Question 1
Recall that the factorial function $n!$ returns $n! = n \times (n-1) \times ...2 \times 1$. Write a function that executes this called *factorial2* that uses a for loop.

Question 2
Consider the polynomial

$$p(x) = a_0 + a_1 x + a_2 x^2 + ...a_n x^n = \sum_{i=0}^{n} a_i x_i$$

Using *enumerate*() in your loop, write a function p such that $p(x, coeff)$ computes the value of the polynomial given a point x and an array of coefficients $coeff$.

Question 3
Compute an approximation for π using Monte Carlo. You may only use rand() for random number generation. **Hint:** If U is a bivariate uniform random variable on the unit square $(0,1)^2$, then the probability that U lies in a subset B of $(0,1)^2$ is equal to the area of B.

Question 4
Let the data generating process for y be

$$y = ax_1 + bx_1^2 + cx_2 + d + \sigma w,$$

where y, x_1, x_2 are scalars, a, b, c, d are parameters to estimate, and $w \sim N(0,1)$ iid.

- First, draw $N = 50$ values for x_1, x_2 from iid normal distributions.
- Then, draw a w vector for the N values to generate y for the simulated data. Use $a = 0.1$, $b = 0.2$, $c = 0.5$, $d = 1.0$, and $\sigma = 0.1$. Repeat this until you have 20 different simulations of the y and N values.
- Finally, calculate ordinary least squares manually for each of the 20 simulations. Plot histograms for each parameter $a, b, c, d,$.

Question 5
Take a random walk starting from $x_0 = 1$:

$$x_{t+1} = \alpha x_t + \sigma \epsilon_{t+1},$$

where $t = 0, ...t_{max}$. Assume that $x_{t_{max}} = 0$ with certainty and that $\{\epsilon_t\}$ is drawn from an iid standard normal. Start with $\sigma = 0.2$ and $\alpha = 1.0$. For a given path $\{x_t\}$ define a first-passage time as $T_a = \min\{t | x_t \leq a\}$.

- Calculate the first-passage time T_0 for 100 simulated random walks (up to $t_{max} = 200$) and plot a histogram.
- Plot the sample mean of T_0 from the simulation for $\alpha \in \{0.8, 1.0, 1.2\}$.

Question 6
Recall that the root of a univariate function $f(\cdot)$ is an x such that $f(x) = 0$. One solution method to find local roots of smooth functions is called Newton's method. Starting with an x_0 guess, a function $f(\cdot)$ and a first derivative $f'(\cdot)$, the algorithm is to repeat

$$x^{n+1} = x^n - \frac{f(x^n)}{f'(x^n}$$

until $|x^{n+1} - x^n|$ is below some tolerance threshold. Code a function that implements Newton's method. The function should accept arguments a function f, its derivative f_prime, a starting guess x_0, a tolerance tol, and a maximum number of iterations maxiter. Test this function with $f(x) = (x-1)^3$ and another function of your choice where you can analytically find the derivative.

Question 7
We consider the capital investment problem of an infinitely lived household with log preferences over consumption. Production is given by $Y_t = Z_t K_t^\theta$, where K_t is current capital, $\theta = 0.36$, and $Z_t \in \{Z_g, Z_b\}$ is the period's current productivity level. Assume that capital depreciates at rate $\delta = 0.025$, and households discount at rate $\beta = 0.99$. Assume further that the states of productivity are $Z_g = 1.25, Z_b = 0.2$, and that transitions between the two states are given by a two-state Markov process:

$$\Pi = \begin{bmatrix} 0.977 & 0.023 \\ 0.074 & 0.926 \end{bmatrix},$$

so, for instance, $P(Z_{t+1} = Z_g | Z_t = Z_g) = 0.977$. The household's value function $V(K, Z)$ is given by

$$V(K, Z) = \max_{K'} \left\{ u(c) + \beta \sum_{Z'} V(K', Z') \Pi(Z', Z) \right\};$$
$$c = ZK^\theta + (1-\delta)K - K'.$$

Set the capital grid to be from 1 to 75 with 100 grid points.

- Write code to solve for value functions and policy functions. Write this code sub-optimally, so that its speed could be improved. Plot the functions and create a Github repository containing your code.
- Pair up with a different member of the class. Clone their repository and optimize the speed of their code. Send a pull request to their repository.
- Once your partner has done the same for your code, review and commit their changes.