#!/usr/bin/env python2

# -\*- coding: utf-8 -\*-

"""

This experiment was created using PsychoPy2 Experiment Builder (v1.85.4),

on Fri Mar 2 13:21:18 2018

If you publish work using this script please cite the PsychoPy publications:

Peirce, JW (2007) PsychoPy - Psychophysics software in Python.

Journal of Neuroscience Methods, 162(1-2), 8-13.

Peirce, JW (2009) Generating stimuli for neuroscience using PsychoPy.

Frontiers in Neuroinformatics, 2:10. doi: 10.3389/neuro.11.010.2008

"""

from \_\_future\_\_ import absolute\_import, division

from psychopy import locale\_setup, sound, gui, visual, core, data, event, logging

from psychopy.constants import (NOT\_STARTED, STARTED, PLAYING, PAUSED,

STOPPED, FINISHED, PRESSED, RELEASED, FOREVER)

import numpy as np # whole numpy lib is available, prepend 'np.'

from numpy import (sin, cos, tan, log, log10, pi, average,

sqrt, std, deg2rad, rad2deg, linspace, asarray)

from numpy.random import random, randint, normal, shuffle

import os # handy system and path functions

import sys # to get file system encoding

# Ensure that relative paths start from the same directory as this script

\_thisDir = os.path.dirname(os.path.abspath(\_\_file\_\_)).decode(sys.getfilesystemencoding())

os.chdir(\_thisDir)

# Store info about the experiment session

expName = 'ColourTesting' # from the Builder filename that created this script

expInfo = {u'gender': u'', u'age': u'', u'handedness': u'', u'participant': u''}

dlg = gui.DlgFromDict(dictionary=expInfo, title=expName)

if dlg.OK == False:

core.quit() # user pressed cancel

expInfo['date'] = data.getDateStr() # add a simple timestamp

expInfo['expName'] = expName

# Data file name stem = absolute path + name; later add .psyexp, .csv, .log, etc

filename = \_thisDir + os.sep + u'data/%s\_%s\_%s' % (expInfo['participant'], expName, expInfo['date'])

# An ExperimentHandler isn't essential but helps with data saving

thisExp = data.ExperimentHandler(name=expName, version='',

extraInfo=expInfo, runtimeInfo=None,

originPath=None,

savePickle=True, saveWideText=True,

dataFileName=filename)

# save a log file for detail verbose info

logFile = logging.LogFile(filename+'.log', level=logging.EXP)

logging.console.setLevel(logging.WARNING) # this outputs to the screen, not a file

endExpNow = False # flag for 'escape' or other condition => quit the exp

# Start Code - component code to be run before the window creation

# Setup the Window

win = visual.Window(

size=(1440, 900), fullscr=True, screen=0,

allowGUI=False, allowStencil=False,

monitor='testMonitor', color=[-1,-1,-1], colorSpace='rgb',

blendMode='avg', useFBO=True)

# store frame rate of monitor if we can measure it

expInfo['frameRate'] = win.getActualFrameRate()

if expInfo['frameRate'] != None:

frameDur = 1.0 / round(expInfo['frameRate'])

else:

frameDur = 1.0 / 60.0 # could not measure, so guess

# Initialize components for Routine "Welcome"

WelcomeClock = core.Clock()

text\_4 = visual.TextStim(win=win, name='text\_4',

text='Hello and welcome to the cognitive task. This is an n-back memory task\nYou will be asked to pay attention to a sequence of coloured squares\nand decide whether you saw the same square n times ago. \nThere 3 blocks with 3 different rules. We will inform you about the new rule.\nPress space to continue.',

font='Arial',

pos=(0, 0), height=0.1, wrapWidth=None, ori=0,

color='white', colorSpace='rgb', opacity=1,

depth=0.0);

# Initialize components for Routine "Instructions"

InstructionsClock = core.Clock()

text = visual.TextStim(win=win, name='text',

text='This is the 1-back test. You will have to press space if you decide the previous coloured square was the same as the current one.\n\nPlease press space only when you decide the coloured square was repeated. Press space to begin.',

font='Arial',

pos=(0, 0), height=0.1, wrapWidth=None, ori=0,

color='white', colorSpace='rgb', opacity=1,

depth=0.0);

# Initialize components for Routine "Fixation"

FixationClock = core.Clock()

Fix = visual.TextStim(win=win, name='Fix',

text='default text',

font='Arial',

pos=[0,0], height=1.0, wrapWidth=None, ori=0,

color=1.0, colorSpace='rgb', opacity=1,

depth=0.0);

# Initialize components for Routine "trial"

trialClock = core.Clock()

polygon = visual.Rect(

win=win, name='polygon',

width=[1.0, 1.0][0], height=[1.0, 1.0][1],

ori=1.0, pos=[0,0],

lineWidth=1, lineColor=1.0, lineColorSpace='rgb',

fillColor=1.0, fillColorSpace='rgb',

opacity=1.0, depth=0.0, interpolate=True)

# Initialize components for Routine "Break"

BreakClock = core.Clock()

text\_2 = visual.TextStim(win=win, name='text\_2',

text="It's now time for a short break. When you are ready to continue please press space. ",

font='Arial',

pos=(0, 0), height=0.1, wrapWidth=None, ori=0,

color='white', colorSpace='rgb', opacity=1,

depth=0.0);

# Initialize components for Routine "Instructions2"

Instructions2Clock = core.Clock()

text\_5 = visual.TextStim(win=win, name='text\_5',

text='This is the 2-back test. You will have to press space if you decide the coloured square two trials ago was the same as the current one.\n\nPlease press space only when you decide the coloured square was repeated. Press space to begin.',

font='Arial',

pos=(0, 0), height=0.1, wrapWidth=None, ori=0,

color='white', colorSpace='rgb', opacity=1,

depth=0.0);

# Initialize components for Routine "Fixation"

FixationClock = core.Clock()

Fix = visual.TextStim(win=win, name='Fix',

text='default text',

font='Arial',

pos=[0,0], height=1.0, wrapWidth=None, ori=0,

color=1.0, colorSpace='rgb', opacity=1,

depth=0.0);

# Initialize components for Routine "trial"

trialClock = core.Clock()

polygon = visual.Rect(

win=win, name='polygon',

width=[1.0, 1.0][0], height=[1.0, 1.0][1],

ori=1.0, pos=[0,0],

lineWidth=1, lineColor=1.0, lineColorSpace='rgb',

fillColor=1.0, fillColorSpace='rgb',

opacity=1.0, depth=0.0, interpolate=True)

# Initialize components for Routine "Break"

BreakClock = core.Clock()

text\_2 = visual.TextStim(win=win, name='text\_2',

text="It's now time for a short break. When you are ready to continue please press space. ",

font='Arial',

pos=(0, 0), height=0.1, wrapWidth=None, ori=0,

color='white', colorSpace='rgb', opacity=1,

depth=0.0);

# Initialize components for Routine "Instructions3"

Instructions3Clock = core.Clock()

text\_6 = visual.TextStim(win=win, name='text\_6',

text='This is the 3-back test. You will have to press space if you decide the coloured square three trials ago was the same as the current one.\n\nPlease press space only when you decide the coloured square was repeated. Press space to begin.',

font='Arial',

pos=(0, 0), height=0.1, wrapWidth=None, ori=0,

color='white', colorSpace='rgb', opacity=1,

depth=0.0);

# Initialize components for Routine "Fixation"

FixationClock = core.Clock()

Fix = visual.TextStim(win=win, name='Fix',

text='default text',

font='Arial',

pos=[0,0], height=1.0, wrapWidth=None, ori=0,

color=1.0, colorSpace='rgb', opacity=1,

depth=0.0);

# Initialize components for Routine "trial"

trialClock = core.Clock()

polygon = visual.Rect(

win=win, name='polygon',

width=[1.0, 1.0][0], height=[1.0, 1.0][1],

ori=1.0, pos=[0,0],

lineWidth=1, lineColor=1.0, lineColorSpace='rgb',

fillColor=1.0, fillColorSpace='rgb',

opacity=1.0, depth=0.0, interpolate=True)

# Initialize components for Routine "Thanks"

ThanksClock = core.Clock()

text\_3 = visual.TextStim(win=win, name='text\_3',

text='Thank you for participanting in the experiment.\nThe experimenter will be with you shortly.',

font='Arial',

pos=(0, 0), height=0.1, wrapWidth=None, ori=0,

color='white', colorSpace='rgb', opacity=1,

depth=0.0);

# Create some handy timers

globalClock = core.Clock() # to track the time since experiment started

routineTimer = core.CountdownTimer() # to track time remaining of each (non-slip) routine

# ------Prepare to start Routine "Welcome"-------

t = 0

WelcomeClock.reset() # clock

frameN = -1

continueRoutine = True

# update component parameters for each repeat

key\_resp\_5 = event.BuilderKeyResponse()

# keep track of which components have finished

WelcomeComponents = [text\_4, key\_resp\_5]

for thisComponent in WelcomeComponents:

if hasattr(thisComponent, 'status'):

thisComponent.status = NOT\_STARTED

# -------Start Routine "Welcome"-------

while continueRoutine:

# get current time

t = WelcomeClock.getTime()

frameN = frameN + 1 # number of completed frames (so 0 is the first frame)

# update/draw components on each frame

# \*text\_4\* updates

if t >= 0.0 and text\_4.status == NOT\_STARTED:

# keep track of start time/frame for later

text\_4.tStart = t

text\_4.frameNStart = frameN # exact frame index

text\_4.setAutoDraw(True)

# \*key\_resp\_5\* updates

if t >= 0.0 and key\_resp\_5.status == NOT\_STARTED:

# keep track of start time/frame for later

key\_resp\_5.tStart = t

key\_resp\_5.frameNStart = frameN # exact frame index

key\_resp\_5.status = STARTED

# keyboard checking is just starting

win.callOnFlip(key\_resp\_5.clock.reset) # t=0 on next screen flip

event.clearEvents(eventType='keyboard')

if key\_resp\_5.status == STARTED:

theseKeys = event.getKeys(keyList=['space'])

# check for quit:

if "escape" in theseKeys:

endExpNow = True

if len(theseKeys) > 0: # at least one key was pressed

key\_resp\_5.keys = theseKeys[-1] # just the last key pressed

key\_resp\_5.rt = key\_resp\_5.clock.getTime()

# a response ends the routine

continueRoutine = False

# check if all components have finished

if not continueRoutine: # a component has requested a forced-end of Routine

break

continueRoutine = False # will revert to True if at least one component still running

for thisComponent in WelcomeComponents:

if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:

continueRoutine = True

break # at least one component has not yet finished

# check for quit (the Esc key)

if endExpNow or event.getKeys(keyList=["escape"]):

core.quit()

# refresh the screen

if continueRoutine: # don't flip if this routine is over or we'll get a blank screen

win.flip()

# -------Ending Routine "Welcome"-------

for thisComponent in WelcomeComponents:

if hasattr(thisComponent, "setAutoDraw"):

thisComponent.setAutoDraw(False)

# check responses

if key\_resp\_5.keys in ['', [], None]: # No response was made

key\_resp\_5.keys=None

thisExp.addData('key\_resp\_5.keys',key\_resp\_5.keys)

if key\_resp\_5.keys != None: # we had a response

thisExp.addData('key\_resp\_5.rt', key\_resp\_5.rt)

thisExp.nextEntry()

# the Routine "Welcome" was not non-slip safe, so reset the non-slip timer

routineTimer.reset()

# ------Prepare to start Routine "Instructions"-------

t = 0

InstructionsClock.reset() # clock

frameN = -1

continueRoutine = True

# update component parameters for each repeat

key\_resp\_2 = event.BuilderKeyResponse()

# keep track of which components have finished

InstructionsComponents = [text, key\_resp\_2]

for thisComponent in InstructionsComponents:

if hasattr(thisComponent, 'status'):

thisComponent.status = NOT\_STARTED

# -------Start Routine "Instructions"-------

while continueRoutine:

# get current time

t = InstructionsClock.getTime()

frameN = frameN + 1 # number of completed frames (so 0 is the first frame)

# update/draw components on each frame

# \*text\* updates

if t >= 0.0 and text.status == NOT\_STARTED:

# keep track of start time/frame for later

text.tStart = t

text.frameNStart = frameN # exact frame index

text.setAutoDraw(True)

# \*key\_resp\_2\* updates

if t >= 0.0 and key\_resp\_2.status == NOT\_STARTED:

# keep track of start time/frame for later

key\_resp\_2.tStart = t

key\_resp\_2.frameNStart = frameN # exact frame index

key\_resp\_2.status = STARTED

# keyboard checking is just starting

win.callOnFlip(key\_resp\_2.clock.reset) # t=0 on next screen flip

event.clearEvents(eventType='keyboard')

if key\_resp\_2.status == STARTED:

theseKeys = event.getKeys(keyList=['space'])

# check for quit:

if "escape" in theseKeys:

endExpNow = True

if len(theseKeys) > 0: # at least one key was pressed

key\_resp\_2.keys = theseKeys[-1] # just the last key pressed

key\_resp\_2.rt = key\_resp\_2.clock.getTime()

# a response ends the routine

continueRoutine = False

# check if all components have finished

if not continueRoutine: # a component has requested a forced-end of Routine

break

continueRoutine = False # will revert to True if at least one component still running

for thisComponent in InstructionsComponents:

if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:

continueRoutine = True

break # at least one component has not yet finished

# check for quit (the Esc key)

if endExpNow or event.getKeys(keyList=["escape"]):

core.quit()

# refresh the screen

if continueRoutine: # don't flip if this routine is over or we'll get a blank screen

win.flip()

# -------Ending Routine "Instructions"-------

for thisComponent in InstructionsComponents:

if hasattr(thisComponent, "setAutoDraw"):

thisComponent.setAutoDraw(False)

# check responses

if key\_resp\_2.keys in ['', [], None]: # No response was made

key\_resp\_2.keys=None

thisExp.addData('key\_resp\_2.keys',key\_resp\_2.keys)

if key\_resp\_2.keys != None: # we had a response

thisExp.addData('key\_resp\_2.rt', key\_resp\_2.rt)

thisExp.nextEntry()

# the Routine "Instructions" was not non-slip safe, so reset the non-slip timer

routineTimer.reset()

# set up handler to look after randomisation of conditions etc

Block1\_1back = data.TrialHandler(nReps=1, method='sequential',

extraInfo=expInfo, originPath=-1,

trialList=data.importConditions('ColourTest1.xlsx'),

seed=None, name='Block1\_1back')

thisExp.addLoop(Block1\_1back) # add the loop to the experiment

thisBlock1\_1back = Block1\_1back.trialList[0] # so we can initialise stimuli with some values

# abbreviate parameter names if possible (e.g. rgb = thisBlock1\_1back.rgb)

if thisBlock1\_1back != None:

for paramName in thisBlock1\_1back.keys():

exec(paramName + '= thisBlock1\_1back.' + paramName)

for thisBlock1\_1back in Block1\_1back:

currentLoop = Block1\_1back

# abbreviate parameter names if possible (e.g. rgb = thisBlock1\_1back.rgb)

if thisBlock1\_1back != None:

for paramName in thisBlock1\_1back.keys():

exec(paramName + '= thisBlock1\_1back.' + paramName)

# ------Prepare to start Routine "Fixation"-------

t = 0

FixationClock.reset() # clock

frameN = -1

continueRoutine = True

routineTimer.add(1.000000)

# update component parameters for each repeat

Fix.setColor('white', colorSpace='rgb')

Fix.setText('+')

Fix.setPos((0, 0))

Fix.setFont('Arial')

Fix.setHeight(0.1)

# keep track of which components have finished

FixationComponents = [Fix]

for thisComponent in FixationComponents:

if hasattr(thisComponent, 'status'):

thisComponent.status = NOT\_STARTED

# -------Start Routine "Fixation"-------

while continueRoutine and routineTimer.getTime() > 0:

# get current time

t = FixationClock.getTime()

frameN = frameN + 1 # number of completed frames (so 0 is the first frame)

# update/draw components on each frame

# \*Fix\* updates

if t >= 0.0 and Fix.status == NOT\_STARTED:

# keep track of start time/frame for later

Fix.tStart = t

Fix.frameNStart = frameN # exact frame index

Fix.setAutoDraw(True)

frameRemains = 0.0 + 1- win.monitorFramePeriod \* 0.75 # most of one frame period left

if Fix.status == STARTED and t >= frameRemains:

Fix.setAutoDraw(False)

# check if all components have finished

if not continueRoutine: # a component has requested a forced-end of Routine

break

continueRoutine = False # will revert to True if at least one component still running

for thisComponent in FixationComponents:

if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:

continueRoutine = True

break # at least one component has not yet finished

# check for quit (the Esc key)

if endExpNow or event.getKeys(keyList=["escape"]):

core.quit()

# refresh the screen

if continueRoutine: # don't flip if this routine is over or we'll get a blank screen

win.flip()

# -------Ending Routine "Fixation"-------

for thisComponent in FixationComponents:

if hasattr(thisComponent, "setAutoDraw"):

thisComponent.setAutoDraw(False)

# ------Prepare to start Routine "trial"-------

t = 0

trialClock.reset() # clock

frameN = -1

continueRoutine = True

routineTimer.add(2.000000)

# update component parameters for each repeat

polygon.setOpacity(1)

polygon.setFillColor(colourtest)

polygon.setPos((0, 0))

polygon.setLineWidth(1)

polygon.setLineColor(colourtest)

polygon.setOri(0)

polygon.setSize((0.20,0.35))

key\_resp\_3 = event.BuilderKeyResponse()

# keep track of which components have finished

trialComponents = [polygon, key\_resp\_3]

for thisComponent in trialComponents:

if hasattr(thisComponent, 'status'):

thisComponent.status = NOT\_STARTED

# -------Start Routine "trial"-------

while continueRoutine and routineTimer.getTime() > 0:

# get current time

t = trialClock.getTime()

frameN = frameN + 1 # number of completed frames (so 0 is the first frame)

# update/draw components on each frame

# \*polygon\* updates

if t >= 0.0 and polygon.status == NOT\_STARTED:

