Assignment 02

그래프를 그리기 위해 **Python3**의 **sympy, numpy,matplotlib module**을 사용합니다.

모듈 정의

In [13]:
```
from sympy import *
import sympy
import numpy
import matplotlib.pyplot as plt
```

1. Define a differentiable function that maps from real number to real number.

- 실수에서 실수로 매핑하는 미분가능함수를 정의합니다.

  $f(x) = \sin$
  x

2. Define a domain of the function.

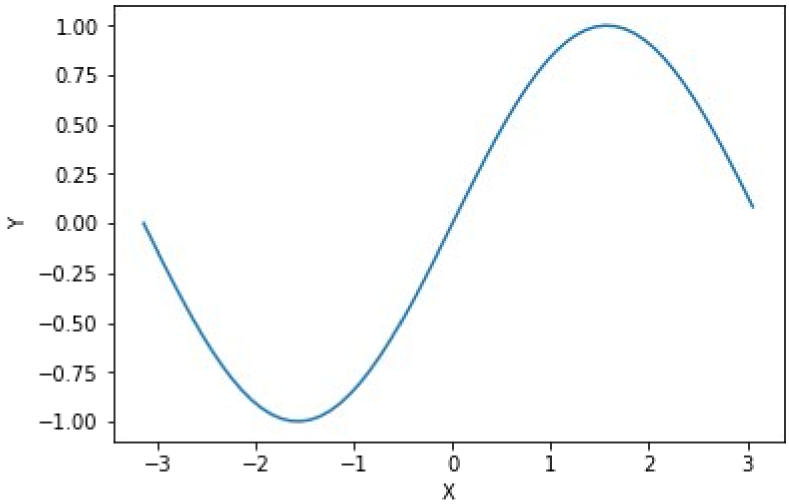
- 도메인을 정의합니다.

- 정의역은 $[-2\pi, 2\pi]$ 으로 정의합니다.

3. Plot the function.

- 그래프로 보입니다.

In [83]:
```
plt.xlabel("X")
plt.ylabel("Y")
x = numpy.arange(-1 * pi , pi , 0.1)
y = [sympy.sin(v) for v in x]
plt.plot(x, y)
plt.show()
```



4. Select a point within the domain.

- 도메인안에 점 하나를 고릅니다.

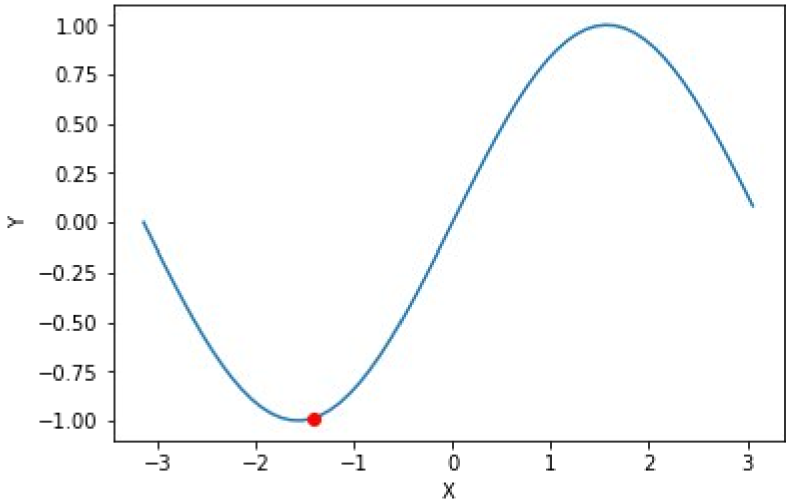
- $x = -\sqrt{2}$ 을 고릅니다

In [57]:
```
select_point_x = sympy.sqrt(2) * -1
select_point_y = sympy.sin(select_point_x)
```

5. Mark the selected point on the function.

- 선택된 점을 함수위에 마킹합니다.

In [82]:
```
plt.xlabel("X")
plt.ylabel("Y")
x = numpy.arange(-1 * pi , pi , 0.1)
y = [sympy.sin(v) for v in x]
plt.plot(x, y)
plt.plot(select_point_x, select_point_y,'ro')
plt.show()
```



6. Define the first-order Taylor approximation at the selected point.

- 선택된 점에 Taylor approximation을 정의합니다.

