0.1 Experiment 1 Code

```
from speriment import *

##### Wording #####

introduction = '''In this experiment, you will be shown made-up words.
Because they're not real words, they're spelled phonetically, so
sometimes the spelling will look weird even though the sounds are
fine. Try to focus only on how they sound. Pronounce the words to
yourself.
<br><br>
You'll be asked to say whether the word could be a
word of English. You don't have to say 'yes' and 'no' an even number
of times. Just answer whatever you think based on the way the word
sounds.'''

instructions = '''During experimental questions, you will not be able
to use your mouse. Use the F key to choose the option on the left, the
J key to choose the option on the right, and the spacebar to move
forward. Make sure your CAPS lock is off.  At the end of the
experiment, you can use the mouse to answer some questions about
yourself.'''

question = '''Based on how it sounds, do you think this word could be
a word of English? <br><br>'''

native_language = "What is your native language?"

daily_language = "What language do you speak at home?"

age = "How old are you?"

sex = "What is your sex?"

strategy = '''What information or strategies did you use to choose
words in the experimental questions?'''

goodbye = '''Thank you for participating in this experiment! Make sure
to click the Complete HIT button on the next page.'''


##### Stimuli #####

rows = get_dicts('items_oct8.csv')
```

```
# condition, onset violation, coda violation
test_conditions = [
        ('GG', 0, 0),
        ('GB', 0, 1),
        ('BG', 1, 0),
        ('BB', 1, 1)]

# good fillers
catch_words = [
     'lbifth',
     'krlisg',
     'psatrl',
     'tmozb',
     'zgdivl',
     'wlramr']

# bad fillers
catch_words2 = [
    'bressic',
    'claffer',
    'spellion',
    'frotion',
    'merly',
    'dellous']

##### Experiment structure #####

with make_experiment(IDGenerator()):

    intro_block = Block(pages = [
        Page(introduction, tags =             'PageType': 'Instructions')])
    instruction_block = Block(pages = [
        Page(instructions, tags = 'PageType': 'Instructions'),
        Page(instructions2, tags = 'PageType': 'Instruction')])

    test_page_groups = [
        [Page(
            [question, "<b>", row[test_condition], "</b>"],
            options = [
                Option("Yes", tags = 'Response': 1),
                Option("No", tags = 'Response': 0)],
            tags =                 'ItemID': str(i),                'OnsetViolatio
            condition = test_condition,
            keyboard = True
        ) for (test_condition, onset, coda) in test_conditions]
```

```
for (i, row) in enumerate(rows)]

catch_trials = [Page(
        [question, '<b>', word, '</b>'],
        options = [
            Option('Yes', tags = 'Response': 1, 'ExcludeMe': 1),
            Option('No', tags = 'Response': 0, 'ExcludeMe': 0)
        ],
        tags = 'PageType': 'Catch',
        condition = 'worst',
        keyboard = True)
    for word in catch_words]

catch_trials2 = [Page(
        [question, '<b>', word, '</b>'],
        options = [
            Option('Yes', tags = 'Response': 1, 'ExcludeMe': 0),
            Option('No', tags = 'Response': 0, 'ExcludeMe': 1)
        ],
        tags = 'PageType': 'Catch',
        condition = 'best',
        keyboard = True)
    for word in catch_words2]

# The test words come in four conditions, but the catch words
# don't, so I create groups of four identical catch words to avoid
# throwing off the Latin Square. I use the "new" method to give
# each copy a unique ID.
def make_catch_groups(catch_trial):
    catch_group = [catch_trial]
    for i in range(1, 4):
        new_trial = catch_trial.new()
        catch_group.append(new_trial)
    return catch_group

catch_groups = [make_catch_groups(catch_trial)
        for catch_trial in catch_trials + catch_trials2]

# The test block includes the test pages and the catch pages.
# They'll be shuffled together.
test_block = Block(
    groups = test_page_groups + catch_groups,
    latin_square = True)

demographics_block = Block(pages = [
```

```
        Page(age,
            options = [Option(i) for i in range(120)],
            ordered = True,
            tags = 'PageType': 'Demographics'),
        Page(sex,
            options = [Option('male'), Option('female'),
            Option('other')],
            tags = 'PageType': 'Demographics'),
        Page(native_language,
            options = [Option()],
            freetext = True,
            tags = 'PageType': 'Demographics'),
        Page(daily_language,
            options = [Option()],
            freetext = True,
            tags = 'PageType': 'Demographics'),
        Page(strategy,
            options = [Option()],
            freetext = True,
            tags = 'PageType': 'Demographics'),
    ])

    thanks_block = Block( pages = [Page(goodbye,
        tags = 'PageType': 'Instructions')])

    experiment = Experiment([
        intro_block,
        instruction_block,
        test_block,
        demographics_block,
        thanks_block
    ])
    experiment.install('cumulativity_experiment')
```

0.2 Experiment 2 Code

```
from speriment import *

##### Wording #####

introduction = '''In this experiment, you will be shown made-up words.
Because they're not real words, they're spelled phonetically, so
sometimes the spelling will look weird even though the sounds are
fine. Try to focus only on how they sound. Pronounce the words to
```

```
yourself.
<br><br>
You'll be asked to say whether the word could be a word of English.
You don't have to say 'yes' and 'no' an even number of times. Just
answer whatever you think based on the way the word sounds.'''

instructions2 = '''Before we start, here are examples of the kind of
words you'll see.
<br><br>
<b>blickity</b> is the kind of word you might want to say "yes" to.
It's not an English word, but it sounds like it could be.
<br><br>
<b>rzbesgathv</b> is the kind of word you might want to say "no" to.
It's not an English word, and it doesn't sound like it could ever be
one.'''

question = '''Based on how it sounds, do you think this word could be
a word of English? <br><br>'''

native_language = "What is your native language?"

daily_language = "What language do you speak at home?"

age = "How old are you?"

sex = "What is your sex?"

strategy = '''What information or strategies did you use to choose
words in the experimental questions?'''

goodbye = '''Thank you for participating in this experiment! Make sure
to click the Complete HIT button on the next page.'''


##### Stimuli #####

rows = get_dicts('items_june17.csv')

# condition, onset violation, coda violation
test_conditions = [
        ('GG', 0, 0),
        ('GB', 0, 1),
        ('BG', 1, 0),
        ('BB', 1, 1)]
```

```
# bad fillers
catch_words = [
        'lbafthrizk',
        'kflisgwevr',
        'psafzotrl',
        'tmuhrizb',
        'zgdokpevf',
        'wlratlumr'
        ]

# good fillers
catch_words2 = [
    'bressic',
    'claffer',
    'spellion',
    'frotion',
    'merly',
    'dellous']

##### Experiment structure #####

with make_experiment(IDGenerator()):

    intro_block = Block(pages = [
        Page(introduction, tags =             'PageType': 'Instructions')])
    instruction_block = Block(pages = [
        Page(instructions, tags = 'PageType': 'Instructions'),
        Page(instructions2, tags = 'PageType': 'Instruction')])

    test_page_groups = [
        [Page(
            [question, "<b>", row[test_condition], "</b>"],
            options = [
                Option("Yes", tags = 'Response': 1),
                Option("No", tags = 'Response': 0)],
            tags =                 'ItemID': str(i),                'OnsetViolatio
            condition = test_condition,
            keyboard = True
        ) for (test_condition, onset, coda) in test_conditions]
    for (i, row) in enumerate(rows)]

    catch_trials = [Page(
            [question, '<b>', word, '</b>'],
            options = [
                Option('Yes', tags = 'Response': 1, 'ExcludeMe': 1),
```

```
            Option('No', tags = 'Response': 0, 'ExcludeMe': 0)
        ],
        tags = 'PageType': 'Catch',
        condition = 'worst',
        keyboard = True)
    for word in catch_words]

catch_trials2 = [Page(
        [question, '<b>', word, '</b>'],
        options = [
            Option('Yes', tags = 'Response': 1, 'ExcludeMe': 0),
            Option('No', tags = 'Response': 0, 'ExcludeMe': 1)
        ],
        tags = 'PageType': 'Catch',
        condition = 'best',
        keyboard = True)
    for word in catch_words2]

# The test words come in four conditions, but the catch words
# don't, so I create groups of four identical catch words to avoid
# throwing off the Latin Square. I use the "new" method to give
# each copy a unique ID.
def make_catch_groups(catch_trial):
    catch_group = [catch_trial]
    for i in range(1, 4):
        new_trial = catch_trial.new()
        catch_group.append(new_trial)
    return catch_group

catch_groups = [make_catch_groups(catch_trial)
        for catch_trial in catch_trials + catch_trials2]

# The test block includes the test pages and the catch pages.
# They'll be shuffled together.
test_block = Block(
    groups = test_page_groups + catch_groups,
    latin_square = True)

demographics_block = Block(pages = [
    Page(age,
        options = [Option(i) for i in range(120)],
        ordered = True,
        tags = 'PageType': 'Demographics'),
    Page(sex,
        options = [Option('male'), Option('female'),
```

```
        Option('other')],
        tags = 'PageType': 'Demographics'),
    Page(native_language,
        options = [Option()],
        freetext = True,
        tags = 'PageType': 'Demographics'),
    Page(daily_language,
        options = [Option()],
        freetext = True,
        tags = 'PageType': 'Demographics'),
    Page(strategy,
        options = [Option()],
        freetext = True,
        tags = 'PageType': 'Demographics'),
])

thanks_block = Block( pages = [Page(goodbye,
    tags = 'PageType': 'Instructions')])

experiment = Experiment([
    intro_block,
    instruction_block,
    test_block,
    demographics_block,
    thanks_block
])
experiment.install('cumulativity_experiment')
```