# keep track of start time/frame for later

polygon.tStart = t

polygon.frameNStart = frameN # exact frame index

polygon.setAutoDraw(True)

frameRemains = 0.0 + 2- win.monitorFramePeriod \* 0.75 # most of one frame period left

if polygon.status == STARTED and t >= frameRemains:

polygon.setAutoDraw(False)

# \*key\_resp\_3\* updates

if t >= 0.0 and key\_resp\_3.status == NOT\_STARTED:

# keep track of start time/frame for later

key\_resp\_3.tStart = t

key\_resp\_3.frameNStart = frameN # exact frame index

key\_resp\_3.status = STARTED

# keyboard checking is just starting

win.callOnFlip(key\_resp\_3.clock.reset) # t=0 on next screen flip

event.clearEvents(eventType='keyboard')

frameRemains = 0.0 + 1- win.monitorFramePeriod \* 0.75 # most of one frame period left

if key\_resp\_3.status == STARTED and t >= frameRemains:

key\_resp\_3.status = STOPPED

if key\_resp\_3.status == STARTED:

theseKeys = event.getKeys(keyList=['space'])

# check for quit:

if "escape" in theseKeys:

endExpNow = True

if len(theseKeys) > 0: # at least one key was pressed

key\_resp\_3.keys.extend(theseKeys) # storing all keys

key\_resp\_3.rt.append(key\_resp\_3.clock.getTime())

# was this 'correct'?

if (key\_resp\_3.keys == str(corresp)) or (key\_resp\_3.keys == corresp):

key\_resp\_3.corr = 1

else:

key\_resp\_3.corr = 0

# a response ends the routine

continueRoutine = False

# check if all components have finished

if not continueRoutine: # a component has requested a forced-end of Routine

break

continueRoutine = False # will revert to True if at least one component still running

for thisComponent in trialComponents:

if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:

continueRoutine = True

break # at least one component has not yet finished

# check for quit (the Esc key)

if endExpNow or event.getKeys(keyList=["escape"]):

core.quit()

# refresh the screen

if continueRoutine: # don't flip if this routine is over or we'll get a blank screen

win.flip()

# -------Ending Routine "trial"-------

for thisComponent in trialComponents:

if hasattr(thisComponent, "setAutoDraw"):

thisComponent.setAutoDraw(False)

# check responses

if key\_resp\_3.keys in ['', [], None]: # No response was made

key\_resp\_3.keys=None

# was no response the correct answer?!

if str(corresp).lower() == 'none':

key\_resp\_3.corr = 1 # correct non-response

else:

key\_resp\_3.corr = 0 # failed to respond (incorrectly)

# store data for Block1\_1back (TrialHandler)

Block1\_1back.addData('key\_resp\_3.keys',key\_resp\_3.keys)

Block1\_1back.addData('key\_resp\_3.corr', key\_resp\_3.corr)

if key\_resp\_3.keys != None: # we had a response

Block1\_1back.addData('key\_resp\_3.rt', key\_resp\_3.rt)

thisExp.nextEntry()

# completed 1 repeats of 'Block1\_1back'

# ------Prepare to start Routine "Break"-------

t = 0

BreakClock.reset() # clock

frameN = -1

continueRoutine = True

# update component parameters for each repeat

key\_resp\_4 = event.BuilderKeyResponse()

# keep track of which components have finished

BreakComponents = [text\_2, key\_resp\_4]

for thisComponent in BreakComponents:

if hasattr(thisComponent, 'status'):

thisComponent.status = NOT\_STARTED

# -------Start Routine "Break"-------

while continueRoutine:

# get current time

t = BreakClock.getTime()

frameN = frameN + 1 # number of completed frames (so 0 is the first frame)

# update/draw components on each frame

# \*text\_2\* updates

if t >= 0.0 and text\_2.status == NOT\_STARTED:

# keep track of start time/frame for later

text\_2.tStart = t

text\_2.frameNStart = frameN # exact frame index

text\_2.setAutoDraw(True)

# \*key\_resp\_4\* updates

if t >= 0.0 and key\_resp\_4.status == NOT\_STARTED:

# keep track of start time/frame for later

key\_resp\_4.tStart = t

key\_resp\_4.frameNStart = frameN # exact frame index

key\_resp\_4.status = STARTED

# keyboard checking is just starting

win.callOnFlip(key\_resp\_4.clock.reset) # t=0 on next screen flip

event.clearEvents(eventType='keyboard')

if key\_resp\_4.status == STARTED:

theseKeys = event.getKeys(keyList=['space'])

# check for quit:

if "escape" in theseKeys:

endExpNow = True

if len(theseKeys) > 0: # at least one key was pressed

key\_resp\_4.keys = theseKeys[-1] # just the last key pressed

key\_resp\_4.rt = key\_resp\_4.clock.getTime()

# a response ends the routine

continueRoutine = False

# check if all components have finished

if not continueRoutine: # a component has requested a forced-end of Routine

break

continueRoutine = False # will revert to True if at least one component still running

for thisComponent in BreakComponents:

if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:

continueRoutine = True

break # at least one component has not yet finished

# check for quit (the Esc key)

if endExpNow or event.getKeys(keyList=["escape"]):

core.quit()

# refresh the screen

if continueRoutine: # don't flip if this routine is over or we'll get a blank screen

win.flip()

# -------Ending Routine "Break"-------

for thisComponent in BreakComponents:

if hasattr(thisComponent, "setAutoDraw"):

thisComponent.setAutoDraw(False)

# check responses

if key\_resp\_4.keys in ['', [], None]: # No response was made

key\_resp\_4.keys=None

thisExp.addData('key\_resp\_4.keys',key\_resp\_4.keys)

if key\_resp\_4.keys != None: # we had a response

thisExp.addData('key\_resp\_4.rt', key\_resp\_4.rt)

thisExp.nextEntry()

# the Routine "Break" was not non-slip safe, so reset the non-slip timer

routineTimer.reset()

# ------Prepare to start Routine "Instructions2"-------

t = 0

Instructions2Clock.reset() # clock

frameN = -1

continueRoutine = True

# update component parameters for each repeat

key\_resp\_6 = event.BuilderKeyResponse()

