SUNWAY

INT'L BUSINESS SCHOOL

Programme Name: BCS(Hons.)

Course Code: CSC 1201

Course Name: Computational Science

Assignment / Lab Sheet **/** Project / Case Study No.: 2

Date of Submission: 20th August 2022

Submitted By:

Student Name: Sharad Chandra Paudel

IUKL ID: 041902900074

Semester: 6th

Intake: September, 2019

Submitted To:

Faculty Name: Prakash Gautam

Department: School Works Pro

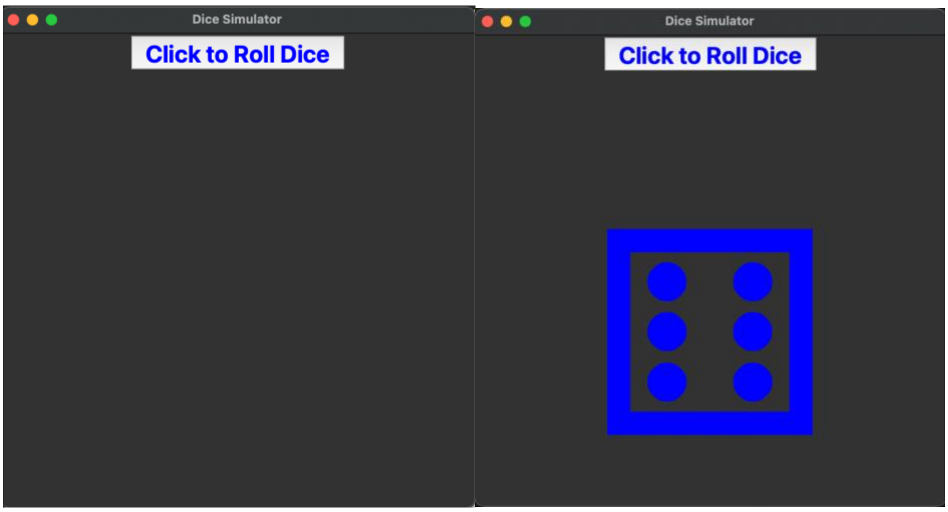
Do simulation of following random evenets and submit the pdf/ipynb file.

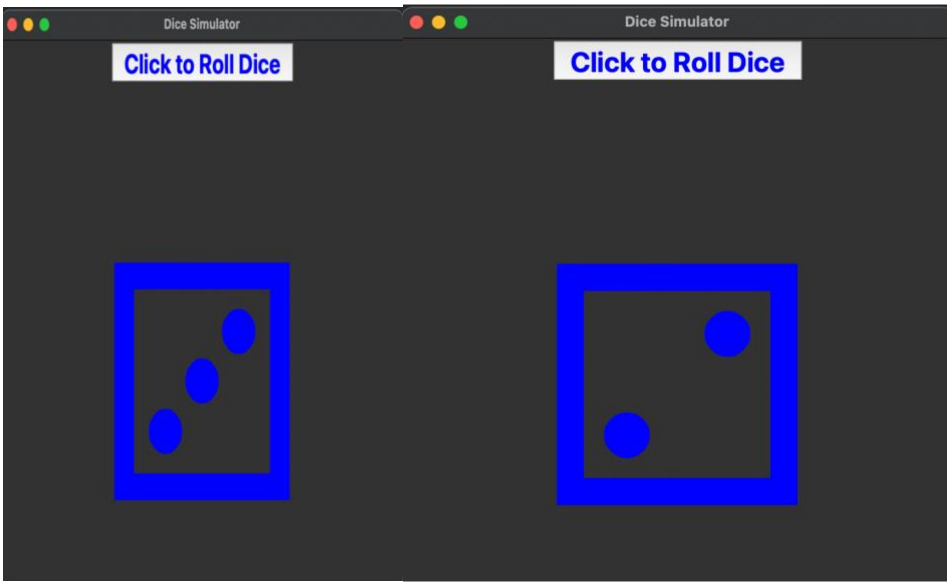
1. **Rolling a die**

Code:

```
from tkinter import *
import random
root = Tk()
root.title ("Dice Simulator")
root. geometry ("500x500")
label = Label (root , font = ("Helvitica", 400 , 'bold') , text = "", fg= 'blue')

def rolldice ():
    dice = ['\u2680', '\u2681' , '\u2682',  '\u2683', '\u2684' , '\u2685']
    label.configure(text = f'{random. choice (dice)}')
    label.pack()

button = Button(root , font = ("Helvitica" , 25 , 'bold'), text = "Click to Roll Dice", command = rolldice , bg =
'white' , fg = 'blue')
button.pack ()

root.mainloop()
```

OUTPUT:





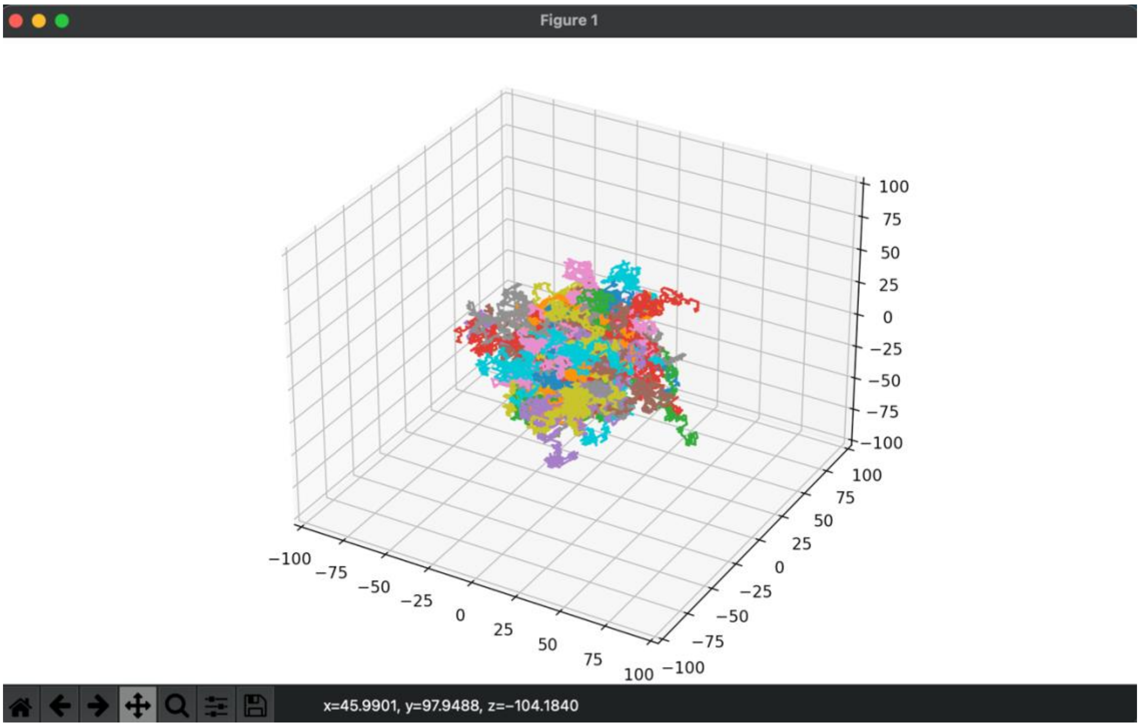
2. Random walk - Drunkerd's walk

Code:

```
import numpy as np
import matplotlib.pyplot as plt
import mpl_toolkits.mplot3d.axes3d as p3
import matplotlib.animation as animation

def path_generator(steps, step):
   path = np.empty((3, steps))
   for i in range(1, steps):
       x_ran, y_ran, z_ran = np.random.rand(3)
       sgnX = (x_ran - 0.5)/abs(x_ran - 0.5)
       sgnY = (y_ran - 0.5)/abs(y_ran - 0.5)
       sgnZ = (z_ran - 0.5)/abs(z_ran - 0.5)
       dis = np.array([step*sgnX, step*sgnY, step*sgnZ])
       path[:, i] = path[:, i - 1] + dis

   return path

fig = plt.figure()
ax = p3.Axes3D(fig)
particles = [path_generator(1000, 1) for i in range(100)]
trajectories = [ax.plot(particle[0, 0:1], particle[1, 0:1], particle[2, 0:1])[0] for particle in particles]

def animate(i):
   global particles, trajectories
```

```
    for trajectory, particle in zip(trajectories, particles):
        trajectory.set_data(particle[0:2, :i])
        trajectory.set_3d_properties(particle[2, :i])

    return trajectories

ax.set_xlim3d([-100, 100])
ax.set_ylim3d([-100, 100])
ax.set_zlim3d([-100, 100])


animacion = animation.FuncAnimation(fig, animate, 1000, interval=50, blit=False)
FFwriter = animation.FFMpegWriter()
plt.show()
```

OUTPUT:



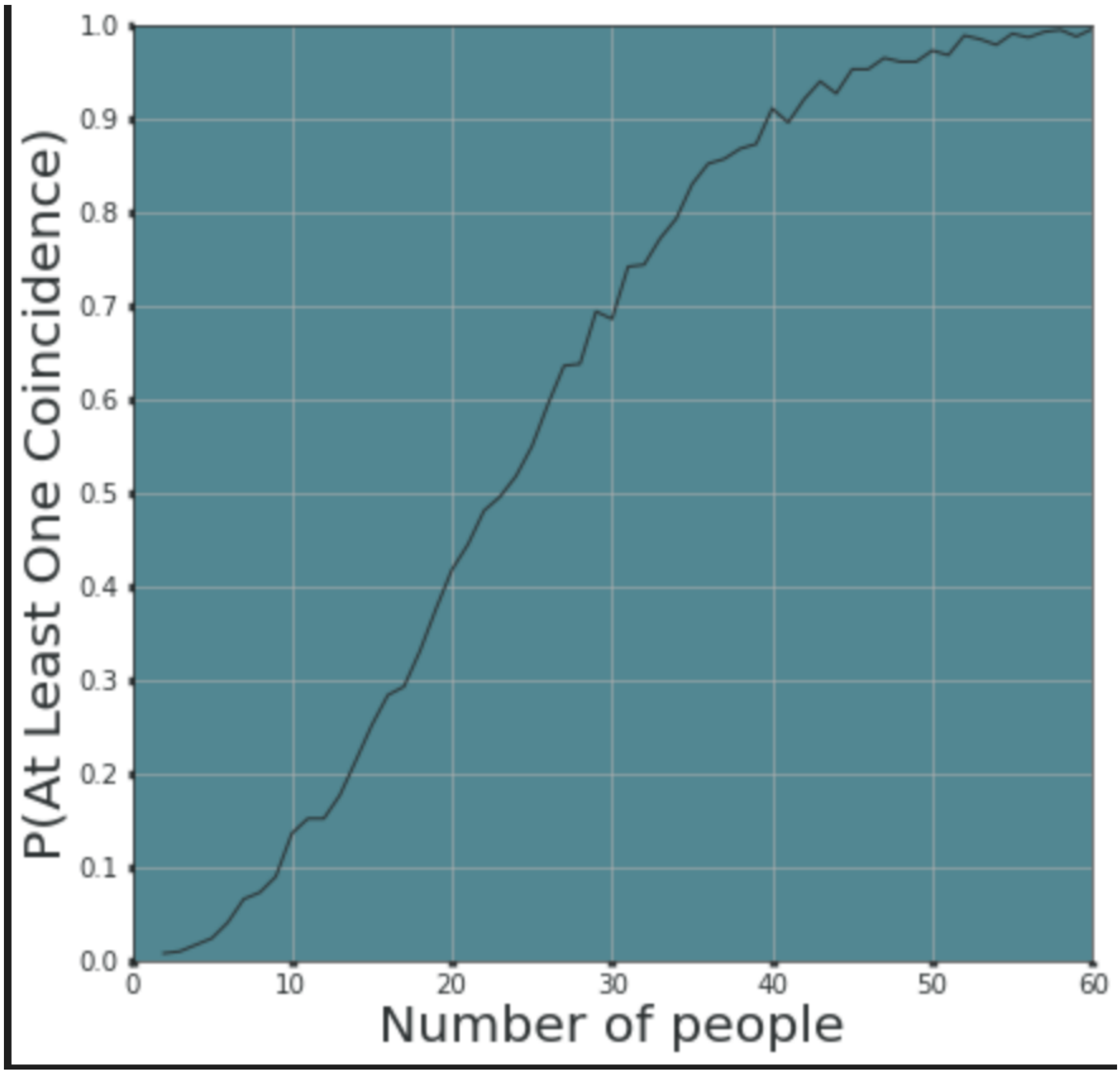
3. Birthday paradox

```
from random import randint

import matplotlib.pyplot as plt
import seaborn as sns

```

```python
MIN_NUM_PEOPLE = 2
MAX_NUM_PEOPLE = 60
NUM_POSSIBLE_BIRTHDAYS = 365
NUM_TRIALS = 1000


def generate_random_birthday():
    birthday = randint(1, NUM_POSSIBLE_BIRTHDAYS)
    return birthday


def generate_k_birthdays(k):
    birthdays = [generate_random_birthday() for _ in range(k)]
    return birthdays


def aloc(birthdays):
    unique_birthdays = set(birthdays)

    num_birthdays = len(birthdays)
    num_unique_birthdays = len(unique_birthdays)
    has_coincidence = (num_birthdays != num_unique_birthdays)

    return has_coincidence


def estimate_p_aloc(k):
    num_aloc = 0
    for _ in range(NUM_TRIALS):
        birthdays = generate_k_birthdays(k)
        has_coincidence = aloc(birthdays)
        if has_coincidence:
            num_aloc += 1

    p_aloc = num_aloc / NUM_TRIALS
    return p_aloc


def estimate_p_aloc_for_range(ks):
    k_probabilities = []
```

```
    for k in ks:
        p_aloc = estimate_p_aloc(k)
        k_probabilities.append(p_aloc)

    return k_probabilities



ks = range(MIN_NUM_PEOPLE, MAX_NUM_PEOPLE + 1)
k_probabilities = estimate_p_aloc_for_range(ks)



fig, ax = plt.subplots(figsize=(10, 10), dpi=49)
ax.set_facecolor('#518792')
ax.xaxis.set_tick_params(width=5, color='#2d3233')
ax.yaxis.set_tick_params(width=5, color='#2d3233')

sns.lineplot(x=ks, y=k_probabilities, color='#2d3233')

plt.xticks(fontsize=15, color='#2d3233')
y_range = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]
plt.yticks(y_range, fontsize=15, color='#2d3233')
plt.grid()
plt.xlim([0, 60])
plt.ylim([0, 1])
plt.xlabel('Number of people', fontsize=30, color='#2d3233')
plt.ylabel('P(At Least One Coincidence)', fontsize=30, color='#2d3233')

plt.show()
```

OUTPUT:



P(At Least One Coincidence)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
10
20
30
40
50
60
Number of people