```
#******************************************************************************
# "Earth Outer Core Radius"
# This Python program calculates the radius of the Earth's outer core using seismic waves
# from earthquakes.
# Written on March 30, 2019 by Veronica Sofia Parra.
#
******************************************************************************

# Data
# Station [Seismic Station, Latitude, Longitude, Earthquake Name, Date, Time (GMT),
#            Magnitude (Mw), Depth (km), Latitude, Longitude, Degrees, Distance (km),
#            S-Wave Recording]

station11 = ['Northview_High_School', 39.52, -87.17, 'S_of_Fiji_Islands', \
             'April_2_2018','GMT_055735', 6.1, 42.0, -24.719, -176.8865, \
            105.23, 11700.37, 'Y']
station12 = ['Eastern_Greene_High_School', 39.04, -86.74, 'S_of_Fiji_Islands', \
             'April_2_2018', 'GMT_055735', 6.1, 42.0, -24.719, -176.8865, \
            105.38, 11717.46, 'N']

station21 = ['State_Center', 41.91, -93.22, 'SE_of_Lata_Solomon_Islands', \
             'July_17_/2018', 'GMT_070253', 6.0, 38.0, -11.5936, 166.432, \
            105.38, 11717.38, 'Y']
station22 = ['MacAlester_College', 44.94, -93.17, 'SE_of_Lata_Solomon_Islands', \
             'July_17_/2018', 'GMT_070253', 6.0, 38.0, -11.5936, 166.432, 105.49, \
             11729.77, 'N']

station31 = ['LASA_Array', 46.69, -106.22, 'W_of Kandrian_Papua_New_Guinea', \
             'July_19_2018', 'GMT_183032', 6.0, 29.6, -6.1139, 148.7302, \
             104.75, 11646.91, 'Y']
station32 = ['Casper', 42.65, -106.52, 'W_of Kandrian_Papua_New_Guinea', \
             'July_19_2018', 'GMT_183032', 6.0, 29.6, -6.1139, 148.7302, \
             104.97, 11671.54, 'N']

station41 = ['Anaktuvuk_Pass', 68.13, -151.81, 'Central_Mid_Atlantic_Ridge', \
             'July_23_2018', 'GMT_103559', 6.0, 10.0, -0.2994, -19.252, \
             104.88, 11661.31, 'Y']
station42 = ['Knifeblade_Ridge', 69.16, -154.78, 'Central_Mid_Atlantic_Ridge', \
             'July_23_2018', 'GMT_103559', 6.0, 10.0, -0.2994, -19.252, \
             105.0, 11674.84, 'N']

station51 = ['Hockley', 39.96, -95.84, 'WNW_of_Ile_Hunter_New_Caledonia', \
             'September_10_2018', 'GMT_193137', 6.3, 12.0, -21.988, \
             170.1584, 104.07, 11571.44, 'Y']
station52 = ['Oklahoma_Geological_Survey_Observatory', 35.91, -95.79, \
             'WNW_of_Ile_Hunter_New_Caledonia', \
```

```
                    'September_10_2018', 'GMT_193137', 6.3, 12.0, -21.988, \
                    170.1584, 105.82, 11766.23, 'N']

station61 = ['Troy_Canyon', 38.35, -115.59, 'Drake_Passage', \
                    'October_29_2018', 'GMT_065421', 6.3, 10.0, -57.434, \
                    -66.3834, 104.30, 11597.61, 'Y']
station62 = ['Dugway', 40.19 , -112.81, 'Drake_Passage', \
                    'October_29_2018', 'GMT_065421', 6.3, 10.0, -57.434, \
                    -66.3834, 105.1, 11686.23, 'N']

station71 = ['Blacksburg', 37.21, -80.42, 'SSE_of_Pangai_Tonga', \
                    'November_10_2018', 'GMT_083321', 6.1, 35.0, -20.4538, \
                    -174.0081, 104.95, 11669.72, 'Y']
station72 = ['Nordonia_Hills_Middle_School', 41.32, -81.54, \
