Question 1

```
# Q1 by bisection method

# defining a function to return the given equation's values
def f1(x):
  return (x**3 - 2*x - 5)

# now defining the bisection method function
def bisection(a, b):
  x = (a + b)/2
  if abs(a - b) < 0.000001:
    return x
  else:
    if f1(a)*f1(x) < 0:
      return bisection(a, x)
    else:
      return bisection(x, b)

# input
x = bisection(2, 3) #choosing a=2, b=3 as f(a)*f(b) < 0 meaning there is a real root in (2,3)

print("Root of the given eqn using bisection method: %.6f" % x)
# output is below:

     Root of the given eqn using bisection method: 2.094552


# Q1 by Newton-Raphson method

# defining the derivative of f(x)
def f_derv(x):
  return (3*(x**2) - 2)

# now defining the Newton-Raphson function:
def newton_raphson(x0):
    x = x0
    while True:
        h = -f1(x)/f_derv(x)
        x += h
        print('x = %0.6f and f(x) = %0.6f' % (x, f1(x)))
        if (abs(h) <= 0.000001):
          break

    print("Root is : ", "%.6f"% x)

# input
x0 = 0.5  # choosing 0.5 as the initial guess

newton_raphson(x0)
```

```
# Output is below:

     x = -4.200000 and f(x) = -70.688000
     x = -2.811783 and f(x) = -21.606742
     x = -1.816923 and f(x) = -7.364198
     x = -0.885174 and f(x) = -3.923215
     x = 10.304820 and f(x) = 1068.652065
     x = 6.929076 and f(x) = 313.821367
     x = 4.719632 and f(x) = 90.690172
     x = 3.320627 and f(x) = 24.973845
     x = 2.517084 and f(x) = 5.913359
     x = 2.169386 and f(x) = 0.870869
     x = 2.097524 and f(x) = 0.033238
     x = 2.094556 and f(x) = 0.000055
     x = 2.094551 and f(x) = 0.000000
     x = 2.094551 and f(x) = -0.000000
     Root is :  2.094551
```

The bisection method has a linear rate of convergence whereas Newton-Raphson method has a quadratic rate of convergence hence the Newton-Raphson method converges faster. The advantage of bisection method is that it is guarenteed to converge even though it may converge slowly, whereas Newton-Raphson method can fail due to some reasons like if the derivative of the function at any point in the interval containing the root is zero and also we have to decide the initial guess and it may happen that the tangent becomes parallel to the x-axis and may not converge.

Question 2

```
# Q2 by Secant Method

from math import cos

def f2(x):
  return (cos(x) - 3*x + 1)

# Secant method function:
def Secant(x0, x1):
    y = x0
    x = x1
    while True:
        h = -f2(x)*(x - y)/(f2(x) - f2(y))
        y = x
        x += h
        print('x = %0.6f and f(x) = %0.6f' % (x, f1(x)))
        if (abs(h) <= 0.000001):
          break

    print("Root is : ", "%.6f"% x)

# input
x0,x1 = 0, 1  # taking x0 as 0 and x1 as 1
```

```
Secant(x0, x1)
# Output is below:

     x = 0.578085 and f(x) = -5.962984
     x = 0.605959 and f(x) = -5.989418
     x = 0.607106 and f(x) = -5.990446
     x = 0.607102 and f(x) = -5.990442
     x = 0.607102 and f(x) = -5.990442
     Root is :  0.607102


# Q2 by Regular Falsi method

# defining the regular falsi function
def regular_falsi(a, b):
  while True:
    h  = (b - a)*abs(f2(a))/(abs(f2(a)) + abs(f2(b)))
    if f2(a+h)*f2(b) < 0:
      a += h
    else:
      b -= h
    print("a = %0.6f, b = %0.6f and f(a) = %0.6f, f(b) = %0.6f" % (a, b, f2(a), f2(b)))
    if abs(h) <= 0.000001:
      break

  if f2(a) < 0.00001:
    print("Root is: ", "%.6f"% a)
  else:
    print("Root is: ", "%.6f"% b)

# Input
a, b = 0, 1  #choosing a=1,b=0 as f(1)<0 and f(0)>0 and f(1)f(0)<0 meaning a root is in (0,1)

regular_falsi(a, b)
# Output is below

     a = 0.578085, b = 1.000000 and f(a) = 0.103255, f(b) = -1.459698
     a = 0.605959, b = 1.000000 and f(a) = 0.004081, f(b) = -1.459698
     a = 0.607057, b = 1.000000 and f(a) = 0.000159, f(b) = -1.459698
     a = 0.607100, b = 1.000000 and f(a) = 0.000006, f(b) = -1.459698
     a = 0.607102, b = 1.000000 and f(a) = 0.000000, f(b) = -1.459698
     a = 0.607102, b = 1.000000 and f(a) = 0.000000, f(b) = -1.459698
     Root is:  0.607102
```