Programming Skills
Lecture 6

PsychoPy

presentation and responses

Recap

- String formatting
- Files
- Dicts

Functions

Write down a typical function:

- with arguments input1 and input2
- that is fruitful

Functions

```
def sum_of_two_numbers(input1, input2):
    output = input1 + input2
    return output


y = sum(4, 5)
```

Functions

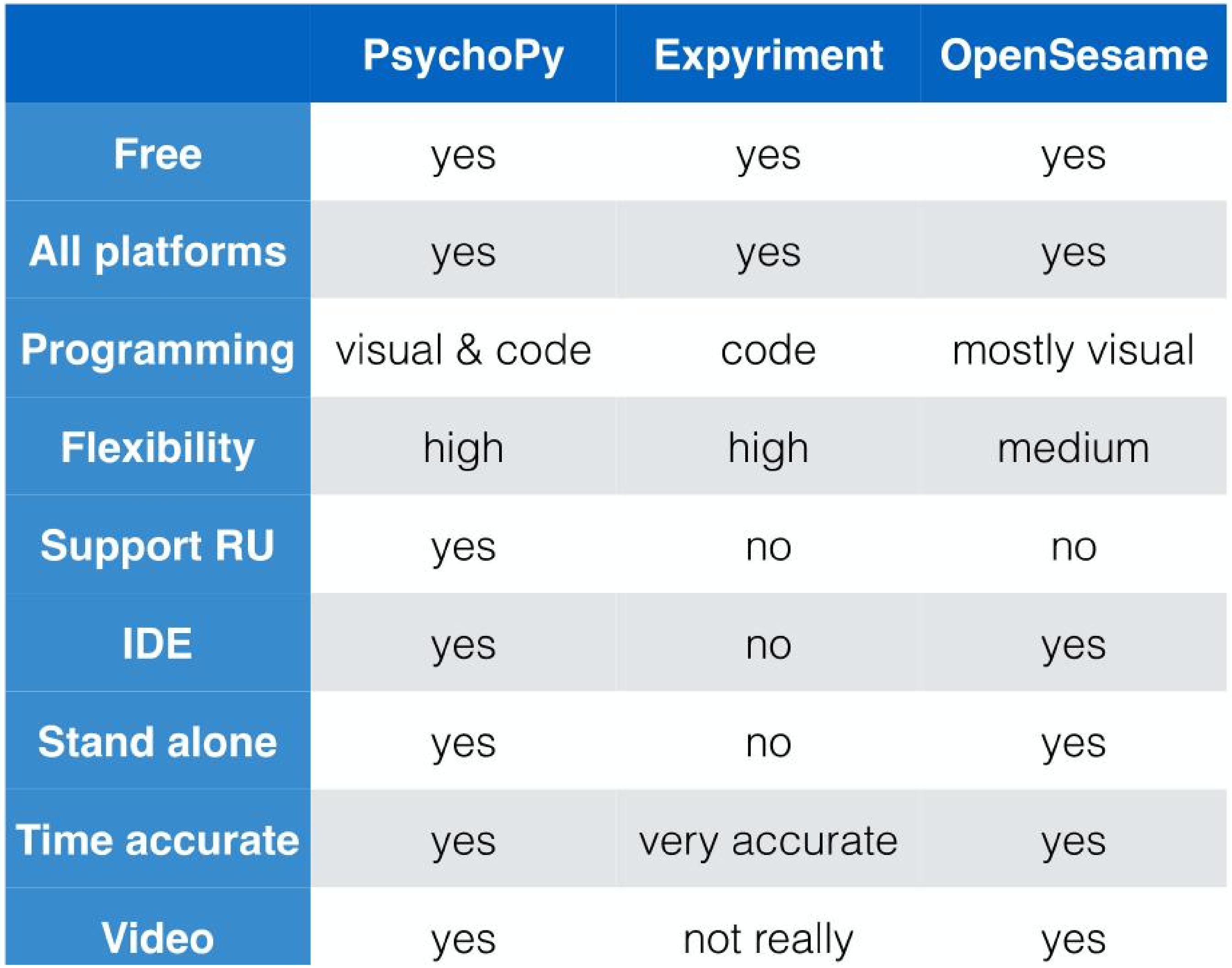
```
def sum_of_two_numbers(input1, input2 = 0):
    output = input1 + input2
    return output


x = sum(4)
y = sum(4, 5)
z = sum(input1=22, input2=5)
```

PsychoPy

Psychology software in Python

expyriment

Comparison



	PsychoPy	Expyriment	OpenSesame
Free	yes	yes	yes
All platforms	yes	yes	yes
Programming	visual & code	code	mostly visual
Flexibility	high	high	medium
Support RU	yes	no	no
IDE	yes	no	yes
Stand alone	yes	no	yes
Time accurate	yes	very accurate	yes
Video	yes	not really	yes