# keep track of which components have finished

Instructions2Components = [text\_5, key\_resp\_6]

for thisComponent in Instructions2Components:

if hasattr(thisComponent, 'status'):

thisComponent.status = NOT\_STARTED

# -------Start Routine "Instructions2"-------

while continueRoutine:

# get current time

t = Instructions2Clock.getTime()

frameN = frameN + 1 # number of completed frames (so 0 is the first frame)

# update/draw components on each frame

# \*text\_5\* updates

if t >= 0.0 and text\_5.status == NOT\_STARTED:

# keep track of start time/frame for later

text\_5.tStart = t

text\_5.frameNStart = frameN # exact frame index

text\_5.setAutoDraw(True)

# \*key\_resp\_6\* updates

if t >= 0.0 and key\_resp\_6.status == NOT\_STARTED:

# keep track of start time/frame for later

key\_resp\_6.tStart = t

key\_resp\_6.frameNStart = frameN # exact frame index

key\_resp\_6.status = STARTED

# keyboard checking is just starting

win.callOnFlip(key\_resp\_6.clock.reset) # t=0 on next screen flip

event.clearEvents(eventType='keyboard')

if key\_resp\_6.status == STARTED:

theseKeys = event.getKeys(keyList=['space'])

# check for quit:

if "escape" in theseKeys:

endExpNow = True

if len(theseKeys) > 0: # at least one key was pressed

key\_resp\_6.keys = theseKeys[-1] # just the last key pressed

key\_resp\_6.rt = key\_resp\_6.clock.getTime()

# a response ends the routine

continueRoutine = False

# check if all components have finished

if not continueRoutine: # a component has requested a forced-end of Routine

break

continueRoutine = False # will revert to True if at least one component still running

for thisComponent in Instructions2Components:

if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:

continueRoutine = True

break # at least one component has not yet finished

# check for quit (the Esc key)

if endExpNow or event.getKeys(keyList=["escape"]):

core.quit()

# refresh the screen

if continueRoutine: # don't flip if this routine is over or we'll get a blank screen

win.flip()

# -------Ending Routine "Instructions2"-------

for thisComponent in Instructions2Components:

if hasattr(thisComponent, "setAutoDraw"):

thisComponent.setAutoDraw(False)

# check responses

if key\_resp\_6.keys in ['', [], None]: # No response was made

key\_resp\_6.keys=None

thisExp.addData('key\_resp\_6.keys',key\_resp\_6.keys)

if key\_resp\_6.keys != None: # we had a response

thisExp.addData('key\_resp\_6.rt', key\_resp\_6.rt)

thisExp.nextEntry()

# the Routine "Instructions2" was not non-slip safe, so reset the non-slip timer

routineTimer.reset()

# set up handler to look after randomisation of conditions etc

Block2\_2back = data.TrialHandler(nReps=1, method='sequential',

extraInfo=expInfo, originPath=-1,

trialList=data.importConditions('ColourTest2.xlsx'),

seed=None, name='Block2\_2back')

thisExp.addLoop(Block2\_2back) # add the loop to the experiment

thisBlock2\_2back = Block2\_2back.trialList[0] # so we can initialise stimuli with some values

# abbreviate parameter names if possible (e.g. rgb = thisBlock2\_2back.rgb)

if thisBlock2\_2back != None:

for paramName in thisBlock2\_2back.keys():

exec(paramName + '= thisBlock2\_2back.' + paramName)

for thisBlock2\_2back in Block2\_2back:

currentLoop = Block2\_2back

# abbreviate parameter names if possible (e.g. rgb = thisBlock2\_2back.rgb)

if thisBlock2\_2back != None:

for paramName in thisBlock2\_2back.keys():

exec(paramName + '= thisBlock2\_2back.' + paramName)

# ------Prepare to start Routine "Fixation"-------

t = 0

FixationClock.reset() # clock

frameN = -1

continueRoutine = True

routineTimer.add(1.000000)

# update component parameters for each repeat

Fix.setColor('white', colorSpace='rgb')

Fix.setText('+')

Fix.setPos((0, 0))

Fix.setFont('Arial')

Fix.setHeight(0.1)

# keep track of which components have finished

FixationComponents = [Fix]

for thisComponent in FixationComponents:

if hasattr(thisComponent, 'status'):

thisComponent.status = NOT\_STARTED

# -------Start Routine "Fixation"-------

while continueRoutine and routineTimer.getTime() > 0:

# get current time

t = FixationClock.getTime()

frameN = frameN + 1 # number of completed frames (so 0 is the first frame)

# update/draw components on each frame

# \*Fix\* updates

if t >= 0.0 and Fix.status == NOT\_STARTED:

# keep track of start time/frame for later

Fix.tStart = t

Fix.frameNStart = frameN # exact frame index

Fix.setAutoDraw(True)

frameRemains = 0.0 + 1- win.monitorFramePeriod \* 0.75 # most of one frame period left

if Fix.status == STARTED and t >= frameRemains:

Fix.setAutoDraw(False)

# check if all components have finished

if not continueRoutine: # a component has requested a forced-end of Routine

break

continueRoutine = False # will revert to True if at least one component still running

for thisComponent in FixationComponents:

if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:

continueRoutine = True

break # at least one component has not yet finished

# check for quit (the Esc key)

if endExpNow or event.getKeys(keyList=["escape"]):

core.quit()

# refresh the screen

if continueRoutine: # don't flip if this routine is over or we'll get a blank screen

win.flip()

# -------Ending Routine "Fixation"-------

for thisComponent in FixationComponents:

if hasattr(thisComponent, "setAutoDraw"):

thisComponent.setAutoDraw(False)

# ------Prepare to start Routine "trial"-------

t = 0

trialClock.reset() # clock

frameN = -1

continueRoutine = True

routineTimer.add(2.000000)

# update component parameters for each repeat

polygon.setOpacity(1)

polygon.setFillColor(colourtest)

polygon.setPos((0, 0))

polygon.setLineWidth(1)

polygon.setLineColor(colourtest)

polygon.setOri(0)

polygon.setSize((0.20,0.35))

key\_resp\_3 = event.BuilderKeyResponse()

