



# PsychoPy Intermediate

Frank Gasking and John Allen

# A quick recap/overview

Quick recap in PsychoPy of:


- ▶ Routines
- ▶ Timelines
- ▶ Basic components
- ▶ Paths (relative/full) - came up last week

**(DEMO)**

# Result files revisited!

How do we add extra information to our data output?

## Start up parameters

- ▶ By default - prompted at the start for **Participant** and **Session ID**
- ▶ To add your own, click the experiment settings icon 

<http://www.psychopy.org/builder/settings.html>

## Add to Excel data file

- ▶ Which we did in the first week
- ▶ Add a new column in your data file - that is it!

## Through embedded Python code

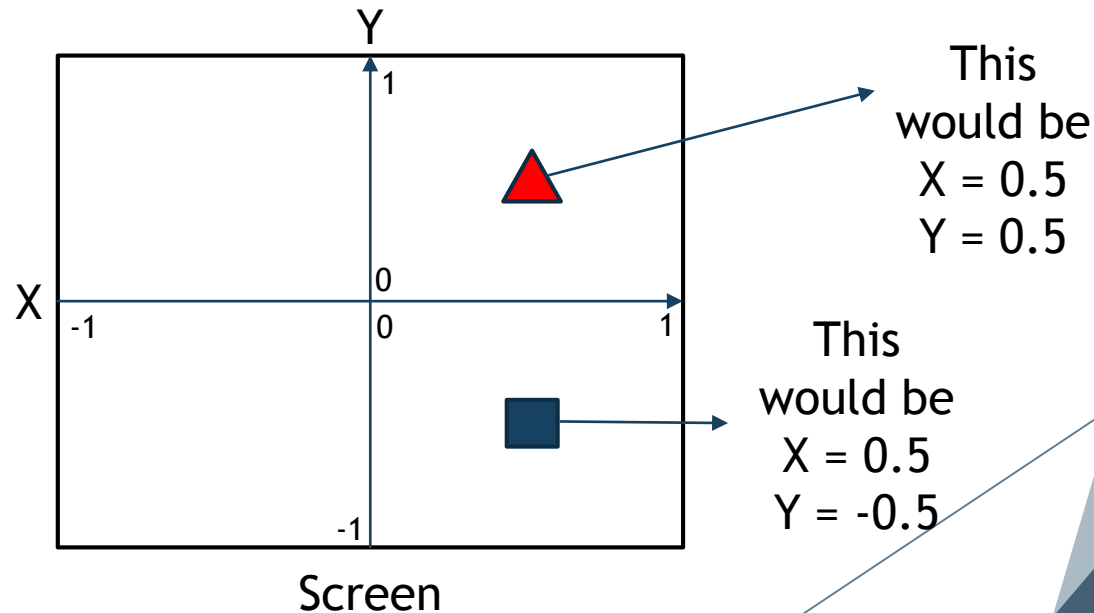
- ▶ Possible to run complex calculations and store into our data file
- ▶ Outside the scope of this session, but it is possible!

# Monitor settings

- PsychoPy has concept of a 'Monitor'
  - Allows you to:
    - store information about multiple monitors
    - keep track of multiple calibrations for the same monitor.
- TOOLS | MONITOR CENTER
- Means you can:
  - specify the size and location of stimuli in units that are independent of your particular setup.e.g.
    - pixels
    - cm of screen
    - degrees of visual angle
  - easy to port programs to different setups as PsychoPy calculate appropriate pixel size for you

# Positioning components

- ▶ Positioning of components within a routine
- ▶ Stacking order is important!
  - Refers to order they are drawn
  - Components rendered at the same time
  - Those at same position will overlap!
- ▶ Screen divided by a coordinate system
  - Other options available too which you can read about here:
  - <http://www.psychopy.org/general/units.html>