- first-order Taylor approximation of f , near point z:
  - $\hat{f}(x)$
    $= f(z)$
    $+ \frac{\partial f}{\partial x_1}(x_1$
    $- z_1) + \cdots$
    $+ \frac{\partial f}{\partial x_n}(x_n$
    $- z_n)$
  - $\hat{f}(x)$ is very close to $f(x)$ when x_i are all near z_i
  - $\hat{f}(x)$ is an affine function of x

- inner product을 사용해 다시 아래 처럼 쓸 수 있다.
  - $\hat{f}(x) = f(z) + \nabla f(z)^T (x - z)$
  - $\nabla f(z) = \left(\frac{\partial f}{\partial x_1}(z) + \cdots + \frac{\partial f}{\partial x_n}(z)\right)$

sin(x) 에대한 Taylor approximation 함수를 만듭니다.

In [85]:
```
diff_x,at_point =symbols('x point') #변수 정의
TaylorApproximation = sympy.sin(at_point) + (diff(sympy.sin(at_point), at_poi
nt) * (diff_x - at_point))
print(TaylorApproximation)
```

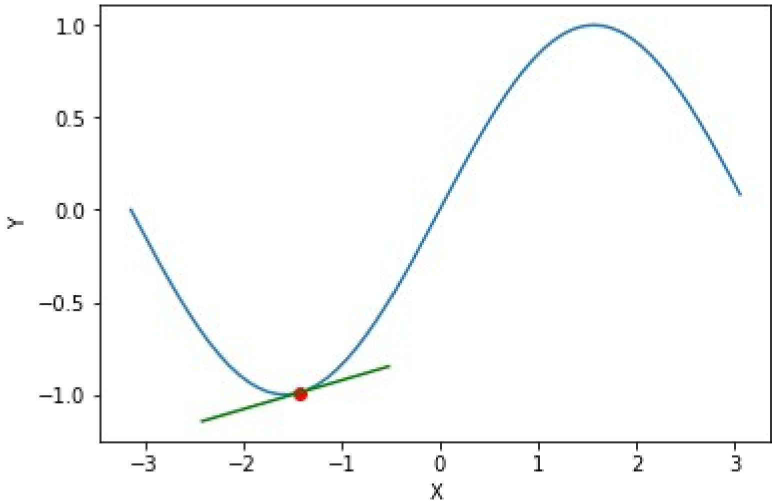
```
(-point + x)*cos(point) + sin(point)
```

선택된 점에 대한 Taylor approximation 함수를 [select_point_x - 1, select_point_x + 1]에 대해서 보입니다.

In [84]:
```
plt.xlabel("X")
plt.ylabel("Y")
x = numpy.arange(-1 * pi , pi , 0.1)
y = [sympy.sin(v) for v in x]

x_ = numpy.arange(select_point_x - 1, select_point_x + 1, 0.1)
y_ = [ TaylorApproximation.subs([(diff_x, v),(at_point,select_point_x)]).eval
f() for v in x_]

plt.plot(x, y)
plt.plot(select_point_x, select_point_y,'ro')
plt.plot(x_, y_,'g')
plt.show()
```



7. Plot the Taylor approximation with the same domain of the original function.

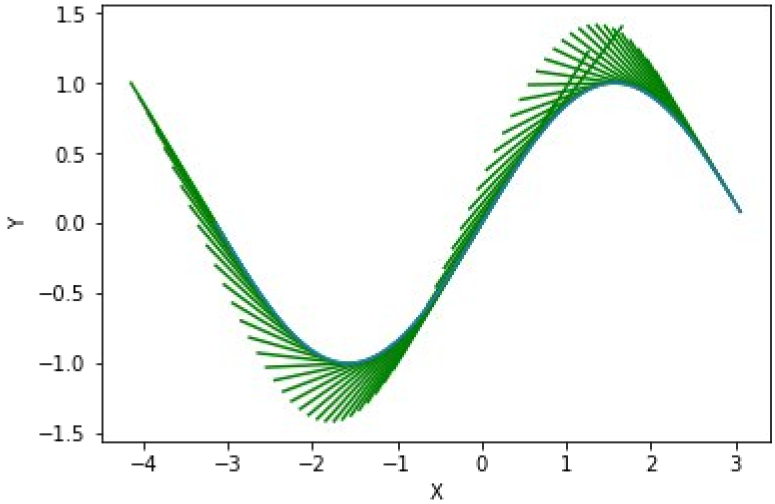
- 원래 함수의 같은 도메인으로 Taylor approximation를 그래프로 보입니다.

In [80]:

```
plt.xlabel("X")
plt.ylabel("Y")
x = numpy.arange(-1 * pi , pi , 0.1)
y = [sympy.sin(v) for v in x]

for domain in x:
    x_ = numpy.arange(domain - 1, domain + 1)
    y_ = [ TaylorApproximation.subs([(diff_x, v),(at_point,domain)]).evalf()
for v in x_]
    plt.plot(x_, y_,'g')


plt.plot(x, y)
plt.show()
```



In [ ]: