How to cook the perfect egg

As an egg cooks, the proteins first denature and then coagulate. When the temperature exceeds a critical point, reactions begin and proceed faster as the temperature increases. In the egg white the proteins start to coagulate for temperatures above 63°C, while in the yolk the proteins start to coagulate for temperatures above 70°C. For a soft boiled egg, the white needs to have been heated long enough to coagulate at a temperature above 63°C, but the yolk should not be heated above 70°C. For a hard boiled egg, the center of the yolk should be allowed to reach 70°C. The following formula expresses the time t it takes (in seconds) for the center of the yolk to reach the temperature T_y (in Celsius degrees):

$$t = \frac{M^{2/3} c \rho^{1/3}}{K \pi^2 (4\pi/3)^{2/3}} \ln\left[0.76 \frac{T_o - T_w}{T_y - T_w}\right]$$

Here, M, ρ, c and K are properties of the egg: M is the mass, ρ is the density, c is the specific heat capacity, and K is thermal conductivity. Relevant values are M = 47 g for a small egg and M = 67 g for a large egg, ρ = 1.038 g cm^{-3}, c = 3.7 J g^{-1}K^{-1}, and K = 5.4 $\times$ 10^{-3} W cm^{-1} K^{-1}. Furthermore, T_w is the temperature (in C degrees) of the boiling water, and T_o is the original temperature (in C degrees) of the egg before being put in the water.

Implement the formula in a program, set T_w = 100°C and T_y = 70°C, and compute t for a small and large egg taken from the fridge (T_o = 4°C) and from room temperature (T_o = 20°C).

```
from math import pi, log

Tw = 100    # C Temperature of the water
Ty = 70     # C Desired temperature of the yolk
rho = 1.038 # g cm^{-3}
M = 67      # g
K = 5.4e-3  # W cm^{-1} K^{-1}
c = 3.7     # J g^{-1} K^{-1}

for To in [4, 20]:
    for M in [47, 67]:
        numerator = (M ** (2/3)) * c * (rho ** (1/3))
        denominator = K * (pi ** 2) * (4 * pi / 3) ** (2/3)
        logarithm = log(0.76 * (To - Tw) / (Ty - Tw))
        t = numerator / denominator * logarithm
        print("Time required for perfect {} g egg when To = {} C is {:.2f} seconds.".format(M, To, t))
```

```
Time required for perfect 47 g egg when To = 4 C is 313.09 seconds.
Time required for perfect 67 g egg when To = 4 C is 396.58 seconds.
Time required for perfect 47 g egg when To = 20 C is 248.86 seconds.
Time required for perfect 67 g egg when To = 20 C is 315.22 seconds.
```

2. Given a matrix, for example: [[5, 6, 7], [8, 3, 2], [8, 2, 1]] define a function that returns a dictionary that associates for each row index (starting from 1) the corresponding list of values in the matrix. Considering the matrix in example, the result would be: {1: [5, 6, 7], 2: [8, 3, 2], 3: [8, 2, 1]}

```
def matrix_to_dictionary(matrix):
    result = "{"
    for i, row in enumerate(matrix):
        result += f"{i + 1}:{row}, "
    # Rimuovo l'ultima virgola e aggiungo la parentesi
    result = result[:-2] + "}"
    return result

test_list = [[5, 6, 7], [8, 3, 2], [8, 2, 1]]
res = matrix_to_dictionary(test_list)
print(res)
```

```
{1:[5, 6, 7], 2:[8, 3, 2], 3:[8, 2, 1]}
```

3. Given a list of integer values, such as: [50, 100, 150, 200, 250, 300] create a dictionary into which, for each element of the list, it's associated a list of 10 random numbers between 0 and the element. For generating the random number, use the randrange from the random package passing the element as parameter.

```
import random

lst = [50, 100, 150, 200, 250, 300]
random_dict = {}

for num in lst:
    random_list = []
```

```
    for _ in range(10):
        random_list.append(random.randrange(num))
    random_dict[num] = random_list

print('{')
for key, value in random_dict.items():
    print(key, ":", value)
print('}')
```

```
{
50 : [47, 11, 10, 38, 45, 42, 44, 34, 1, 12]
100 : [21, 53, 24, 85, 73, 64, 25, 7, 20, 24]
150 : [17, 68, 59, 47, 138, 136, 37, 60, 32, 7]
200 : [73, 46, 44, 115, 39, 33, 198, 187, 60, 148]
250 : [214, 188, 3, 73, 112, 27, 168, 217, 101, 46]
300 : [54, 17, 208, 251, 285, 241, 161, 284, 53, 239]
}
```

4. Create a function that takes in input a dictionary like the one generated in the Exercise 3. This function should calculate the minimum and the maximum for each list in the dictionary. The return value should be another dictionary into which, for each key of the input dictionay is associated a tuple formed by the minimum and the maximum of the list. For calculating the minimum and maximum use the min(list) and max(list) functions.

```
def transform_dictionary(d):
    new_dict = {}
    for k, l in d.items():
        minimum = min(l)
        maximum = max(l)
        new_dict[k] = (minimum, maximum)
    return new_dict

print(transform_dictionary(random_dict))
```

```
{50: (1, 47), 100: (7, 85), 150: (7, 138), 200: (33, 198), 250: (3, 217), 300: (17, 285)}
```

5. Create the same function using the list comprehension. Also, in this version the lists which contains at least an element lower than 10 should not be included.

```
def transform_dictionary(d):
    new_dict = {}
    for k, lst in d.items():
        if all(x >= 10 for x in lst):
            new_dict[k] = (min(lst), max(lst))
    return new_dict

print(transform_dictionary(random_dict))
```

```
{200: (33, 198), 300: (17, 285)}
```