Question 1

```
def driver_Q1():
    '''
    Demonstrating approximation via interpolation
    '''
    f = lambda x: 1/(1+(10*x)**2)

    n = 19 # number of sample points to use

    # choose a basis - comment out the other


    x_i=lambda i: -1+(i-1)*(2/(n-1))
    xs=[]
    ys=[]
    for i in range(1,n+1):
        xs.append(x_i(i))
        ys.append(f(x_i(i)))
    basis = monomial_basis(n)

    coeffs = get_interpolation_coefficients(xs, ys, basis)
    polynomial_text = ' + '.join([f'{c:.2f}x^{i}' for i, c in enumerate(coeffs)])

    # test the function on a finer grid
    zs = np.linspace(xs[0], xs[-1], 1001)
    zs_eval = interp_eval(zs, coeffs, basis)
    print(zs_eval)

    plt.figure(3, figsize=(16,6))
    r=np.linspace(-1,1,1001)
    plt.plot(r, f(r), label='True function')
    plt.plot(zs, zs_eval, label='Interpolating Polynomial')
    plt.plot(xs, ys, 'k.', label='sample points')
    plt.text(-.5, .5,  'n=19')
    #plt.text(-1, -1, '$' + polynomial_text + '$')

    plt.legend(bbox_to_anchor=(1.04,1), loc="upper left")
    plt.title('Polynomial approximation by interpolation')
    #plt.show()

def monomial_basis(n):
    return [lambda x, i=i: x**i for i in range(n)]


def get_interpolation_coefficients(xs, ys, basis):
```

```
    M = np.array([[phi(x) for phi in basis] for x in xs])
    return la.solve(M, ys)

def interp_eval(zs, coeffs, basis):
    return sum(c*phi(zs) for c, phi in zip(coeffs, basis))

driver_Q1()
```

Question 2

```
def driver_Q2():
    '''
    Demonstrating approximation via interpolation
    '''
    f = lambda x: 1/(1+(10*x)**2)

    n = 100 # number of sample points to use

    # choose a basis - comment out the other

    x_i=lambda i: -1+(i-1)*(2/(n-1))
    xs=[]
    ys=[]
    for i in range(1,n+1):
        xs.append(x_i(i))
        ys.append(f(x_i(i)))
    basis = lagrange_basis(xs)

    coeffs = get_interpolation_coefficients(xs, ys, basis)
    polynomial_text = ' + '.join([f'{c:.2f}x^{i}' for i, c in enumerate(coeffs)])

    # test the function on a finer grid
    zs = np.linspace(xs[0], xs[-1], 1001)
    zs_eval = interp_eval(zs, coeffs, basis)
    print(zs_eval)

    plt.figure(3, figsize=(16,6))
    r=np.linspace(-1,1,1001)
    plt.plot(r, f(r), label='True function')
    plt.plot(zs, zs_eval, label='Interpolating Polynomial')
    plt.plot(xs, ys, 'k.', label='sample points')
    plt.text(-.5, .5,  'n=100')
    #plt.text(-1, -1, '$' + polynomial_text + '$')

    plt.legend(bbox_to_anchor=(1.04,1), loc="upper left")
    plt.title('Polynomial approximation by interpolation')
    #plt.xlim(-.5,.5)
    #plt.ylim(-1,2)
    #plt.show()

def lagrange_polynomial(x, x_i, other_points):
    # f_i(x_j) = 0 for i =/= j
    # f_j(x_j) = 1
    # phi(x) = product of (x - x_j)/(x_i - x_j)
```

```
    y = 1 # start with 1 when making a product in a loop
    for x_i in other_points:
        #y *= (x - x_j)/(x_i - x_j)
        y *= (x-x_i)
    return y

def lagrange_basis(points):
    return [lambda x, x_i=x_i, other_points=remove_point_from_list(x_i, points):
                lagrange_polynomial(x, x_i, other_points)
            for x_i in points]

def get_interpolation_coefficients(xs, ys, basis):
    M = np.array([[phi(x) for phi in basis] for x in xs])
    return la.solve(M, ys)

def interp_eval(zs, coeffs, basis):
    return sum(c*phi(zs) for c, phi in zip(coeffs, basis))

driver_Q2()
```

Question 3

```
def driver_Q3():
    '''
    Demonstrating approximation via interpolation
    '''
    f = lambda x: 1/(1+(10*x)**2)

    n = 400 # number of sample points to use

    # choose a basis - comment out the other


    x_i=lambda i: np.cos(((2*i-1)*math.pi)/(2*n))
    xs=[]
    ys=[]
    for i in range(1,n+1):
        xs.append(x_i(i))
        ys.append(f(x_i(i)))

    basis = monomial_basis(n)

    coeffs = get_interpolation_coefficients(xs, ys, basis)
    polynomial_text = ' + '.join([f'{c:.2f}x^{i}' for i, c in enumerate(coeffs)])

    # test the function on a finer grid
    zs = np.linspace(xs[0], xs[-1], 1001)
    zs_eval = interp_eval(zs, coeffs, basis)
    print(zs_eval)

    plt.figure(3, figsize=(16,6))
    r=np.linspace(-1,1,1001)
    plt.plot(r, f(r), label='True function')
    plt.plot(zs, zs_eval, label='Interpolating Polynomial')
    plt.plot(xs, ys, 'k.', label='sample points')
    plt.text(-.5, .5,  'n=400')

    plt.legend(bbox_to_anchor=(1.04,1), loc="upper left")
    plt.title('Polynomial approximation by interpolation')

def monomial_basis(n):
    return [lambda x, i=i: x**i for i in range(n)]

def get_interpolation_coefficients(xs, ys, basis):
    M = np.array([[phi(x) for phi in basis] for x in xs])
    return la.solve(M, ys)
```

```
def interp_eval(zs, coeffs, basis):
    return sum(c*phi(zs) for c, phi in zip(coeffs, basis))

driver_Q3()
```