                    'SSE_of_Pangai_Tonga', 'November_10_2018', 'GMT_083321', 6.1, \
                    35.0, -20.4538, -174.0081, 105.13, 11689.69, 'N']

station81 = ['Blacksbury', 37.21, -80.42, \
                    'E_of_Visokoi_Island_South_Geogia_and_South_\
                    Sandwich_Islands', 'November_15_2018', \
                    'GMT_200222', 6.4, 15.0, -56.7065, -25.546, 104.71, \
                    11642.94, 'Y']
station82 = ['Tazewall', 36.54, -83.55, \
                    'E_of_Visokoi_Island_South_Geogia_and_South_\
                    Sandwich_Islands', 'November_15_2018', \
                    'GMT_200222', 6.4, 15.0, -56.7065, -25.546, 105.31, \
                    11709.34, 'N']

station91 = ['Erie', 42.12, -79.99, 'SE_of_Pacific_Rise', 'November_15_2018', \
                    'GMT_230901', 6.3, 10.0, -56.2363, -122.0441, 104.56, 11626.33, 'Y']
station92 = ['Minisink_Valley_Middle_School', 41.38, -74.52, \
                    'SE_of_Pacific_Rise', 'November_15_2018', 'GMT_230901', 6.3, \
                    10.0, -56.2363, -122.0441, 105.55, 11735.57, 'N']

station101 = ['Contact_Creek', 58.26, -155.89, 'SE_of_Easter_Island', \
                     'December_19_2018', 'GMT_013740', 6.2, 10.0, -36.118, \
                     -101.019, 104.88, 11661.60, 'Y']
station102 = ['Pilot_Point', 57.57, -157.57, 'SE_of_Easter_Island', \
                     'December_19_2018', 'GMT_013740', 6.2, 10.0, -36.118, \
                     -101.019, 105.00, 11674.46, 'N']

earthquake1 = (station11, station12)
earthquake2 = (station21, station22)
earthquake3 = (station31, station32)
earthquake4 = (station41, station42)
earthquake5 = (station51, station52)
```

```
earthquake6 = (station61, station62)
earthquake7 = (station71, station72)
earthquake8 = (station81, station82)
earthquake9 = (station91, station92)
earthquake10 = (station101, station102)

import numpy as np

# Verify data meets requirements.
# Criteria 1: Is the moment magnitude of the earthquake => 6 Mw?
if earthquake1 [0] [6] < 6.0 or earthquake1 [1] [6] < 6.0:
    print 'The earthquake has a moment magnitude less than 6 Mw.'
if earthquake2 [0] [6] < 6.0 or earthquake2 [1] [6] < 6.0:
    print 'The earthquake has a moment magnitude less than 6 Mw.'
if earthquake3 [0] [6] < 6.0 or earthquake3 [1] [6] < 6.0:
    print 'The earthquake has a moment magnitude less than 6 Mw.'
if earthquake4 [0] [6] < 6.0 or earthquake4 [1] [6] < 6.0:
    print 'The earthquake has a moment magnitude less than 6 Mw.'
if earthquake5 [0] [6] < 6.0 or earthquake5 [1] [6] < 6.0:
    print 'The earthquake has a moment magnitude less than 6 Mw.'
if earthquake6 [0] [6] < 6.0 or earthquake6 [1] [6] < 6.0:
    print 'The earthquake has a moment magnitude less than 6 Mw.'
if earthquake7 [0] [6] < 6.0 or earthquake7 [1] [6] < 6.0:
    print 'The earthquake has a moment magnitude less than 6 Mw.'
if earthquake8 [0] [6] < 6.0 or earthquake8 [1] [6] < 6.0:
    print 'The earthquake has a moment magnitude less than 6 Mw.'
if earthquake9 [0] [6] < 6.0 or earthquake9 [1] [6] < 6.0:
    print 'The earthquake has a moment magnitude less than 6 Mw.'
if earthquake10 [0] [6] < 6.0 or earthquake10 [1] [6] < 6.0:
    print 'The earthquake has a moment magnitude less than 6 Mw.'