# keep track of which components have finished

trialComponents = [polygon, key\_resp\_3]

for thisComponent in trialComponents:

if hasattr(thisComponent, 'status'):

thisComponent.status = NOT\_STARTED

# -------Start Routine "trial"-------

while continueRoutine and routineTimer.getTime() > 0:

# get current time

t = trialClock.getTime()

frameN = frameN + 1 # number of completed frames (so 0 is the first frame)

# update/draw components on each frame

# \*polygon\* updates

if t >= 0.0 and polygon.status == NOT\_STARTED:

# keep track of start time/frame for later

polygon.tStart = t

polygon.frameNStart = frameN # exact frame index

polygon.setAutoDraw(True)

frameRemains = 0.0 + 2- win.monitorFramePeriod \* 0.75 # most of one frame period left

if polygon.status == STARTED and t >= frameRemains:

polygon.setAutoDraw(False)

# \*key\_resp\_3\* updates

if t >= 0.0 and key\_resp\_3.status == NOT\_STARTED:

# keep track of start time/frame for later

key\_resp\_3.tStart = t

key\_resp\_3.frameNStart = frameN # exact frame index

key\_resp\_3.status = STARTED

# keyboard checking is just starting

win.callOnFlip(key\_resp\_3.clock.reset) # t=0 on next screen flip

event.clearEvents(eventType='keyboard')

frameRemains = 0.0 + 1- win.monitorFramePeriod \* 0.75 # most of one frame period left

if key\_resp\_3.status == STARTED and t >= frameRemains:

key\_resp\_3.status = STOPPED

if key\_resp\_3.status == STARTED:

theseKeys = event.getKeys(keyList=['space'])

# check for quit:

if "escape" in theseKeys:

endExpNow = True

if len(theseKeys) > 0: # at least one key was pressed

key\_resp\_3.keys.extend(theseKeys) # storing all keys

key\_resp\_3.rt.append(key\_resp\_3.clock.getTime())

# was this 'correct'?

if (key\_resp\_3.keys == str(corresp)) or (key\_resp\_3.keys == corresp):

key\_resp\_3.corr = 1

else:

key\_resp\_3.corr = 0

# a response ends the routine

continueRoutine = False

# check if all components have finished

if not continueRoutine: # a component has requested a forced-end of Routine

break

continueRoutine = False # will revert to True if at least one component still running

for thisComponent in trialComponents:

if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:

continueRoutine = True

break # at least one component has not yet finished

# check for quit (the Esc key)

if endExpNow or event.getKeys(keyList=["escape"]):

core.quit()

# refresh the screen

if continueRoutine: # don't flip if this routine is over or we'll get a blank screen

win.flip()

# -------Ending Routine "trial"-------

for thisComponent in trialComponents:

if hasattr(thisComponent, "setAutoDraw"):

thisComponent.setAutoDraw(False)

# check responses

if key\_resp\_3.keys in ['', [], None]: # No response was made

key\_resp\_3.keys=None

# was no response the correct answer?!

if str(corresp).lower() == 'none':

key\_resp\_3.corr = 1 # correct non-response

else:

key\_resp\_3.corr = 0 # failed to respond (incorrectly)

# store data for Block2\_2back (TrialHandler)

Block2\_2back.addData('key\_resp\_3.keys',key\_resp\_3.keys)

Block2\_2back.addData('key\_resp\_3.corr', key\_resp\_3.corr)

if key\_resp\_3.keys != None: # we had a response

Block2\_2back.addData('key\_resp\_3.rt', key\_resp\_3.rt)

thisExp.nextEntry()

# completed 1 repeats of 'Block2\_2back'

# ------Prepare to start Routine "Break"-------

t = 0

BreakClock.reset() # clock

frameN = -1

continueRoutine = True

# update component parameters for each repeat

key\_resp\_4 = event.BuilderKeyResponse()

# keep track of which components have finished

BreakComponents = [text\_2, key\_resp\_4]

for thisComponent in BreakComponents:

if hasattr(thisComponent, 'status'):

thisComponent.status = NOT\_STARTED

# -------Start Routine "Break"-------

while continueRoutine:

# get current time

t = BreakClock.getTime()

frameN = frameN + 1 # number of completed frames (so 0 is the first frame)

# update/draw components on each frame

# \*text\_2\* updates

if t >= 0.0 and text\_2.status == NOT\_STARTED:

# keep track of start time/frame for later

text\_2.tStart = t

text\_2.frameNStart = frameN # exact frame index

text\_2.setAutoDraw(True)

# \*key\_resp\_4\* updates

if t >= 0.0 and key\_resp\_4.status == NOT\_STARTED:

# keep track of start time/frame for later

key\_resp\_4.tStart = t

key\_resp\_4.frameNStart = frameN # exact frame index

key\_resp\_4.status = STARTED

# keyboard checking is just starting

win.callOnFlip(key\_resp\_4.clock.reset) # t=0 on next screen flip

event.clearEvents(eventType='keyboard')

if key\_resp\_4.status == STARTED:

theseKeys = event.getKeys(keyList=['space'])

# check for quit:

if "escape" in theseKeys:

endExpNow = True

if len(theseKeys) > 0: # at least one key was pressed

key\_resp\_4.keys = theseKeys[-1] # just the last key pressed

key\_resp\_4.rt = key\_resp\_4.clock.getTime()

# a response ends the routine

continueRoutine = False

# check if all components have finished

if not continueRoutine: # a component has requested a forced-end of Routine

break

continueRoutine = False # will revert to True if at least one component still running

for thisComponent in BreakComponents:

if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:

continueRoutine = True

break # at least one component has not yet finished

# check for quit (the Esc key)

if endExpNow or event.getKeys(keyList=["escape"]):

core.quit()

# refresh the screen

if continueRoutine: # don't flip if this routine is over or we'll get a blank screen

win.flip()

# -------Ending Routine "Break"-------

for thisComponent in BreakComponents:

if hasattr(thisComponent, "setAutoDraw"):

thisComponent.setAutoDraw(False)

