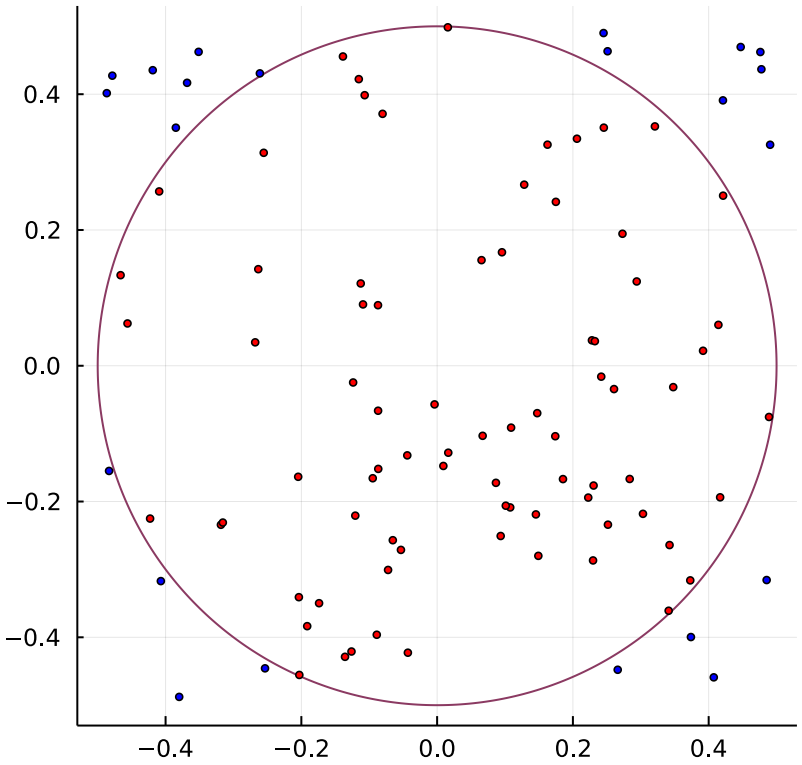
```
using Plots , Distributions , PlutoUI
```

Ejercicio 3

Escribe un algoritmo que estime el valor de π y que te permita visualizar algo similar al gráfico de la Figura 2, asegúrate de incluir el conteo del número de puntos rojos, número de puntos totales, y la respectiva estimación de π.

```
circunferencia (generic function with 1 method)
```

```
begin
    n=100                           #Cantidad de puntos
    x=rand(Uniform(-1/2, 1/2), 1, n) #Forma un arreglo aleatorio con n columnas y una
                                     #fila, esto en el rango -1/2 a 1/2
    y=rand(Uniform(-1/2, 1/2), 1, n) #Otro arreglo pero esta vez para y
    function circunferencia(r)       #Se define la función de la circunferencia
        theta= LinRange(0, 2*pi, 500)#Rango correspondiente
        r*sin.(theta),r*cos.(theta)  #Coordenadas de la circunferencia
    end
end
```



```
3.12
```

```
begin
    estimacion=4*sumin/n
end
```

```
("Con", 100, "puntos totales, la estimación de π es", 3.12, "Hay", 78, "puntos rojos", 22,
```

En promedio, ¿cuántos puntos necesitas generar para obtener una precisión de ±0.01?

Esto es una pregunta difícil de contestar, esto debido a la naturalidad azarosa de la construcción de las coordenadas, con mucha suerte incluso se podría obtener una estimación de 3.12 con 100 puntos, con 10,000 puntos 13/20 tuvo esa precisión, incluso podría decir que (con muy pequeña probabilidad) todos los puntos caigan fuera del círculo y la estimación termine siendo 0.

```
3.1424
```

```
aproxpi(10000)
```

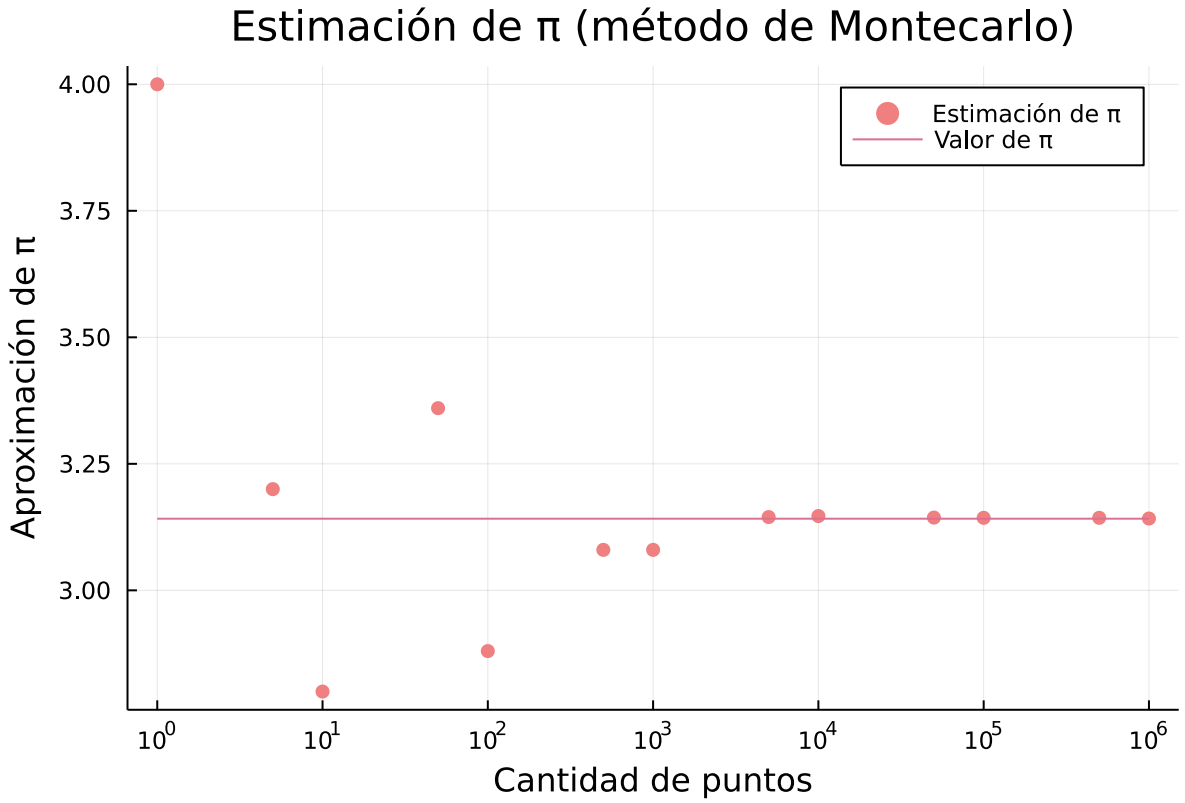
```
aproxpi (generic function with 1 method)
```

```
begin
function aproxpi(n)      #Función aproxpi dependiente de n, n es la cantidad de puntos
    X=rand(Uniform(-1/2, 1/2), 1, n)#Arreglo de una columna con 2 filas, entradas de
                                    # -1/2 a 1/2
    Y=rand(Uniform(-1/2, 1/2), 1, n)
    p = 4*count(X.^2 .+ Y.^2 .< 1/4)/n
    #P es la aproximación, cuenta si los puntos están dentro del círculo, divide
    entre el total de puntos y multiplica por 4
end
end
```

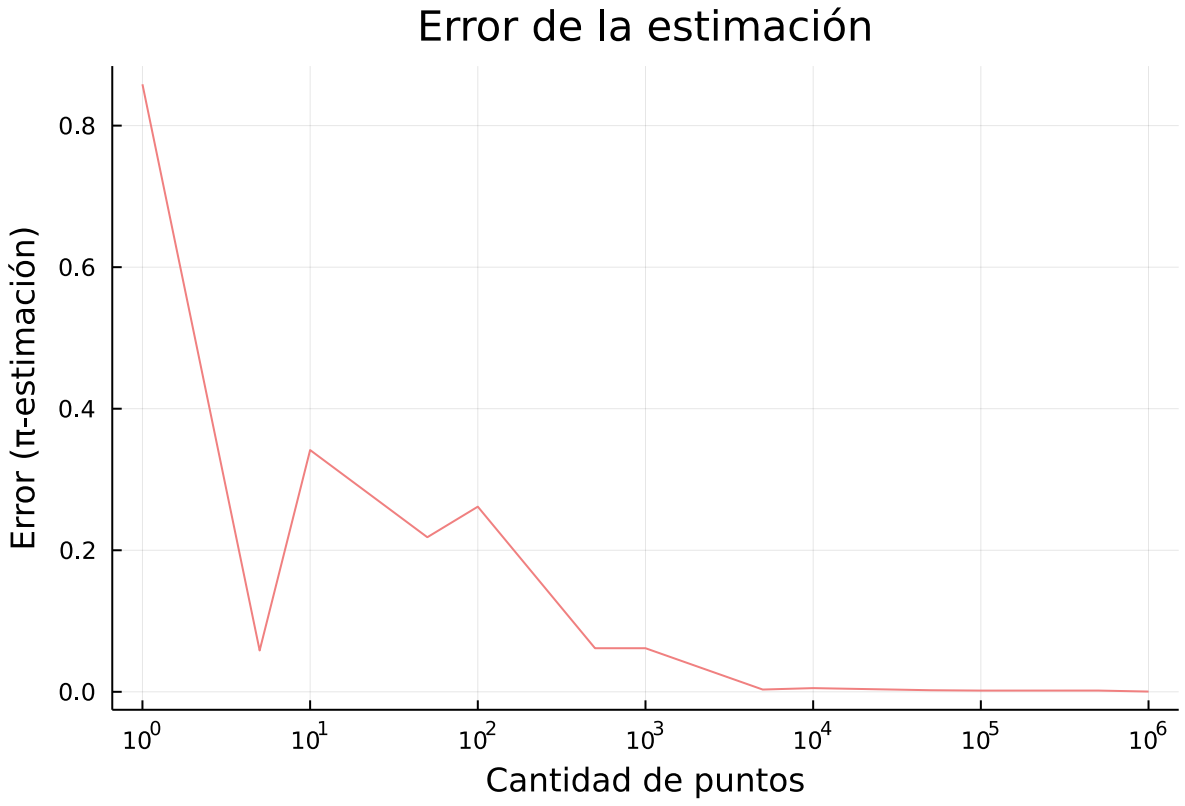
Realiza una gráfica del error de la estimación en función del número de puntos comparando contra el valor predeterminado de π de Julia

```
[0.858407, 0.0584073, 0.341593, 0.218407, 0.261593, 0.0615927, 0.0615927, 0.00320735, 0.00
```

```
begin
    A=[1,5,10,50,100,500,1000,5000,10000,50000,100000,500000,1000000] #Arreglo para
    observar los diferentes valores de la aproximación
    B=aproxpi.(A) #Aplicacamos la funcion aproxpi al vector A
    C=A./A.*pi   #No se nos ocurrió manera más elegante para graficar pi ):
    D=abs.(pi.-B)
end
```




```
begin
    scatter(A,B, xaxis=:log, xlabel = "Cantidad de puntos", ylabel = "Aproximación de π", color=:"lightcoral", markerstrokewidth=0, title="Estimación de π (método de Montecarlo)", label="Estimación de π") #Grafica la aproxpi de A, con escala logarítmica
    plot!(A,C, linecolor = :"palevioletred", label="Valor de π") #Valor real de Pi
end
```




```
plot(A,D, xaxis=:log, xlabel = "Cantidad de puntos", ylabel = "Error (π-estimación)", color=:"lightcoral", label=false, title="Error de la estimación") #Grafica del valor absoluto de la resta de pi menor la estimación con el método de montecarlo.
```