In [1]:
```
import numpy as np
import tqdm
import math
```

In [2]:
```
def mark_pores_slow(x, pore_centers, pore_radii):
    I, J, K = x.shape
    for i in tqdm.trange(I):
        for j in range(J):
            for k in range(K):
                position = np.array([i, j, k])
                for pore_center, pore_radius in zip(pore_centers, pore_radii):
                    delta = pore_center - position
                    distance = np.sqrt(np.dot(delta, delta))
                    if distance <= pore_radius:
                        x[i, j, k] = 1
    return x
```

In [3]:
```
def mark_pores_fast(x, pore_centers, pore_radii):
    """
    This is for you to implement. It should yield the same result as the above function
    mark_pores_slow, but it should run faster.
    Hint: If you do it correctly, you should be able to change the below variable 'res'
    from 100 to 1000 without a significant slow-down of mark_pores_fast.

    Args:
        x: Numpy array of spatial domain in 3D. Initialized with 0.
        pore_centers: List of numpy arrays representing the centers of spherical pores
        pore_radii: List of pore radii

    Returns:
        x: Each field in x is 1 if in pore, 0 else.
    """
    # run for each pores
    for r in tqdm.trange(len(pore_centers)):

        # Create bounding box around each pores in order to reduce complexity
        pore_center, pore_radius = pore_centers[r], pore_radii[r]

        # Calculate bounding box minimum value for each pores
        lower_limit = [math.floor(i) for i in pore_center-pore_radius]
        lower_limit = [0 if i<0 else i for i in lower_limit]

        # Calculate bounding box maximum value for each pores
        upper_limit = [math.ceil(i) for i in pore_center+pore_radius+1]
        upper_limit =  [dim_size if i>dim_size else i for i, dim_size in zip(upper_limi

        # run for each points withing the bounding box
        for i in range(lower_limit[0], upper_limit[0]):
            for j in range(lower_limit[1], upper_limit[1]):
                for k in range(lower_limit[2], upper_limit[2]):
                    position = np.array([i, j, k])
                    delta = pore_center - position
                    distance = np.sqrt(np.dot(delta, delta))
                    if distance <= pore_radius:
                        x[i, j, k] = 1
    return x
```

In [4]:

```
if __name__ == '__main__':

    # initialize domain
    res = 100
    x = np.zeros((res, res, res))

    # sample some random pores
    n_pores = 10
    pore_radii = [np.random.uniform(3, 10) for n_pore in range(n_pores)]
    pore_centers = [np.random.rand(3) * x.shape for n_pore in range(n_pores)]

    # mark pores in domain
    pore_marker_slow = mark_pores_slow(x, pore_centers, pore_radii)
    pore_slow_2 = np.copy(pore_marker_slow)
    pore_marker_fast = mark_pores_fast(x, pore_centers, pore_radii)

    print("Same result from both method:" , (pore_marker_fast==pore_slow_2).all())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 1
00/100 [01:44<00:00,  1.05s/it]
100%|████████████████████████████████████████████████████████████████████████████|
10/10 [00:00<00:00, 19.71it/s]
Same result from both method: True
```