# Experiment_Code_Jupyter_Notebook

September 20, 2023

[1]:
```
import numpy as np
import sounddevice as sd
import time
import random
import json
import matplotlib.pyplot as plt
import os
random.seed(43)
```

[2]:
```
def generate_sinusoidal_tone(frequency, duration, spl, sample_rate):
    num_samples = int((duration / 1000) * sample_rate)
    t = np.linspace(0, duration / 1000, num_samples, False)
    amplitude = 10**((spl - 94) / 20)
    tone = amplitude * np.sin(2 * np.pi * frequency * t)
    return tone
```

[3]:
```
def test_Audio_devices():
    devices = sd.query_devices()
    print("Available audio devices:")
    for i, device in enumerate(devices):
        if("speaker" in device['name'].lower()):
            print("Device Number: "+str(i)+" Device Name: "+device['name'])
    my_sound_device = devices[int(input("Enter the number corresponding to the␣
↪audio device you want to use"))]

    try:
        test_tone = generate_sinusoidal_tone(1000, 250, 65,␣
↪my_sound_device['default_samplerate'])
        sd.play(test_tone, device=my_sound_device['index'])
        sd.wait()
        print(my_sound_device)
        user_value = int(input("Did you hear?\t 1 for yes 0 for no"))
        if(user_value == 1):
            return my_sound_device
        else:
            print("Choose another audio device\t")
            return test_Audio_devices()
    except Exception as e:
        print(e)
        print("There was an error testing this device. Choose another device")
        return test_Audio_devices()
```

[4]:
```
def value_exists(value, list_of_dicts):
    for dictionary in list_of_dicts:
        if value in dictionary.values():
            return True
    return False
```

[5]:
```
selected_sound_device = test_Audio_devices()
```

```
Available audio devices:
Device Number: 4 Device Name: Speaker/Headphone (Realtek High
Device Number: 9 Device Name: Speaker/Headphone (Realtek High Definition Audio)
Device Number: 10 Device Name: Speaker/Headphone (Realtek High Definition Audio)
Device Number: 15 Device Name: Speakers (Realtek HD Audio output)
Enter the number corresponding to the audio device you want to use9
{'name': 'Speaker/Headphone (Realtek High Definition Audio)', 'index': 9,
'hostapi': 1, 'max_input_channels': 0, 'max_output_channels': 2,
'default_low_input_latency': 0.0, 'default_low_output_latency': 0.12,
'default_high_input_latency': 0.0, 'default_high_output_latency': 0.24,
'default_samplerate': 44100.0}
Did you hear?    1 for yes 0 for no1
```

[6]:
```
step_size = 25 #Step Size in Hz
SPL = 70 #in dB
duration = 250 #duration in ms
standard_freq = 1000 #in HZ
```

[7]:
```
def play_tone(tone1):
    sd.play(tone1, device=selected_sound_device['index'])
    sd.wait()
def play_3AFC_stimuli(reference_frequency, test_frequency):



    test_tone = []
    frequencies = [reference_frequency, test_frequency, reference_frequency]
    random.shuffle(frequencies)

    for i, myfreq in enumerate(frequencies):
        time.sleep(2)
        print("Playing frequency "+str(i))

        test_tone_1 = generate_sinusoidal_tone(myfreq, duration, SPL,␣
 ↪selected_sound_device['default_samplerate'])
```

```
        play_tone(test_tone_1)
        if(myfreq == test_frequency):
            test_tone = test_tone_1.tolist()

    time.sleep(2)
    correct_response = frequencies.index(test_frequency)
    if(reference_frequency == test_frequency):
        correct_response = 9
    return correct_response, test_frequency, test_tone
```

[8]:
```
def play_2AFC_stimuli(standard_frequency, test_frequency):
    frequencies = [test_frequency, standard_frequency]
    random.shuffle(frequencies)
    tones = []
    for i,freq in enumerate(frequencies):
        time.sleep(2)
        print("Playing frequency "+str(i))
        test_tone_1 = generate_sinusoidal_tone(freq, duration, SPL,␣
 ↪selected_sound_device['default_samplerate'])
        play_tone(test_tone_1)
        tones.append(test_tone_1)
    greater_response = frequencies.index(max(frequencies))
    if(standard_frequency == test_frequency):
        greater_response = 9
    return greater_response, test_frequency, tones[frequencies.
 ↪index(test_frequency)].tolist()
```

[9]:
```
def get_user_response_3AFC():
    user_input = int(input("Which tone contains test frequency? If all tones␣
 ↪are equal, answer 9\t user input: \t"))
    return user_input
def get_user_response_2AFC():
    user_input = int(input("Which tone is greater? If the tones are equal,␣
 ↪answer 9\t user input: \t"))
    return user_input
```

