Chris Piech and Mehran Sahami
Code in Place

Section Handout #3: Animals, cute or scary?

In psychology, the Implicit Association Test (IAT) is a demonstration of cognitive interference where a delay in the reaction time of a task occurs due to implicit associations users have. In this section we are going to implement a working version of IAT to test associations of animals as either cute or scary. Our IAT test has two phases:

The first phase is the "control phase". The user is shown animal names with a ***stereotypical*** association of cute or scary. Each time the user has to write the name of the association. Count how many times they type the association correctly in 10 seconds. Here are a few examples:

```
Association test, standard
You will be asked to answer association questions.
You should associate animals as follows:
puppy => cute
panda => cute
spider => scary
bat => scary
Press enter to start...

puppy
Type the association: cute

panda
Type the association: cute

bat
Type the association: cute
Correct answer was scary
```

The second phase is the "experiment phase". The user is shown animal names with an ***anti-stereotypical*** association. Each time the user has to write the name of the association. Count how many times they type the association correctly in 10 seconds. Here are a few examples:

```
Association test, flipped
You will be asked to answer association questions.
You should associate animals as follows:
puppy => scary
panda => scary
spider => cute
bat => cute
Press enter to start...
```

```
panda
Type the association: scary

bat
Type the association: cute

puppy
Type the association: cute
Correct answer was scary
```

The mismatch in stimuli in this second phase slows users down.

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 #2**.

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(is_phase_1):
    if is_phase_1:
        print('Association test, standard')
    else:
        print('Association test, flipped')
    print('You will be asked to answer three questions.')
    print('You should associate animals as follows:')
    print('puppy', get_association('puppy', is_phase_1))
    print('panda', get_association('panda', is_phase_1))
    print('spider', get_association('spider', is_phase_1))
    print('bat', get_association('bat', is_phase_1))
    input('Press enter to start... ')

def random_animal():
    return random.choice(['puppy', 'panda', 'spider', 'bat'])

def get_association(animal, is_phase_1):
    if animal == 'puppy' or animal == 'panda':
        return 'cute'
    if animal == 'bat' or animal == 'spider':
        return 'scary'
```

Milestone #1: Fix get association

Right now the get_association function is incomplete. If the second parameter is False, the answers should be flipped!

Milestone #2: 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`:

```
puppy
Type the association: cute
Correct answer was scary
```

It should take a single boolean parameter, `is_phase_1`. If `is_phase_1` is `True`, then the association should be the stereotypical one. Otherwise the association should be the anti-stereotypical one. Your function should return a boolean which is True if the user answered correctly. False otherwise.

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.