
- The encoding and recognition block should be separated by a distractor task.
- In the recognition block double the number of stimuli should be shown to the participants compared to the encoding block – 50% previously seen (old) and 50% unseen (new) stimuli.
- Stimuli sets should be counterbalanced across participants.

Trial requirements

- The interval between trials (also known as the inter-trial-interval: ITI) should be filled with a fixation cross – the length of time the fixation trial is on the screen should vary such that it's offset is unpredictable.

Use your imagination

There are several aspects of the experiment that you can change freely; this means that you will be able to test your own specific hypothesis. Here is a list of somethings you can change but it isn't comprehensive; if it isn't specified above – you can change it.

- Nature of the stimuli -> words, pictures, letters, symbols etc.
- Nature of the task -> how will you define what shallow and deep processing is?
- Nature of retrieval -> you can choose to look at implicit or explicit memory based on how you measure retrieval.
- Attention check -> How do you know your participants were paying attention, what steps can you take to identify this.

N-back task

The n-back task is a classical assessment of working memory capacity. Initially developed by Kirchner (1958), the task presents a continuous stream of stimuli, and the participant must identify when the same stimulus (or stimulus belonging to the same category) is presented n steps earlier in the sequence.

You will need to build a behavioural n-back task using PsychoPy that conforms to the requirements outlined below and allow you to measure reaction and accuracy for each condition.

Design

This again is a simple 'flat' design the number of experimental conditions you have can vary but you should have at least two:

- Low n – there should be a low number of steps between the identical stimuli.
- High n – there should be a high number of steps between the identical stimuli.

Experimental requirements

Your experiment must include the following

- An information and consent sheet.
- 30 trials in each condition.
- A practice block where feedback is given to the participants based on their performance.
- Feedback in practice trials should be – correct/incorrect/no-response, too slow.

Trial requirements

- The interval between trials (also known as the inter-trial-interval: ITI) should be filled with a fixation cross – the length of time the fixation trial is on the screen should vary such that it's offset is unpredictable.

Use your imagination

There are several aspects of the experiment that you can change freely and systematically; this means that you will be able to test your own specific hypothesis. Here is a list of somethings you can change but it isn't comprehensive; if it isn't specified above – you can change it.

- Nature of the stimuli -> words, pictures, letters, symbols etc.
- What participants are trying to identify is 'the same' –> colour, meaning, location, look.
- Number of blocks and n's -> you can have just low and high, or more intervals.

Mental rotation

This measures an individual's capacity to represent an image and rotate it in their mind. The nature of the stimuli and the complexity of shapes is often found to cause a change in accuracy in reaction times of mental rotation. Typically, an image is presented on the screen and the participant has to chose which of four possible targets is the same as the cue image but rotated in space. Usually, individual differences are reported in mental rotation, but you should focus your experiment on within-subject manipulations. Although the first reported experimental protocol of mental rotation is decades old (Vandenberg & Kuse, 1978), the task is still used today.

You will need to build a behavioural mental rotation task using PsychoPy that conform to the requirements outlined below and allow you to measure reaction and accuracy for each condition.

Design

This again is a simple 'flat' design the number of experimental conditions you have can vary but you should have at least one factor and 4 conditions:

- Orientation -> the correct answer should be rotated at four different orientations: 0°, 90°, 180°, 270°.

Experimental requirements

Your experiment must include the following

- An information and consent sheet.
- 30 trials in each condition.
- A practice block where feedback is given to the participants based on their performance.
- Feedback in practice trials should be – correct/incorrect.

Trial requirements

- The interval between trials (also known as the inter-trial-interval: ITI) should be filled with a fixation cross – the length of time the fixation trial is on the screen should vary such that it's offset is unpredictable.
- The correct response should be one of four possible; three should be distractors.
- The cue, target and distractors should all be presented on the screen at the same time.

Use your imagination

There are several aspects of the experiment that you can change freely; this means that you will be able to test your own specific hypothesis. Here is a list of some things you can change but it isn't comprehensive; if it isn't specified above – you can change it.

- Nature of the stimuli -> pictures, letters, symbols, abstract shapes, etc.
- Similarity of target and distractors.

Explain Your Experiment

Submit final version with complete portfolio by - 12.12.25 10am

1500 word summary explanation of your experiment

Instructions

Based on your chosen experiment that you have built in PsychoPy, provide a 1500-word summary by answering the questions below. This summary should explain what design decisions you have made and why you made them – specifically what research question are you addressing with your design choices. Make sure you briefly review the available evidence that suggests that your task is underpinned by a specific neural region or network. Use the following subsections to write your summary:

Aims

(~150 words)

What are the overall research aims of your experiment? What do you hope to test and find out about cognition and/or the brain from your results.

Background & rationale

(~700 words)

This section should include references and provide a succinct summary of key findings in the field that have used similar experimental designs. It is important that you include here an overview of research that has linked brain structure and/or function to experiments similar to yours. Importantly, you should also provide some key theoretical rationale for your experiment.

Predictions

(~150 words)

Your predictions don't have to be comprehensive but they should include critical hypotheses - what hypothesis is the key test of your theory?

Methods & design rationale

(~500 words)

Here you should outline the details of your experimental design and procedure. It is important that you include a rationale for any decisions you made about the design structure and procedural set up. If, for example, you added another condition, or changed the nature of the stimuli systematically - explain here why you did this - what are these novel manipulations meant to test?

How to Submit

> ! Portfolio Deadline
>
> **You should submit your portfolio and your experiment no later than 12.12.25 10am**

Written component

Your write up of your experiment should form part of your portfolio that should be submitted as one document on MyLearning here (log in to MyLearning required)

Experiment

As part of your portfolio you should also submit your experiment built in PsychoPy. To do so you should zip your folder (help on how to zip a file) and upload your compressed folder to MyLearning here.

Your folder should include:
- A PsychoPy experiment file
- A spreadsheet for trials and blocks.
- A readme file with a brief outline of the experiment.
- Any stimuli needed for your experiment to run.

Your experiment should be standalone - the folder should contain everything needed to run the experiment (except an installation of PsychoPy)

Resources

There are multiple sources of information that you should use to help you along your journey to designing building building and explaining an experiment in PsychoPy. Here are some we've listed but make sure you look for your own too.

PsychoPy Specific

- PsychoPy Experiment Recipe Book
- Susan Benear's - Coding Outreach Group Summer Workshop
- PsychoPy Forum – use it, create an account, ask questions and help others.
- Excellent introductory information from the University of Nottingham
- Peirce, J., Hirst, R., & MacAskill, M. (2022). Building experiments in PsychoPy. Sage.
  - Available electronically from MDX library - sign in required.
- Slides Jon & Alex's lecture on PsychoPy.

Python Specific

- Download and install Python
- Introduction to Python online
- learnpython.org
- Lutz, M. (2013). Learning python: Powerful object-oriented programming. " O'Reilly Media, Inc.".
  - Available electronically from MDX library - sign in required.

Broader Programing Resources

- R and RStudio
  - R is a programming language commonly used in data science and statistics.
  - R studio is a desktop environment for using coding programs like R and Python.
- PsyTeachR - is an amazing set of resources made freely available by the Psychology department at Glasgow University - highly recommended!

References

Craik, F. I., & Tulving, E. (1975). Depth of processing and the retention of words in episodic memory. *Journal of Experimental Psychology: General*, *104*(3), 268.

Eriksen, B. A., & Eriksen, C. W. (1974). Effects of noise letters upon the identification of a target letter in a nonsearch task. *Perception & Psychophysics*, *16*(1), 143–149.

Kirchner, W. K. (1958). Age differences in short-term retention of rapidly changing information. *Journal of Experimental Psychology*, *55*(4), 352.

Posner, M. I., Snyder, C. R., & Davidson, B. J. (1980). Attention and the detection of signals. *Journal of Experimental Psychology: General*, *109*(2), 160.

Vandenberg, S. G., & Kuse, A. R. (1978). Mental rotations, a group test of three-dimensional spatial visualization. *Perceptual and Motor Skills*, *47*(2), 599–604.