# check responses

if key\_resp\_4.keys in ['', [], None]: # No response was made

key\_resp\_4.keys=None

thisExp.addData('key\_resp\_4.keys',key\_resp\_4.keys)

if key\_resp\_4.keys != None: # we had a response

thisExp.addData('key\_resp\_4.rt', key\_resp\_4.rt)

thisExp.nextEntry()

# the Routine "Break" was not non-slip safe, so reset the non-slip timer

routineTimer.reset()

# ------Prepare to start Routine "Instructions3"-------

t = 0

Instructions3Clock.reset() # clock

frameN = -1

continueRoutine = True

# update component parameters for each repeat

key\_resp\_7 = event.BuilderKeyResponse()

# keep track of which components have finished

Instructions3Components = [text\_6, key\_resp\_7]

for thisComponent in Instructions3Components:

if hasattr(thisComponent, 'status'):

thisComponent.status = NOT\_STARTED

# -------Start Routine "Instructions3"-------

while continueRoutine:

# get current time

t = Instructions3Clock.getTime()

frameN = frameN + 1 # number of completed frames (so 0 is the first frame)

# update/draw components on each frame

# \*text\_6\* updates

if t >= 0.0 and text\_6.status == NOT\_STARTED:

# keep track of start time/frame for later

text\_6.tStart = t

text\_6.frameNStart = frameN # exact frame index

text\_6.setAutoDraw(True)

# \*key\_resp\_7\* updates

if t >= 0.0 and key\_resp\_7.status == NOT\_STARTED:

# keep track of start time/frame for later

key\_resp\_7.tStart = t

key\_resp\_7.frameNStart = frameN # exact frame index

key\_resp\_7.status = STARTED

# keyboard checking is just starting

win.callOnFlip(key\_resp\_7.clock.reset) # t=0 on next screen flip

event.clearEvents(eventType='keyboard')

if key\_resp\_7.status == STARTED:

theseKeys = event.getKeys(keyList=['space'])

# check for quit:

if "escape" in theseKeys:

endExpNow = True

if len(theseKeys) > 0: # at least one key was pressed

key\_resp\_7.keys = theseKeys[-1] # just the last key pressed

key\_resp\_7.rt = key\_resp\_7.clock.getTime()

# a response ends the routine

continueRoutine = False

# check if all components have finished

if not continueRoutine: # a component has requested a forced-end of Routine

break

continueRoutine = False # will revert to True if at least one component still running

for thisComponent in Instructions3Components:

if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:

continueRoutine = True

break # at least one component has not yet finished

# check for quit (the Esc key)

if endExpNow or event.getKeys(keyList=["escape"]):

core.quit()

# refresh the screen

if continueRoutine: # don't flip if this routine is over or we'll get a blank screen

win.flip()

# -------Ending Routine "Instructions3"-------

for thisComponent in Instructions3Components:

if hasattr(thisComponent, "setAutoDraw"):

thisComponent.setAutoDraw(False)

# check responses

if key\_resp\_7.keys in ['', [], None]: # No response was made

key\_resp\_7.keys=None

thisExp.addData('key\_resp\_7.keys',key\_resp\_7.keys)

if key\_resp\_7.keys != None: # we had a response

thisExp.addData('key\_resp\_7.rt', key\_resp\_7.rt)

thisExp.nextEntry()

# the Routine "Instructions3" was not non-slip safe, so reset the non-slip timer

routineTimer.reset()

# set up handler to look after randomisation of conditions etc

Block3\_3back = data.TrialHandler(nReps=1, method='sequential',

extraInfo=expInfo, originPath=-1,

trialList=data.importConditions('ColourTest3.xlsx'),

seed=None, name='Block3\_3back')

thisExp.addLoop(Block3\_3back) # add the loop to the experiment

thisBlock3\_3back = Block3\_3back.trialList[0] # so we can initialise stimuli with some values

# abbreviate parameter names if possible (e.g. rgb = thisBlock3\_3back.rgb)

if thisBlock3\_3back != None:

for paramName in thisBlock3\_3back.keys():

exec(paramName + '= thisBlock3\_3back.' + paramName)

for thisBlock3\_3back in Block3\_3back:

currentLoop = Block3\_3back

# abbreviate parameter names if possible (e.g. rgb = thisBlock3\_3back.rgb)

if thisBlock3\_3back != None:

for paramName in thisBlock3\_3back.keys():

exec(paramName + '= thisBlock3\_3back.' + paramName)

# ------Prepare to start Routine "Fixation"-------

t = 0

FixationClock.reset() # clock

frameN = -1

continueRoutine = True

routineTimer.add(1.000000)

# update component parameters for each repeat

Fix.setColor('white', colorSpace='rgb')

Fix.setText('+')

Fix.setPos((0, 0))

Fix.setFont('Arial')

Fix.setHeight(0.1)

# keep track of which components have finished

FixationComponents = [Fix]

for thisComponent in FixationComponents:

if hasattr(thisComponent, 'status'):

thisComponent.status = NOT\_STARTED

# -------Start Routine "Fixation"-------

while continueRoutine and routineTimer.getTime() > 0:

# get current time

t = FixationClock.getTime()

frameN = frameN + 1 # number of completed frames (so 0 is the first frame)

# update/draw components on each frame

# \*Fix\* updates

if t >= 0.0 and Fix.status == NOT\_STARTED:

# keep track of start time/frame for later

Fix.tStart = t

Fix.frameNStart = frameN # exact frame index

Fix.setAutoDraw(True)

frameRemains = 0.0 + 1- win.monitorFramePeriod \* 0.75 # most of one frame period left

if Fix.status == STARTED and t >= frameRemains:

Fix.setAutoDraw(False)

# check if all components have finished

if not continueRoutine: # a component has requested a forced-end of Routine

break

continueRoutine = False # will revert to True if at least one component still running

for thisComponent in FixationComponents:

if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:

continueRoutine = True

break # at least one component has not yet finished

# check for quit (the Esc key)

if endExpNow or event.getKeys(keyList=["escape"]):

core.quit()

# refresh the screen

if continueRoutine: # don't flip if this routine is over or we'll get a blank screen

win.flip()