Installing PsychoPy

Instructions here:

http://www.psychopy.org/installation.html

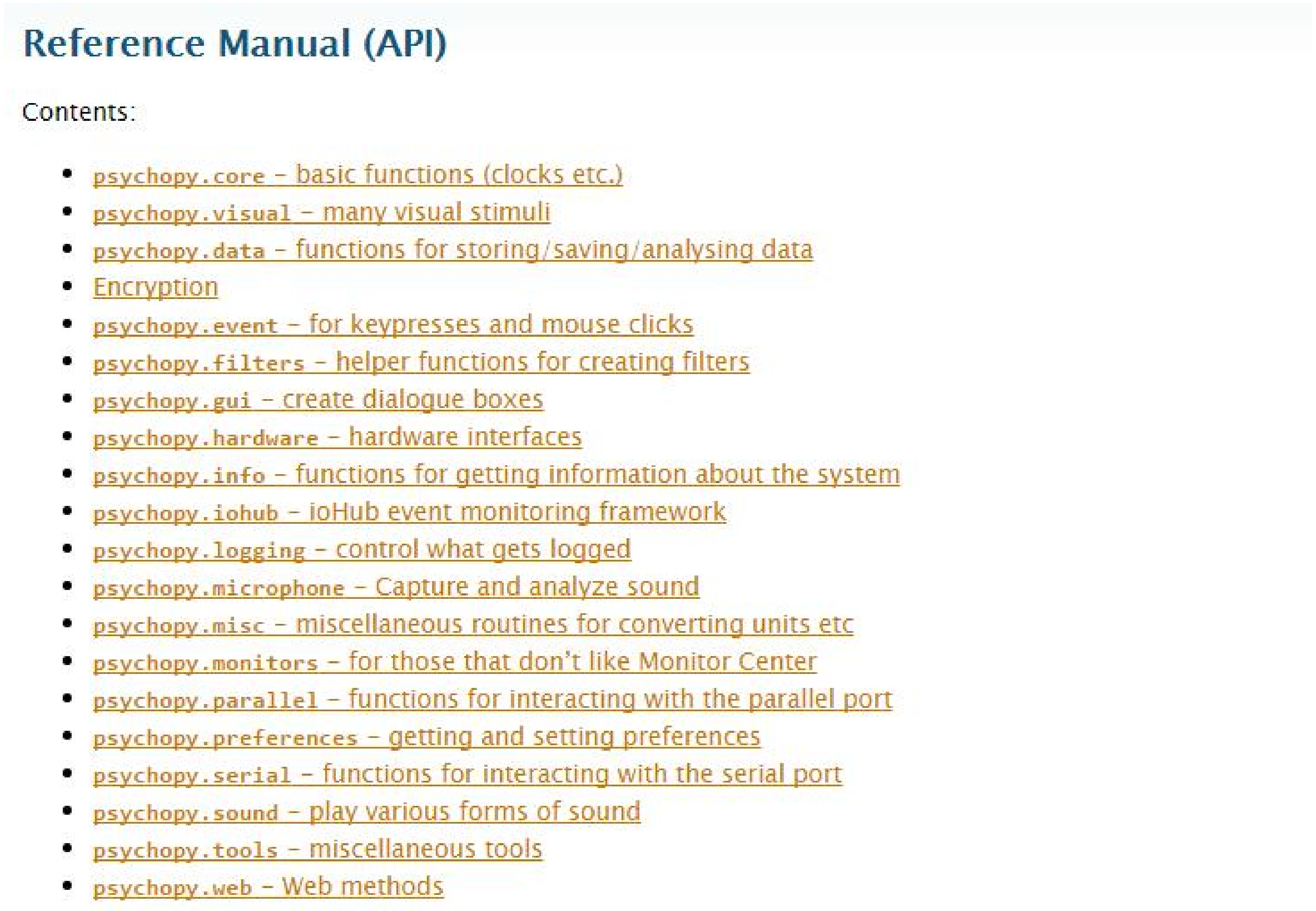
Download the psychopy library (make sure to download all the dependencies)

OR

Download the stand-alone!

http://sourceforge.net/projects/psychpy/files/

PsychoPy modules



Reference Manual (API)

Contents:

- psychopy.core – basic functions (clocks etc.)
- psychopy.visual – many visual stimuli
- psychopy.data – functions for storing/saving/analysing data
- Encryption
- psychopy.event – for keypresses and mouse clicks
- psychopy.filters – helper functions for creating filters
- psychopy.gui – create dialogue boxes
- psychopy.hardware – hardware interfaces
- psychopy.info – functions for getting information about the system
- psychopy.iohub – ioHub event monitoring framework
- psychopy.logging – control what gets logged
- psychopy.microphone – Capture and analyze sound
- psychopy.misc – miscellaneous routines for converting units etc
- psychopy.monitors – for those that don't like Monitor Center
- psychopy.parallel – functions for interacting with the parallel port
- psychopy.preferences – getting and setting preferences
- psychopy.serial – functions for interacting with the serial port
- psychopy.sound – play various forms of sound
- psychopy.tools – miscellaneous tools
- psychopy.web – Web methods

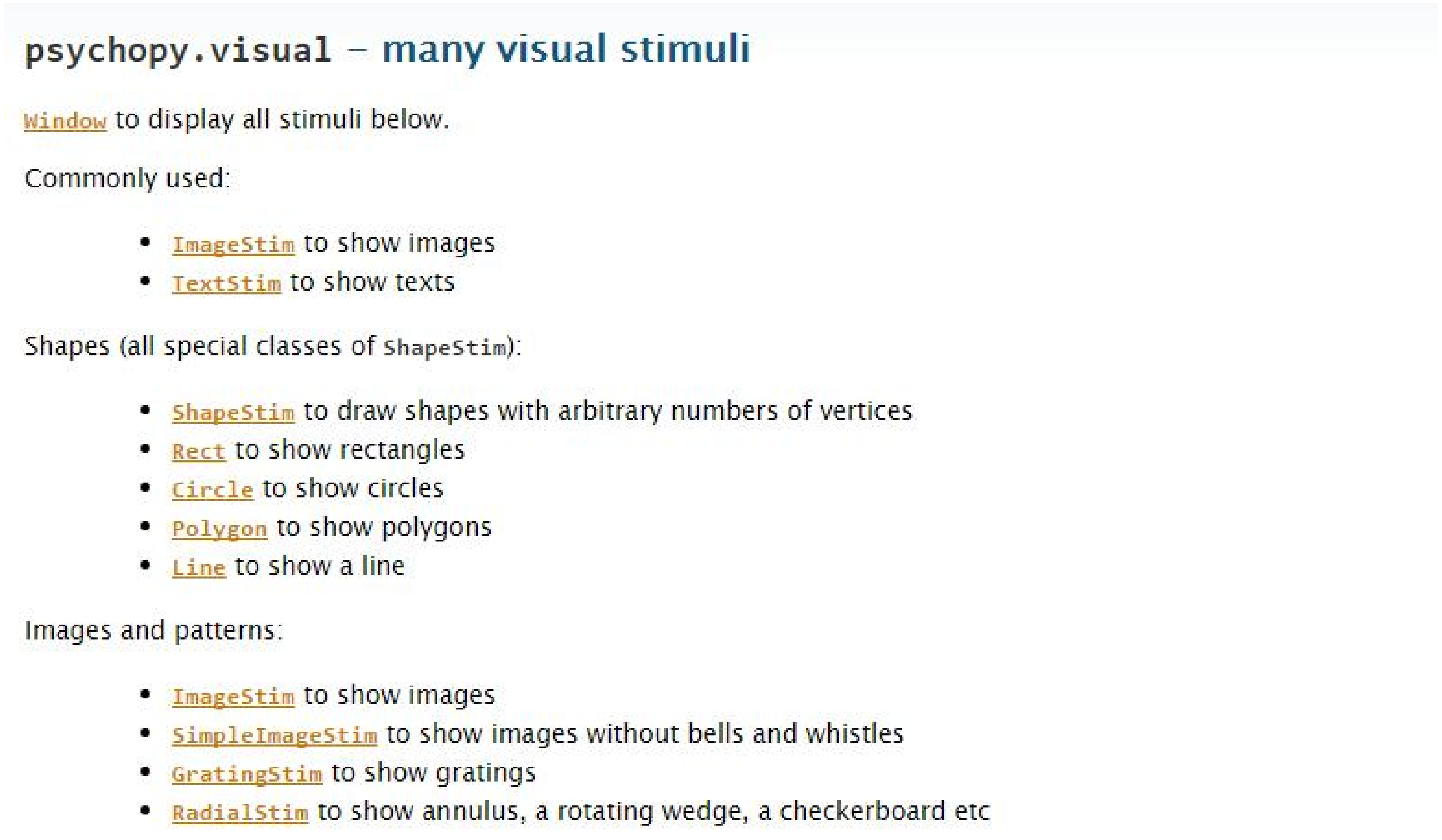
Basic presentation of stimuli

```
from psychopy import visual, core

win = visual.Window()
msg = visual.TextStim(win, text=‘Hello World’)

msg.draw()
win.flip()
core.wait(1)
win.close()
```

Basic presentation of stimuli

```
from psychopy import visual, core

win = visual.window()
msg = visual.TextStim(win, text=‘Hello World’)

msg.draw()
win.flip()
core.wait(1)
win.close()
```

Reference manual



psychopy.visual – many visual stimuli

Window to display all stimuli below.

