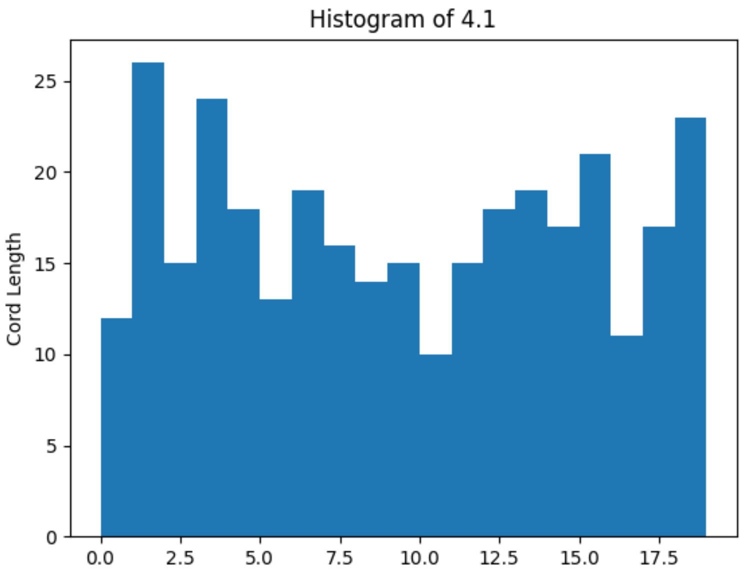
Q4. Saying Random is not enough

4.1

```python

# 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 = 1000    #iterations

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

    plt.hist(cord_len, bins = range(0,R))
    plt.title("Histogram of 4.1")
    plt.ylabel("Cord Length")
    plt.savefig("Q4/Q4(1).png")
    plt.show()
```

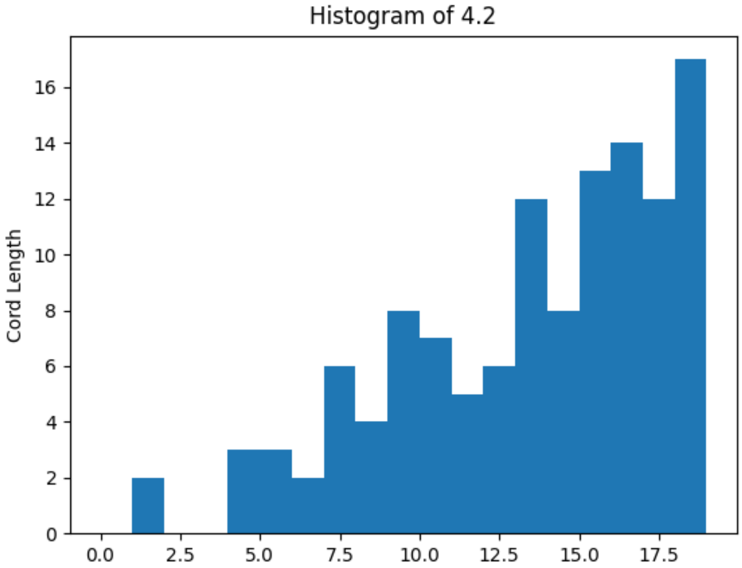
Figure 1: Histogram of 4.1



4.2

```python
# 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

    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

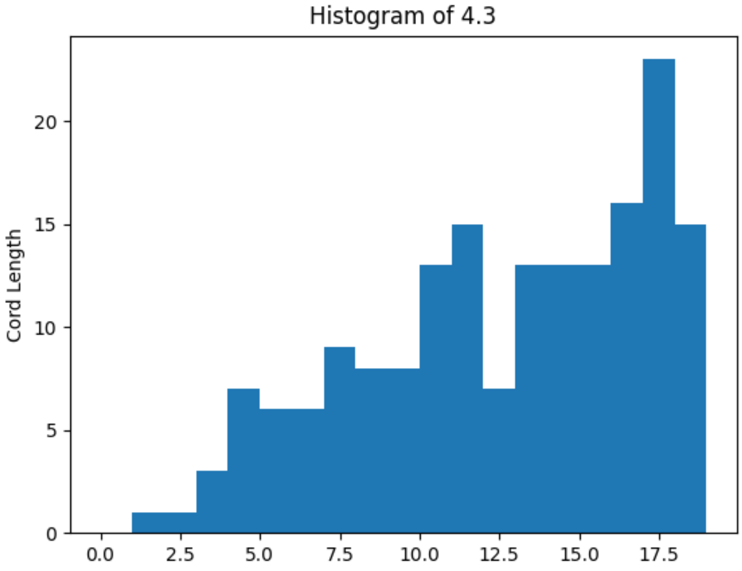


4.3

```
# 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

def find_cords3(R):

    I = 1000
    cord_len = []

    for a in range(I):
        point = random_point(R)
        if p_to_o(point) <= R:
            cord_len.append(cal_cord(R,point))
        else:
            a = a - 1

    plt.hist(cord_len, bins = range(0,R))
    plt.title("Histogram of 4.3")
    plt.ylabel("Cord Length")
    plt.savefig("Q4/Q4(3).png")
    plt.show()
```

Figure 3: Histogram of 4.3


Histogram of 4.3
Cord Length