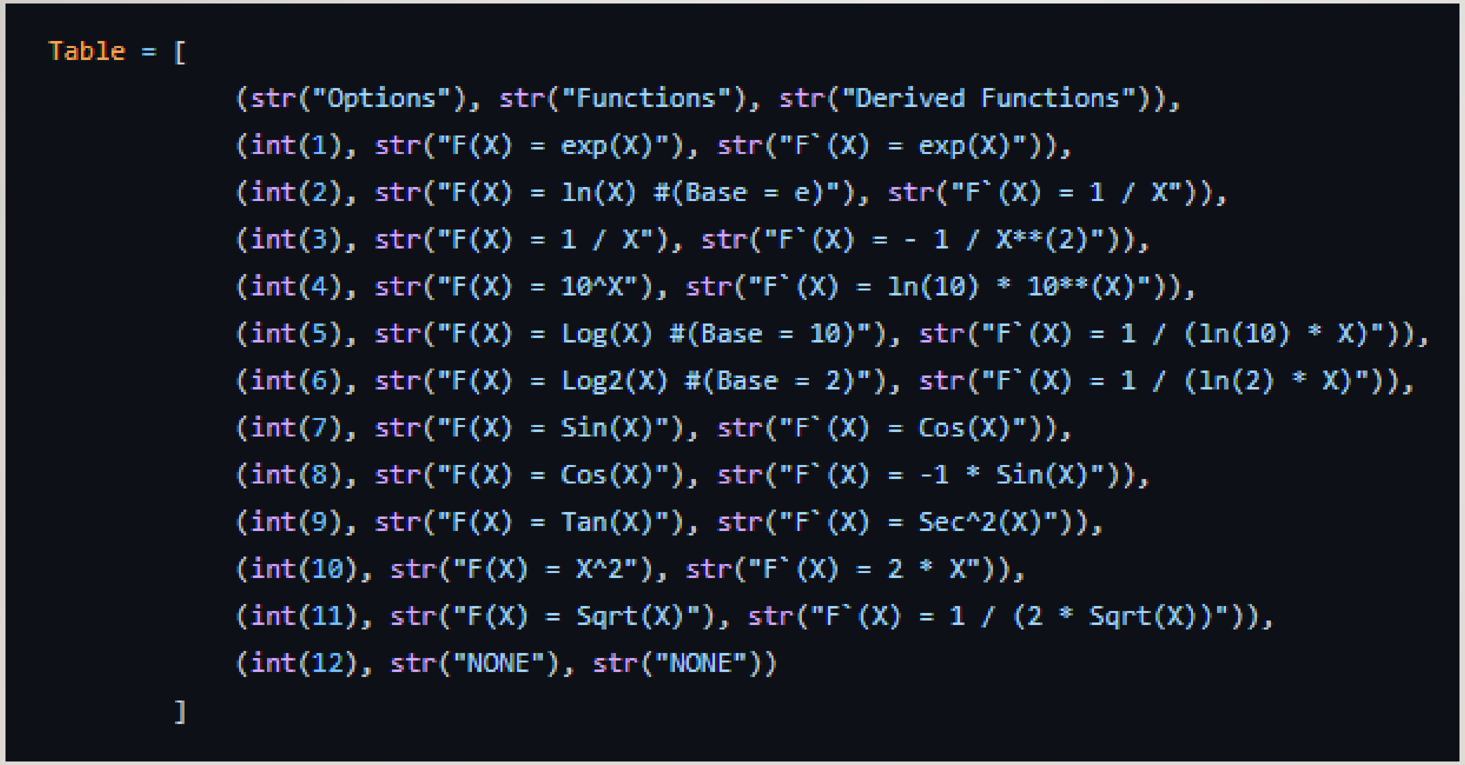
NUMERICAL DIFFERENTIATION

BY: EDWARD AND BRILIAN

INTRODUCTION

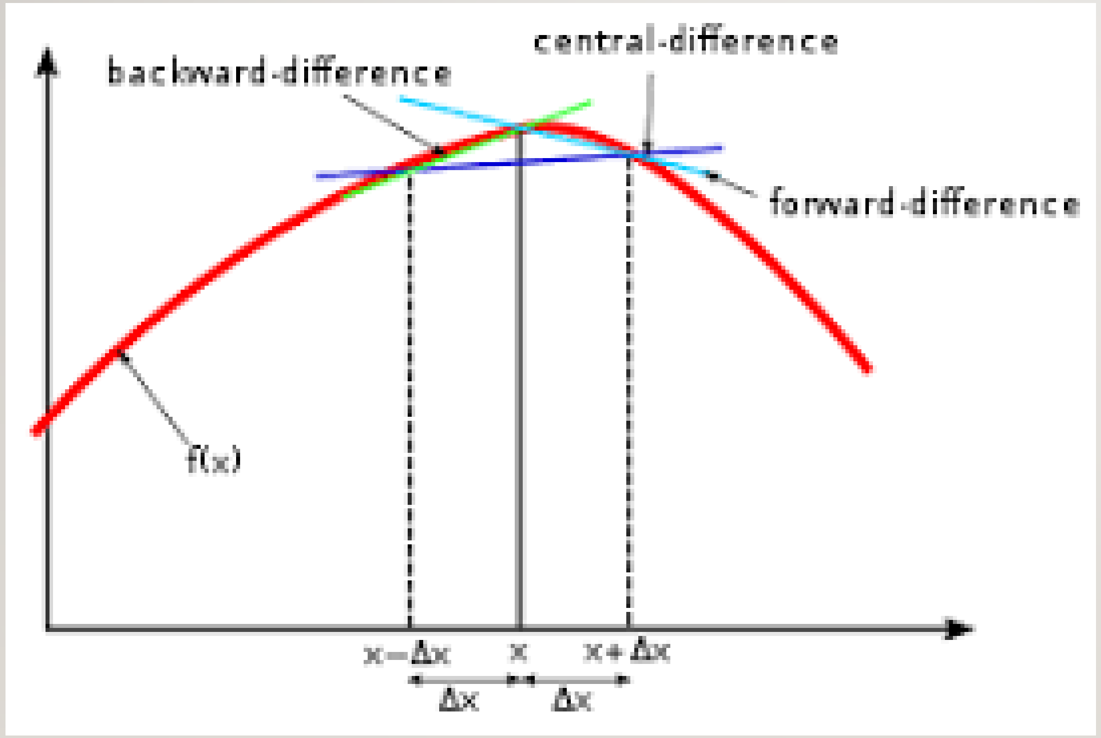
- Numerical differentiation is the process of finding the numerical value of a derivative of a given function at a given point
- It involves forward, central and backward differences for derivatives

FUNCTIONS WE USE



```
Table = [
            (str("Options"), str("Functions"), str("Derived Functions")),
            (int(1), str("F(X) = exp(X)"), str("F`(X) = exp(X)")),
            (int(2), str("F(X) = ln(X) #(Base = e)"), str("F`(X) = 1 / X")),
            (int(3), str("F(X) = 1 / X"), str("F`(X) = - 1 / X**(2)")),
            (int(4), str("F(X) = 10^X"), str("F`(X) = ln(10) * 10**(X)")),
            (int(5), str("F(X) = Log(X) #(Base = 10)"), str("F`(X) = 1 / (ln(10) * X)")),
            (int(6), str("F(X) = Log2(X) #(Base = 2)"), str("F`(X) = 1 / (ln(2) * X)")),
            (int(7), str("F(X) = Sin(X)"), str("F`(X) = Cos(X)")),
            (int(8), str("F(X) = Cos(X)"), str("F`(X) = -1 * Sin(X)")),
            (int(9), str("F(X) = Tan(X)"), str("F`(X) = Sec^2(X)")),
            (int(10), str("F(X) = X^2"), str("F`(X) = 2 * X")),
            (int(11), str("F(X) = Sqrt(X)"), str("F`(X) = 1 / (2 * Sqrt(X))")),
            (int(12), str("NONE"), str("NONE"))
        ]
```

FINITE DIFFERENCE



FORWARD DIFFERENCE APPROXIMATION

- Forward differences are useful in solving ordinary differential equations by single-step predictor-corrector methods

$$I'(t_0) = (I_1 - I_0) / (t_1 - t_0)$$

BACKWARD DIFFERENCE APPROXIMATION

- Approximation of the derivative of that function using information from already computed time points, thereby increasing the accuracy of the approximation.

