Deber 1

Conjunto de Ejercicios 1

Resuelva los siguientes ejercicios, tome en cuenta que debe mostrar el desarrollo completo del ejercicio.

Ejercicio 1

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

a. = , * = 22/7

```
p=math.pi
pp=22/7
EA=abs(p-pp)
print(EA)
ER=(EA)/p
print(ER)
porcentaje =ER*100
print(porcentaje ,'%')
```

```
0.0012644892673496777
0.0004024994347707008
0.04024994347707008 %
```

b. = , * = 3.1416

```
p=math.pi
pp=3.1416
EA=abs(p-pp)
print(EA)
ER=(EA)/p
print(ER)
porcentaje =ER*100
print(porcentaje ,'%')
```

```
7.346410206832132e-06
2.3384349967961744e-06
0.00023384349967961744 %
```

c. $= \ , \ * = 2.718$

```
p=math.e
pp=2.718
EA=abs(p-pp)
print(EA)
ER=(EA)/p
print(ER)
porcentaje =ER*100
print(porcentaje ,'%')
```

```
0.0002818284590451192
0.00010367889601972718
0.010367889601972718 %
```

d. $= \sqrt{2}, \ * = 1.414$

```
p=math.sqrt(2)
pp=1.414
EA=abs(p-pp)
print(EA)
ER=(EA)/p
print(ER)
porcentaje =ER*100
print(porcentaje ,'%')
```

```
0.00021356237309522186
0.00015101140222192286
0.015101140222192286 %
```

Ejercicio 2

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

a. $= {}^{1}$, $* = 22000$

```
p=(math.e)**10
pp=22000
EA=abs(p-pp)
print(EA)
ER=(EA)/p
print(ER)
porcentaje =ER*100
print(porcentaje ,'%')
```

```
26.465794806703343
0.0012015452253326688
0.12015452253326687 %
```

b. $= 10$, $* = 1400$

```
p=math.pi*10
pp=1400
EA=abs(p-pp)
print(EA)
ER=(EA)/p
print(ER)
porcentaje =ER*100
print(porcentaje ,'%')
```

```
1368.5840734641022
43.5633840657307
4356.33840657307 %
```

c. $= 8!$, $* = 39900$

```
p=40320
pp=39900
EA=abs(p-pp)
print(EA)
ER=(EA)/p
print(ER)
porcentaje =ER*100
print(porcentaje ,'%')
```

```
420
0.010416666666666666
1.0416666666666665 %
```

d. $= 9!, \ * = \sqrt{18}\ (9/\ )9$

```
p=362880
pp=359536.8728419483
EA=abs(p-pp)
print(EA)
ER=(EA)/p
print(ER)
porcentaje =ER*100
print(porcentaje ,'%')
```

```
3343.127158051706
0.00921276223008076
0.921276223008076 %
```

Ejercicio 3

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

a.

```
ER=1/10000
p=math.pi
#caso 1
pp1=p-ER*(abs(p))
#caso 2
pp2=ER*(abs(p))+p
print('[',pp1,' , ',pp2,']')
```

```
[ 3.141278494324434  ,  3.141906812855152 ]
```

```
b.
```

```
ER=1/10000
p=math.e
#caso 1
pp1=p-ER*(abs(p))
#caso 2
pp2=ER*(abs(p))+p
print('[',pp1,' , ',pp2,']')
```

```
[ 2.718010000276199  ,  2.718553656641891 ]
```

c. $\sqrt{2}$

```
ER=1/10000
p=math.sqrt(2)
#caso 1
pp1=p-ER*(abs(p))
#caso 2
pp2=ER*(abs(p))+p
print('[',pp1,' , ',pp2,']')
```

```
[ 1.4140721410168577  ,  1.4143549837293325 ]
```

```
d. √7
```

```
ER=1/10000
p=math.sqrt(7)
#caso 1
pp1=p-ER*(abs(p))
#caso 2
pp2=ER*(abs(p))+p
print('[',pp1,' , ',pp2,']')
```

```
[ 2.6454867359334844  ,  2.646015886195697 ]
```

Ejercicio 4

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)** [(13/14) / (5/7)] / (2e-5.4)

```
a=round((13/14),5)
aa=round((13/14),3)
b=round((5/7),5)
bb=round((5/7),3)
c=round(((2*math.e)-5.4),5)
cc=round(((2*math.e)-5.4),3)
abc=(a/b)/c
aabbcc=(aa/bb)/cc
EA=abs(abc-aabbcc)
print(EA)
ER=EA/abs(abc)
print(ER)
```

```
0.39230130163035426
0.011032802852634603
```

- **b)** -10 + 6e - 3/61

```
a=round((-10*math.pi),5)
aa=round((-10*math.pi),3)
b=round((6*math.e),5)
bb=round((6*math.e),3)
c=round(((-1*3)/61),5)
cc=round(((-1*3)/61),3)
abc=a+b+c
aabbcc=aa+bb+cc
EA=abs(abc-aabbcc)
print(EA)
ER=EA/abs(abc)
print(ER)
```

```
0.0004199999999983106
2.7712857842165417e-05
```

- **c)** (2/9) * (9/11)

```
a=round((2/9),5)
aa=round((2/9),3)
b=round((9/11),5)
bb=round((9/11),3)

ab=a*b
aabb=aa*bb
EA=abs(ab-aabb)
print(EA)
ER=EA/abs(ab)
print(ER)
```

```
0.0002199596000000137
0.001209792586327024
```

