```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
This experiment was created using PsychoPy2 Experiment Builder (v1.90.3),
    on 四  7/19 20:25:18 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, clock
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__))
os.chdir(_thisDir)

# Store info about the experiment session
expName = 'untitled.py'
expInfo = {'participant': '', 'session': '001'}
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=(1024, 768), fullscr=True, screen=0,
    allowGUI=False, allowStencil=False,
    monitor='testMonitor', color=[0,0,0], 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 "trial"
trialClock = core.Clock()

# 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 "trial"-------
```

```
t = 0
trialClock.reset()  # clock
frameN = -1
continueRoutine = True
# update component parameters for each repeat
# keep track of which components have finished
trialComponents = []
for thisComponent in trialComponents:
    if hasattr(thisComponent, 'status'):
        thisComponent.status = NOT_STARTED

# -------Start Routine "trial"-------
while continueRoutine:
    # 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

    # 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)
# the Routine "trial" 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()
```