# Criteria 2: Is the depth of the earthquake < 50 km?
if earthquake1 [0] [7] > 50.0 or earthquake1 [1] [7] > 50.0:
    print 'The earthquake had a depth greater than 50 km.'
if earthquake2 [0] [7] > 50.0 or earthquake2 [1] [7] > 50.0:
    print 'The earthquake had a depth greater than 50 km.'
if earthquake3 [0] [7] > 50.0 or earthquake3 [1] [7] > 50.0:
    print 'The earthquake had a depth greater than 50 km.'
if earthquake4 [0] [7] > 50.0 or earthquake4 [1] [7] > 50.0:
    print 'The earthquake had a depth greater than 50 km.'
if earthquake5 [0] [7] > 50.0 or earthquake5 [1] [7] > 50.0:
    print 'The earthquake had a depth greater than 50 km.'
if earthquake6 [0] [7] > 50.0 or earthquake6 [1] [7] > 50.0:
    print 'The earthquake had a depth greater than 50 km.'
if earthquake7 [0] [7] > 50.0 or earthquake7 [1] [7] > 50.0:
    print 'The earthquake had a depth greater than 50 km.'
```

```
if earthquake8 [0] [7] > 50.0 or earthquake8 [1] [7] > 50.0:
    print 'The earthquake had a depth greater than 50 km.'
if earthquake9 [0] [7] > 50.0 or earthquake9 [1] [7] > 50.0:
    print 'The earthquake had a depth greater than 50 km.'
if earthquake10 [0] [7] > 50.0 or earthquake10 [1] [7] > 50.0:
    print 'The earthquake had a depth greater than 50 km.'

# Criteria 3: Did the seismic stations measured an S-Wave?
if earthquake1 [0] [12] != 'Y' and earthquake1 [1] [12] != 'N':
    print 'The seismic stations did not measure the S-Wave Shadowing Zone.'
if earthquake2 [0] [12] != 'Y' and earthquake2 [1] [12] != 'N':
    print 'The seismic stations did not measure the S-Wave Shadowing Zone.'
if earthquake3 [0] [12] != 'Y' and earthquake3 [1] [12] != 'N':
    print 'The seismic stations did not measure the S-Wave Shadowing Zone.'
if earthquake4 [0] [12] != 'Y' and earthquake4 [1] [12] != 'N':
    print 'The seismic stations did not measure the S-Wave Shadowing Zone.'
if earthquake5 [0] [12] != 'Y' and earthquake5 [1] [12] != 'N':
    print 'The seismic stations did not measure the S-Wave Shadowing Zone.'
if earthquake6 [0] [12] != 'Y' and earthquake6 [1] [12] != 'N':
    print 'The seismic stations did not measure the S-Wave Shadowing Zone.'
if earthquake7 [0] [12] != 'Y' and earthquake7 [1] [12] != 'N':
    print 'The seismic stations did not measure the S-Wave Shadowing Zone.'
if earthquake8 [0] [12] != 'Y' and earthquake8 [1] [12] != 'N':
    print 'The seismic stations did not measure the S-Wave Shadowing Zone.'
if earthquake9 [0] [12] != 'Y' and earthquake9 [1] [12] != 'N':
    print 'The seismic stations did not measure the S-Wave Shadowing Zone.'
if earthquake10 [0] [12] != 'Y' and earthquake10 [1] [12] != 'N':
    print 'The seismic stations did not measure the S-Wave Shadowing Zone.'

