Q4. Saying Random is not enough

4.1

1. *random_theta()*: The function finds random θ_1 and θ_2. θ is chosen randomly from degrees(ϕ) and then converted to radians.

$$\theta_1 = \phi_1.\frac{\pi}{180}$$
$$\theta_2 = \phi_2.\frac{\pi}{180}$$

2. *cord(R)*: The function takes the radius as the input and returns length of the cord between θ_1 and θ_2. It first calls *random_theta()*, and find the absolute value of their difference ($\theta = |\theta_1 - \theta_2|$). Then the length of the cord in calculated using the following formuls,

$$\text{cord length} = 2R.\sin(\theta/2)$$

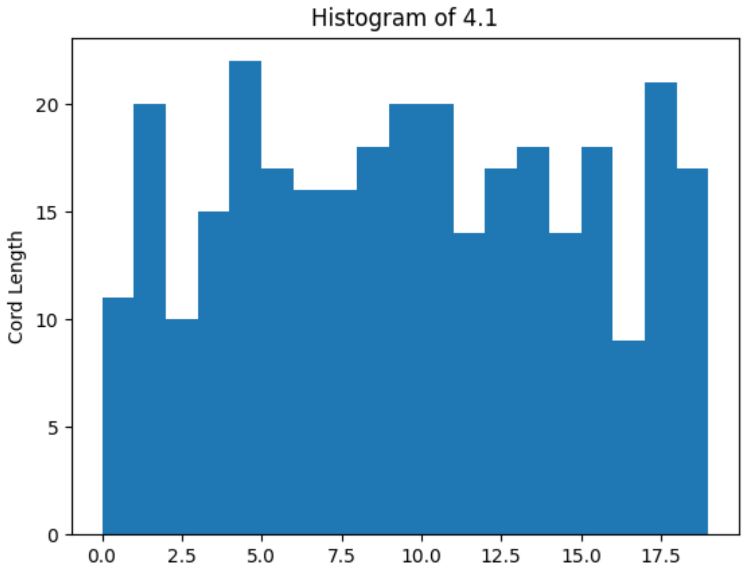
3. *find_cords1(R)*: The function takes radius R as input, then draw the histogram as the output. It iterates 1000 times, calls *cord(R)* for each iteration and append the result in *cord_len*. Then we calculate the value of appropriate bins for the histogram, and then plot it using matplotlib.

```


def find_bins(n, val_range):
    interval = m.sqrt(n)
    width_interval = val_range/interval
    return width_interval

# 4.1 -------------------------------------------------------------------------------------------
def random_theta():
    theta1 = np.random.uniform(0,360)*(m.pi/180)     #random theta 1
    theta2 = np.random.uniform(0,360)*(m.pi/180)     #random theta 2
    return (theta1,theta2)

def cord(R):
    angle = random_theta()
    theta = abs(angle[0] - angle[1])     #theta between the two radius(theta1 and theta2)
    l = 2*R*m.sin(theta/2)  #length of cord = 2rsin(theta/2)
    return(l)

def find_cords1(R):

    cord_len = []
    I = 100     #iterations

    for _ in range(I):
        cord_len.append(cord(R))
```

4.2

```
    #Finding bins
    rng = max(cord_len) - min(cord_len)
    width = int(find_bins(I, rng))

    #bin val list
    bin_lst = [x*width for x in range(int(m.sqrt(I)))]
    print(len(cord_len))

    arr=plt.hist(cord_len, bins = bin_lst)
    for i in range(int(m.sqrt(I))-1):
        plt.text(arr[1][i],arr[0][i],str(arr[0][i]))
    plt.title("Histogram of 4.1")
```

Figure 1: Histogram of 4.1

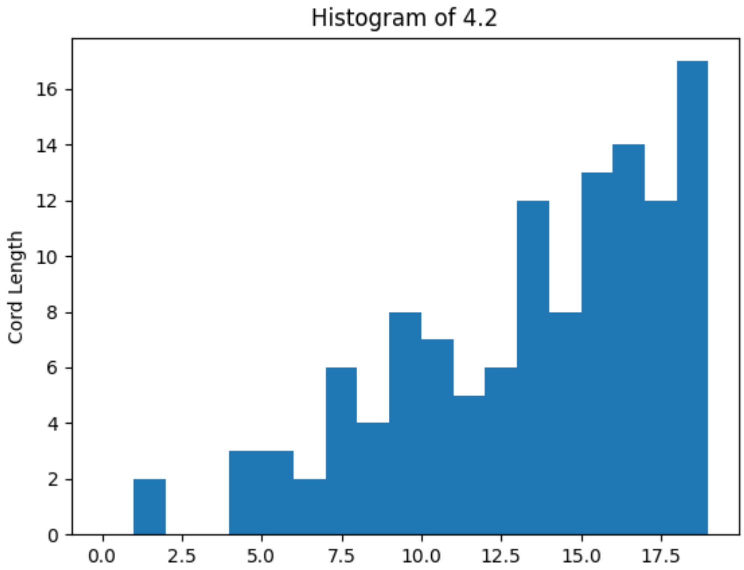



```
    plt.ylabel("Cord Length")
    plt.xlabel("bins")
    #plt.savefig("Q4/Q4(1).png")
    plt.show()

# 4.2
    ------------------------------------------------------------------------------------------

def random_cord(R):

    theta = np.random.uniform(0,360)*(m.pi/180)
    point_at_radius = np.random.uniform(0,R)    #point at R

    #cartesian cordinates at the picked point
    x = point_at_radius*m.cos(theta)
    y = point_at_radius*m.sin(theta)

    #finding base line from center to point_at_radius using distance formula
    base = m.sqrt(x**2+y**2)

    #perpendicular using pythagorean theorem, p = sqrt(h^2-b^2)
    perp = m.sqrt(R**2-base**2)

    #length of the cord
    l = 2*perp
```

4.3

```

    return l

def find_cords2(R):

    cord_len = []
    I = 1000    #iterations

    for _ in range(I):
        cord_len.append(random_cord(R))

    plt.hist(cord_len, bins = range(0,R))
    plt.title("Histogram of 4.2")
    plt.ylabel("Cord Length")
    plt.savefig("Q4/Q4(2).png")
    plt.show()
```

Figure 2: Histogram of 4.2




```

# 4.3 ------------------------------------------------------------------------------------------

def p_to_o(cord):
    return m.sqrt(cord[0]**2+cord[1]**2)

def random_point(R):

    x = np.random.uniform(-R,R)  #random point x
    y = np.random.uniform(-R,R)  #random point y

    return (x,y)

def cal_cord(R,pnt):
    #finding adjacent from the random point.
    adj = p_to_o(pnt)

    #length of opposite
    opp = m.sqrt(R**2-adj**2)

    #length of the cord
    l = 2*opp

    return l
```

Figure 3: Histogram of 4.3