[ ]:
```

```

[10]:
```
##Major Code for 2AFC
```

[11]:
```
initially_below_2AFC = []
initially_above_2AFC = []
frequency_for_initially_below_2AFC = []
frequency_for_initially_above_2AFC = []
map_For_initially_below_2AFC = []
map_For_initially_above_2AFC = []
```

```
initially_below_reversal_count_2AFC = 0
reversals_for_initially_below_2AFC = []
reversals_for_initially_above_2AFC = []
initially_above_reversal_count_2AFC = 0
trials_2AFC = 0
correct_count_initially_below_2AFC = 0
correct_count_initially_above_2AFC = 0
incorrects_between_consecutive_corrects_asc_2AFC = []
incorrects_between_consecutive_corrects_desc_2AFC = []

def start_below_reference_2AFC():
    global step_size
    global SPL
    global duration
    global standard_freq
    global trials_2AFC

    initial_freq = 935
    correct_response, test_frequency, test_tone = 
 ↪play_2AFC_stimuli(standard_freq, initial_freq)
    user_input = get_user_response_2AFC()
    trials_2AFC += 1
    if(user_input == correct_response):
        test_case = {"test_tone": test_tone, "test_frequency": test_frequency, 
 ↪"user_response": "correct"}
        initially_below_2AFC.append(test_case)
        frequency_for_initially_below_2AFC.append(test_frequency)
        map_For_initially_below_2AFC.append({"frequency": test_frequency, 
 ↪"user_response": "correct"})

    else:
        test_case = {"test_tone": test_tone, "test_frequency": test_frequency, 
 ↪"user_response": "incorrect"}
        initially_below_2AFC.append(test_case)
        frequency_for_initially_below_2AFC.append(test_frequency)
        map_For_initially_below_2AFC.append({"frequency": test_frequency, 
 ↪"user_response": "incorrect"})
def start_above_reference_2AFC():
    global step_size
    global SPL
    global duration
    global standard_freq
    global trials_2AFC

    initial_freq = 1065
    correct_response, test_frequency, test_tone = 
 ↪play_2AFC_stimuli(standard_freq, initial_freq)
```

```
    user_input = get_user_response_2AFC()
    trials_2AFC += 1
    if(user_input == correct_response):
        test_case = {"test_tone": test_tone, "test_frequency": test_frequency,␣
↪"user_response": "correct"}
        initially_above_2AFC.append(test_case)
        frequency_for_initially_above_2AFC.append(test_frequency)
        map_For_initially_above_2AFC.append({"frequency": test_frequency,␣
↪"user_response": "correct"})

    else:
        test_case = {"test_tone": test_tone, "test_frequency": test_frequency,␣
↪"user_response": "incorrect"}
        initially_above_2AFC.append(test_case)
        frequency_for_initially_above_2AFC.append(test_frequency)
        map_For_initially_above_2AFC.append({"frequency": test_frequency,␣
↪"user_response": "incorrect"})

def last_response_is_incorrect_asc_2AFC(last_response_of_selected_series):
    global step_size
    global SPL
    global duration
    global standard_freq
    global trials_2AFC
    global initially_below_reversal_count_2AFC
    global step_size_record_correct
    global step_size_record_incorrect
    global min_step_size
    global max_step_size
    test_frequency = last_response_of_selected_series["test_frequency"]



    if(last_response_of_selected_series["test_frequency"] < standard_freq):
        test_frequency = last_response_of_selected_series["test_frequency"] -␣
↪step_size
    elif(last_response_of_selected_series["test_frequency"] > standard_freq):
        test_frequency = last_response_of_selected_series["test_frequency"] +␣
↪step_size
    else:
        test_frequency = last_response_of_selected_series["test_frequency"] -␣
↪step_size
```

```
    correct_response, test_frequency, test_tone =␣
↪play_2AFC_stimuli(standard_freq, test_frequency)

    trials_2AFC = trials_2AFC + 1
    user_input = get_user_response_2AFC()

    if(user_input==correct_response):


        test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"correct"}
        initially_below_2AFC.append(test_case_1)
        frequency_for_initially_below_2AFC.append(test_frequency)
        map_For_initially_below_2AFC.append({"frequency": test_frequency,␣
↪"user_response": "correct"})

    else:

        test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"incorrect"}
        initially_below_2AFC.append(test_case_1)
        frequency_for_initially_below_2AFC.append(test_frequency)
        map_For_initially_below_2AFC.append({"frequency": test_frequency,␣
↪"user_response": "incorrect"})

def last_response_is_correct_asc_2AFC(last_response_of_selected_series):
    global correct_count_initially_below_2AFC
    global step_size
    global SPL
    global duration
    global standard_freq
    global trials_2AFC
    global initially_below_reversal_count_2AFC
    global step_size_record_correct
    global step_size_record_incorrect
    global min_step_size
    global max_step_size

    if(correct_count_initially_below_2AFC < 3 and␣
↪correct_count_initially_below_2AFC >=1):
        test_frequency = last_response_of_selected_series['test_frequency']
        correct_response, test_frequency, test_tone =␣
↪play_2AFC_stimuli(standard_freq, test_frequency)



        user_input = get_user_response_2AFC()

        trials_2AFC = trials_2AFC + 1

        if (user_input!=correct_response):

            test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"incorrect"}
            frequency_for_initially_below_2AFC.append(test_frequency)
            initially_below_2AFC.append(test_case_1)
            incorrects_between_consecutive_corrects_asc_2AFC.append(test_case_1)
            map_For_initially_below_2AFC.append({"frequency": test_frequency,␣
↪"user_response": "incorrect"})

            correct_count_initially_below_2AFC = 0


        else:

            correct_count_initially_below_2AFC += 1
            test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"correct"}
            initially_below_2AFC.append(test_case_1)
            frequency_for_initially_below_2AFC.append(test_frequency)
            map_For_initially_below_2AFC.append({"frequency": test_frequency,␣
↪"user_response": "correct"})
        return
    elif(correct_count_initially_below_2AFC == 0):




        test_frequency = last_response_of_selected_series["test_frequency"]



        if(last_response_of_selected_series["test_frequency"] < standard_freq):
            test_frequency = last_response_of_selected_series["test_frequency"]␣
↪+ step_size
```

```
        elif(last_response_of_selected_series["test_frequency"] >␣
↪standard_freq):
            test_frequency = last_response_of_selected_series["test_frequency"]␣
↪- step_size
        else:
            test_frequency = last_response_of_selected_series["test_frequency"]␣
↪+ step_size



        correct_response, test_frequency, test_tone =␣
↪play_2AFC_stimuli(standard_freq, test_frequency)



        user_input = get_user_response_2AFC()

        trials_2AFC = trials_2AFC + 1

        if (user_input==correct_response):


            test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"correct"}
            initially_below_2AFC.append(test_case_1)
            frequency_for_initially_below_2AFC.append(test_frequency)
            map_For_initially_below_2AFC.append({"frequency": test_frequency,␣
↪"user_response": "correct"})


        else:

            test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"incorrect"}

            initially_below_2AFC.append(test_case_1)
            frequency_for_initially_below_2AFC.append(test_frequency)
            map_For_initially_below_2AFC.append({"frequency": test_frequency,␣
↪"user_response": "incorrect"})


def initially_below_2afc_trial():
    global correct_count_initially_below_2AFC
    global step_size
    global SPL
    global duration
    global standard_freq
    global trials_2AFC
    global initially_below_reversal_count_2AFC
    if(initially_below_2AFC ):
        last_response_of_selected_series = initially_below_2AFC[-1]
        if(last_response_of_selected_series['user_response']=="incorrect"):
            ␣
 ↪last_response_is_incorrect_asc_2AFC(last_response_of_selected_series)
            return
        elif(last_response_of_selected_series['user_response']=="correct"):
            if(correct_count_initially_below_2AFC  ==0):
                correct_count_initially_below_2AFC  = 1
            if(correct_count_initially_below_2AFC  >=1 and␣
 ↪correct_count_initially_below_2AFC <3):
                last_response_of_selected_series = initially_below_2AFC[-1]
                ␣
 ↪if(last_response_of_selected_series['user_response']=="correct"):
                    ␣
 ↪last_response_is_correct_asc_2AFC(last_response_of_selected_series)
            elif(correct_count_initially_below_2AFC >= 3):
                correct_count_initially_below_2AFC = 0
                last_response_of_selected_series = initially_below_2AFC[-1]
                ␣
 ↪if(last_response_of_selected_series['user_response']=="correct"):
                    ␣
 ↪last_response_is_correct_asc_2AFC(last_response_of_selected_series)

            return



    else:
        start_below_reference_2AFC()

def last_response_is_incorrect_desc_2AFC(last_response_of_selected_series):
    #user unable to perceive the difference, make the difference go up, making␣
 ↪it easier to differnetiate
    global step_size
    global SPL
    global duration
    global standard_freq
    global trials_2AFC
    global initially_above_reversal_count_2AFC
    global step_size_record_correct
    global step_size_record_incorrect
    global min_step_size
    global max_step_size
    test_frequency = last_response_of_selected_series["test_frequency"]


    if(last_response_of_selected_series["test_frequency"] < standard_freq):
        test_frequency = last_response_of_selected_series["test_frequency"] -␣
↪step_size
    elif(last_response_of_selected_series["test_frequency"] > standard_freq):
        test_frequency = last_response_of_selected_series["test_frequency"] +␣
↪step_size
    else:
        test_frequency = last_response_of_selected_series["test_frequency"] +␣
↪step_size



    correct_response, test_frequency, test_tone =␣
↪play_2AFC_stimuli(standard_freq, test_frequency)

    trials_2AFC = trials_2AFC + 1
    user_input = get_user_response_2AFC()

    if(user_input==correct_response):

        test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"correct"}
        initially_above_2AFC.append(test_case_1)
        frequency_for_initially_above_2AFC.append(test_frequency)
        map_For_initially_above_2AFC.append({"frequency": test_frequency,␣
↪"user_response": "correct"})

    else:

        test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"incorrect"}
        initially_above_2AFC.append(test_case_1)
        frequency_for_initially_above_2AFC.append(test_frequency)
        map_For_initially_above_2AFC.append({"frequency": test_frequency,␣
↪"user_response": "incorrect"})

def last_response_is_correct_desc_2AFC(last_response_of_selected_series):
    global correct_count_initially_above_2AFC
    global step_size
    global SPL
    global duration
    global standard_freq
    global trials_2AFC
    global initially_above_reversal_count_2AFC
    global step_size_record_correct
    global step_size_record_incorrect
    global min_step_size
    global max_step_size

    if(correct_count_initially_above_2AFC < 3 and␣
↪correct_count_initially_above_2AFC >=1):

        test_frequency = last_response_of_selected_series['test_frequency']
        correct_response, test_frequency, test_tone =␣
↪play_2AFC_stimuli(standard_freq, test_frequency)



        user_input = get_user_response_2AFC()

        trials_2AFC = trials_2AFC + 1

        if (user_input!=correct_response):

            test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"incorrect"}
            incorrects_between_consecutive_corrects_desc_2AFC.
↪append(test_case_1)
            initially_above_2AFC.append(test_case_1)
            frequency_for_initially_above_2AFC.append(test_frequency)
            map_For_initially_above_2AFC.append({"frequency": test_frequency,␣
↪"user_response": "incorrect"})
            correct_count_initially_above_2AFC = 0


        else:
            correct_count_initially_above_2AFC += 1
            test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"correct"}
            initially_above_2AFC.append(test_case_1)
            frequency_for_initially_above_2AFC.append(test_frequency)
            map_For_initially_above_2AFC.append({"frequency": test_frequency,␣
↪"user_response": "correct"})
        return
    elif(correct_count_initially_above_2AFC == 0):


        test_frequency = last_response_of_selected_series["test_frequency"]
        if(last_response_of_selected_series["test_frequency"] < standard_freq):
            test_frequency = last_response_of_selected_series["test_frequency"]␣
↪+ step_size
        elif(last_response_of_selected_series["test_frequency"] >␣
↪standard_freq):
            test_frequency = last_response_of_selected_series["test_frequency"]␣
↪- step_size
        else:
            test_frequency = last_response_of_selected_series["test_frequency"]␣
↪- step_size


        correct_response, test_frequency, test_tone =␣
↪play_2AFC_stimuli(standard_freq, test_frequency)



        user_input = get_user_response_2AFC()

        trials_2AFC = trials_2AFC + 1

        if (user_input==correct_response):


            test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"correct"}
            initially_above_2AFC.append(test_case_1)
            frequency_for_initially_above_2AFC.append(test_frequency)
            map_For_initially_above_2AFC.append({"frequency": test_frequency,␣
↪"user_response": "correct"})


        else:

            test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"incorrect"}
            initially_above_2AFC.append(test_case_1)
            frequency_for_initially_above_2AFC.append(test_frequency)
            map_For_initially_above_2AFC.append({"frequency": test_frequency,␣
↪"user_response": "incorrect"})


def initially_below_2afc_trial():
    global correct_count_initially_below_2AFC
    global step_size
    global SPL
    global duration
    global standard_freq
    global trials_2AFC
    global initially_below_reversal_count_2AFC
    if(initially_below_2AFC ):
        last_response_of_selected_series = initially_below_2AFC[-1]
        if(last_response_of_selected_series['user_response']=="incorrect"):
           ␣
 ↪last_response_is_incorrect_asc_2AFC(last_response_of_selected_series)
            return
        elif(last_response_of_selected_series['user_response']=="correct"):
            if(correct_count_initially_below_2AFC  ==0):
                correct_count_initially_below_2AFC  = 1
            if(correct_count_initially_below_2AFC  >=1 and␣
 ↪correct_count_initially_below_2AFC <3):
                last_response_of_selected_series = initially_below_2AFC[-1]
               ␣
 ↪if(last_response_of_selected_series['user_response']=="correct"):
                   ␣
 ↪last_response_is_correct_asc_2AFC(last_response_of_selected_series)
            elif(correct_count_initially_below_2AFC >= 3):
                correct_count_initially_below_2AFC = 0
                last_response_of_selected_series = initially_below_2AFC[-1]
               ␣
 ↪if(last_response_of_selected_series['user_response']=="correct"):
                   ␣
 ↪last_response_is_correct_asc_2AFC(last_response_of_selected_series)

            return


    else:
        start_below_reference_2AFC()

def initially_above_2afc_trial():
    global correct_count_initially_above_2AFC
    global step_size
    global SPL
    global duration
    global standard_freq
    global trials_2AFC
    global initially_above_reversal_count_2AFC
    if(initially_above_2AFC ):
        last_response_of_selected_series = initially_above_2AFC[-1]
        if(last_response_of_selected_series['user_response']=="incorrect"):
                ␣
↪last_response_is_incorrect_desc_2AFC(last_response_of_selected_series)
            return
        elif(last_response_of_selected_series['user_response']=="correct"):
            if(correct_count_initially_above_2AFC  ==0):
                correct_count_initially_above_2AFC  = 1
            if(correct_count_initially_above_2AFC  >=1 and␣
↪correct_count_initially_above_2AFC <3):
                last_response_of_selected_series = initially_above_2AFC[-1]
                ␣
↪if(last_response_of_selected_series['user_response']=="correct"):
                    ␣
↪last_response_is_correct_desc_2AFC(last_response_of_selected_series)
            elif(correct_count_initially_above_2AFC >= 3):
                correct_count_initially_above_2AFC = 0
                last_response_of_selected_series = initially_above_2AFC[-1]
                ␣
↪if(last_response_of_selected_series['user_response']=="correct"):
                    ␣
↪last_response_is_correct_desc_2AFC(last_response_of_selected_series)

            return



    else:
        start_above_reference_2AFC()
```

