Taylor approximation

import library

In [ ]:

```
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 $f(x) = cos(x)$

In [ ]:

```
def function(x):

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

    return y
```

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

In [ ]:

```
def derivative_function(x):

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    fnt = x
    y_prime = np.gradient(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 [ ]:

```
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 [ ]:
```
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()
```

In [ ]:
```
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()
```

In [ ]:
```
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()
```

In [ ]:
```
def function_result_04():

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

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

In [ ]:
```
def function_result_05():

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

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

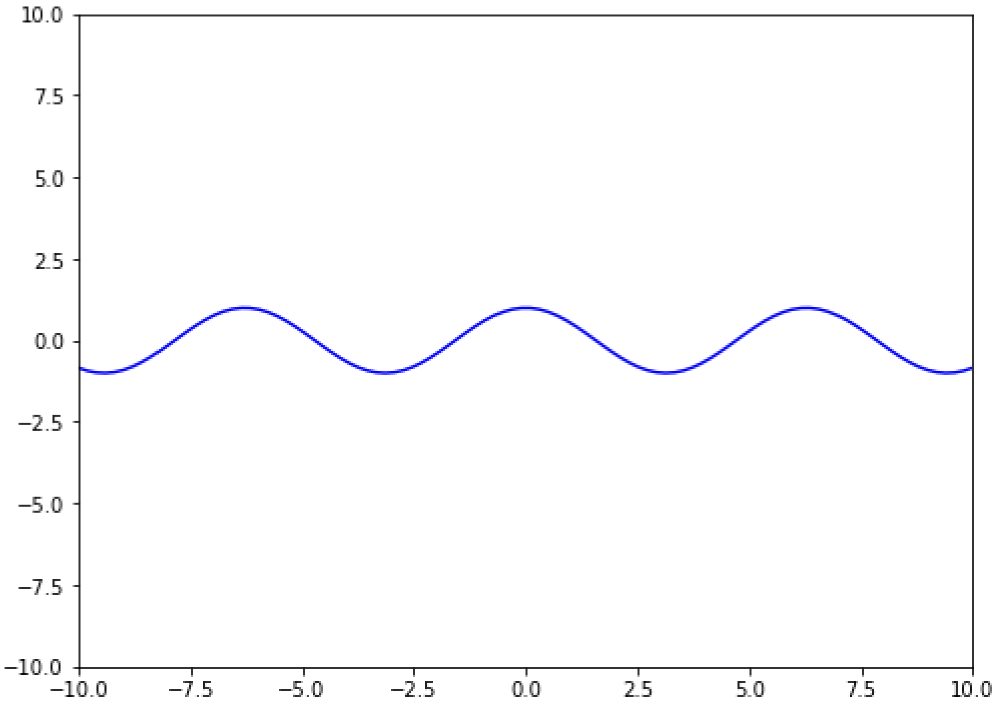
---

results

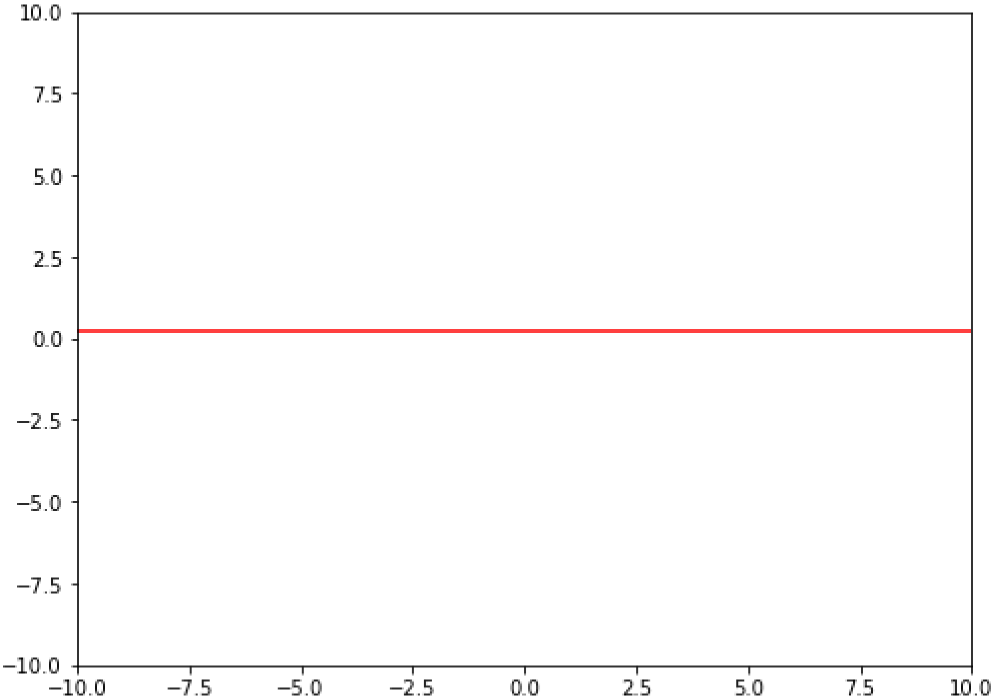
In [ ]:

```
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]
**************************************************
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-23-b732c121165c> in <module>
      8     print(title)
      9     print('**************************************************')
---> 10     eval(name_function)

<string> in <module>

<ipython-input-10-5b347b716b23> in function_result_03()
      6     x0      = 1
      7     y0      = function(x0)
----> 8     y_hat   = approximate_function(x, x0)
      9
     10     plt.figure(figsize=(8,6))

<ipython-input-21-50f17bee3c26> in approximate_function(x, x0)
      4     # complete the blanks
      5     #
----> 6     y_hat = function(x0) + derivative_function(x0)(x-x0)
      7     #
      8     # ++++++++++++++++++++++++++++++++++++++++++++++++++

TypeError: 'list' object is not callable
```

In [ ]: