In [1]:
```python
from numpy import * #That takes all definitions from the module and places them into your current namespace
from sympy import * #That takes all definitions from the module and places them into your current namespace
from sympy import init_printing #enable best printer available in ur enviornment
import math
```

In [2]:
```python
#defining the variable and function which need to be used  further
x = Symbol('x')
```

In [3]:
```python
fun = exp(x)
```

In [4]:
```python
fun
```

Out[4]: e^x

In [5]:
```python
# definition for expressing any function in the term of taylor series
def tay_ser_def(fun,order,value):
    # value = about which value fun need to be expand
    # order = upto what order fun need to expressed in term of series
    #fun    = any function which need to be expressed in the term of taylor series
    a          = value
    fun_tay     =0
    for i in range (order):
        fun_tay = fun_tay+  diff(fun, (x,i)).subs(x,a)*(x-a)**(i)/factorial(i)
    return fun_tay
```

In [6]:
```python
tay_ser_def(fun,10,0)
```

Out[6]:
$$\frac{x^9}{362880} + \frac{x^8}{40320} + \frac{x^7}{5040} + \frac{x^6}{720} + \frac{x^5}{120} + \frac{x^4}{24} + \frac{x^3}{6} + \frac{x^2}{2} + x + 1$$

In [17]:
```python
# Python program to implement Runge Kutta method
# A sample differential equation "dy / dx = (x - y)/2"
def dydx(x, y):
    return x+y

# Finds value of y for a given x using step size h
# and initial value y0 at x0.
def rungeKutta(x0, y0, x, h):
    # Count number of iterations using step size or
```

```
    # step height h
    n = (int)((x - x0)/h)
    # Iterate for number of iterations
    y = y0
    for i in range(1, n + 1):
        "Apply Runge Kutta Formulas to find next value of y"
        k1 = n * dydx(x0, y)
        k2 = n * dydx(x0 + 0.5 * h, y + 0.5 * k1)
        k3 = n * dydx(x0 + 0.5 * h, y + 0.5 * k2)
        k4 = n * dydx(x0 + h, y + k3)

        # Update next value of y
        y = y + (1.0 / 6.0)*(k1 + 2 * k2 + 2 * k3 + k4)

        # Update next value of x
        x0 = x0 + h
    return y
```

In [20]:

```
# Driver method
x0 = 0
y0 = 1
x = 1
n = 2
print ('The value of y at x is:'), rungeKutta(x0, y0, x, n)
```

```
The value of y at x is:
```

Out[20]:

```
(None, 1)
```

In [10]:

```
print('Enter initial conditions:')
x0 = float(input('x0 = '))
y0 = float(input('y0 = '))
```

```
Enter initial conditions:
x0 = 0
y0 = 1
```

In [15]:

```
print('Enter calculation point: ')
x = float(input('x = '))

print('Enter number of steps:')
h = int(input('h = '))
```

```
Enter calculation point:
x = 1
```

```
Enter number of steps:
h = 2
```

In [24]:

```python
# RK-4 method python program

# function to be solved
def f(x,y):
    return x+y

# or
# f = lambda x: x+y

# RK-4 method
def rk4(x0,y0,xn,n):

    # Calculating step size
    h = (xn-x0)/n

    print('\n--------SOLUTION--------')
    print('-------------------------')
    print('x0\ty0\tyn')
    print('-------------------------')
    for i in range(n):
        k1 = h * (f(x0, y0))
        k2 = h * (f((x0+h/2), (y0+k1/2)))
        k3 = h * (f((x0+h/2), (y0+k2/2)))
        k4 = h * (f((x0+h), (y0+k3)))
        k = (k1+2*k2+2*k3+k4)/6
        yn = y0 + k
        print('%.4f\t%.4f\t%.4f'% (x0,y0,yn) )
        print('-------------------------')
        y0 = yn
        x0 = x0+h

    print('\nAt x=%.4f, y=%.4f' %(xn,yn))

# Inputs
print('Enter initial conditions:')
x0 = float(input('x0 = '))
y0 = float(input('y0 = '))

print('Enter calculation point: ')
xn = float(input('xn = '))
```

```python
print('Enter number of steps:')
step = int(input('Number of steps = '))

# RK4 method call
rk4(x0,y0,xn,step)
```

```
Enter initial conditions:
x0 = 0
y0 = 1
Enter calculation point:
xn = 1
Enter number of steps:
Number of steps = 2

--------SOLUTION--------
-------------------------
x0      y0      yn
-------------------------
0.0000  1.0000  1.7969
-------------------------
0.5000  1.7969  3.4347
-------------------------

At x=1.0000, y=3.4347
```

In [21]:

```python
\t??
```

```
Object `t` not found.
```

In [ ]:

```python
\t
```

In [ ]:

```python

```