Taylor approximation

Taylor Expansion에 대해 이해한다. 어떤 함수가 주어질 때 특정 점에서 이 함수에 대해 근사를 하는 것이다. 이를 특정한 점에서의 first order taylor approximation $\hat{f}(x) = f(z) + f'(z)(x - z)$ 로 정의가 된다.

import library

In [1]:

```
import numpy as np
import matplotlib.image as img
import matplotlib.pyplot as plt
from matplotlib import cm
import matplotlib.colors as colors
```

define a function $f(x) = cos(x)$

x가 input일 때 output y cosx가 output y가 되도록한다.

In [2]:

```
def function(x):

    # ++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    y = np.cos(x)
    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++

    return y
```

define the derivative $f'(x)$ of function $f(x)$

In [3]:

```
def derivative_function(x): # cos(x) 미분함수

    # ++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    y_prime = - np.sin(x)
    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++

    return y_prime
```

define the first order Taylor approxation of the function at x_0

- $\hat{f}(x) = f(x_0) + f'(x_0)(x - x_0)$

In [4]:

```
def approximate_function(x, x0):

    # ++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    y_hat = function(x0) + derivative_function(x0) * (x-x0)
    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++

    return y_hat
```

functions for presenting the results

In [5]:

```
def function_result_01():

    x = np.linspace(-10, 10, 100)
    y = function(x)

    plt.figure(figsize=(8,6))
    plt.plot(x, y, 'b')
    plt.xlim([-10, 10])
    plt.ylim([-10, 10])
    plt.show()
```

function_result_01() : $cos(x)$

In [6]:

```
def function_result_02():

    x       = np.linspace(-10, 10, 100)
    y_prime = derivative_function(x)

    plt.figure(figsize=(8,6))
    plt.plot(x, y_prime, 'r')
    plt.xlim([-10, 10])
    plt.ylim([-10, 10])
    plt.show()
```

function_result_02() : $cos(x)$ 미분함수 그리기, $-sin(x)$

In [7]:

```
def function_result_03():

    x = np.linspace(-10, 10, 100)
    y = function(x)

    x0      = 1
    y0      = function(x0)
    y_hat   = approximate_function(x, x0)

    plt.figure(figsize=(8,6))
    plt.plot(x, y, 'b')
    plt.plot(x, y_hat, 'r')
    plt.plot(x0, y0, 'go')
    plt.xlim([-10, 10])
    plt.ylim([-10, 10])
    plt.show()
```

function_result_03() : x에 대해 $cos(x)$, 1에서의 근사함수 그리기

In [8]:

```
def function_result_04():

    x1      = -1
    x2      = 1
    value1  = function(x1)
    value2  = function(x2)

    print('value1 = ', value1)
    print('value2 = ', value2)
```

function_result_04() : -1, 1에 대한 $cos(x)$ 값 출력

In [9]:

```
def function_result_05():

    x1      = -1
    x2      = 1
    value1  = derivative_function(x1)
    value2  = derivative_function(x2)

    print('value1 = ', value1)
    print('value2 = ', value2)
```

function_result_05() : -1, 1에 대해 $cos(x)$ 미분 함수에 대한 값 출력

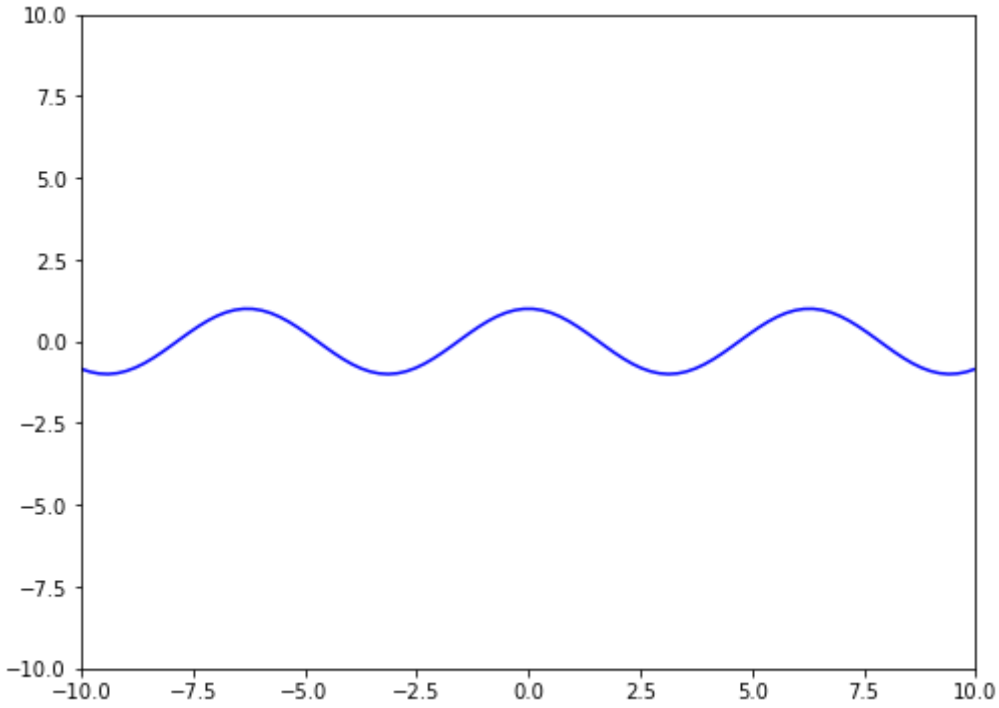
Define function result 01 - 05

results

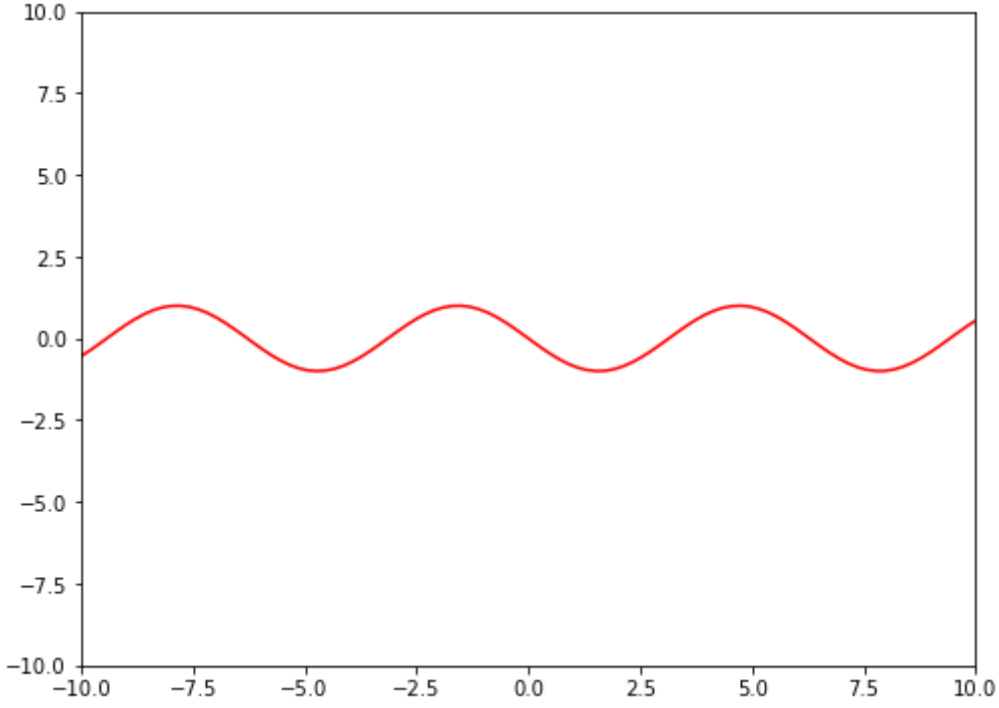
In [10]:

```
number_result = 5

for i in range(number_result):
    title = '## [RESULT {:02d}]'.format(i+1)
    name_function = 'function_result_{:02d}()'.format(i+1)

    print('************************************************')
    print(title)
    print('************************************************')
    eval(name_function)
```

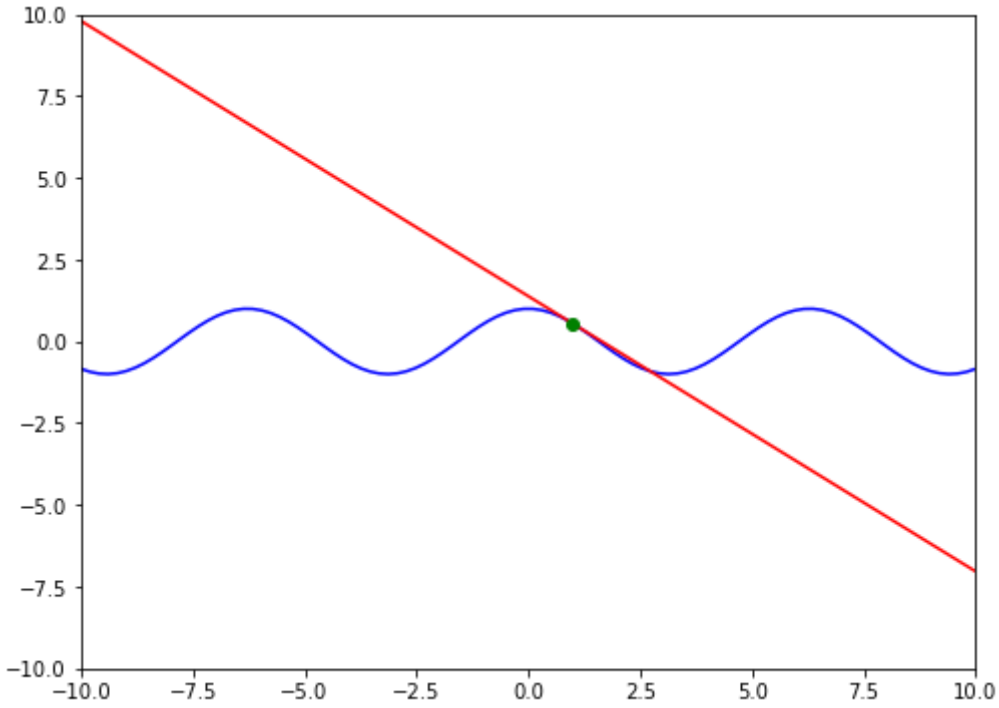
```
******************************************************
## [RESULT 01]
******************************************************
```




```
******************************************************
## [RESULT 02]
******************************************************
```



```
******************************************************
## [RESULT 03]
******************************************************
```




```
******************************************************
## [RESULT 04]
******************************************************
value1 =  0.5403023058681398
value2 =  0.5403023058681398
******************************************************
## [RESULT 05]
******************************************************
value1 =  0.8414709848078965
value2 =  -0.8414709848078965
```