[12]:
```
initially_below_3AFC = []
initially_above_3AFC = []
frequency_for_initially_below_3AFC = []
frequency_for_initially_above_3AFC = []
map_For_initially_below_3AFC = []
map_For_initially_above_3AFC = []
initially_below_reversal_count_3AFC = 0
reversals_for_initially_below_3AFC = []
reversals_for_initially_above_3AFC = []
initially_above_reversal_count_3AFC = 0
trials_3AFC = 0
correct_count_initially_below_3AFC = 0
correct_count_initially_above_3AFC = 0
incorrects_between_consecutive_corrects_asc_3AFC = []
incorrects_between_consecutive_corrects_desc_3AFC = []
def start_below_reference_3AFC():
    global step_size
    global SPL
    global duration
    global standard_freq
    global trials_3AFC

    initial_freq = 935
    correct_response, test_frequency, test_tone =␣
 ↪play_3AFC_stimuli(standard_freq, initial_freq)
    user_input = get_user_response_3AFC()
    trials_3AFC += 1
    if(user_input == correct_response):
        test_case = {"test_tone": test_tone, "test_frequency": test_frequency,␣
 ↪"user_response": "correct"}
        initially_below_3AFC.append(test_case)
        frequency_for_initially_below_3AFC.append(test_frequency)
        map_For_initially_below_3AFC.append({"frequency": test_frequency,␣
 ↪"user_response": "correct"})

    else:
        test_case = {"test_tone": test_tone, "test_frequency": test_frequency,␣
 ↪"user_response": "incorrect"}
        initially_below_3AFC.append(test_case)
        frequency_for_initially_below_3AFC.append(test_frequency)
        map_For_initially_below_3AFC.append({"frequency": test_frequency,␣
 ↪"user_response": "incorrect"})
def start_above_reference_3AFC():
    global step_size
    global SPL
    global duration
    global standard_freq
    global trials_3AFC

    initial_freq = 1065
    correct_response, test_frequency, test_tone =␣
 ↪play_3AFC_stimuli(standard_freq, initial_freq)
    user_input = get_user_response_3AFC()
    trials_3AFC += 1
    if(user_input == correct_response):
        test_case = {"test_tone": test_tone, "test_frequency": test_frequency,␣
 ↪"user_response": "correct"}
        initially_above_3AFC.append(test_case)
        frequency_for_initially_above_3AFC.append(test_frequency)
        map_For_initially_above_3AFC.append({"frequency": test_frequency,␣
 ↪"user_response": "correct"})

    else:
        test_case = {"test_tone": test_tone, "test_frequency": test_frequency,␣
 ↪"user_response": "incorrect"}
```

```
        initially_above_3AFC.append(test_case)
        frequency_for_initially_above_3AFC.append(test_frequency)
        map_For_initially_above_3AFC.append({"frequency": test_frequency,␣
↪"user_response": "incorrect"})

def last_response_is_incorrect_asc_3AFC(last_response_of_selected_series):
    global step_size
    global SPL
    global duration
    global standard_freq
    global trials_3AFC
    global initially_below_reversal_count_3AFC
    global step_size_record_correct
    global step_size_record_incorrect
    global min_step_size
    global max_step_size
    test_frequency = last_response_of_selected_series["test_frequency"]



    if(last_response_of_selected_series["test_frequency"] < standard_freq):
        test_frequency = last_response_of_selected_series["test_frequency"] -␣
↪step_size
    elif(last_response_of_selected_series["test_frequency"] > standard_freq):
        test_frequency = last_response_of_selected_series["test_frequency"] +␣
↪step_size
    else:
        test_frequency = last_response_of_selected_series["test_frequency"] -␣
↪step_size











    correct_response, test_frequency, test_tone =␣
↪play_3AFC_stimuli(standard_freq, test_frequency)

    trials_3AFC = trials_3AFC + 1
    user_input = get_user_response_3AFC()

    if(user_input==correct_response):
        test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"correct"}
        initially_below_3AFC.append(test_case_1)
        frequency_for_initially_below_3AFC.append(test_frequency)
        map_For_initially_below_3AFC.append({"frequency": test_frequency,␣
↪"user_response": "correct"})

    else:

        test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"incorrect"}
        initially_below_3AFC.append(test_case_1)
        frequency_for_initially_below_3AFC.append(test_frequency)
        map_For_initially_below_3AFC.append({"frequency": test_frequency,␣
↪"user_response": "incorrect"})

def last_response_is_correct_asc_3AFC(last_response_of_selected_series):
    global correct_count_initially_below_3AFC
    global step_size
    global SPL
    global duration
    global standard_freq
    global trials_3AFC
    global initially_below_reversal_count_3AFC
    global step_size_record_correct
    global step_size_record_incorrect
    global min_step_size
    global max_step_size

    if(correct_count_initially_below_3AFC < 3 and␣
↪correct_count_initially_below_3AFC >=1):

        test_frequency = last_response_of_selected_series['test_frequency']
        correct_response, test_frequency, test_tone =␣
↪play_3AFC_stimuli(standard_freq, test_frequency)



        user_input = get_user_response_3AFC()

        trials_3AFC = trials_3AFC + 1

        if (user_input!=correct_response):
            test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"incorrect"}
            frequency_for_initially_below_3AFC.append(test_frequency)
            initially_below_3AFC.append(test_case_1)
            incorrects_between_consecutive_corrects_asc_3AFC.append(test_case_1)
            map_For_initially_below_3AFC.append({"frequency": test_frequency,␣
↪"user_response": "incorrect"})

            correct_count_initially_below_3AFC = 0


        else:

            correct_count_initially_below_3AFC += 1
            test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"correct"}
            initially_below_3AFC.append(test_case_1)
            frequency_for_initially_below_3AFC.append(test_frequency)
            map_For_initially_below_3AFC.append({"frequency": test_frequency,␣
↪"user_response": "correct"})
        return
    elif(correct_count_initially_below_3AFC == 0):




        test_frequency = last_response_of_selected_series["test_frequency"]



        if(last_response_of_selected_series["test_frequency"] < standard_freq):
            test_frequency = last_response_of_selected_series["test_frequency"]␣
↪+ step_size
        elif(last_response_of_selected_series["test_frequency"] >␣
↪standard_freq):
            test_frequency = last_response_of_selected_series["test_frequency"]␣
↪- step_size
        else:
            test_frequency = last_response_of_selected_series["test_frequency"]␣
↪+ step_size



        correct_response, test_frequency, test_tone =␣
↪play_3AFC_stimuli(standard_freq, test_frequency)
```

```
        user_input = get_user_response_3AFC()

        trials_3AFC = trials_3AFC + 1

        if (user_input==correct_response):

            test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"correct"}
            initially_below_3AFC.append(test_case_1)
            frequency_for_initially_below_3AFC.append(test_frequency)
            map_For_initially_below_3AFC.append({"frequency": test_frequency,␣
↪"user_response": "correct"})


        else:

            test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"incorrect"}

            initially_below_3AFC.append(test_case_1)
            frequency_for_initially_below_3AFC.append(test_frequency)
            map_For_initially_below_3AFC.append({"frequency": test_frequency,␣
↪"user_response": "incorrect"})


def initially_below_3afc_trial():
    global correct_count_initially_below_3AFC
    global step_size
    global SPL
    global duration
    global standard_freq
    global trials_3AFC
    global initially_below_reversal_count_3AFC
    if(initially_below_3AFC ):
        last_response_of_selected_series = initially_below_3AFC[-1]
        if(last_response_of_selected_series['user_response']=="incorrect"):
            ␣
↪last_response_is_incorrect_asc_3AFC(last_response_of_selected_series)
            return
        elif(last_response_of_selected_series['user_response']=="correct"):
            if(correct_count_initially_below_3AFC  ==0):
                correct_count_initially_below_3AFC  = 1
            if(correct_count_initially_below_3AFC  >=1 and␣
↪correct_count_initially_below_3AFC <3):
```

```
                last_response_of_selected_series = initially_below_3AFC[-1]
                ␣
↪if(last_response_of_selected_series['user_response']=="correct"):
                    ␣
