Question 1

```
# using log base 10 in the questions
from math import log

def trapezoidal(a, b, n):
  # initialising the integral by adding in values of y(a) and y(b)
  I = log(a, 10) + log(b, 10)

  # the spacing
  h = (b - a)/n

  # adding the values from y_1 to y_n-1
  for i in range(1, n):
    I += 2*(log(a+i*h, 10))

  I *= h/2
  return I

# input (a is lower limit, b is upper limit, n is the no. of intervals)
a, b, n = 4, 5.2, 10
print("Definite integral using trapezoidal rule is:", trapezoidal(a,b,n))
# Output below
```

```
Definite integral using trapezoidal rule is: 0.7937939788929803
```

```
# using simpson's 1/3 rule

def simpsons_one_third_rule(a, b, n):
  # initialising the integral to 0
  I = 0

  # the spacing
  h = (b - a)/n

  # storing the values of x and y in these empty lists
  x = list()
  y = list()
  for i in range(n+1):
    x.append(a + i*h)
    y.append(log(x[i], 10))

  # now calculating the integral I by adding the terms in
  I += y[0] + y[n]
```

```
  for i in range(1, n):
    if i % 2 != 0:
      I += 4 * y[i]
    else:
      I += 2 * y[i]

  I *= (h / 3)
  return I

# input (a is lower limit, b is upper limit, n is the no. of intervals)
a, b, n = 4, 5.2, 10

print("Definite integral using Simpson's 1/3rd rule is:", simpsons_one_third_rule(a, b, n))
# Output below
Definite integral using Simpson's 1/3rd rule is: 0.7938240348015837
```

Question 2

```
import numpy as np

def newton_cotes(a, b, n):
  # the spacing
  h = (b - a)/n

  # creating an array initialised to zeros to store Σf(x)*c^n values
  C = np.zeros((n+1,))

  for i in range(n+1):
    # initialising the lagrange interpolating polynomial to 1
    p = np.poly1d([0, 1])

    # initialising the denominator in the L(x) expression to 1
    deno = 1

    # finding the values of x in each iteration
    x = a + i*h

    # running a for loop to calculate the integral of all lagrange interpolating
    # values at the respective i values
    for j in range(n+1):
      if j != i:
        # creating a polynomial (x - 1*j)
```

```
        p1 = np.poly1d([1, -1*j])

        # multiplying the above polynomial with the initial polynomial until we
        # get the lagrange interpolating poly
        p = np.polymul(p, p1)

        # calculating the denominator term also ((i*(i-1)*(i-2)...*(i-n)) term)
        deno *= (i - j)

    # calculating the integral from 0 to n on the lagrange polynomial calculated
    I = p.integ()
    val = (I(n) - I(0))/deno

    val /= n
    C[i] = val * log(x, 10)

  sum = np.sum(C)
  return (b - a)*sum


# input
a, b, n = 4, 5.2, 10
print("Definite integral using Newton-Cotes method is:", newton_cotes(a, b, n))
# Output below
```

```
Definite integral using Newton-Cotes method is: 0.7938240432012428
```