- **d)** √(13+11) / √(13-11)

```
a=round(math.sqrt(13+11),5)
aa=round(math.sqrt(13+11),3)
b=round(math.sqrt(13-11),5)
bb=round(math.sqrt(13-11),3)

ab=a/b
aabb=aa/bb
EA=abs(ab-aabb)
print(EA)
ER=EA/abs(ab)
print(ER)
```

```
0.0005286161584390214
0.00015259794027043354
```

Ejercicio 5

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

x - (1/3)x^3 + (1/5)x^5. Calcule los errores absoluto y relativo en las siguientes aproximaciones de mediante el polinomio en lugar del arcotangente:

a. 4[arctan(1/2) + arctan(1/3)]

b. 16 arctan(1/5) - 4 arctan(1/239)

```
import math

def arctan_approx(x):
  return x - (1/3)*x**3 + (1/5)*x**5

def approx_pi(formula):
  # Evaluar la fórmula dada
  approx_pi = eval(formula)

  # Calcular el error absoluto y relativo
  error_absoluto = abs(approx_pi - math.pi)
  error_relativo = error_absoluto / math.pi

  return approx_pi, error_absoluto, error_relativo

# Ejemplo
formula_a = "4*(arctan_approx(1/2) + arctan_approx(1/3))"
formula_b = "16*arctan_approx(1/5) - 4*arctan_approx(1/239)"

result_a = approx_pi(formula_a)
result_b = approx_pi(formula_b)

print("Aproximación a usando los primeros 3 términos de la serie de Maclaurin:")
print(f"Valor aproximado de pi: {result_a[0]}")
print(f"Error absoluto: {result_a[1]}")
print(f"Error relativo: {result_a[2]}\n")

print("Aproximación b usando los primeros 3 términos de la serie de Maclaurin:")
print(f"Valor aproximado de pi: {result_b[0]}")
print(f"Error absoluto: {result_b[1]}")
print(f"Error relativo: {result_b[2]}")
```

```
Aproximación a usando los primeros 3 términos de la serie de Maclaurin:
Valor aproximado de pi: 3.1455761316872426
Error absoluto: 0.003983478097449478
Error relativo: 0.0012679804598147663

Aproximación b usando los primeros 3 términos de la serie de Maclaurin:
Valor aproximado de pi: 3.1416210293250346
Error absoluto: 2.837573524150372e-05
Error relativo: 9.032277055104427e-06
```

Ejercicio 6

El número se puede definir por medio de = ∞ (1/ n!), donde ! = (− 1) 2 1 para 0 y n=0 ! 0! = 1. Calcule los errores absoluto y relativo en la siguiente aproximación de :

a. Σ (1/n!)

```
import math

def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)

def approximate_e(n):
    sum = 0
    for i in range(n+1):
        sum += 1/factorial(i)
    return sum

# Calcular la aproximación para n=5 y n=10
approx_5 = approximate_e(5)


error_abs_5 = abs(approx_5 - math.e)
error_rel_5 = error_abs_5 / math.e
print(error_abs_5)

print(error_rel_5)
```

```
0.0016151617923787498
0.0005941848175817597
```

b. Σ ¹ (1/n!)

```
approx_10 = approximate_e(10)
```

```
error_abs_10 = abs(approx_10 - math.e)
error_rel_10 = error_abs_10 / math.e
print(error_abs_10)
print(error_rel_10)
```

```
2.7312660577649694e-08
1.0047766310211053e-08
```

Ejercicio 7

Suponga que dos puntos(,) y (,) se encuentran en línea recta con . Existen dos fórmulas para encontrar la intersección de la línea:

= x y –x y y = y –y y –y

a. Use los datos (,) = (1.31,3.24) y (,) = (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?

```
 X0=1.31
X1=1.92
Y0=3.24
Y1=5.76
#PRIMER CASO
 Xp=((X0*Y1)-(X1*Y0))/(Y1-Y0)
Xpp=round(Xp,3)
EA=abs(Xp-Xpp)
print('-------------')

print('Primer Caso')
print('-------------')

print('Error abosoluto: ',EA)
ER=EA/abs(Xp)
print('Error relativo: ',ER)
print('X: ',Xp)
#SEGUNDO CASO

Xs=X0-(((X1-X0)*Y0)/(Y1-Y0))
Xss=round(Xs,3)
print('-------------')
```

```
print('Segundo Caso')
print('----------------')


print(Xs)
EA=abs(Xs-Xss)
print('Error abosoluto: ',EA)
ER=EA/abs(Xs)
print('Error relativo: ',ER)
print('X: ',Xs)
```

```
-------------
Primer Caso
-------------
Error abosoluto:  0.00028571428571433355
Error relativo:  0.0005434782608696562
X:  0.5257142857142857
-------------
Segundo Caso
----------------
0.5257142857142858
Error abosoluto:  0.0002857142857142225
Error relativo:  0.00054347826086944 45
X:  0.5257142857142858
```

Si bien ambas ecuaciones son válidas para determinar la intersección en el eje x, la segunda ecuación demostró ser numéricamente más estable, resultando en una menor propagación de errores de redondeo y, por tanto, en una mayor precisión