# Exercise 1:

## Modify ordering and positions

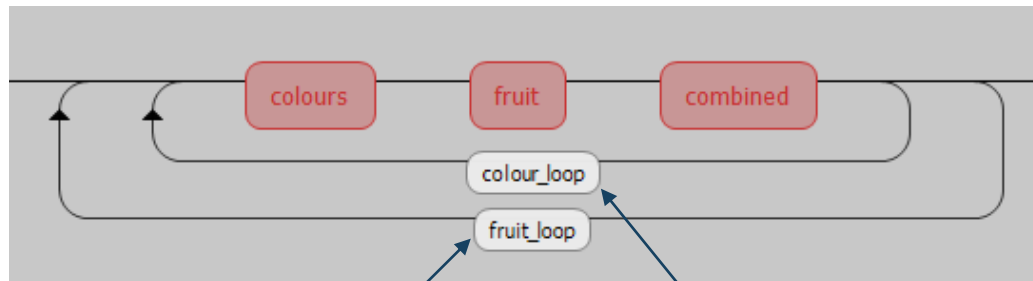
We have given you a sample screen with a bunch of random objects

- ▶ Open and run `ex1-positions/experiment.psyexp`
- ▶ Press space to finish the demo at any time
- ▶ Now try and modify the layering order of the objects
  - ▶ Right click the item in the flow and use the move options.
- ▶ Also try playing with positioning using what we have shown you on the previous slide.
- ▶ Try adding some more polygon or image objects if you like.
  - Get familiar to the stacking and positioning in PsychoPy

We'll take **5 minutes** for this task.

# Nested loops and lists

- ▶ Repeated stimuli with different conditions can cause repetition (DEMO - combined-data.xlsx)
- ▶ Using nested loops/lists in PsychoPy, we can reduce this
- ▶ A nested loop in PsychoPy:



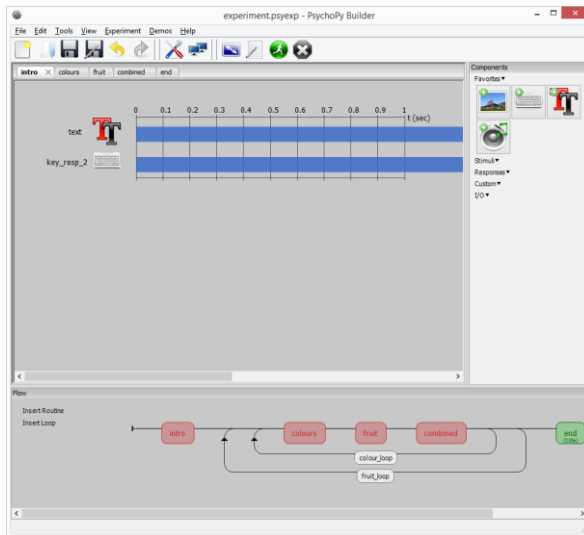
Outer loop  
(runs first)

Inner loop  
(runs second, repeats)

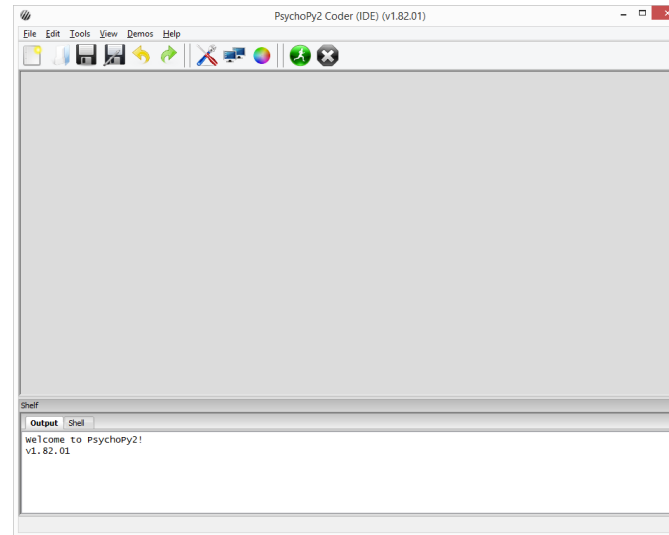
Lets look at the example in action... (DEMO)

# Using Coder instead of Builder

- ▶ Writing pure Python code using PsychoPy library
- ▶ Too advanced to cover in detail on this course
- ▶ Unlikely you will ever need to use it
- ▶ Sometimes we do if an experiment is outside scope of builder



Builder view



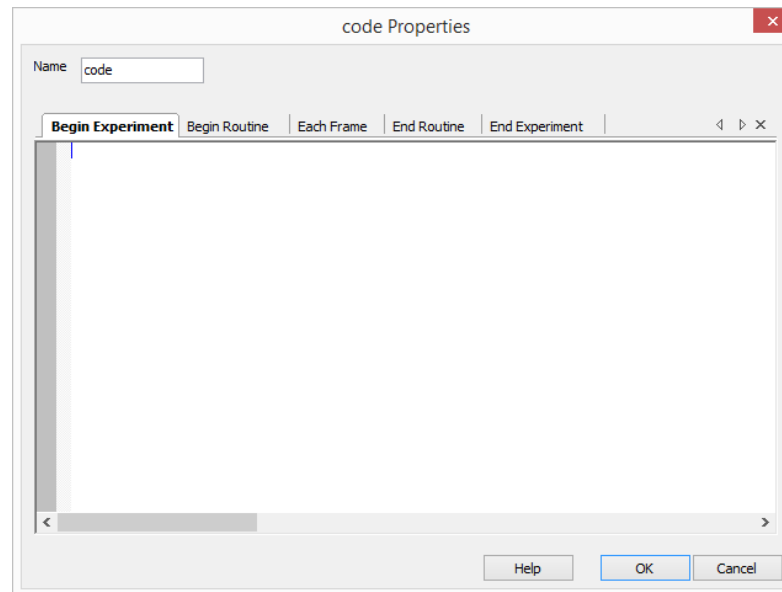
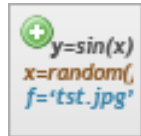
Coder view

Here is a quick demo of something which can be achieved with Coder... (DEMO)



# Using code in builder

- ▶ The card game could have been done in builder
  - Structure/nested loops required would have been complex
- ▶ However, even using builder can occasionally require us to have to dip into code
  - Often to produce something builder cannot handle.
  - A limit to what a builder interface can achieve.
- ▶ We can drop in snippets of code using **code components**

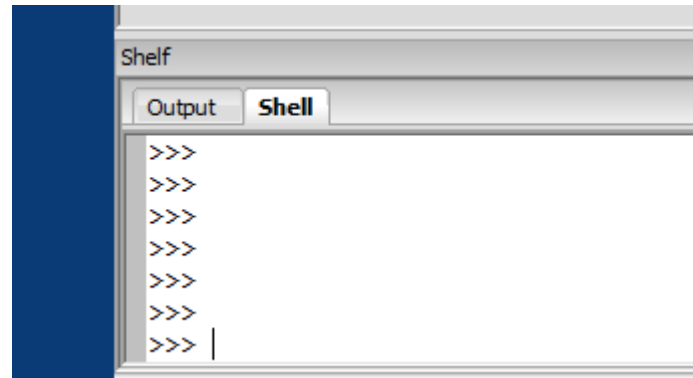


# Tip: Testing simple Python code

- ▶ You can also test bits of Python code via the “Shell” tab
- ▶ In the coder view...

## Demo

```
firstname = "John"  
firstname.upper()  
firstname.lower()
```



# Exercise 2:

## Using snippets of code in Builder

Open up `ex2-code-blocks/experiment.psyexp` and spend **5 minutes** by doing the following:

- ▶ Run it first and see what happens

**Where are the numbers coming from for the calculation?**

- ▶ Look at the “`trial`” routine
  - Click onto the code block called “`calculation`”
  - Then the “`Begin routine`” tab
  - See how the calculation is being done
  - What is `int()` doing, and why do we use it?
- ▶ How do we then print that calculation in the Text component?
  - Open up the Text component - what is `str()` doing?
  - Remember why “text” is “**set every repeat**”?

**A number of new concepts we have just thrown at you there  
- without warning!**

**Lets cover them briefly and some more!....**

# A whistle stop tour of simple programming concepts...

Will give a bit of context for the use of `int()` and `str()`

# Variables and data types

- ▶ Containers for data (i.e. "answer" in the last exercise)
- ▶ Or otherwise known as variables
  - `age = 10`
  - `age = 15`
- ▶ Can use in calculations, concatenations (joining), loops and more...
  - `age = age + 5`
  - `first_name = "Frank"`
  - `Full_name = first_name + " Gasking"`
- ▶ Should give meaningful names for reference
- ▶ No spaces in variables (ever!) - use "\_" or "-" if you must have some spaces.