↪last_response_is_correct_asc_3AFC(last_response_of_selected_series)
            elif(correct_count_initially_below_3AFC >= 3):
                correct_count_initially_below_3AFC = 0
                last_response_of_selected_series = initially_below_3AFC[-1]
                ␣
↪if(last_response_of_selected_series['user_response']=="correct"):
                    ␣
↪last_response_is_correct_asc_3AFC(last_response_of_selected_series)

            return


    else:
        start_below_reference_3AFC()

def last_response_is_incorrect_desc_3AFC(last_response_of_selected_series):
    #user unable to perceive the difference, make the difference go up, making␣
↪it easier to differnetiate
    global step_size
    global SPL
    global duration
    global standard_freq
    global trials_3AFC
    global initially_above_reversal_count_3AFC
    global step_size_record_correct
    global step_size_record_incorrect
    global min_step_size
    global max_step_size
    test_frequency = last_response_of_selected_series["test_frequency"]


    if(last_response_of_selected_series["test_frequency"] < standard_freq):
        test_frequency = last_response_of_selected_series["test_frequency"] -␣
↪step_size
    elif(last_response_of_selected_series["test_frequency"] > standard_freq):
        test_frequency = last_response_of_selected_series["test_frequency"] +␣
↪step_size
    else:
        test_frequency = last_response_of_selected_series["test_frequency"] +␣
↪step_size
```

```
    correct_response, test_frequency, test_tone =␣
↪play_3AFC_stimuli(standard_freq, test_frequency)

    trials_3AFC = trials_3AFC + 1
    user_input = get_user_response_3AFC()

    if(user_input==correct_response):

        test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"correct"}
        initially_above_3AFC.append(test_case_1)
        frequency_for_initially_above_3AFC.append(test_frequency)
        map_For_initially_above_3AFC.append({"frequency": test_frequency,␣
↪"user_response": "correct"})

    else:

        test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"incorrect"}
        initially_above_3AFC.append(test_case_1)
        frequency_for_initially_above_3AFC.append(test_frequency)
        map_For_initially_above_3AFC.append({"frequency": test_frequency,␣
↪"user_response": "incorrect"})

def last_response_is_correct_desc_3AFC(last_response_of_selected_series):
    global correct_count_initially_above_3AFC
    global step_size
    global SPL
    global duration
    global standard_freq
    global trials_3AFC
    global initially_above_reversal_count_3AFC
    global step_size_record_correct
    global step_size_record_incorrect
    global min_step_size
    global max_step_size

    if(correct_count_initially_above_3AFC < 3 and␣
↪correct_count_initially_above_3AFC >=1):

        test_frequency = last_response_of_selected_series['test_frequency']
        correct_response, test_frequency, test_tone =␣
↪play_3AFC_stimuli(standard_freq, test_frequency)
```

```
        user_input = get_user_response_3AFC()

        trials_3AFC = trials_3AFC + 1

        if (user_input!=correct_response):

            test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"incorrect"}
            incorrects_between_consecutive_corrects_desc_3AFC.
↪append(test_case_1)
            initially_above_3AFC.append(test_case_1)
            frequency_for_initially_above_3AFC.append(test_frequency)
            map_For_initially_above_3AFC.append({"frequency": test_frequency,␣
↪"user_response": "incorrect"})
            correct_count_initially_above_3AFC = 0


        else:
            #user is correct within corrects

            correct_count_initially_above_3AFC += 1
            test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"correct"}
            initially_above_3AFC.append(test_case_1)
            frequency_for_initially_above_3AFC.append(test_frequency)
            map_For_initially_above_3AFC.append({"frequency": test_frequency,␣
↪"user_response": "correct"})
        return
    elif(correct_count_initially_above_3AFC == 0):


        test_frequency = last_response_of_selected_series["test_frequency"]


        if(last_response_of_selected_series["test_frequency"] < standard_freq):
            test_frequency = last_response_of_selected_series["test_frequency"]␣
↪+ step_size
        elif(last_response_of_selected_series["test_frequency"] >␣
↪standard_freq):
            test_frequency = last_response_of_selected_series["test_frequency"]␣
↪- step_size
        else:
            test_frequency = last_response_of_selected_series["test_frequency"]␣
↪- step_size
```

```
        correct_response, test_frequency, test_tone =␣
↪play_3AFC_stimuli(standard_freq, test_frequency)



        user_input = get_user_response_3AFC()

        trials_3AFC = trials_3AFC + 1

        if (user_input==correct_response):


            test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"correct"}
            initially_above_3AFC.append(test_case_1)
            frequency_for_initially_above_3AFC.append(test_frequency)
            map_For_initially_above_3AFC.append({"frequency": test_frequency,␣
↪"user_response": "correct"})


        else:

            test_case_1 = {"test_tone": test_tone, "test_frequency":␣
↪test_frequency, "user_response":"incorrect"}
            initially_above_3AFC.append(test_case_1)
            frequency_for_initially_above_3AFC.append(test_frequency)
            map_For_initially_above_3AFC.append({"frequency": test_frequency,␣
↪"user_response": "incorrect"})


def initially_below_3afc_trial():
    global correct_count_initially_below_3AFC
    global step_size
    global SPL
    global duration
    global standard_freq
    global trials_3AFC
    global initially_below_reversal_count_3AFC
    if(initially_below_3AFC ):
        last_response_of_selected_series = initially_below_3AFC[-1]
        if(last_response_of_selected_series['user_response']=="incorrect"):
            ␣
↪last_response_is_incorrect_asc_3AFC(last_response_of_selected_series)
            return
        elif(last_response_of_selected_series['user_response']=="correct"):
            if(correct_count_initially_below_3AFC  ==0):
                correct_count_initially_below_3AFC  = 1
            if(correct_count_initially_below_3AFC  >=1 and␣
 ↪correct_count_initially_below_3AFC <3):
                last_response_of_selected_series = initially_below_3AFC[-1]
               ␣
 ↪if(last_response_of_selected_series['user_response']=="correct"):
                   ␣
 ↪last_response_is_correct_asc_3AFC(last_response_of_selected_series)
            elif(correct_count_initially_below_3AFC >= 3):
                correct_count_initially_below_3AFC = 0
                last_response_of_selected_series = initially_below_3AFC[-1]
               ␣
 ↪if(last_response_of_selected_series['user_response']=="correct"):
                   ␣
 ↪last_response_is_correct_asc_3AFC(last_response_of_selected_series)

            return


    else:
        start_below_reference_3AFC()

def initially_above_3afc_trial():
    global correct_count_initially_above_3AFC
    global step_size
    global SPL
    global duration
    global standard_freq
    global trials_3AFC
    global initially_above_reversal_count_3AFC
    if(initially_above_3AFC ):
        last_response_of_selected_series = initially_above_3AFC[-1]
        if(last_response_of_selected_series['user_response']=="incorrect"):
           ␣
 ↪last_response_is_incorrect_desc_3AFC(last_response_of_selected_series)
            return
        elif(last_response_of_selected_series['user_response']=="correct"):
            if(correct_count_initially_above_3AFC  ==0):
                correct_count_initially_above_3AFC  = 1
            if(correct_count_initially_above_3AFC  >=1 and␣
 ↪correct_count_initially_above_3AFC <3):
                last_response_of_selected_series = initially_above_3AFC[-1]
               ␣
 ↪if(last_response_of_selected_series['user_response']=="correct"):
```

```
                   ␣
↪last_response_is_correct_desc_3AFC(last_response_of_selected_series)
          elif(correct_count_initially_above_3AFC >= 3):
              correct_count_initially_above_3AFC = 0
              last_response_of_selected_series = initially_above_3AFC[-1]
               ␣
↪if(last_response_of_selected_series['user_response']=="correct"):
                   ␣
↪last_response_is_correct_desc_3AFC(last_response_of_selected_series)

          return


    else:
        start_above_reference_3AFC()
```

[13]:
```
def find_transition_points(frequency_array):
    # Initialize variables to track direction and transition points
    direction = None  # 'up', 'down', or None (initial state)
    transition_points = []

    # Iterate through the frequency array to find transitions
    for i in range(1, len(frequency_array)):
        current_frequency = frequency_array[i]
        previous_frequency = frequency_array[i - 1]

        if current_frequency > previous_frequency:
            new_direction = 'up'
        elif current_frequency < previous_frequency:
            new_direction = 'down'
        else:
            new_direction = direction  # Use the previous direction for␣
↪repeated values

        if direction is None:
            direction = new_direction
        elif direction != new_direction:
            # Direction changed, record the transition point
            transition_points.append(previous_frequency)
            direction = new_direction

    # Print the transition points
    return transition_points
```

[14]:
```
initially_below_reversal_count_2AFC = 0
initially_above_reversal_count_2AFC = 0
def run_2afc():
    global reversals_for_initially_above_2AFC
    global reversals_for_initially_below_2AFC
    global initially_above_reversal_count_2AFC
    global initially_below_reversal_count_2AFC
    choice = np.random.randint(0,2)
    if(choice == 0):
        initially_above_2afc_trial()
        reversals_for_initially_above_2AFC =␣
↪find_transition_points(frequency_for_initially_above_2AFC)
        initially_above_reversal_count_2AFC =␣
↪len(reversals_for_initially_above_2AFC)

    else:
        initially_below_2afc_trial()
        reversals_for_initially_below_2AFC =␣
↪find_transition_points(frequency_for_initially_below_2AFC)
        initially_below_reversal_count_2AFC =␣
↪len(reversals_for_initially_below_2AFC)
def run_2afc_experiment():

    global initially_below_reversal_count_2AFC
    global initially_above_reversal_count_2AFC
    while(initially_below_reversal_count_2AFC< 12 or␣
↪initially_above_reversal_count_2AFC < 12):
        run_2afc()
    print("End of Experiment 2AFC")
```

[15]:
```
initially_below_reversal_count_3AFC = 0
initially_above_reversal_count_3AFC = 0
def run_3afc():
    global reversals_for_initially_above_3AFC
    global reversals_for_initially_below_3AFC
    global initially_above_reversal_count_3AFC
    global initially_below_reversal_count_3AFC
    choice = np.random.randint(0,2)
    if(choice == 0):
        initially_above_3afc_trial()
        reversals_for_initially_above_3AFC =␣
↪find_transition_points(frequency_for_initially_above_3AFC)
        initially_above_reversal_count_3AFC =␣
↪len(reversals_for_initially_above_3AFC)

    else:
        initially_below_3afc_trial()
        reversals_for_initially_below_3AFC =␣
↪find_transition_points(frequency_for_initially_below_3AFC)
```

```
        initially_below_reversal_count_3AFC =␣
↪len(reversals_for_initially_below_3AFC)
def run_3afc_experiment():

    global initially_below_reversal_count_3AFC
    global initially_above_reversal_count_3AFC
    while(initially_below_reversal_count_3AFC < 12 or␣
↪initially_above_reversal_count_3AFC < 12):
            run_3afc()
    print("End of Experiment 3AFC")
```

[16]:
```
#Running the 2AFC experiment
```

[17]:
```
run_2afc_experiment()
```

```
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
```

```
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    9
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
```

```
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    9
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
```

```
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    9
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    9
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    9
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
```

```
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    9
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
```

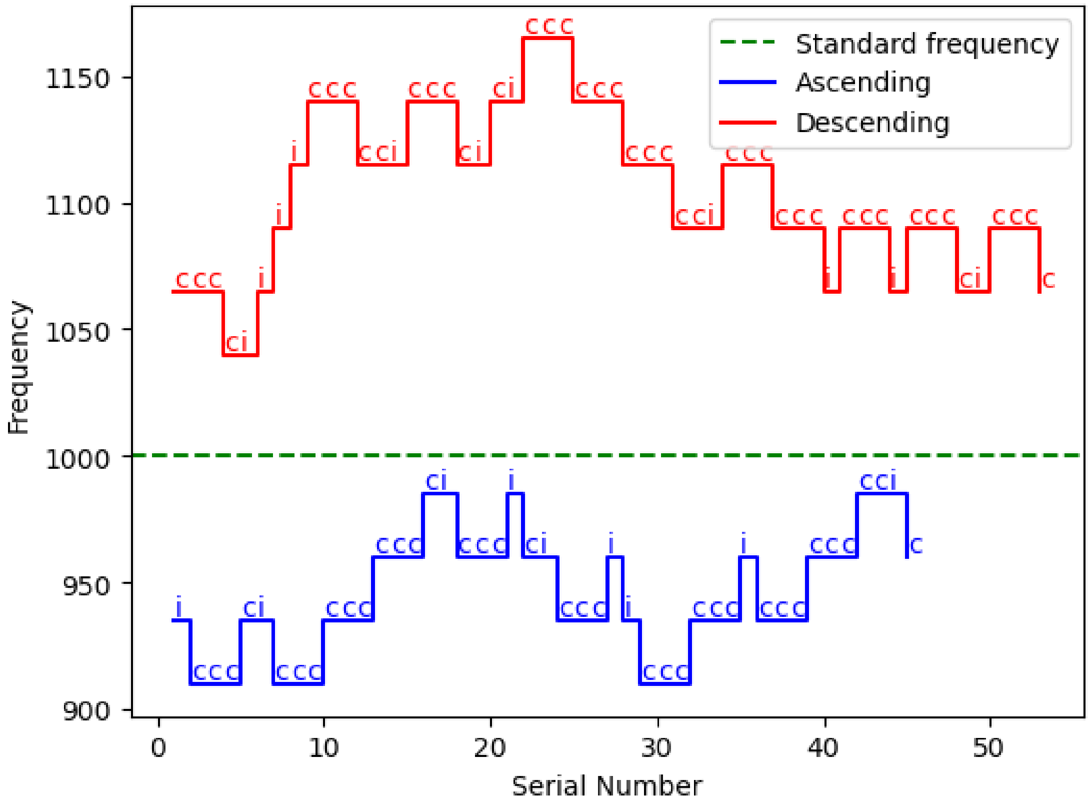
```
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
```

```
Which tone is greater? If the tones are equal, answer 9  user input:    9
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    1
Playing frequency 0
Playing frequency 1
Which tone is greater? If the tones are equal, answer 9  user input:    0
End of Experiment 2AFC
```

[18]:
```python
average_desc_2AFC = np.average(reversals_for_initially_above_2AFC[:12])
average_asc_2AFC = np.average(reversals_for_initially_below_2AFC[:12])
difference_threshold_2AFC = abs(average_desc_2AFC - average_asc_2AFC)/2
print("The difference threshold is\t"+str(difference_threshold_2AFC))
subfolder_path = '2AFC_Results'

if not os.path.exists(subfolder_path):
    os.makedirs(subfolder_path)

file_paths = [
    '2AFC_Ascending_Trials.json',
    '2AFC_Descending_Trials.json',
    '2AFC_Ascending_frequencies.json',
    '2AFC_Descedning_frequencies.json',
    '2AFC_Descedning_Transition_Points.json',
    '2AFC_Ascending_Transition_Points.json',
    'frequency_mapping_Ascending_2AFC.json',
    'frequency_mapping_Descending_2AFC.json',
    'total_number_of_trials_2AFC.json',
    'reversals_ascending_2AFC.json',
    'reversals_descending_2AFC.json',
    'difference_threshold_2AFC.json'
]
```