# Criteria 4: Are both seismic stations within 500 km apart?
# Distance between the seismic stations is calculated using the equation of
#    a spherical earth projected to a plane.
# Earth’s radius is assumed to be 6378 km
beta = float(earthquake1 [0] [1] - earthquake1 [1] [1])
beta_radian = np.radians(beta)
mean_beta = float(earthquake1 [0] [1] + earthquake1 [1] [1])/2
mean_beta_radian = np.radians(mean_beta)
gamma = float(earthquake1 [0] [2] - earthquake1 [1] [2])
gamma_radian = np.radians(gamma)
station_distance = 6378 * np.sqrt((beta_radian)**2 + \
                                    (np.cos(mean_beta_radian) * \
                                    gamma_radian)**2)
if station_distance > 500:
print 'The distance between the seismic stations is greater than 500 km.'
beta = float(earthquake2 [0] [1] - earthquake2 [1] [1])
beta_radian = np.radians(beta)
```

```
mean_beta = float(earthquake2 [0] [1] + earthquake2 [1] [1])/2
mean_beta_radian = np.radians(mean_beta)
gamma = float(earthquake2 [0] [2] - earthquake2 [1] [2])
gamma_radian = np.radians(gamma)
station_distance = 6378 * np.sqrt((beta_radian)**2 + \
                                  (np.cos(mean_beta_radian) * \
                                  gamma_radian)**2)
if station_distance > 500:
    print 'The distance between the seismic stations is greater than 500 km.'
beta = float(earthquake3 [0] [1] - earthquake3 [1] [1])
beta_radian = np.radians(beta)
mean_beta = float(earthquake3 [0] [1] + earthquake3 [1] [1])/2
mean_beta_radian = np.radians(mean_beta)
gamma = float(earthquake3 [0] [2] - earthquake3 [1] [2])
gamma_radian = np.radians(gamma)
station_distance = 6378 * np.sqrt((beta_radian)**2 + \
                                  (np.cos(mean_beta_radian) * \
                                  gamma_radian)**2)
if station_distance > 500:
    print 'The distance between the seismic stations is greater than 500 km.'
beta = float(earthquake4 [0] [1] - earthquake4 [1] [1])
beta_radian = np.radians(beta)
mean_beta = float(earthquake4 [0] [1] + earthquake4 [1] [1])/2
mean_beta_radian = np.radians(mean_beta)
gamma = float(earthquake4 [0] [2] - earthquake4 [1] [2])
gamma_radian = np.radians(gamma)
station_distance = 6378 * np.sqrt((beta_radian)**2 + \
                                  (np.cos(mean_beta_radian) * \
                                  gamma_radian)**2)
if station_distance > 500:
    print 'The distance between the seismic stations is greater than 500 km.'
beta = float(earthquake5 [0] [1] - earthquake5 [1] [1])
beta_radian = np.radians(beta)
mean_beta = float(earthquake5 [0] [1] + earthquake5 [1] [1])/2
mean_beta_radian = np.radians(mean_beta)
gamma = float(earthquake5 [0] [2] - earthquake5 [1] [2])
gamma_radian = np.radians(gamma)
station_distance = 6378 * np.sqrt((beta_radian)**2 + \
                                  (np.cos(mean_beta_radian) * \
                                  gamma_radian)**2)
if station_distance > 500:
    print 'The distance between the seismic stations is greater than 500 km.'
beta = float(earthquake6 [0] [1] - earthquake6 [1] [1])
beta_radian = np.radians(beta)
mean_beta = float(earthquake6 [0] [1] + earthquake6 [1] [1])/2
mean_beta_radian = np.radians(mean_beta)
```

```
gamma = float(earthquake6 [0] [2] - earthquake6 [1] [2])
gamma_radian = np.radians(gamma)
station_distance = 6378 * np.sqrt((beta_radian)**2 + \
                                   (np.cos(mean_beta_radian) * \
                                   gamma_radian)**2)
if station_distance > 500:
    print 'The distance between the seismic stations is greater than 500 km.'
beta = float(earthquake7 [0] [1] - earthquake7 [1] [1])
beta_radian = np.radians(beta)
mean_beta = float(earthquake7 [0] [1] + earthquake7 [1] [1])/2
mean_beta_radian = np.radians(mean_beta)
gamma = float(earthquake7 [0] [2] - earthquake7 [1] [2])
gamma_radian = np.radians(gamma)
station_distance = 6378 * np.sqrt((beta_radian)**2 + \
                                   (np.cos(mean_beta_radian) * \
                                   gamma_radian)**2)
if station_distance > 500:
    print 'The distance between the seismic stations is greater than 500 km.'