Commonly used:

- ImageStim to show images
- TextStim to show texts

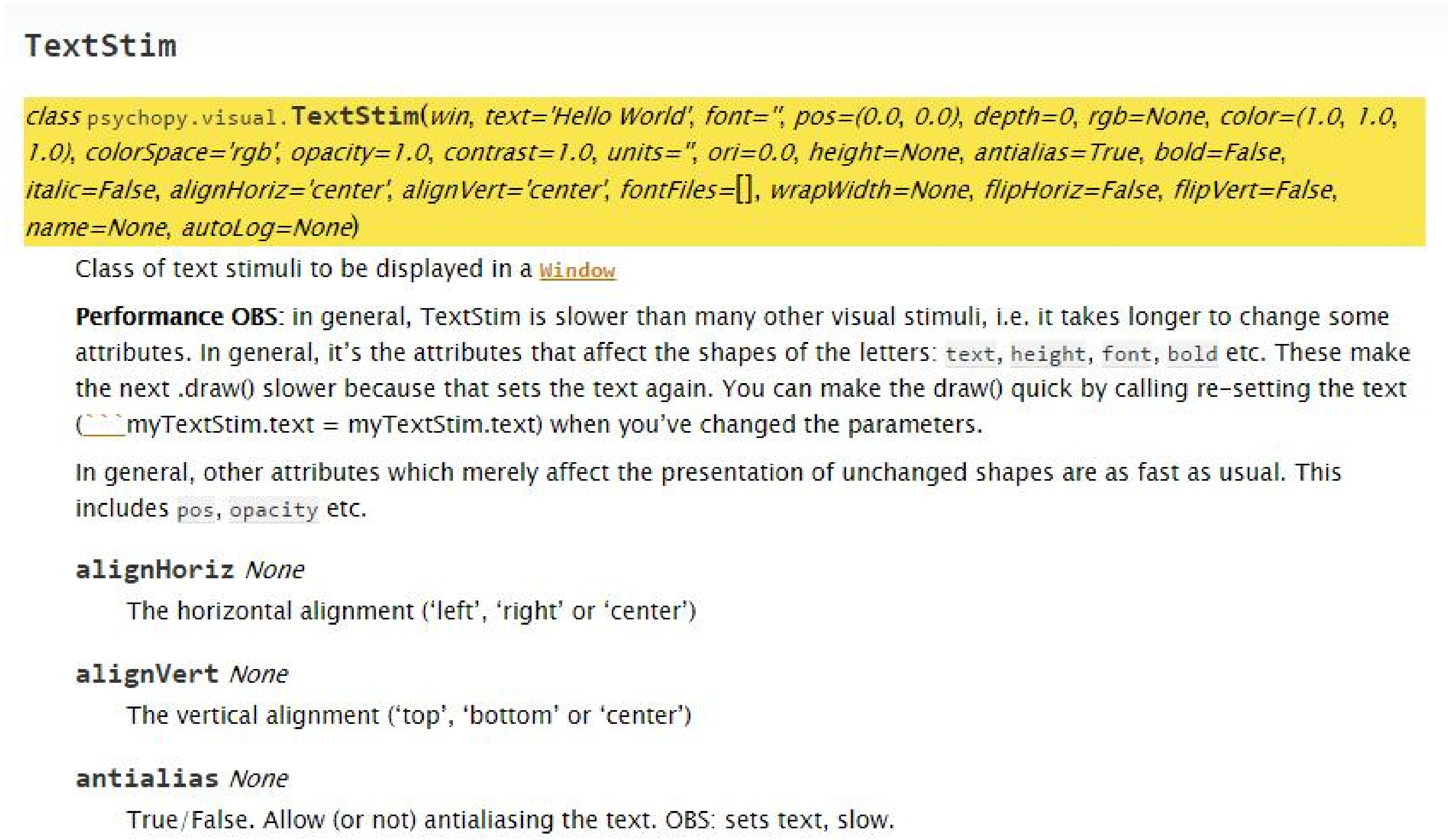
Shapes (all special classes of ShapeStim):

- ShapeStim to draw shapes with arbitrary numbers of vertices
- Rect to show rectangles
- Circle to show circles
- Polygon to show polygons
- Line to show a line

Images and patterns:

- ImageStim to show images
- SimpleImageStim to show images without bells and whistles
- GratingStim to show gratings
- RadialStim to show annulus, a rotating wedge, a checkerboard etc

Reference manual



TextStim

class `psychopy.visual.`**`TextStim`**(*win, text='Hello World', font='', pos=(0.0, 0.0), depth=0, rgb=None, color=(1.0, 1.0, 1.0), colorSpace='rgb', opacity=1.0, contrast=1.0, units='', ori=0.0, height=None, antialias=True, bold=False, italic=False, alignHoriz='center', alignVert='center', fontFiles=[], wrapWidth=None, flipHoriz=False, flipVert=False, name=None, autoLog=None*)

Class of text stimuli to be displayed in a `Window`

Performance OBS: in general, TextStim is slower than many other visual stimuli, i.e. it takes longer to change some attributes. In general, it's the attributes that affect the shapes of the letters: `text`, `height`, `font`, `bold` etc. These make the next .draw() slower because that sets the text again. You can make the draw() quick by calling re-setting the text (```myTextStim.text = myTextStim.text) when you've changed the parameters.

In general, other attributes which merely affect the presentation of unchanged shapes are as fast as usual. This includes `pos`, `opacity` etc.

`alignHoriz` *None*

The horizontal alignment ('left', 'right' or 'center')

`alignVert` *None*

The vertical alignment ('top', 'bottom' or 'center')

`antialias` *None*

True/False. Allow (or not) antialiasing the text. OBS: sets text, slow.

Basic presentation of stimuli

```
from psychopy import visual, core

win = visual.window()
msg = visual.TextStim(win, text=‘Hello World’)

msg.draw()
win.flip()
core.wait(1)
win.close()
```

Why flip?

- Monitors run on specific frequencies / refresh rates
- 60 Hz = 60 frames per second
- CRT monitors offer better timing than LCD
- Flip synchronizes presentation with screen refresh
- Script continues only when flip occurs

What you want to do

- Timing: think in frames
- Draw stimuli in video memory that you need for the following frame
- Flip once all stimuli for next frame are drawn
- As long as you don’t flip again, stimuli will stay on screen

`visual.Window()` parameters

`size = (x, y)`

size of the window in numbers of pixels

`fullscr = True`

Should presentation window be full screen?

`color = ('cornflowerblue')`

Background color of window (see http://www.w3schools.com/html/html_colornames.asp

`visual.Window()` some methods

`.setMouseVisible(False)`

Don’t show mouse pointer during the experiment

`.close()`

Close the window

More on
http://www.psychopy.org/api/visual/window.html

`visual.TextStim(win)` parameters

`win = windowObject`
required: the stimulus must know in which window to draw itself

`text`
Text to be rendered

`color`
Color, same as with Window()

`height, ori, bold, italic, alignHoriz, wrapWidth`
Other useful function to control lay-out

`pos, size`
Position, size on the screen

`units`
Controls what the position and size values mean

Units

- (0,0) is always the center of the screen
- Negative values are down/left
- Positive values are up/right
- Exact values depend on units chosen for stimulus: `pix, height, norm`