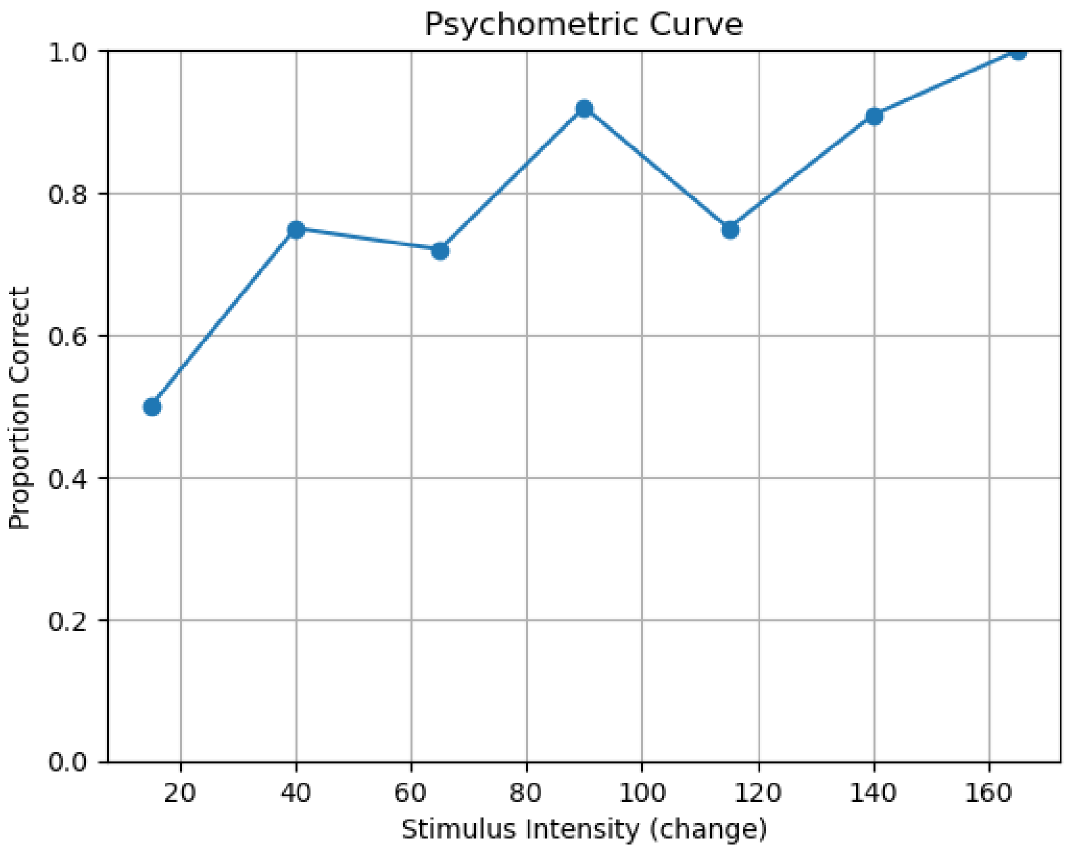
```python
for file_path, data in zip(file_paths, [
    initially_below_2AFC,
    initially_above_2AFC,
    frequency_for_initially_below_2AFC,
    frequency_for_initially_above_2AFC,
    reversals_for_initially_above_2AFC,
    reversals_for_initially_below_2AFC,
    map_For_initially_below_2AFC,
    map_For_initially_above_2AFC,
    trials_2AFC,
    reversals_for_initially_below_2AFC,
    reversals_for_initially_above_2AFC,
    difference_threshold_2AFC
]):
    with open(os.path.join(subfolder_path, file_path), 'w') as json_file:
        json.dump(data, json_file)

list1 = []
list2 = []
with open(os.path.join(subfolder_path, 'frequency_mapping_Ascending_2AFC.
 ↪json'), 'r') as json_file:
    list1 = json.load(json_file)
with open(os.path.join(subfolder_path, 'frequency_mapping_Descending_2AFC.
 ↪json'), 'r') as json_file:
    list2 = json.load(json_file)

frequencies1 = [entry["frequency"] for entry in list1]
labels1 = [entry["user_response"] for entry in list1]
frequencies2 = [entry["frequency"] for entry in list2]
labels2 = [entry["user_response"] for entry in list2]

index1 = range(1, len(frequencies1) + 1)
index2 = range(1, len(frequencies2) + 1)

fig, ax = plt.subplots()
plt.axhline(y=standard_freq, color='green', linestyle='--', label='Standard␣
 ↪frequency')

ax.step(index1, frequencies1, label="Ascending", where='post', color='blue')
ax.step(index2, frequencies2, label="Descending", where='post', color='red')


for i, (x, y, label) in enumerate(zip(index1, frequencies1, labels1)):
    ax.text(x, y, label[0], ha='left', va='bottom',color="blue")

for i, (x, y, label) in enumerate(zip(index2, frequencies2, labels2)):
    ax.text(x, y, label[0], ha='left', va='bottom', color="red")

ax.set_xlabel("Serial Number")
ax.set_ylabel("Frequency")
plt.legend(loc='upper right')
plt.show()
```

```
The difference threshold is     77.5
```



```
[19]: combined_list = list1+list2
      combined_list.sort(key=lambda x: x["frequency"])

      total_instances = []
      for values in combined_list:
          frequency = values["frequency"]
          response = values["user_response"]
          change = abs(values["frequency"] - standard_freq)

          if(response == "correct"):

              if(not value_exists(change,total_instances)):
```

```
            total_instances.append({"change": change, "instances_count": 1,␣
 ↪"correct_count": 1})

        elif(value_exists(change,total_instances)):
            for myvalue in total_instances:
                if(myvalue["change"] == change):
                    myvalue["correct_count"] = myvalue["correct_count"]+1
                    myvalue["instances_count"] = myvalue["instances_count"]+1

    else:
        if(not value_exists(change,total_instances)):
            total_instances.append({"change":change, "instances_count": 1,␣
 ↪"correct_count": 0})
        else:
            for myvalue in total_instances:
                if(myvalue["change"] == change):
                    myvalue["instances_count"] = myvalue["instances_count"]+1
total_instances.sort(key=lambda x: x["change"])
plt.plot([entry["change"] for entry in total_instances],␣
 ↪[entry["correct_count"]/entry["instances_count"] for entry in␣
 ↪total_instances], marker='o', linestyle='-')
plt.xlabel('Stimulus Intensity (change)')
plt.ylabel('Proportion Correct')
plt.title('Psychometric Curve')
plt.grid(True)
plt.ylim(0, 1)  # Set the y-axis limits to [0, 1]
plt.show()
print(total_instances)
```



```
[{'change': 15, 'instances_count': 6, 'correct_count': 3}, {'change': 40,
'instances_count': 16, 'correct_count': 12}, {'change': 65, 'instances_count':
25, 'correct_count': 18}, {'change': 90, 'instances_count': 25, 'correct_count':
23}, {'change': 115, 'instances_count': 12, 'correct_count': 9}, {'change': 140,
'instances_count': 11, 'correct_count': 10}, {'change': 165, 'instances_count':
3, 'correct_count': 3}]
```

[20]:
```
run_3afc_experiment()
```

```
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    9
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
```

```
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
```

```
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
```

```
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    9
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
```

```
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    9
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
```

```
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
```

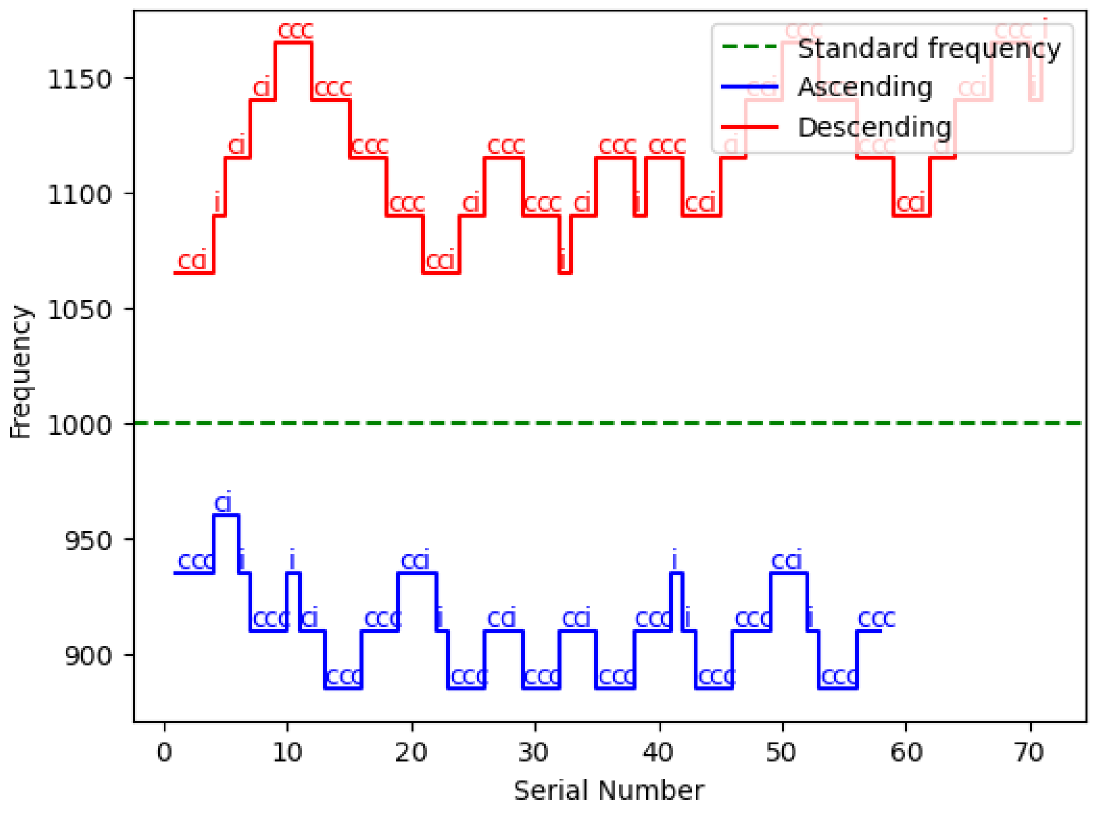
```
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
```

```
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
```

```
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    1
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    2
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
Playing frequency 0
Playing frequency 1
Playing frequency 2
Which tone contains test frequency? If all tones are equal, answer 9     user
input:    0
End of Experiment 3AFC
```

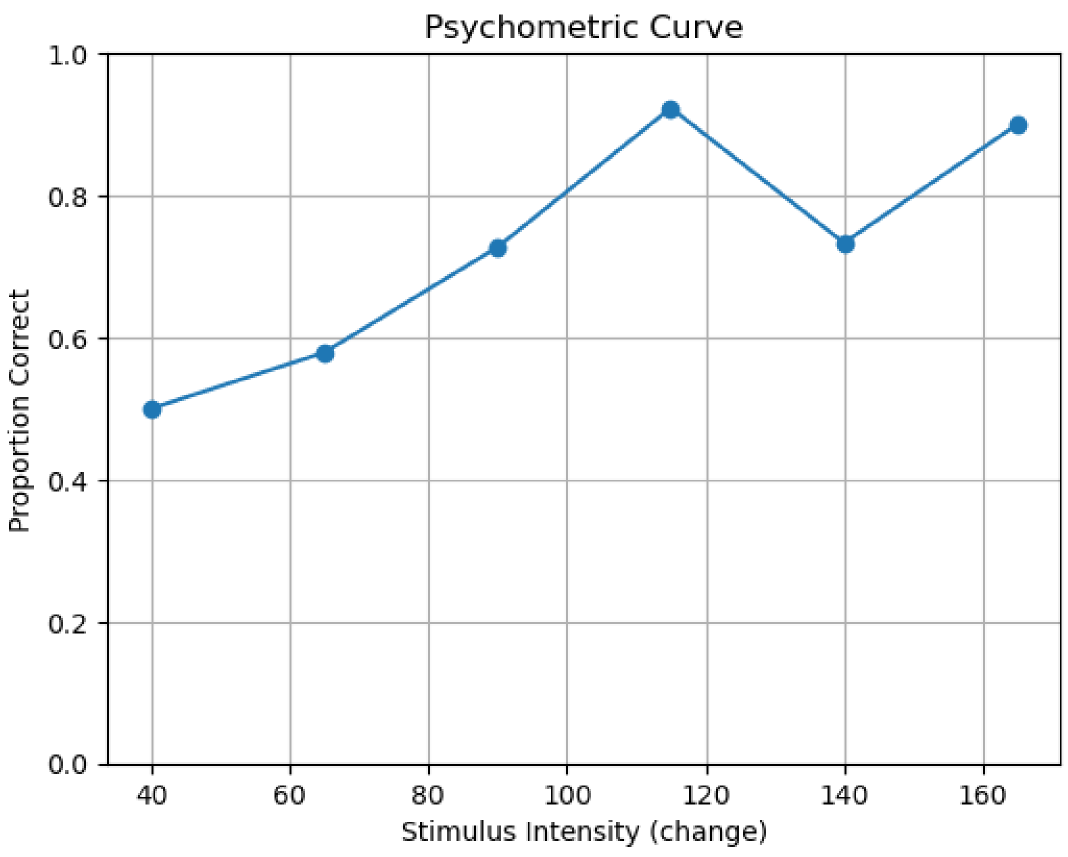
[21]:
```
average_desc_3AFC = np.average(reversals_for_initially_above_3AFC[:12])
average_asc_3AFC = np.average(reversals_for_initially_below_3AFC[:12])
difference_threshold_3AFC = abs(average_desc_3AFC - average_asc_3AFC)/2
print("The difference threshold is\t"+str(difference_threshold_3AFC))
subfolder_path = '3AFC_Results'

if not os.path.exists(subfolder_path):
    os.makedirs(subfolder_path)

file_paths = [
    '3AFC_Ascending_Trials.json',
    '3AFC_Descending_Trials.json',
    '3AFC_Ascending_frequencies.json',
    '3AFC_Descedning_frequencies.json',
    '3AFC_Descedning_Transition_Points.json',
    '3AFC_Ascending_Transition_Points.json',
    'frequency_mapping_Ascending_3AFC.json',
    'frequency_mapping_Descending_3AFC.json',
    'total_number_of_trials_3AFC.json',
    'reversals_ascending_3AFC.json',
    'reversals_descending_3AFC.json',
    'difference_threshold_3AFC.json'
]

for file_path, data in zip(file_paths, [
    initially_below_3AFC,
    initially_above_3AFC,
    frequency_for_initially_below_3AFC,
    frequency_for_initially_above_3AFC,
    reversals_for_initially_above_3AFC,
    reversals_for_initially_below_3AFC,
    map_For_initially_below_3AFC,
    map_For_initially_above_3AFC,
    trials_3AFC,
    reversals_for_initially_below_3AFC,
    reversals_for_initially_above_3AFC,
    difference_threshold_3AFC
]):
    with open(os.path.join(subfolder_path, file_path), 'w') as json_file:
        json.dump(data, json_file)

list1 = []
list2 = []
with open(os.path.join(subfolder_path, 'frequency_mapping_Ascending_3AFC.
 ↪json'), 'r') as json_file:
    list1 = json.load(json_file)
with open(os.path.join(subfolder_path, 'frequency_mapping_Descending_3AFC.
 ↪json'), 'r') as json_file:
    list2 = json.load(json_file)

frequencies1 = [entry["frequency"] for entry in list1]
labels1 = [entry["user_response"] for entry in list1]
frequencies2 = [entry["frequency"] for entry in list2]
labels2 = [entry["user_response"] for entry in list2]

index1 = range(1, len(frequencies1) + 1)
index2 = range(1, len(frequencies2) + 1)

fig, ax = plt.subplots()
plt.axhline(y=standard_freq, color='green', linestyle='--', label='Standard␣
 ↪frequency')

ax.step(index1, frequencies1, label="Ascending", where='post', color='blue')
ax.step(index2, frequencies2, label="Descending", where='post', color='red')


for i, (x, y, label) in enumerate(zip(index1, frequencies1, labels1)):
    ax.text(x, y, label[0], ha='left', va='bottom',color="blue")

for i, (x, y, label) in enumerate(zip(index2, frequencies2, labels2)):
    ax.text(x, y, label[0], ha='left', va='bottom', color="red")

ax.set_xlabel("Serial Number")
ax.set_ylabel("Frequency")
plt.legend(loc='upper right')
plt.show()
```

```
The difference threshold is     102.5
```



```
[22]: combined_list = list1+list2
      combined_list.sort(key=lambda x: x["frequency"])

      total_instances = []
      for values in combined_list:
          frequency = values["frequency"]
          response = values["user_response"]
          change = abs(values["frequency"] - standard_freq)

          if(response == "correct"):

              if(not value_exists(change,total_instances)):

                  total_instances.append({"change": change, "instances_count": 1,␣
       ↪"correct_count": 1})

              elif(value_exists(change,total_instances)):
                  for myvalue in total_instances:
                      if(myvalue["change"] == change):
                          myvalue["correct_count"] = myvalue["correct_count"]+1
                          myvalue["instances_count"] = myvalue["instances_count"]+1
```

```
    else:
        if(not value_exists(change,total_instances)):
            total_instances.append({"change":change, "instances_count": 1,␣
↪"correct_count": 0})
        else:
            for myvalue in total_instances:
                if(myvalue["change"] == change):
                    myvalue["instances_count"] = myvalue["instances_count"]+1
total_instances.sort(key=lambda x: x["change"])
plt.plot([entry["change"] for entry in total_instances],␣
↪[entry["correct_count"]/entry["instances_count"] for entry in␣
↪total_instances], marker='o', linestyle='-')
plt.xlabel('Stimulus Intensity (change)')
plt.ylabel('Proportion Correct')
plt.title('Psychometric Curve')
plt.grid(True)
plt.ylim(0, 1)  # Set the y-axis limits to [0, 1]
plt.show()
print(total_instances)
```



```
[{'change': 40, 'instances_count': 2, 'correct_count': 1}, {'change': 65,
'instances_count': 19, 'correct_count': 11}, {'change': 90, 'instances_count':
44, 'correct_count': 32}, {'change': 115, 'instances_count': 39,
'correct_count': 36}, {'change': 140, 'instances_count': 15, 'correct_count':
11}, {'change': 165, 'instances_count': 10, 'correct_count': 9}]
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: