**1IEE14 - Laboratorio 12**

- Materiales permitidos: Wiki del curso, apuntes de clase, consultar foros, tutoriales o documentación de python online.
- Está prohibido el uso de cualquier modelo de lenguaje como ChatGPT o alguno similar. A cualquier alumno que se le detecte que ha consultado un modelo de lenguaje se le pondrá nota 0(cero) en el laboratorio.
- Usted debe subir a Paideia 1 solo archivo comprimido (.zip o .rar) con el nombre L12_CODIGOPUCP.zip o L12_CODIGOPUCP.rar. Este archivo comprimido debe tener archivos de python(extensión .py) para cada pregunta. No se aceptarán soluciones en Jupyter notebook.
- El horario máximo permitido para subir el archivo es hasta las 10:00:00 pm. Pasada esa hora, habrá una penalidad de 2 puntos por cada minuto extra que se demore en entregar su archivo.

**Pregunta 1 (5 puntos)**

Escriba un programa en Python que calcule el factorial de un número tanto en serial como en paralelo. Para ello:

a) (1 punto) Escriba una función que reciba como parámetro de entrada el argumento n y retorne n! (Implementación serial)

No use recursividad porque para números grandes le saldrá error. Lo que tiene que hacer es iterar de 1 al n y en cada iteración ir acumulando el producto.

Imprima el tiempo de ejecución para n = 80 000

b) (3 puntos) Escriba una función que permita paralelizar el cálculo del factorial de un número. Use 2 procesos.

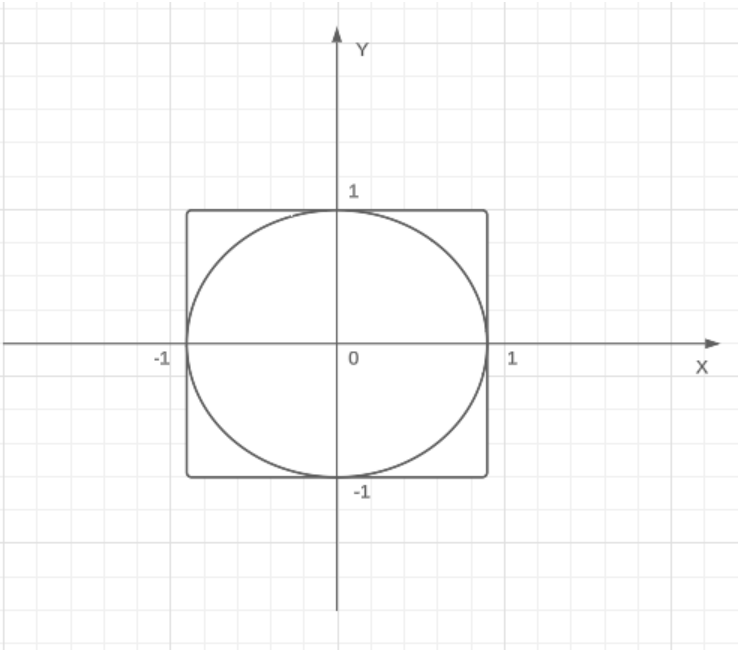
Imprima el tiempo de ejecución de esta función para n = 80 000. Calcule el Speedup.

c) (1 punto) Investigue sobre la función assert() y úsela al final de su archivo para que el programa corrobore que el resultado de la función de la parte a) es exactamente igual a la función de la parte b)

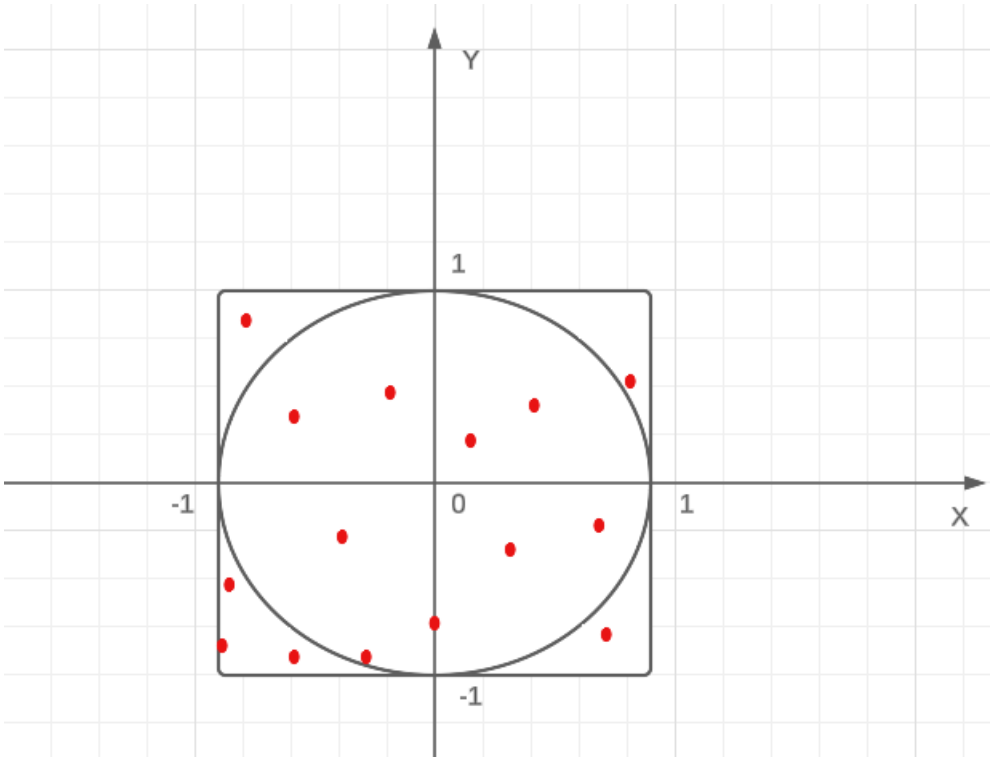
**Pregunta 2 (5 puntos)**

Calcule el valor de Pi usando el método de Montecarlo. A continuación se explica el procedimiento matemático y luego cómo hacer el algortimo en Python:

Explicación matemática: Para calcular el valor de Pi, imagine que se tiene un círculo de radio 1m dentro de un cuadrado cuyo lado mide 2m:




A continuación, se genera n puntos aleatorios dentro del cuadrado. Algunos puntos caerán dentro del círculo y otros fuera del círculo:



Como el radio del círculo es 1, entonces el área del círculo es:

$$Área\ de\ círculo = \pi * 1^2 = \pi$$

Asimismo, el área del círculo puede ser calculado de la siguiente manera:

$$Área\ de\ círculo = \frac{Número\ de\ muestras\ dentro\ del\ círculo}{Número\ de\ muestras\ totales}\ x\ 4$$

El motivo por el que se multiplica por 4 es para escalarlo al área del cuadrado. Como el cuadrado tiene lado 2, su área es 4.

Por tanto, Pi se puede aproximar de la siguiente manera:

$$Pi = \frac{Número\ de\ muestras\ dentro\ del\ círculo}{Número\ de\ muestras\ totales}\ x\ 4$$

Mientras más grande sea el número de muestras, más preciso será el valor de Pi calculado.

Pasos para implementar el algoritmo en Python:

- Defina un número alto de muestras. Por ejemplo, n = 10 000 000
- Haga un for loop para iterar. Por cada muestra:
  a) Genere una coordenada aleatoria. Puede usar la función random.uniform() para generar valores aleatorios entre -1 y 1.
  b) Si la coordenada generada de manera aleatoria está dentro del círculo, aumente en 1 un contador que almacene la cantidad de muestras que han caído dentro del círculo. **Nota**: Si un punto cae justo en el borde del círculo, también considérelo como si estuviera adentro del círculo.
- Terminado el bucle for, aplique la fórmula para calcular Pi en base a los datos obtenidos.

En base a la explicación dada, implemente lo siguiente:

a. (2.0 puntos) Cree un archivo llamado pregunta2a.py y calcule el valor de Pi usando el algoritmo que se le ha explicado. Use 10 000 000 muestras. Su programa debe imprimir el valor de Pi calculado.

b. (3.0 puntos) Cree un archivo llamado pregunta2b.py y use multiprocessing para paralelizar el algoritmo:

- Divida el cuadrado en 4 partes iguales
- Escriba una función que reciba como parámetro de entrada el área que va a procesar. Dentro de esa área, va a generar puntos aleatorios y va a seguir el mismo procedimiento que se hizo anteriormente.
- Use multiprocessing para llamar a esa función. Serán 4 procesos, cada proceso se ocupará de analizar solo su área asignada.
- Terminados los 4 procesos, junte los resultados obtenidos y obtenga el valor de Pi.

En esta pregunta, no hay necesidad de medir el tiempo de ejecución. Su programa debe imprimir el valor de Pi calculado.

**Pregunta 3 (5 puntos)**

El método de criptografía RSA, es un sistema que permitir encriptar información y protegerla antes de enviarla. Uno de sus pasos para encriptar consiste en usar un número natural lo suficientemente grande de tal manera que es difícil poder factorizarlos en sus números primos. Por ejemplo:

- Si se quiere factorizar 6, sabemos que la respuesta es 2 x 3. Calcular esto es muy fácil, por eso en un escenario real nunca se usaría un número tan pequeño.
- Si se quiere factorizar 667, la respuesta es 23 x 29. Esto es un poquito más complicado, pero con un programa de Python se puede hacer.
- Si se quiere factorizar 10403, la respuesta correcta es 101 x 103.

Como podrá observar, a medida que el número aumenta su cantidad de dígitos, se vuelve más complicado obtener los factores primos que generan ese número. Una aplicación real usa números que contienen entre 1024 – 4096 dígitos.

A usted se le pide escribir un programa en Python que obtenga los 2 factores primos que, cuando se multiplican, el resultado es el número X = 1000000016000000063

El algoritmo sugerido para obtener los 2 factores primos es el siguiente: Dado un número n, verificar si es divisible entre cualquier número natural dentro del intervalo:

$[\ 2\ ,\ \sqrt{n}\ ]$

a) (2 puntos) Escriba una función que reciba como parámetro de entrada un número *n* y retorne una lista con los dos números primos que componen el número ingresado. Use esta función para imprimir los 2 valores que componen el número X solicitado

b) (3 puntos) Escriba una función divida la tarea de la parte en a) en 2 procesos con el objetivo de que aprovechar 2 procesadores y la tarea acabe más rápido. Esta función debe retornar lo mismo que retorna la función hecha en la parte a)