# -------Ending Routine "Fixation"-------

for thisComponent in FixationComponents:

if hasattr(thisComponent, "setAutoDraw"):

thisComponent.setAutoDraw(False)

# ------Prepare to start Routine "trial"-------

t = 0

trialClock.reset() # clock

frameN = -1

continueRoutine = True

routineTimer.add(2.000000)

# update component parameters for each repeat

polygon.setOpacity(1)

polygon.setFillColor(colourtest)

polygon.setPos((0, 0))

polygon.setLineWidth(1)

polygon.setLineColor(colourtest)

polygon.setOri(0)

polygon.setSize((0.20,0.35))

key\_resp\_3 = event.BuilderKeyResponse()

# keep track of which components have finished

trialComponents = [polygon, key\_resp\_3]

for thisComponent in trialComponents:

if hasattr(thisComponent, 'status'):

thisComponent.status = NOT\_STARTED

# -------Start Routine "trial"-------

while continueRoutine and routineTimer.getTime() > 0:

# get current time

t = trialClock.getTime()

frameN = frameN + 1 # number of completed frames (so 0 is the first frame)

# update/draw components on each frame

# \*polygon\* updates

if t >= 0.0 and polygon.status == NOT\_STARTED:

# keep track of start time/frame for later

polygon.tStart = t

polygon.frameNStart = frameN # exact frame index

polygon.setAutoDraw(True)

frameRemains = 0.0 + 2- win.monitorFramePeriod \* 0.75 # most of one frame period left

if polygon.status == STARTED and t >= frameRemains:

polygon.setAutoDraw(False)

# \*key\_resp\_3\* updates

if t >= 0.0 and key\_resp\_3.status == NOT\_STARTED:

# keep track of start time/frame for later

key\_resp\_3.tStart = t

key\_resp\_3.frameNStart = frameN # exact frame index

key\_resp\_3.status = STARTED

# keyboard checking is just starting

win.callOnFlip(key\_resp\_3.clock.reset) # t=0 on next screen flip

event.clearEvents(eventType='keyboard')

frameRemains = 0.0 + 1- win.monitorFramePeriod \* 0.75 # most of one frame period left

if key\_resp\_3.status == STARTED and t >= frameRemains:

key\_resp\_3.status = STOPPED

if key\_resp\_3.status == STARTED:

theseKeys = event.getKeys(keyList=['space'])

# check for quit:

if "escape" in theseKeys:

endExpNow = True

if len(theseKeys) > 0: # at least one key was pressed

key\_resp\_3.keys.extend(theseKeys) # storing all keys

key\_resp\_3.rt.append(key\_resp\_3.clock.getTime())

# was this 'correct'?

if (key\_resp\_3.keys == str(corresp)) or (key\_resp\_3.keys == corresp):

key\_resp\_3.corr = 1

else:

key\_resp\_3.corr = 0

# a response ends the routine

continueRoutine = False

# check if all components have finished

if not continueRoutine: # a component has requested a forced-end of Routine

break

continueRoutine = False # will revert to True if at least one component still running

for thisComponent in trialComponents:

if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:

continueRoutine = True

break # at least one component has not yet finished

# check for quit (the Esc key)

if endExpNow or event.getKeys(keyList=["escape"]):

core.quit()

# refresh the screen

if continueRoutine: # don't flip if this routine is over or we'll get a blank screen

win.flip()

# -------Ending Routine "trial"-------

for thisComponent in trialComponents:

if hasattr(thisComponent, "setAutoDraw"):

thisComponent.setAutoDraw(False)

# check responses

if key\_resp\_3.keys in ['', [], None]: # No response was made

key\_resp\_3.keys=None

# was no response the correct answer?!

if str(corresp).lower() == 'none':

key\_resp\_3.corr = 1 # correct non-response

else:

key\_resp\_3.corr = 0 # failed to respond (incorrectly)

# store data for Block3\_3back (TrialHandler)

Block3\_3back.addData('key\_resp\_3.keys',key\_resp\_3.keys)

Block3\_3back.addData('key\_resp\_3.corr', key\_resp\_3.corr)

if key\_resp\_3.keys != None: # we had a response

Block3\_3back.addData('key\_resp\_3.rt', key\_resp\_3.rt)

thisExp.nextEntry()

# completed 1 repeats of 'Block3\_3back'

# ------Prepare to start Routine "Thanks"-------

t = 0

ThanksClock.reset() # clock

frameN = -1

continueRoutine = True

# update component parameters for each repeat

# keep track of which components have finished

ThanksComponents = [text\_3]

for thisComponent in ThanksComponents:

if hasattr(thisComponent, 'status'):

thisComponent.status = NOT\_STARTED

# -------Start Routine "Thanks"-------

while continueRoutine:

# get current time

t = ThanksClock.getTime()

frameN = frameN + 1 # number of completed frames (so 0 is the first frame)

# update/draw components on each frame

# \*text\_3\* updates

if t >= 0.0 and text\_3.status == NOT\_STARTED:

# keep track of start time/frame for later

text\_3.tStart = t

text\_3.frameNStart = frameN # exact frame index

text\_3.setAutoDraw(True)

# check if all components have finished

if not continueRoutine: # a component has requested a forced-end of Routine

break

continueRoutine = False # will revert to True if at least one component still running

for thisComponent in ThanksComponents:

if hasattr(thisComponent, "status") and thisComponent.status != FINISHED:

continueRoutine = True

break # at least one component has not yet finished

# check for quit (the Esc key)

if endExpNow or event.getKeys(keyList=["escape"]):

core.quit()

# refresh the screen

if continueRoutine: # don't flip if this routine is over or we'll get a blank screen

win.flip()

# -------Ending Routine "Thanks"-------

for thisComponent in ThanksComponents:

if hasattr(thisComponent, "setAutoDraw"):

thisComponent.setAutoDraw(False)

# the Routine "Thanks" was not non-slip safe, so reset the non-slip timer

routineTimer.reset()

# these shouldn't be strictly necessary (should auto-save)

thisExp.saveAsWideText(filename+'.csv')

thisExp.saveAsPickle(filename)

logging.flush()

# make sure everything is closed down

thisExp.abort() # or data files will save again on exit

win.close()

core.quit()