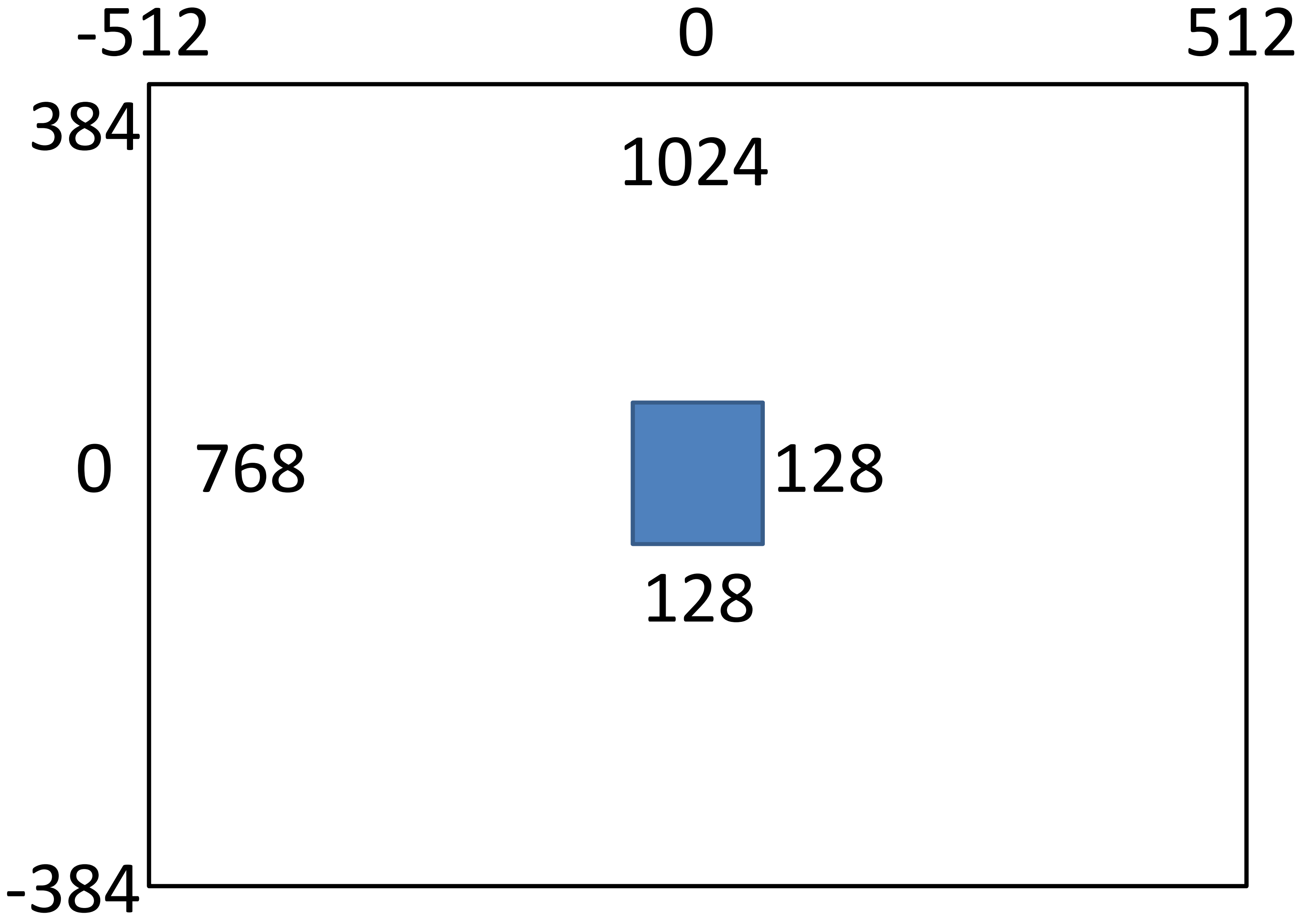
Units

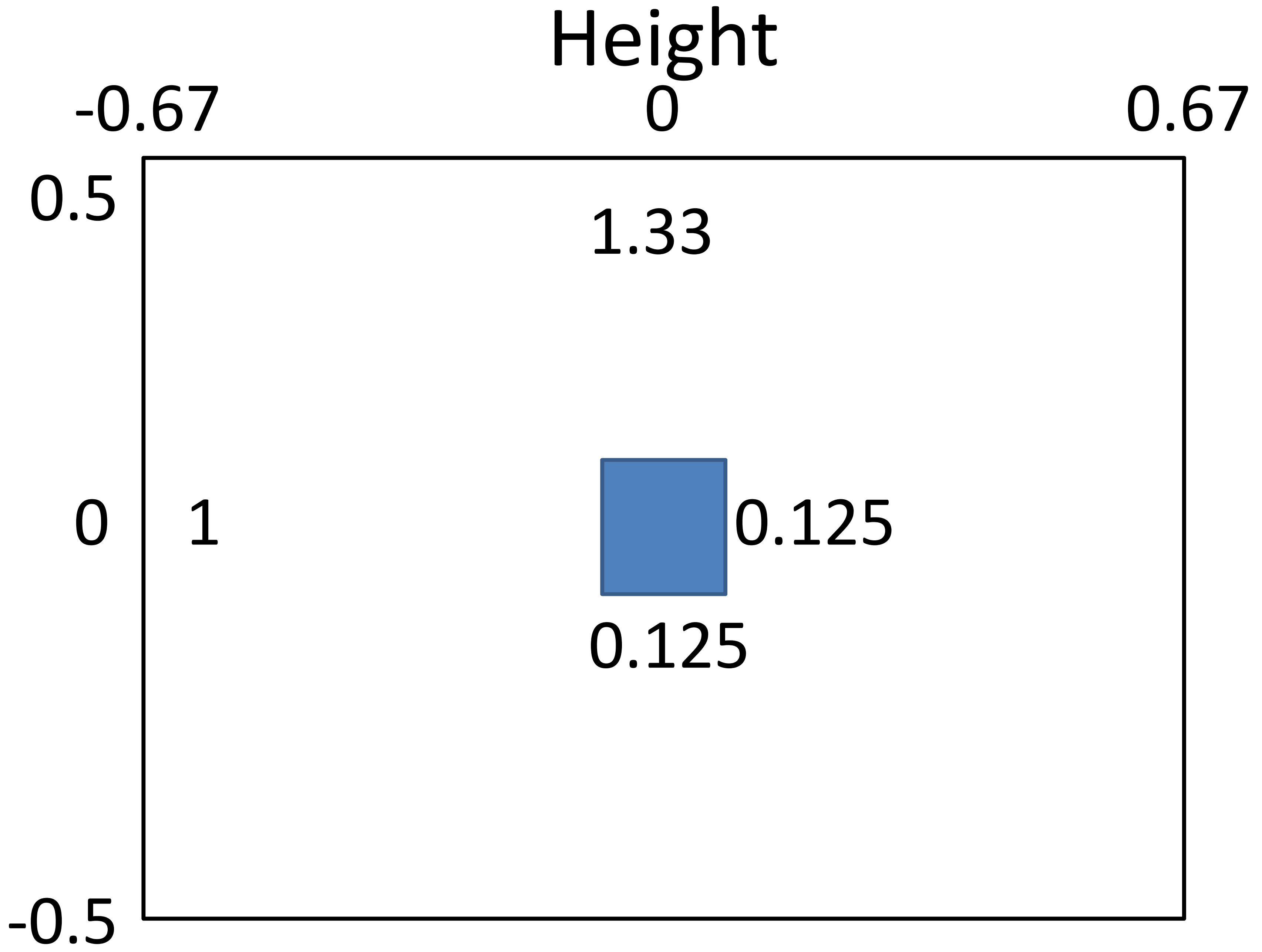
