Tarea2

November 5, 2025

Tarea N2

Métodos Númericos

Nombre: Lenin Amangandi

Enlace GitHub

0.1 Pregunta N1

Calcule los errores absoluto y relativo en las aproximaciones de por $*$.

[17]:
```
import math

def error_absoluto(p, p_aprox):
    return abs(p - p_aprox)

def error_relativo(p, p_aprox):
    return error_absoluto(p, p_aprox) / abs(p)

def mostrar_resultados(p, p_aprox):
    ea = error_absoluto(p, p_aprox)
    er = error_relativo(p, p_aprox)
    print(f"Valor real (p): {p}")
    print(f"Aproximación (p*): {p_aprox}")
    print(f"Error absoluto: {ea}")
    print(f"Error relativo: {er}")
    print()

def intervalo_error(p, error_max):
    inferior = p * (1 - error_max)
    superior = p * (1 + error_max)
    return inferior, superior
```

a)

$$p = \pi, \qquad p^* = \frac{22}{7}$$

[18]:
```
mostrar_resultados(math.pi, 22/7)
```

```
Valor real (p): 3.141592653589793
Aproximación (p*): 3.142857142857143
Error absoluto: 0.0012644892673496777
Error relativo: 0.0004024994347707008
```

b)

$$p = \pi, \qquad p^* = 3.1416$$

[19]:
```
mostrar_resultados(math.pi, 3.1416)
```

```
Valor real (p): 3.141592653589793
Aproximación (p*): 3.1416
Error absoluto: 7.346410206832132e-06
Error relativo: 2.3384349967961744e-06
```

[ ]:
```

```

c)

$$p = e, \qquad p^* = 2.718$$

[20]:
```
mostrar_resultados(math.e, 2.718)
```

```
Valor real (p): 2.718281828459045
Aproximación (p*): 2.718
Error absoluto: 0.0002818284590451192
Error relativo: 0.00010367889601972718
```

d)

$$p = \sqrt{2}, \qquad p^* = 1.414$$

[21]:
```
mostrar_resultados(math.sqrt(2), 1.414)
```

```
Valor real (p): 1.4142135623730951
Aproximación (p*): 1.414
Error absoluto: 0.00021356237309522186
Error relativo: 0.00015101140222192286
```

0.2 Pregunta N2

Calcule los errores absoluto y relativo en las aproximaciones de por $*$.

a)

$$p = e^{10}, \qquad p^* = 22000$$

[22]:
```
mostrar_resultados(math.e**10, 22000)
```

```
Valor real (p): 22026.465794806703
Aproximación (p*): 22000
Error absoluto: 26.465794806703343
Error relativo: 0.0012015452253326688
```

b)

$$p = 10^{\pi}, \qquad p^* = 1400$$

[23]:
```
mostrar_resultados(10**math.pi, 1400)
```

```
Valor real (p): 1385.4557313670107
Aproximación (p*): 1400
Error absoluto: 14.544268632989315
Error relativo: 0.010497822704619136
```

c)

$$p = 8!, \qquad p^* = 39900$$

[ ]:
```
mostrar_resultados(math.factorial(8), 39900)
```

```
Valor real (p): 40320
Aproximación (p*): 39900
Error absoluto: 420
Error relativo: 0.010416666666666666
```

d)

$$p = 9!, \qquad p^* = \sqrt{18\pi}\left(\frac{9}{e}\right)^9$$

[27]:
```
p_real = math.factorial(9)
p_aprox = math.sqrt(18 * math.pi) * ((9 / math.e) ** 9)
mostrar_resultados(p_real, p_aprox)
```

```
Valor real (p): 362880
Aproximación (p*): 359536.87284194835
Error absoluto: 3343.1271580516477
Error relativo: 0.009212762230080598
```

0.3 Pregunta N3

Encuentre el intervalo más largo en el que se debe encontrar $*$ para aproximarse a con error relativo máximo de 10^−4 para cada valor de .

[38]:
```
import math
def redondear_3_digitos(x):
    if x == 0:
        return 0
```

```
    else:
        return round(x, 3 - int(math.floor(math.log10(abs(x)))) - 1)

def error_absoluto(p, p_aprox):
    return abs(p - p_aprox)

def error_relativo(p, p_aprox):
    return error_absoluto(p, p_aprox) / abs(p)

def mostrar_resultados(p, p_aprox):
    ea = error_absoluto(p, p_aprox)
    er = error_relativo(p, p_aprox)
    print(f"Valor real (p): {p}")
    print(f"Aproximado (p*): {p_aprox}")
    print(f"Error absoluto: {ea}")
    print(f"Error relativo: {er}\n")
```

a)

π

[39]:
```
error_max = 1e-4

a_inf, a_sup = intervalo_error(math.pi, error_max)
print(f"3.a    [{a_inf}, {a_sup}]")
```

```
3.a    [3.141278494324434, 3.141906812855152]
```

b)

e

[40]:
```
b_inf, b_sup = intervalo_error(math.e, error_max)
print(f"3.b e   [{b_inf}, {b_sup}]")
```

```
3.b e   [2.718010000276199, 2.718553656641891]
```

c)

$\sqrt{2}$

[31]:
```
c_inf, c_sup = intervalo_error(math.sqrt(2), error_max)
print(f"3.c √2   [{c_inf}, {c_sup}]")
```

```
3.c √2   [1.4140721410168577, 1.4143549837293325]
```

d)

$\sqrt[3]{7}$

[32]:
```
d_inf, d_sup = intervalo_error(7**(1/3), error_max)
print(f"3.d  7   [{d_inf}, {d_sup}]")
```

```
3.d  7   [1.9127398896541117, 1.9131224758906662]
```

0.4 Pregunta N4

Use la aritmética de redondeo de tres dígitos para realizar lo siguiente. Calcule los errores absoluto y relativo con el valor exacto determinado para por lo menos cinco dígitos.

a)

$$\frac{\frac{13}{14} - \frac{5}{7}}{2e - 5.4}$$

[41]:
```
num = redondear_3_digitos(13/14)
num = redondear_3_digitos(num - redondear_3_digitos(5/7))
den = redondear_3_digitos((2 * math.e) - 5.4)
aprox_a = redondear_3_digitos(num / den)
exact_a = (13/14 - 5/7) / ((2 * math.e) - 5.4)
mostrar_resultados(exact_a, aprox_a)
```

```
Valor real (p): 5.860620417858059
Aproximado (p*): 5.87
Error absoluto: 0.009379582141940901
Error relativo: 0.0016004418428738562
```

b)

$$-10\pi + 6e - \frac{3}{61}$$

[42]:
```
aprox_b = redondear_3_digitos(-10 * math.pi)
aprox_b = redondear_3_digitos(aprox_b + redondear_3_digitos(6 * math.e))
aprox_b = redondear_3_digitos(aprox_b - redondear_3_digitos(3/61))
exact_b = -10*math.pi + 6*math.e - 3/61
mostrar_resultados(exact_b, aprox_b)
```

```
Valor real (p): -15.155415893012512
Aproximado (p*): -15.1
Error absoluto: 0.05541589301251193
Error relativo: 0.003656507574830839
```

c)

$$\left(\frac{2}{9}\right) \cdot \left(\frac{9}{11}\right)$$

[43]:
```
aprox_c = redondear_3_digitos(2/9)
aprox_c = redondear_3_digitos(aprox_c * redondear_3_digitos(9/11))
exact_c = (2/9)*(9/11)
mostrar_resultados(exact_c, aprox_c)
```

```
Valor real (p): 0.18181818181818182
Aproximado (p*): 0.182
Error absoluto: 0.0001818181818181719
Error relativo: 0.0009999999999999454
```

d)

$$\frac{\sqrt{13}+\sqrt{11}}{\sqrt{13}-\sqrt{11}}$$

[45]:
```
aprox_d = redondear_3_digitos(math.sqrt(13))
aprox_d = redondear_3_digitos((aprox_d + redondear_3_digitos(math.sqrt(11))) /
                              (math.sqrt(13) - math.sqrt(11)))
exact_d = (math.sqrt(13) + math.sqrt(11)) / (math.sqrt(13) - math.sqrt(11))
mostrar_resultados(exact_d, aprox_d)
```

```
Valor real (p): 23.95826074310141
Aproximado (p*): 24.0
Error absoluto: 0.04173925689859104
Error relativo: 0.0017421655664470355
```

0.5 Pregunta N5

Los primeros tres términos diferentes a cero de la serie de Maclaurin para la función arcotangente son:

$$x - \frac{1}{3}x^3 + \frac{1}{5}x^5$$

Calcule los errores absoluto y relativo en las siguientes aproximaciones de mediante el polinomio en lugar del arcotangente:

a)

$$4\left[\arctan\left(\frac{1}{2}\right) + \arctan\left(\frac{1}{3}\right)\right]$$

[48]:
```
def arctan_aprox(x):
    return x - (x**3)/3 + (x**5)/5
real_a5 = 4 * (math.atan(1/2) + math.atan(1/3))
aprox_a5 = 4 * (arctan_aprox(1/2) + arctan_aprox(1/3))
mostrar_resultados(real_a5, aprox_a5)
```

```
Valor real (p): 3.141592653589793
Aproximado (p*): 3.1455761316872426
Error absoluto: 0.003983478097449478
Error relativo: 0.0012679804598147663
```

b)

$$16\arctan\left(\frac{1}{5}\right) - 4\arctan\left(\frac{1}{239}\right)$$

[50]:
```
real_b5 = 16 * math.atan(1/5) - 4 * math.atan(1/239)
aprox_b5 = 16 * arctan_aprox(1/5) - 4 * arctan_aprox(1/239)
mostrar_resultados(real_b5, aprox_b5)
```

```
Valor real (p): 3.1415926535897936
Aproximado (p*): 3.1416210293250346
Error absoluto: 2.837573524105963e-05
Error relativo: 9.032277054963067e-06
```

0.6 Pregunta N6

El número se puede definir por medio de

$$e = \sum_{n=0}^{\infty} \frac{1}{n!}$$

donde

a)

$$\sum_{n=0}^{5} \frac{1}{n!}$$

[52]:
```
aprox_a6 = sum(1 / math.factorial(i) for i in range(6))
mostrar_resultados(math.e, aprox_a6)
```

```
Valor real (p): 2.718281828459045
Aproximado (p*): 2.716666666666667
Error absoluto: 0.0016151617923783057
Error relativo: 0.0005941848175815963
```

b)

$$\sum_{n=0}^{10} \frac{1}{n!}$$

[53]:
```
aprox_b6 = sum(1 / math.factorial(i) for i in range(11))
mostrar_resultados(math.e, aprox_b6)
```

```
Valor real (p): 2.718281828459045
Aproximado (p*): 2.7182818011463845
Error absoluto: 2.7312660577649694e-08
Error relativo: 1.0047766310211053e-08
```

0.7 Pregunta N7

Suponga que dos puntos se encuentran en línea recta con Existen dos fórmulas para encontrar la intersección de la línea:

$$x = \frac{x_0 y_1 - x_1 y_0}{y_1 - y_0} \quad \text{y} \quad x = x_0 - \frac{(x_1 - x_0) y_0}{y_1 - y_0}$$

a) Use los datos

$$(x_0, y_0) = (1.31, 3.24) \quad \text{y} \quad (x_1, y_1) = (1.93, 5.76)$$

y la aritmética de redondeo de tres dígitos para calcular la intersección con de ambas maneras. ¿Cuál método es mejor y por qué?

[60]:
```
import math

def round_sig(x, sig=3):
    if x == 0:
        return 0.0
    else:
        return float(round(x, sig - int(math.floor(math.log10(abs(x)))) - 1))

x0, y0 = 1.31, 3.24
x1, y1 = 1.93, 5.76

x_exact = (x0*y1 - x1*y0) / (y1 - y0)

print("Exacto:", x_exact)
```

```
Exacto: 0.5128571428571429
```

- Primer Metodo

[61]:
```
a = round_sig(x0 * y1)
b = round_sig(x1 * y0)
numer = round_sig(a - b)
denom = round_sig(y1 - y0)
xA = round_sig(numer / denom)

print("Metodo A:", xA)
```

```
Metodo A: 0.516
```

- Segundo Metodo

[62]:
```
dx = round_sig(x1 - x0)
mult = round_sig(dx * y0)
denom = round_sig(y1 - y0)
div = round_sig(mult / denom)
xB = round_sig(x0 - div)
print("Metodo B:", xB)
```

```
Metodo B: 0.512
```