Variables store different data types - which we'll cover now...

# Simple data types

## ► Integer

- `age = 10`

## ► Float

- `pi = 3.14`

## ► Boolean

- `bilingual = True`
- `or bilingual = 1` (in some languages is valid)

## ► Char

- `initial = "M" ... number = "1"`

## ► String

- `first_name = "John"`
- `phone_number = "+44 01227 888999"`

# Operators

You will probably remember these from Maths...

## ► Math

- `+`, `-`, `*`, `/`, `mod`
- `total = number1 * number2`

## ► Boolean

- `AND`, `OR`, `NOT`

## ► Comparison

- `=`, `==` (equivalence)
- `===` (exact equivalence)
- `<>`, `!=` (not equivalence)
- `<`, `>`, `<=`, `>=`

## ► BIDMAS/BODMAS

- `((1+1)+(2+2)) * 2`

# Conditional statements

- ▶ Making decisions
  - E.g. `if ... then ... else`
- ▶ Uses data types, variables and operators

```
if age < 18:  
    print("Sorry, you're too young to drink!")  
elif age < 100:  
    print("Have a drink on us!")  
else:  
    print("Wow, have two drinks on us!")
```

**Question** - How do we make decisions in PsychoPy?


**Answer** - Using code blocks!



# Exercise 3:

## Using conditionals and variables

Open up file `ex3-conditionals/experiment.psyexp` and spend the next **5-10 minutes** by doing the following:

- ▶ Run it and see what happens
- ▶ How is the correct **gender based** Excel data file read?
  - Look at the “`setup`” code block under routine “`start`”
  - Go to the “Begin Experiment” tab
  - Read the comments (in green) and see what is happening.
  - How does it work out which data file to use?
  - How are we then using that data file reference?
- ▶ Where is “`gender`” set and how to we create new prompts?
  - Click onto this icon  and take a look
- ▶ Allow “undisclosed” as another gender option.
  - Via entering “`u`” or “`undisclosed`” for gender
  - Create a copy of one of the excel sheets and create a mixture of male and female names to list.
  - Have a go at updating the “`setup`” code block to include this option to set the correct data file. We’ll help if you get stuck.

# Adding functionality via code blocks

You can add functionality to PsychoPy in a number of ways:

- ▶ Not just by creating our own snippets of code like we did previously
- ▶ But also by bringing in other people's ready-made code
- ▶ Thanks to PsychoPy being built on Python, we can:
  - Import from libraries/3<sup>rd</sup> party code
  - Use custom **functions** (either our own or written)

# What are functions?

- ▶ A “container” of reusable code that does “something”
- ▶ Defined under a given (sensible) name
- ▶ Benefits:
  - ▶ Write once, use many times
  - ▶ Reduces redundancy and repetition
  - ▶ Declutters your program
  - ▶ Only runs when called
- ▶ Variables can be passed to a function
  - ▶ Called “parameters”
- ▶ Functions always give back something
  - ▶ A value of some kind, depending what the purpose of the function is.
- ▶ `int()` and `str()` are Python functions!
  - ▶ They take a value
  - ▶ They give back either an integer or string value

# Exercise 4:

## Adding functions to your experiment

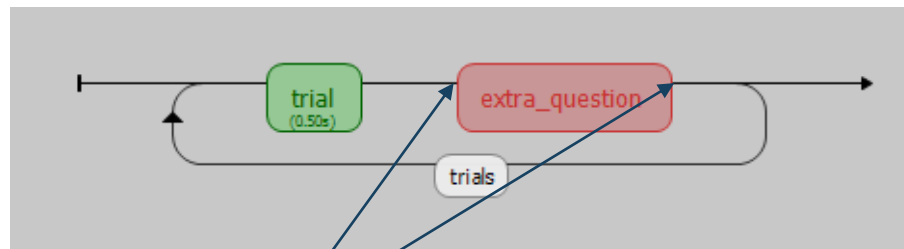
Open `ex4-import-code/experiment.psyexp` and spend the next **5-10 minutes** by doing the following:

- ▶ Run experiment to check what happens
- ▶ Select the "intro" routine and then click on the "code" component, and then "begin experiment" tab
  - Read the comments in green.
  - What is single line of code doing?
  - **Hint** - look for something similarly named inside the experiment folder.
- ▶ Select "my\_function\_test" routine, then click on the "text\_3" component and observe what is in the Text box
  - `getRandomMonthID()` will fetch a number from between 0 and 11
  - `getMonth()` takes a number from between 0 and 11 and gives back a text month
  - "\$" is used to tell PsychoPy that we are about to run a small snippet of Python code
- ▶ Now create a Excel data source which will load in a month numbers 1-12
  - Feed it into "trials\_2" loop as a data source
  - Replace `getRandomMonthID()` with the header from the Excel file

# Branching in PsychoPy

- ▶ Sometimes we may want to skip a particular routine
- ▶ Maybe the user should only be prompted for a particular question if they answered previously with a particular answer?
- ▶ Conditionals in a Code block come in handy here!

Example:



So how do we bypass this routine based on an answer under “trial”?

Time for a quick demo...

`demos/3-Branching-example/experiment.psyexp`

# Creating user breaks

- ▶ It is good to ensure the user gets a break.
- ▶ Especially if you have something like 200 trials!
- ▶ Method is very similar to “Branching” like before

One last demo... `demos/4-Break-demo/experiment.psyexp`

# Error messages / debugging

Fixing errors can be tricky, especially when its with an unfamiliar system, or someone else's experiment!

Here are some tips if you encounter any issues:

- ▶ Read the error message carefully
  - Look for references to line numbers which indicate the line at fault
- ▶ Be wary of case sensitivity: Python is very particular
  - e.g. "if" (not "If")
- ▶ Check the experiment .log file under /data for any extra hints
- ▶ Google the exact error message as a starting point
- ▶ Check the online help/manuals
- ▶ Check forums/communities for similar problems
- ▶ If you get really stuck - pop to the support hatch  
(Offices - Frank - A1.6, John - A1.2)

# General best practices

- ▶ Label your routines/loops and components clearly
- ▶ Good naming convention
  - "Age = 10" is meaningful in comparison to "a = 10"
  - "Question\_display" is meaningful as a routine name than "routine\_1"
  - Be careful of using RESERVED WORDS
    - Some of you encountered this problem last week (with "image")
    - These are Python language terms, which cannot be re-used
    - i.e. def, list, dict
    - PsychoPy should warn you if a name is already in use.
- ▶ Make regular backups!!
  - We cannot stress this enough!
  - Don't just rely on USB sticks (can be unreliable)
- ▶ Check your data output and ensure its solid
  - Make sure you are getting the data you need to later analyse
  - Worse thing that could happen is to check after your tests and find you haven't got what you need.
  - Basically do some quick analysis early to make sure!



# Where can I get extra help?

## ► PsychoPy:

- PsychoPy built-in HELP
  - Remember, there are separate help sections for Builder and Coder!
- PsychoPy website - <http://www.psychopy.org/index.html>
- Google groups forum - <https://groups.google.com/forum/#!forum/psychopy-users>
- Pre-made scripts by us - <http://www.kent.ac.uk/psychology/technical/experiments.html>

## ► Python:

- <http://www.python.org/doc/>
- Google is your friend!

## ► Anything else:

- Psychology Technical Team (A1.6)
- Check out FAQ handout in the directory (will grow over time)

# Exercise 5:

## Create an experiment

For the rest of the lesson (hopefully we have left enough time!) - we have given you an experiment specification handout.

See (or via print out):

`ex5-image-ratings/Exercise-docs.docx`

We are going to let you build a complete experiment from this, with deliberately vague instructions to test what you have learnt So far.

Follow the instructions and we will come round and help and periodically demonstrate on the main screen. If you don't manage to start or finish, then try having a go at home!

**We will upload the solution onto Moodle later on.**

# Feedback

Please take a moment to leave any feedback for future workshops at:

<https://goo.gl/lBg0Em>