beta = float(earthquake8 [0] [1] - earthquake8 [1] [1])
beta_radian = np.radians(beta)
mean_beta = float(earthquake8 [0] [1] + earthquake8 [1] [1])/2
mean_beta_radian = np.radians(mean_beta)
gamma = float(earthquake8 [0] [2] - earthquake8 [1] [2])
gamma_radian = np.radians(gamma)
station_distance = 6378 * np.sqrt((beta_radian)**2 + \
                                   (np.cos(mean_beta_radian) * \
                                   gamma_radian)**2)
if station_distance > 500:
    print 'The distance between the seismic stations is greater than 500 km.'
beta = float(earthquake9 [0] [1] - earthquake9 [1] [1])
beta_radian = np.radians(beta)
mean_beta = float(earthquake9 [0] [1] + earthquake9 [1] [1])/2
mean_beta_radian = np.radians(mean_beta)
gamma = float(earthquake9 [0] [2] - earthquake9 [1] [2])
gamma_radian = np.radians(gamma)
station_distance = 6378 * np.sqrt((beta_radian)**2 + \
                                   (np.cos(mean_beta_radian) * \
                                   gamma_radian)**2)
if station_distance > 500:
    print 'The distance between the seismic stations is greater than 500 km.'
beta = float(earthquake10 [0] [1] - earthquake10 [1] [1])
beta_radian = np.radians(beta)
mean_beta = float(earthquake10 [0] [1] + earthquake10 [1] [1])/2
mean_beta_radian = np.radians(mean_beta)
gamma = float(earthquake10 [0] [2] - earthquake10 [1] [2])
gamma_radian = np.radians(gamma)
```

```
station_distance = 6378 * np.sqrt((beta_radian)**2 + \
                        (np.cos(mean_beta_radian) * \
                        gamma_radian)**2)
if station_distance > 500:
    print 'The distance between the seismic stations is greater than 500 km.'

# Calculated initial radius of the Earth's outer core
# Earth's radius is assumed to be 6378 km
angle_theta = float
angle_theta = (earthquake1 [0] [10] + earthquake1 [1] [10])/2
angle_theta_radian = np.radians(angle_theta)
initial_radius = 6378 * np.cos(angle_theta_radian/2)
# Calculate correction for radius of Earth's outer core using Snell's Law
# At the critial angle, angle alpha = 90 degrees
# S-wave maximum velocities are V1 = 7.499 and V2 = 7.500 at outer core surface
V1 = 7.49
V2 = 7.50
angle_alpha = np.arcsin(V1/V2)
distance = (earthquake1 [0] [11] + earthquake1 [1] [11])/2
corr = (distance/2) / (np.tan(angle_alpha))
# Final calculated radius of Earth's outer core
radius = float(initial_radius - corr)
# Average Radius, Angle, and Distance
total_radius = float
total_radius = 0.0
total_radius = total_radius + radius
total_angle = float
total_angle = 0.0
total_angle = (total_angle + angle_theta)
total_distance = float
total_distance = 0.0
total_distance = (total_distance + distance)

angle_theta = (earthquake2 [0] [10] + earthquake2 [1] [10])/2
angle_theta_radian = np.radians(angle_theta)
initial_radius = 6378 * np.cos(angle_theta_radian/2)
# Calculate correction for radius of Earth's outer core using Snell's Law
# At the critial angle, angle_alpha = 90 degrees
# S-wave maximum velocities are V1 = 7.499 and V2 = 7.500 at outer core surface
V1 = 7.49
V2 = 7.50
angle_alpha = np.arcsin(V1/V2)
distance = (earthquake2 [0] [11] + earthquake2 [1] [11])/2
corr = (distance/2) / (np.tan(angle_alpha))
# Final calculated radius of Earth's outer core
radius = initial_radius - corr
```

```
# Average Radius, Angle, and Distance
total_radius = float(total_radius + radius)
total_angle = float(total_angle + angle_theta)
