```
import numpy as np
import pandas as pd

formula1Data = []       #list of values from formula 1
formula1RelErrors = []  #relative errors from formula 1`
formula2Data = []       #list of values from formula 2
formula2RelErrors = []  #relative errors from formula 2

valueToFind = 5         #value in -exponent to find
trueValue = np.e**-5    #value to compare to

#factorial formula, could have use the gamma function in the numpy library, but
# this is probably more resource efficient
def factorial(value):
    result = 1
    while value > 0:
        result *= value
        value -= 1
    return result

#formula 1:
previousValue = 0       #Used to add up terms in series
sign = 1                #sign of current term, either 1 or 01
for i in range(1,21):
    #find next term in series
    term = sign * ((valueToFind**(i-1))/factorial(i - 1))
    #add term to previous term
    value = previousValue + term
    #store term
    formula1Data.append(value)
    formula1RelErrors.append(100 * np.abs((value - previousValue)/value))
    #update previous value and sign
    previousValue = value
    sign *= -1


#formula 2:
previousValue = 0       #set to zero to not preserve data from above
for i in range(1, 21):
    #find next term in demoninator
    term = (valueToFind**(i-1)/factorial(i-1))
    #add next term to previous denominator
    denominator = previousValue + term
    #store 1/denominator
    formula2Data.append(denominator**-1)
    #handles first iteration where I can't take 0**-1
    previousValueInverse = 0 if previousValue == 0 else previousValue**-1
    formula2RelErrors.append(100 * np.abs(denominator**-1 - previousValueInverse)/(denominator**-1))
    #update values
    previousValue = denominator

#plotting

data = {
    ('Formula 1', 'Value'): formula1Data,
    ('Formula 1', 'true error %'): [np.abs((x - trueValue)/trueValue) * 100 for x in formula1Data],
    ('Formula 1', 'approx error %'): formula1RelErrors,
    ('',''): '',
    ('Formula 2', 'Value'): formula2Data,
    ('Formula 2', 'true error %'): [np.abs((x - trueValue)/trueValue) * 100 for x in formula2Data],
    ('Formula 2', 'approx error %'): formula2RelErrors
}

dataFrame = pd.DataFrame(data)

dataFrame.index = range(1, len(formula1Data)+1)
dataFrame.index.name = 'Terms'

print("Expected value: " + str(np.e**-5))
print(dataFrame)

#Looking at the table, the second fomula has the least error. Also notable is how the
# error converges much much more quickly and consistently than the first formula
```