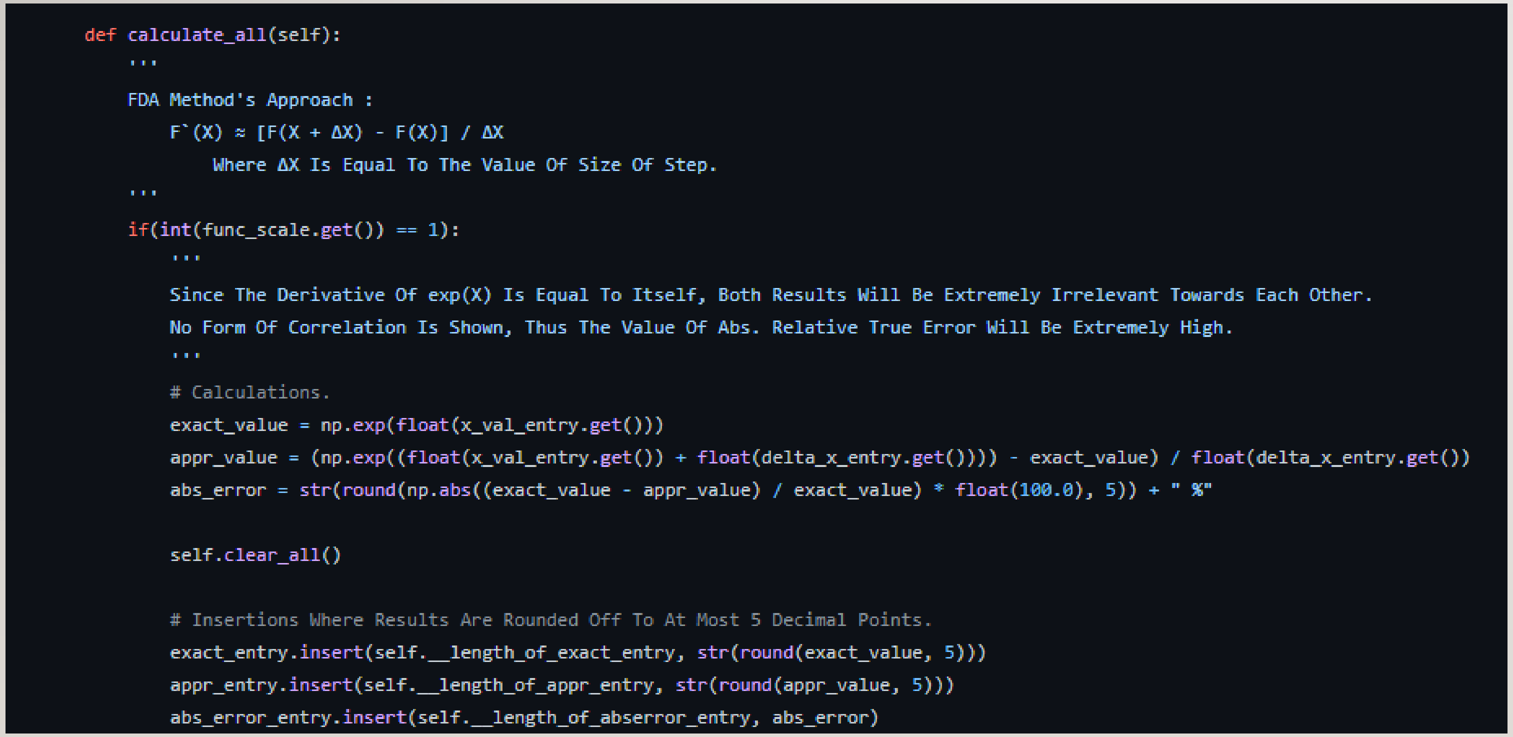
$$I'(t_0) = (I_0 - I_{-1}) / (t_0 - t_{-1})$$

CENTRAL DIFFERENCE APPROXIMATION

- If the data values are equally spaced, the central difference is an average of the forward and backward differences.

$$I'(t_0) = (I_1 - I_{-1}) / (t_1 - t_{-1})$$

IMPLEMENTATION



```
def calculate_all(self):
    '''
    FDA Method's Approach :
        F`(X) ≈ [F(X + ΔX) - F(X)] / ΔX
            Where ΔX Is Equal To The Value Of Size Of Step.
    '''
    if(int(func_scale.get()) == 1):
        '''
        Since The Derivative Of exp(X) Is Equal To Itself, Both Results Will Be Extremely Irrelevant Towards Each Other.
        No Form Of Correlation Is Shown, Thus The Value Of Abs. Relative True Error Will Be Extremely High.
        '''
        # Calculations.
        exact_value = np.exp(float(x_val_entry.get()))
        appr_value = (np.exp((float(x_val_entry.get()) + float(delta_x_entry.get()))) - exact_value) / float(delta_x_entry.get())
        abs_error = str(round(np.abs((exact_value - appr_value) / exact_value) * float(100.0), 5)) + " %"

        self.clear_all()

        # Insertions Where Results Are Rounded Off To At Most 5 Decimal Points.
        exact_entry.insert(self.__length_of_exact_entry, str(round(exact_value, 5)))
        appr_entry.insert(self.__length_of_appr_entry, str(round(appr_value, 5)))
        abs_error_entry.insert(self.__length_of_abserror_entry, abs_error)
```

1. Differentiation of the functions
2. Input the x value to the derived function
3. Approximate value of f(x)
4. Compare approx. value with exact result and calculate error percentage