Chris Piech and Mehran Sahami
Code in Place

Section #3
April 29, 2020

Section Handout #3: Social Science

In psychology, the Stroop effect is a demonstration of cognitive interference where a delay in the reaction time of a task occurs due to a mismatch in stimuli. The corresponding paper is one of the most cited in the history of Psychology. In this section we are going to implement a working version of a Stroop test. Our stroop test has two phases:

The first phase is the "control phase". The user is shown color names written in the ***same*** font-color. Each time the user has to write the name of the color. Count how many times they type the name of the color correctly in 10 seconds. Here are a few examples:

```
pink
What color ink is the word written in? pink

red
What color ink is the word written in? red

blue
What color ink is the word written in? green
Correct answer was blue.
```

The second phase is the "experiment phase". The user is shown color names written in a font color which is ***different*** from the color name. Each time the user has to write the name of the font-color. Count how many times they type the name of the font-color correctly in 10 seconds.

```
blue
What color ink is the word written in? red

red
What color ink is the word written in? blue

pink
What color ink is the word written in? pink
Correct answer was: blue
```

Note that the font colors in the example are red then blue, then blue. The mismatch in stimuli in this second phase slows users down. Colorblind? See the section at the end.

This is a big, real-world program. It is going to give you practice with key skills. You don't need to get through the whole program! **Our goal is to hit milestone #1**.

Starter Code

Unlike past programs, this time we start with a small amount of starter code. To begin, think about what parameters and returns each function expects:

```
def print_intro():
    print('This is the Stroop test! Name the font-color used:')
    print_in_color('red', 'red')
    print_in_color('blue', 'blue')
    print_in_color('pink', 'pink')

def random_color_name():
    return random.choice(['red', 'blue', 'pink'])

def print_in_color(text, font_color):
    if font_color == 'pink': # magenta is a confusing name...
        font_color = 'magenta'
    print(colored(text, font_color, attrs=['bold']))
```

Milestone #1: Run a Single Test

As an intermediate milestone, write a function:

```
def run_single_test(is_phase_1):
```

Which displays a single experiment like this one. In this example `is_phase_1` is `False`:

```
blue
What color ink is the word written in? red
```

It should take a single boolean parameter, `is_phase_1`. If `is_phase_1` is `True`, then the color printed should be the same as the font-color. Otherwise the font-color should be random. Your function should return a boolean which is True if the user answered correctly. False otherwise.

For this milestone, don't worry about making sure that a randomly chosen font color is different from the written color. Also don't worry about telling the user the correct answer.

Milestone #2: Run Multiple Tests

If you have time, add in a function `run_phase(is_phase_1)` which runs tests for 10 seconds and then returns the number of correct responses. The parameter is the same as run_single_test.

To measure time, use the function `time.time()` making sure to `import time`. This function returns the number of seconds since Jan 1st, 1970. If you call `time.time()` twice, the difference in the two values will be the time elapsed between the two calls.

Bringing it all together

To touch up your program, complete the main function: run phase 1, then phase 2. After, report the number of correct responses in each phase.

As a final touch-up, make sure that randomly chosen font-colors in the experiment phase are different from the written color name.

Installing font-color modules

This example uses the "termcolor" library. A library is a body of already written code which you import and use, and in this case the termcolor library has code to print to the terminal in color.

In order for the Stroop Test to work properly, you need to install termcolor on your machine. To install termcolor, you should first open a "terminal" window: the easiest way to do this is to use the “Terminal” tab within PyCharm on the lower-left (next to the “Run” and “Python Console” tabs). Type the following command shown in bold below into the Terminal. On Windows, type “py” instead of “python3”):

```
> python3 -m pip install termcolor
...prints stuff...
Successfully installed termcolor-1.1.0
```

Most python code works exactly the same on Windows, Linux and Mac. In the case of terminal colors, Windows users need to do one extra step (and it is safe to include this step in code for Mac and Windows). Install the module named colorama

```
> python3 -m pip install colorama
```

Then, at the top of your code include the code:

```
from colorama import init
init(autoreset=True)
```

Not everyone can see colors!

It is very common for people to distinguish colors differently! Roughly 8% of men (and 1% of women) have some form of “color blindness”. Famous examples include Keanu Reeves, Mark Zuckerberg and Mark Twain. The most common form is for users to have trouble distinguishing red and green however there are several types, and some people can’t see color at all. Color-blind programmers can write this Stroop test program, but might have trouble taking the test. How could you improve this test for folks who are colorblind?

If you are having trouble with this example, see our alternate section problem: Animal Association Test (which is truly similar to the Stroop Test).

Solution to all milestones

```
PHASE_TIME_S = 10

def main():
    print_intro()
    n_correct_control = run_tests(True)
    n_correct_flipped = run_tests(False)
    print('\nDone!')
    print('control', n_correct_control)
    print('flipped', n_correct_flipped)

def run_tests(is_control):
    start_time = time.time()
    n_correct = 0
    while time.time() - start_time < PHASE_TIME_S:
        correct = run_single_test(is_control)
        if correct:
            n_correct += 1
    return n_correct

def run_single_test(is_control):
    print('')
    color_name = random_color_name()
    font_color = get_font_color(color_name, is_control)
    print_in_color(color_name, font_color)
    response = input('What color ink is the word written in? ')
    if response != font_color:
        print('Correct answer was: ' + font_color)
    return response == font_color

def get_font_color(color_name, is_control):
    if is_control:
        return color_name
    while True:
        next_choice = random_color_name()
        if next_choice != color_name:
            return next_choice


(this doesn’t include the starter code)
```