



TECNOLÓGICO
NACIONAL DE MÉXICO®



Métodos numéricos



Docente: M.M. Jorge Manuel Pool Cen

Actividad: BITÁCORA

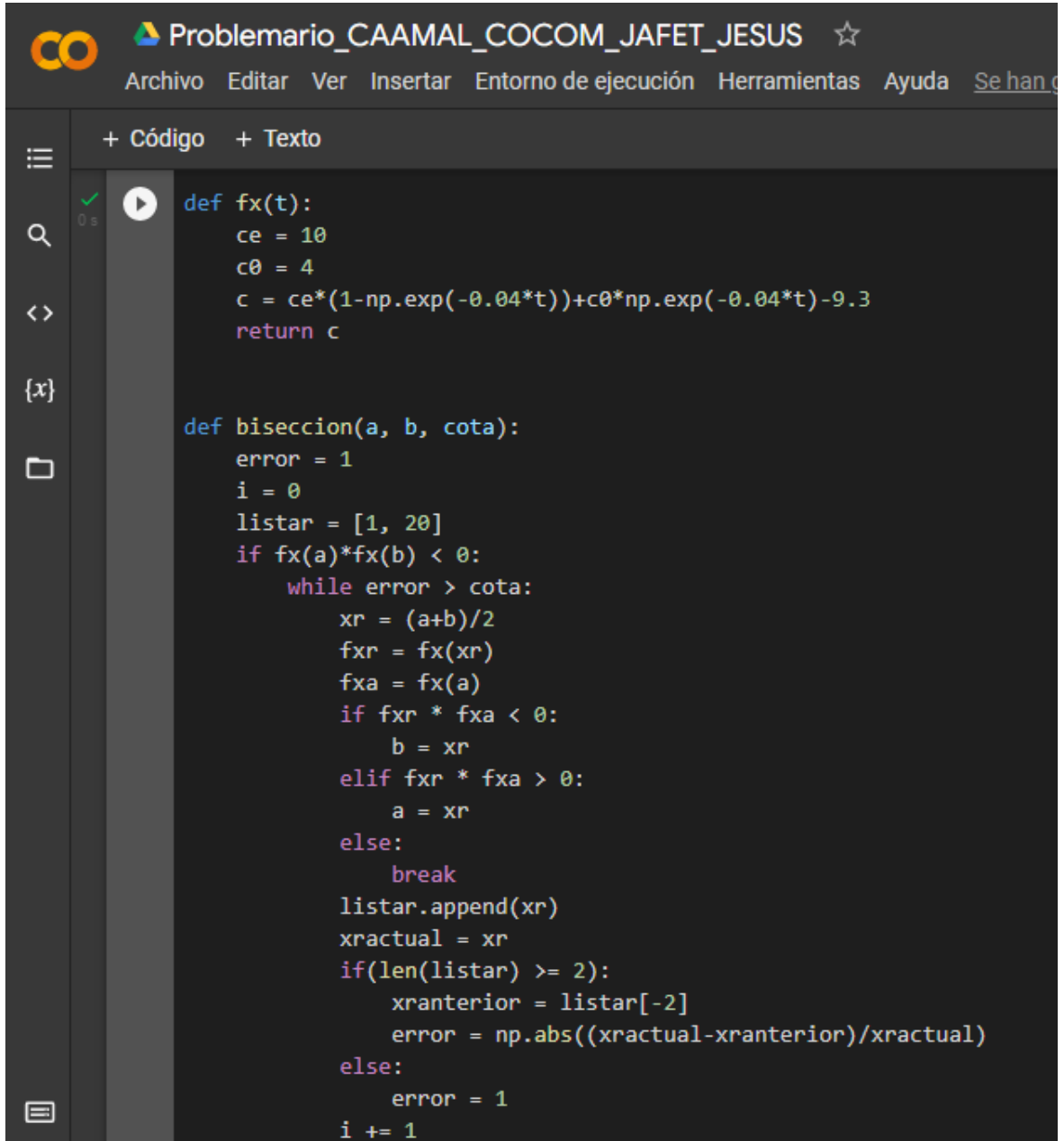
Alumno: Jafet Jesús Caamal Cocom

Matricula: 20070004

4° A

Fecha: 15/03/2021

EJERCICIO 1



The screenshot shows a Jupyter Notebook window titled "Problemario_CAAMAL_COCOM_JAFET_JESUS". The interface includes a top menu bar with options: Archivo, Editar, Ver, Insertar, Entorno de ejecución, Herramientas, Ayuda, and a link to "Se han c...". Below the menu bar, there are tabs for "+ Código" and "+ Texto". The left sidebar contains icons for a menu, search, expand/collapse, a variable {x}, and a file explorer. The main area displays Python code for a function `fx(t)` and a bisection method function `biseccion(a, b, cota)`. The `fx(t)` function calculates a value `c` based on `ce`, `c0`, and `t`. The `biseccion` function implements a bisection algorithm to find the root of `fx` within a given interval `[a, b]` and tolerance `cota`. It uses a list `listar` to store intermediate values and calculates the relative error.

```
def fx(t):
    ce = 10
    c0 = 4
    c = ce*(1-np.exp(-0.04*t))+c0*np.exp(-0.04*t)-9.3
    return c

def biseccion(a, b, cota):
    error = 1
    i = 0
    listar = [1, 20]
    if fx(a)*fx(b) < 0:
        while error > cota:
            xr = (a+b)/2
            fxr = fx(xr)
            fxa = fx(a)
            if fxr * fxa < 0:
                b = xr
            elif fxr * fxa > 0:
                a = xr
            else:
                break
        listar.append(xr)
        xractual = xr
        if(len(listar) >= 2):
            xranterior = listar[-2]
            error = np.abs((xractual-xranterior)/xractual)
        else:
            error = 1
        i += 1
```



+ Código + Texto



0 s

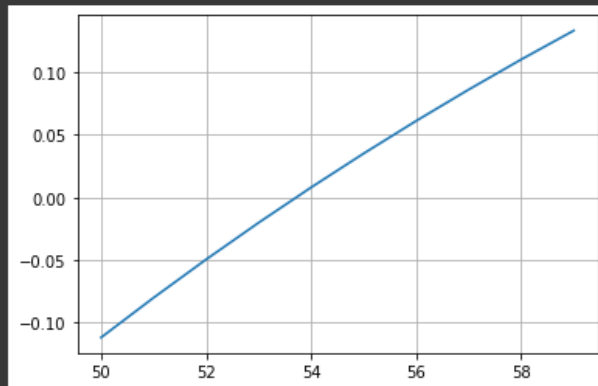
```
        i += 1
    else:
        print("No hay solución esa region")
    return listar
```

```
raices = biseccion(50, 60, 0.001)
print(raices)
print("X:", raices[-1])
```

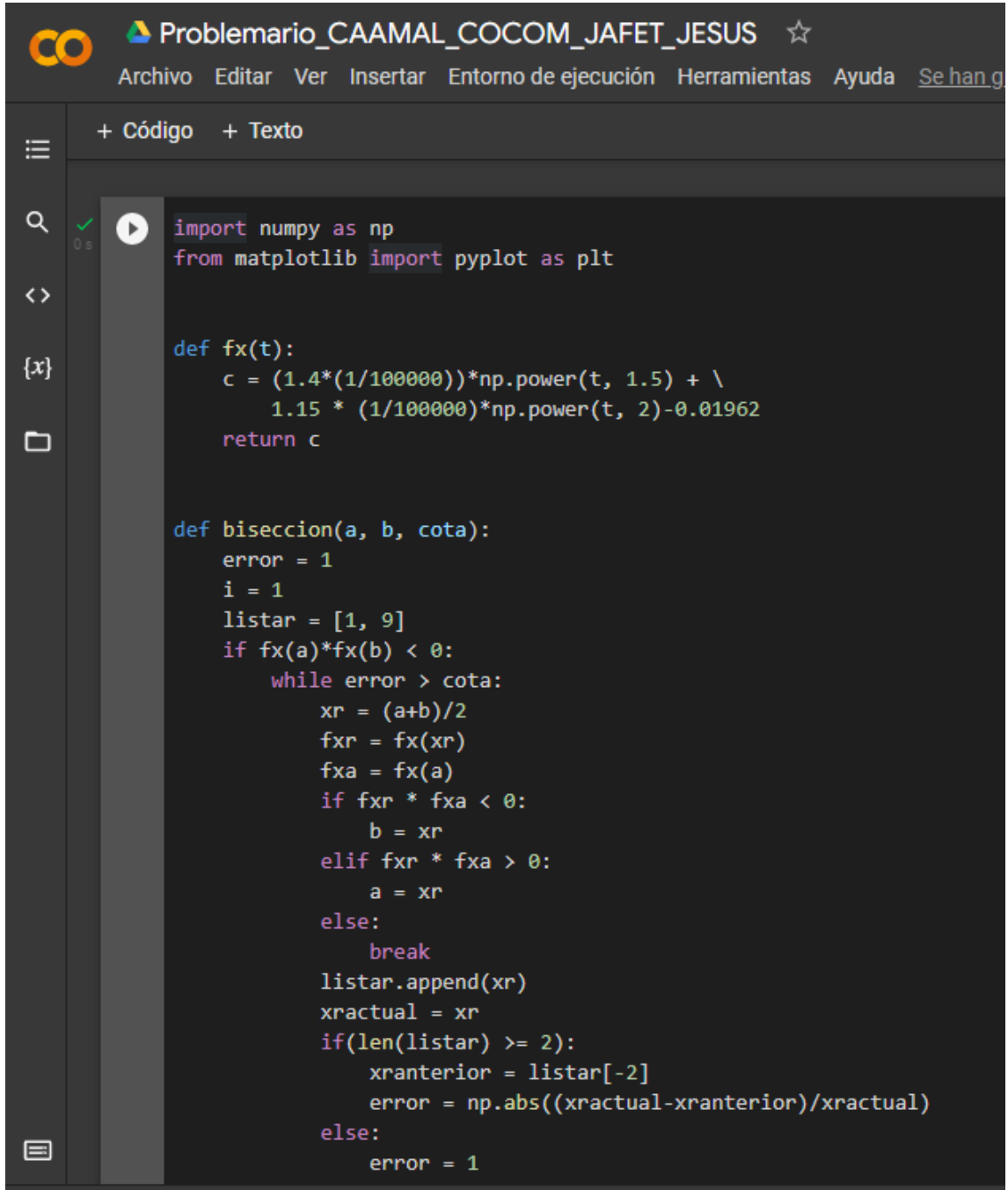
```
vectorx = np.arange(50, 60, 1)
plt.plot(vectorx, fx(vectorx))
plt.grid("X")
plt.grid("Y")
plt.show()
```



```
[1, 20, 55.0, 52.5, 53.75, 53.125, 53.4375, 53.59375, 53.671875, 53.7109375]
X: 53.7109375
```



EJERCICIO 2



The screenshot shows a Jupyter Notebook window titled "Problemario_CAAMAL_COCOM_JAFET_JESUS". The interface includes a top menu bar with options: Archivo, Editar, Ver, Insertar, Entorno de ejecución, Herramientas, Ayuda, and a link "Se han g...". Below the menu is a toolbar with icons for file operations and a "+ Código" button. The main area displays Python code for a bisection method. The code defines a function `fx(t)` and a function `biseccion(a, b, cota)`. The `fx(t)` function calculates a value `c` based on a formula involving `np.power`. The `biseccion` function implements the bisection algorithm, including a `while` loop to refine the interval and a `break` statement when the error is within the specified tolerance.

```
import numpy as np
from matplotlib import pyplot as plt

def fx(t):
    c = (1.4*(1/100000))*np.power(t, 1.5) + \
        1.15 * (1/100000)*np.power(t, 2)-0.01962
    return c

def biseccion(a, b, cota):
    error = 1
    i = 1
    listar = [1, 9]
    if fx(a)*fx(b) < 0:
        while error > cota:
            xr = (a+b)/2
            fxr = fx(xr)
            fxa = fx(a)
            if fxr * fxa < 0:
                b = xr
            elif fxr * fxa > 0:
                a = xr
            else:
                break
            listar.append(xr)
            xractual = xr
            if(len(listar) >= 2):
                xranterior = listar[-2]
                error = np.abs((xractual-xranterior)/xractual)
            else:
                error = 1
```



+ Código + Texto



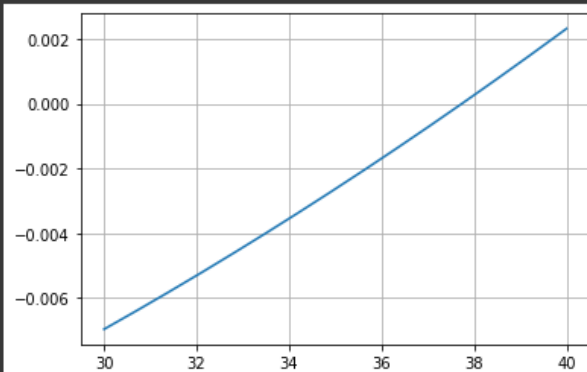
0 s

```
        else:
            error = 1
            i += 1
    else:
        print("No hay solución esa region")
    return listar
```

```
raices = biseccion(30, 40, 0.001)
print(raices)
print("X:", raices[-1])
# Grafica
vectorx = np.arange(30, 40, 0.001)
plt.plot(vectorx, fx(vectorx))
plt.grid("X")
plt.grid("Y")
plt.show()
```

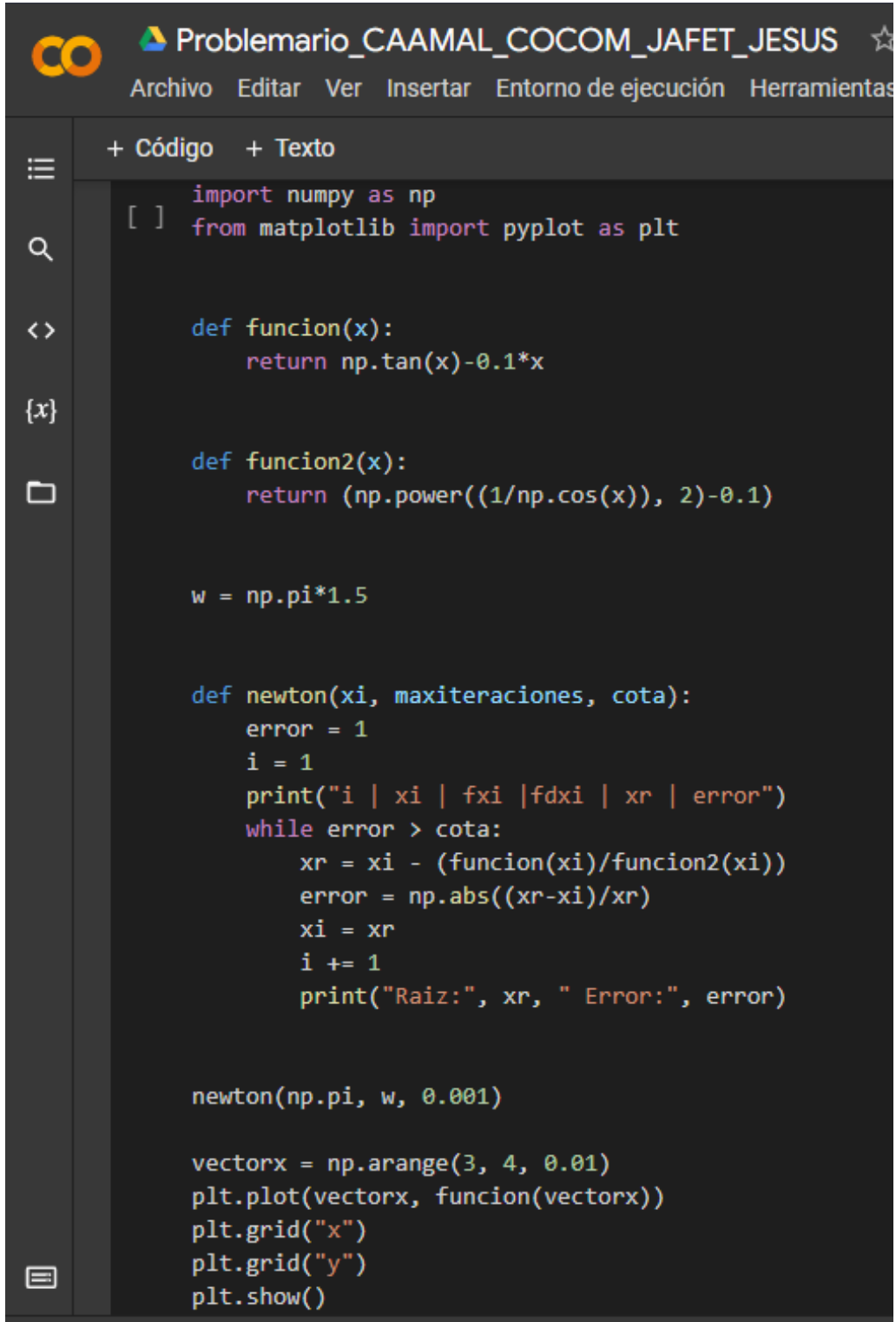


```
[1, 9, 35.0, 37.5, 38.75, 38.125, 37.8125, 37.65625, 37.734375, 37.7734375, 37.75390625]
X: 37.75390625
```



✓ 0 s completado a la

EJERCICIO 3



The screenshot shows a Jupyter Notebook window titled "Probleuario_CAAMAL_COCOM_JAFET_JESUS". The interface includes a top menu bar with "Archivo", "Editar", "Ver", "Insertar", "Entorno de ejecución", and "Herramientas". On the left, there is a sidebar with icons for file management and a panel with "+ Código" and "+ Texto" tabs. The main area displays the following Python code:

```
import numpy as np
from matplotlib import pyplot as plt

def funcion(x):
    return np.tan(x)-0.1*x

def funcion2(x):
    return (np.power((1/np.cos(x)), 2)-0.1)

w = np.pi*1.5

def newton(xi, maxiteraciones, cota):
    error = 1
    i = 1
    print("i | xi | fxi |fdxi | xr | error")
    while error > cota:
        xr = xi - (funcion(xi)/funcion2(xi))
        error = np.abs((xr-xi)/xr)
        xi = xr
        i += 1
        print("Raiz:", xr, " Error:", error)

newton(np.pi, w, 0.001)

vectorx = np.arange(3, 4, 0.01)
plt.plot(vectorx, funcion(vectorx))
plt.grid("x")
plt.grid("y")
plt.show()
```




Problematario_CAAMAL_COCOM_JAFET_JESUS ☆

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda [Se han guardado todos los cambios](#)



+ Código + Texto

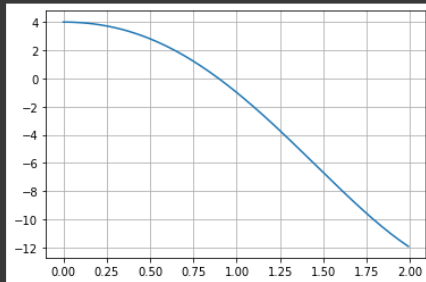


```
else:
    print("No hay solución esa región")
return listar

raices = biseccion(0, 2, 0.001)
# raices = biseccion(-2, -1, 0.001) No hay solución
print(raices)
print("La raíz es: ", raices[-1])

vectorx = np.arange(0, 2, 0.01)
plt.plot(vectorx, fx(vectorx))
plt.grid("x")
plt.grid("y")
plt.show()
```

[1, 20, 1.0, 0.5, 0.75, 0.875, 0.9375, 0.90625, 0.890625, 0.8984375, 0.89453125, 0.896484375, 0.8955078125, 0.89501953125]
La raíz es: 0.89501953125



EJERCICIO 5

Ejercicio 5

$f(x) = \tan(x) - 3.5$ en el intervalo $[0, \pi]$
Error = 0.05

i	x_i	x_s	x_r	$f(x_i)$	$f(x_r)$	AxB	Error
1	0	π	1.57	-3.5	-3.47	-	1
2	1.57	1.57	0.785	-3.5	-3.486	+	1
3	0.785	0.785	0.785	-3.486	-3.486	+	1
4	1.178	0.785	0.981	-3.47	-3.486	-	0.3333
5	1.374	0.981	1.1775	-3.47	-3.4	+	0.142

EJERCICIO 6

Ejercicio 6

Encuentra la raíz de $f(x) = \tan(x) - 0.1x$
Intervalo $[\pi, 1.5\pi]$ Error: 0.0001

i	a	b	x_r	$f(x_r)$	$f(a)$	AxB	Error
1	3.1416	1.5π	3.9269	-0.3141	0.6071	-	0.1949
2	3.1416	3.9269	3.5342	-0.3141	0.0606	-	0.1110
3	3.1416	3.5332	3.3378	-0.3141	-0.1350	+	0.0587
4	3.3378	3.5332	3.436	-0.1350	-0.0403	+	0.028
5	3.436	3.5343	3.4851	-0.0403	-0.0091	-	0.0140

EJERCICIO 7

Ejercicio 7

Hallar la raíz de la Función $F(x) = x^4 - 2x^3 - 4x^2 + 4 = 0$ $F(x) = x^4 - 2x^3 - 4x^2$
 Error = 0.001

a $[-2, -1]$
 b $[0, 2]$

i	a	b	x_i	$F(x_i)$	$F(a)$	$A \times B$	Error
1	0	2	1	4	-1	-	1
2	0	1	0.5	4	2.81	+	1
3	0.5	1	0.75	2.8125	1.22	+	6.33
4	0.75	1	0.875	1.2226	1.18	+	0.14
5	0.875	1	0.9375	1.1830	-0.39	-	0.06

EJERCICIO 8

Ejercicio 8

Sea $F(x) = \tan(x) - 6$, y utilice $x_0 = 0$, $x_1 = 0.44$
 para determinar la raíz. Error = 0.001

i	x_{i-1}	x_i	$F(x_{i-1})$	$F(x_i)$	x_{i+1}	Error
0	0	0.44	-6	-0.7578	0.5036	0.1263
1	0.44	0.5036	-0.7578	-99.4156	0.4394	0.1459
2	0.3036	0.4394	-99.4156	-0.8109	0.4388	0.0013
3	0.4394	0.4388	-0.8109	-0.8631	0.4487	0.0217
4	0.4388	0.4487	-0.8631	-0.1510	0.4472	0.0032