total_distance = float(total_distance + distance)

angle_theta = (earthquake3 [0] [10] + earthquake3 [1] [10])/2
angle_theta_radian = np.radians(angle_theta)
initial_radius = 6378 * np.cos(angle_theta_radian/2)
# Calculate correction for radius of Earth's outer core using Snell's Law
# At the critial angle, angle_alpha = 90 degrees
# S-Wave maximum velocities are V1 = 7.499 and V2 = 7.500 at outer core surface
V1 = 7.49
V2 = 7.50
angle_alpha = np.arcsin(V1/V2)
distance = (earthquake3 [0] [11] + earthquake3 [1] [11])/2
corr = (distance/2) / (np.tan(angle_alpha))
# Final calculated radius of Earth's outer core
radius = initial_radius - corr
# Average Radius, Angle, and Distance
total_radius = float(total_radius + radius)
total_angle = float(total_angle + angle_theta)
total_distance = float(total_distance + distance)

angle_theta = (earthquake4 [0] [10] + earthquake4 [1] [10])/2
angle_theta_radian = np.radians(angle_theta)
initial_radius = 6378 * np.cos(angle_theta_radian/2)
# Calculate correction for radius of Earth's outer core using Snell's Law
# At the critial angle, angle_alpha = 90 degrees
# S-wave maximum velocities are V1 = 7.499 and V2 = 7.500 at outer core surface
V1 = 7.49
V2 = 7.50
angle_alpha = np.arcsin(V1/V2)
distance = (earthquake4 [0] [11] + earthquake4 [1] [11])/2
corr = (distance/2) / (np.tan(angle_alpha))
# Final calculated radius of Earth's outer core
radius = initial_radius - corr
# Average Radius, Angle, and Distance
total_radius = float(total_radius + radius)
total_angle = float(total_angle + angle_theta)
total_distance = float(total_distance + distance)

angle_theta = (earthquake5 [0] [10] + earthquake5 [1] [10])/2
angle_theta_radian = np.radians(angle_theta)
initial_radius = 6378 * np.cos(angle_theta_radian/2)
# Calculate correction for radius of Earth's outer core using Snell's Law
# At the critial angle, angle_alpha = 90 degrees
```

```
# S-Wave maximum velocities are V1 = 7.499 and V2 = 7.500 at outer core surface
V1 = 7.49
V2 = 7.50
angle_alpha = np.arcsin(V1/V2)
distance = (earthquake5 [0] [11] + earthquake5 [1] [11])/2
corr = (distance/2) / (np.tan(angle_alpha))
# Final calculated radius of Earth's outer core
radius = initial_radius - corr
# Average Radius, Angle, and Distance
total_radius = float(total_radius + radius)
total_angle = float(total_angle + angle_theta)
total_distance = float(total_distance + distance)

angle_theta = (earthquake6 [0] [10] + earthquake6 [1] [10])/2
angle_theta_radian = np.radians(angle_theta)
initial_radius = 6378 * np.cos(angle_theta_radian/2)
# Calculate correction for radius of Earth's outer core using Snell's Law
# At the critial angle, angle_alpha = 90 degrees
# S-Wave maximum velocities are V1 = 7.499 and V2 = 7.500 at outer core surface
V1 = 7.49
V2 = 7.50
angle_alpha = np.arcsin(V1/V2)
distance = (earthquake6 [0] [11] + earthquake6 [1] [11])/2
corr = (distance/2) / (np.tan(angle_alpha))
# Final calculated radius of Earth's outer core
radius = initial_radius - corr
# Average Radius, Angle, and Distance
total_radius = float(total_radius + radius)
total_angle = float(total_angle + angle_theta)
total_distance = float(total_distance + distance)

angle_theta = (earthquake7 [0] [10] + earthquake7 [1] [10])/2
angle_theta_radian = np.radians(angle_theta)
initial_radius = 6378 * np.cos(angle_theta_radian/2)
