


	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

		FORMATO DE INFORME DE PRÁCTICA DE LABORATORIO / TALLERES / CENTROS DE SIMULACIÓN – PARA ESTUDIANTES
CARRERA: Computación		ASIGNATURA: Simulación
NRO. PRÁCTICA:	TÍTULO PRÁCTICA: Simulación COVID-19	
OBJETIVO ALCANZADO: Comprensión de la simulación que se puede realizar sobre datos real para observar de manera rápida y eficiente como se pueden desarrollar las diferentes situaciones		
ACTIVIDADES DESARROLLADAS		
<p>1. Vamos a utilizar SIMPY para esta actividad</p> <p>Empezamos por importar las librerías necesarias</p> <pre># Imports import simpy import random import pandas as pd import numpy as np from matplotlib import pyplot as plt</pre>		
<p>2. Empezamos instanciando las variables iniciales</p> <pre># Instanciar Variables horas_diarias = 9 minutos_diarios = horas_diarias * 60 TIEMPO_TOTAL = minutos_diarios * 1 NUM_MEDICOS = 6 TIEMPO_VACUNACION = 5 #Hasta 10 INTERVALO_LLEGADA = 5 POST_VACUNA = 20 CERTIFICACION = 2 #Hasta 3 TOTAL_DIAS = 100 ESPERA_INICIAL = 6</pre> <p>Así mismo tendremos las variables que se utilizarán para almacenar los valores que pasan por la simulación</p> <pre>vacunados = 0 no_vacunados = 0 pacientes = 0 bandera = 1 proceso = pd.DataFrame(columns=['dia', 'pacientes', 'vacunados', 'no_vacunados'])</pre> <p>Tenemos contadores y un dataframe para mantener los valores respecto al día</p>		
<p>3. Creamos la clase principal</p> <p>Para instanciar esta clase necesitamos las variables principales, entre las principales tenemos el número de médicos que serán los recursos necesarios</p> <pre># Clase class Vacunacion(object): def __init__(self, environment, num_medicos, tiempo_vacunacion): self.env = environment self.medicos = simpy.Resource(environment, num_medicos) self.numero_graves = 0 self.numero_muertos = 0 self.falla = False env.process(self.revisar())</pre> <p>Así mismo crearemos las actividades necesarias, tales como la vacunación y su posterior revisión</p>		

```
def vacunar(self, paciente):
    complicaciones = random.randint(1,100)
    global vacunados
    global no_vacunados
    global bandera
    if(complicaciones<95):
        tVacunacion = random.randint(TIEMPO_VACUNACION, TIEMPO_VACUNACION+5)
        print("El paciente %s empieza la vacunación a la hora: %.2f." %(paciente, env.now))
        yield self.env.timeout(tVacunacion)
        vacunados+=1
        bandera = 1
    else:
        print("*** El %s No puede vacunarse" % (paciente))
        no_vacunados+=1
        bandera = 0

def revisar(self, paciente):
    global bandera
    print('Empieza espera: %s a la hora %.2f.' % (paciente, env.now))
    tCertificacion = random.randint(CERTIFICACION, CERTIFICACION+1)
    if bandera == 1:
        try:
            yield env.timeout(tCertificacion + POST_VACUNA)
            print("[%s] se retira vacunado y certificado a las %.2f." % (paciente, env.now))
        except simpy.Interrupt:
            print("[%s] tuvo complicaciones a las %.2f." % (paciente, env.now))
            self.numero_graves += 1
            self.falla = False
    else:
        print("[%s] se retira sin ser vacunado a las %.2f." % (paciente, env.now))
```

Entonces en el método *vacunar* pondremos las complicaciones que pueden existir, siendo el 95% de los pacientes que será transmitido a la vacunación, los demás serán remitidos a la salida. Así mismo cada paso será guardado en las variables.

El método de *revisar* será ejecutado después de la vacunación, se utilizará interrupciones para cuando existan complicaciones y hará que se retire.

```
def complicacion(self):
    while True:
        traslado_ambulancia = random.randint(5,15)
        atencion_medica = random.randint(300, 1440)
        alta = random.randint(60, 120)
        fallece = random.randint(1,10)

        yield self.env.timeout(traslado_ambulancia + atencion_medica + alta)
        if fallece == 1:
            self.numero_muertos += 1
            self.proceso.interrupt()
```

El método de complicación hará que se interrumpa el proceso

4. Procesos

```
# Procesos
def llegada_paciente(env, nombre, vacunacion):
    print("Llega paciente: %s a la hora %.2f." % (nombre, env.now))
    with vacunacion.medicos.request() as medico:
        yield medico
        #print('Empieza vacunación: %s a la hora %.2f.' % (nombre, env.now))
        yield env.process(vacunacion.vacunar(nombre))
        print('La vacunación de %s terminó a la hora: %.2f.'%(nombre, env.now))
        env.process(vacunacion.revisar(nombre))
        print("[%s] se retira a las %.2f." % (nombre, env.now))
```

El proceso de llegada de los pacientes recibirá parámetros como el nombre del paciente y la clase vacunación. El recurso de Médico será utilizado para simular un número limitado de médicos en la zona. Entonces entra el paciente y utiliza un médico.

```
def ejecutar_simulacion(env, num_medicos, tiempo_vacunacion, intervalo):
    global pacientes
    global ESPERA_INICIAL
    vacunacion = Vacunacion(env, num_medicos, tiempo_vacunacion)
    for i in range(ESPERA_INICIAL):
        env.process(llegada_paciente(env, 'Paciente-%d' % (i+1), vacunacion))

    while True:
        yield env.timeout(random.randint(intervalo-3, intervalo+2))
        i+=1
        env.process(llegada_paciente(env, 'Paciente-%d' % (i+1), vacunacion))
        pacientes = i+1
```

Se ejecuta la simulación llamando a los parámetros necesarios. Utilizando variables globales podremos seguir almacenando los valores de cada paso de la simulación. Aquí crearemos la clase vacunación, en la que pondremos nuestros parámetros iniciales y crearemos un inicio programado, en el que llegarán una cantidad inicial de pacientes. Posteriormente llegarán uno por uno cada cierto tiempo.

5. Simulación

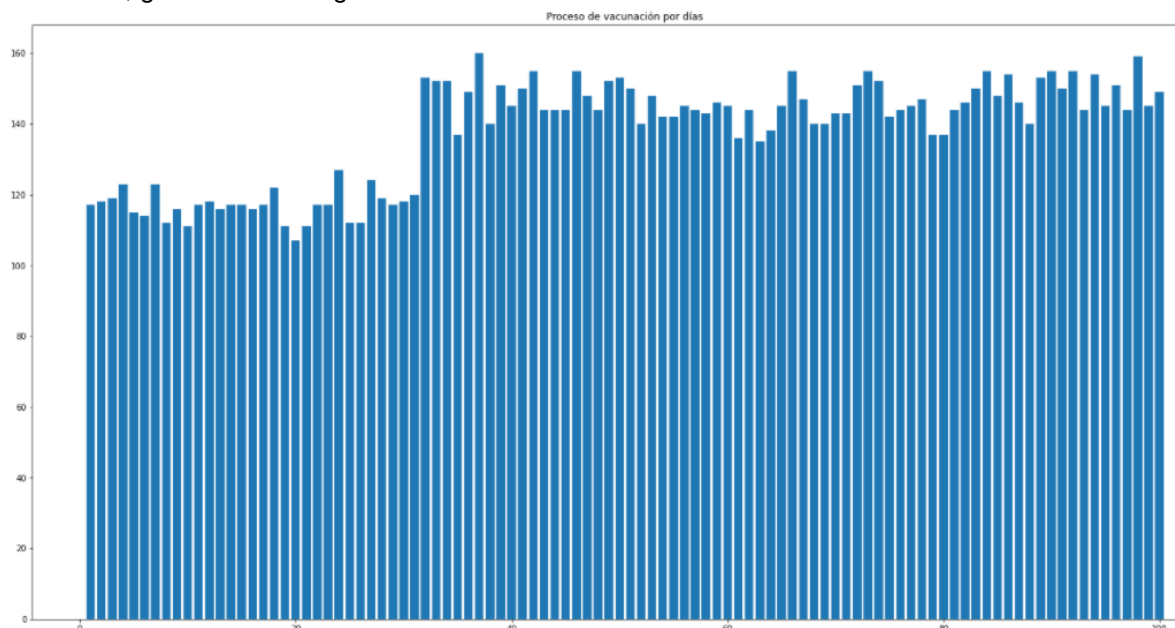
```
# Simulación
print("Iniciando vacunacion")
random.seed(76)

for i in range(TOTAL_DIAS):
    env = simpy.Environment()
    print("###Los médicos del día ", i, " son: ", NUM_MEDICOS)
    env.process(ejecutar_simulacion(env, NUM_MEDICOS, TIEMPO_VACUNACION, INTERVALO_LLEGADA))
    env.run(until=minutos_diarios)
    dia = [i+1, pacientes, vacunados, no_vacunados]
    proceso.loc[len(proceso)] = dia
    vacunados = 0
    no_vacunados = 0
    if i >= 30:
        NUM_MEDICOS = 10
        INTERVALO_LLEGADA = 4
    else:
        NUM_MEDICOS = 6
        INTERVALO_LLEGADA = 5
```

La simulación se realizará hasta que se cumpla la condición de tiempo, estará cumpliendo un día en horas y posteriormente se ejecutarán varios días hasta cumplir con la cantidad total de días. Cuando los días sean mayores a un cierto número los médicos aumentarán y el intervalo de llegada variará.

6. Gráfica

Por último, generamos una gráfica de la simulación con los resultados obtenidos



	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

RESULTADO(S) OBTENIDO(S): Conocimiento sobre la simulación que se puede realizar utilizando herramientas de Python.

CONCLUSIONES: Con herramientas de este tipo podremos efectuar simulaciones precisas y útiles para la visualización de diferentes ambientes

Nombre de los estudiantes: Alejandro Enríquez