Screen

Aspect ratio: 4:3

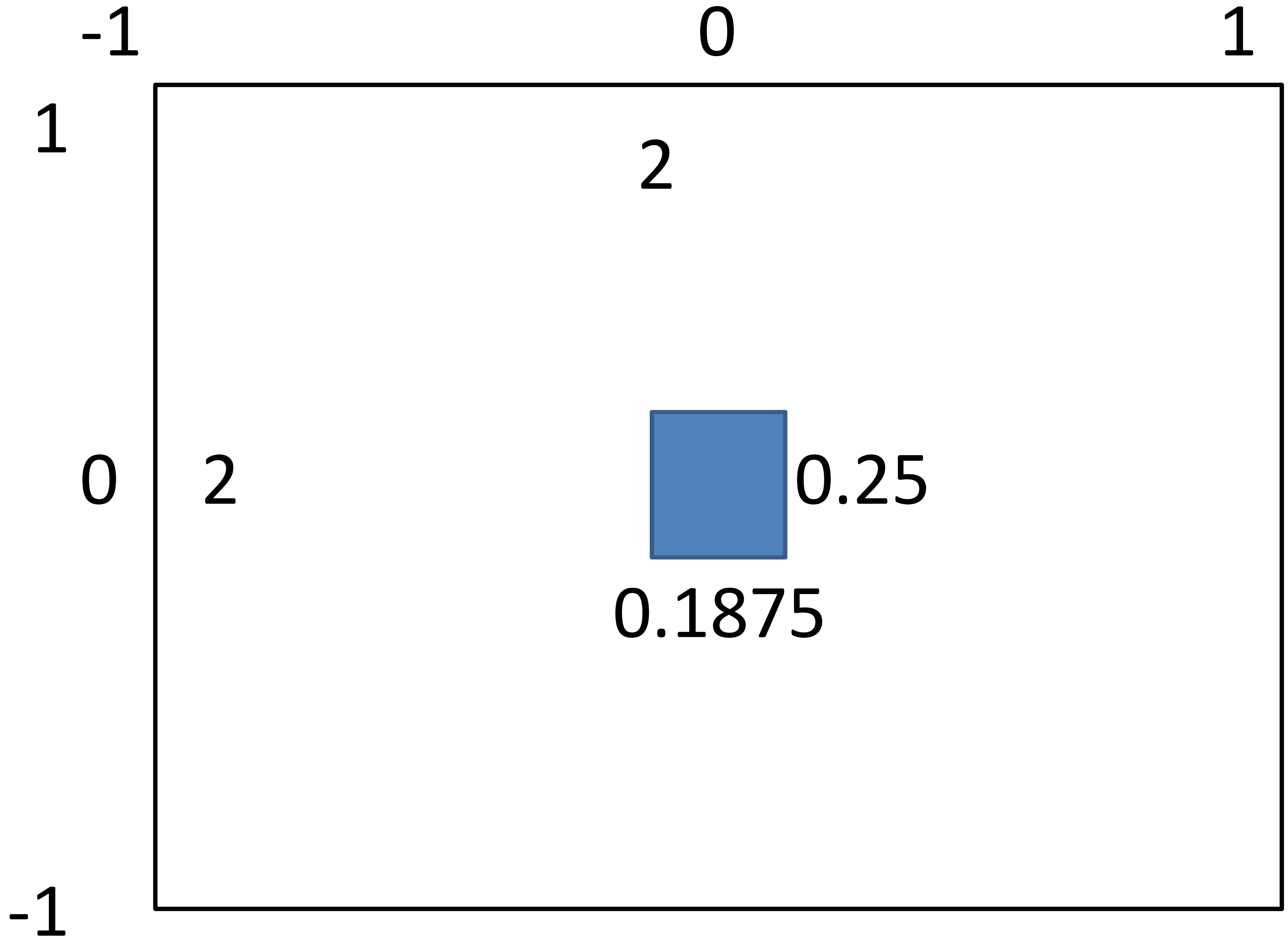
(widescreen: 16:10)

Pix




Height
-0.67
0
0.67
0.5
1.33
0
1
0.125
0.125
-0.5

Norm



`visual.TextStim()` some methods

`.setText("sometext")`

Change the text contents of a TextStim

`.setSize(), .setColor()`

Change size or color of a TextStim

`.draw()`

Draw the stimulus to the video buffer

More on:
http://www.psychopy.org/api/visual/textstim.html

Presenting multiple stimuli

- Draw them all before the `win.flip()` call
- Or automatize with:

```
stim.setAutoDraw(True)
```

so it will get drawn every flip

Responses

- `psychopy.events` module
- Includes functionality to record mouse and keyboard

Wait for keyboard response

```
from psychopy import visual, core, event

win = visual.Window()
msg = visual.TextStim(win, text=“hi guys!”)

msg.draw()
win.flip()
respond = event.waitKeys()
win.close()
```

`event.waitKeys()` parameters

`maxWait`

response window in seconds

`keyList`

list of keys that can be used to respond

`timeStamped`

return exact time of keypress

Note: halts execution of script until key is pressed

Response window and record latency

```
from psychopy import visual, core, event

win = visual.Window()
msg = visual.TextStim(win, text=“hi guys!”)

msg.draw()
win.flip()
respond = event.waitKeys(maxWait=6.0,
  timeStamped=True)
win.close()
```

Get relative latency

```
from psychopy import visual, core, event

win = visual.Window()
msg = visual.TextStim(win, text=“hi guys!”)

msg.draw()
win.flip()
clock = core.Clock()
respond = event.waitKeys(maxWait=6.0,
  timeStamped=clock)
win.close()
```

event.waitKeys()

Returns:

```
[('k', 2.45454)]
```

Coping with non-response in response windows

```
respond = event.waitKeys(maxWait=6.0,
  timeStamped=clock)
responseTuple = respond[0] # (‘k’, 2.543)
```

Problem: `respond == None`

```
respond = event.waitKeys(maxWait=6.0,
  timeStamped=clock)
if respond:
    responseTuple = respond[0] # (‘k’,
  2.543)
else:
    # Do whatever is needed when response
  window is not met
```

Saving data

- There is a `psychopy.data` module for advanced data storage, which includes a lot of fancy stuff
- We use pure python, things you already know

Saving response data

```
from psychopy import visual, core, event

win = visual.Window()
msg = visual.TextStim(win, text="hi guys!")

msg.draw()
win.flip()
clock = core.Clock()
respond= event.waitKeys(maxWait=6.0, timeStamped=clock)

if respond:
    response, latency = response[0]
else:
    response, latency = -1, -1

outputFile = open('data.txt', 'a')
outputFile.write("{}\t{}\n".format(response, latency)
outputFile.close()

win.close()
```

The experiment loop

```
# open your window
# prepare the stimulus objects

For trial in trials :
    # update stimulus objects based on current trial
    # draw your stuff
    # flip your window
    # get response
    # write data (incl trial nr, presented stimuli)
    # inter trial interval with core.Wait()

# close your window
```

Considerations for creating stim objects

- Stim object cost memory
- Just create a single place holder stim and then change the contents at each iteration of the experiment loop

Recap

- Presenting stimuli
- Getting response and latency
- Saving data

This week's homework

Program a gender categorization task using PsychoPy

- 20 trials, 5 male names, 5 female names, each gets used twice
- Use one list containing all names, use `random.shuffle()` to randomize its order then loop through the list to get the stimulus for the respective trial
- Display one line of instruction before the task starts
- Present key reminders during the task
- Record response and latency
- Save trial number, presented stimulus, response and latency data to a txt file (one row per response)