# Calculate correction for radius of Earth's outer core using Snell's Law
# At the critial angle, angle_alpha = 90 degrees
# S-Wave maximum velocities are V1 = 7.499 and V2 = 7.500 at outer core surface
V1 = 7.49
V2 = 7.50
angle_alpha = np.arcsin(V1/V2)
distance = (earthquake7 [0] [11] + earthquake7 [1] [11])/2
corr = (distance/2) / (np.tan(angle_alpha))
# Final calculated radius of Earth's outer core
radius = initial_radius - corr
# Average Radius, Angle, and Distance
total_radius = float(total_radius + radius)
```

```
total_angle = float(total_angle + angle_theta)
total_distance = float(total_distance + distance)

angle_theta = (earthquake8 [0] [10] + earthquake8 [1] [10])/2
angle_theta_radian = np.radians(angle_theta)
initial_radius = 6378 * np.cos(angle_theta_radian/2)
# Calculate correction for radius of Earth's outer core using Snell's Law
# At the critial angle, angle_alpha = 90 degrees
# S-Wave maximum velocities are V1 = 7.499 and V2 = 7.500 at outer core surface
V1 = 7.49
V2 = 7.50
angle_alpha = np.arcsin(V1/V2)
distance = (earthquake8 [0] [11] + earthquake8 [1] [11])/2
corr = (distance/2) / (np.tan(angle_alpha))
# Final calculated radius of Earth's outer core
radius = initial_radius - corr
# Average Radius, Angle, and Distance
total_radius = float(total_radius + radius)
total_angle = float(total_angle + angle_theta)
total_distance = float(total_distance + distance)

angle_theta = (earthquake9 [0] [10] + earthquake9 [1] [10])/2
angle_theta_radian = np.radians(angle_theta)
initial_radius = 6378 * np.cos(angle_theta_radian/2)
# Calculate correction for radius of Earth's outer core using Snell's Law
# At the critial angle, angle_alpha = 90 degrees
# S-Wave maximum velocities are V1 = 7.499 and V2 = 7.500 at outer core surface
V1 = 7.49
V2 = 7.50
angle_alpha = np.arcsin(V1/V2)
distance = (earthquake9 [0] [11] + earthquake9 [1] [11])/2
corr = (distance/2) / (np.tan(angle_alpha))
# Final calculated radius of Earth's outer core
radius = initial_radius - corr
# Average Radius, Angle, and Distance
total_radius = float(total_radius + radius)
total_angle = float(total_angle + angle_theta)
total_distance = float(total_distance + distance)

angle_theta = (earthquake10 [0] [10] + earthquake10 [1] [10])/2
angle_theta_radian = np.radians(angle_theta)
initial_radius = 6378 * np.cos(angle_theta_radian/2)
# Calculate correction for radius of Earth's outer core using Snell's Law
# At the critial angle, angle_alpha = 90 degrees
# S-Wave maximum velocities are V1 = 7.499 and V2 = 7.500 at outer core surface
V1 = 7.49
```

```
V2 = 7.50
angle_alpha = np.arcsin(V1/V2)
distance = (earthquake10 [0] [11] + earthquake10 [1] [11])/2
corr = (distance/2) / (np.tan(angle_alpha))
# Final calculated radius of Earth's outer core
radius = initial_radius - corr
# Average Radius, Angle, and Distance
total_radius = float(total_radius + radius)
total_angle = float(total_angle + angle_theta)
total_distance = float(total_distance + distance)

# Results
ave_radius = float(total_radius/10)
ave_angle = float(total_angle/10)
ave_distance = float(total_distance/10)

# Print the results
    print 'The calculated average outer core radius is', ave_radius, 'km.'
    print 'The average angle from the epicenter to the start of the S-wave \
          shadowing zone is', ave_angle, 'degrees.'
    print 'The average distance from the epicenter to the start of the \
          S-wave shadowing zone is', ave_distance, 'km.'
```