Example 1:

rights = [1,2,4]

valuations = [[11, 11, 22, 33, 44], [11, 22, 44, 55, 66], [11, 33, 22, 11, 66]]

y = 0.5

Iteration 1:

1. Initialization:  
   players\_chosen\_objects = [0,0,0]   
   remaining\_objects = [0,1,2,3,4]
2. Computing: player’s right / number of objects he chosen + y   
   - player 0: 1 / (0 + 0.5) = 2  
   - player 1: 2 / (0 + 0.5) = 4  
   - player 2: 4 / (0 + 0.5) = 8
3. The player with highest portion is the one that will choose now:  
   His valuations: [11, 33, 22, 11, 66] so object 4 is the most object he wants.  
   Now we will remove object 4: remaining\_objects = [0,1,2,3]  
   Update the valuations: [[11, 11, 22, 33, 0], [11, 22, 44, 55, 0], [11, 33, 22, 11, 0]]  
   Also update the number of objects player 2 took: players\_chosen\_objects = [0,0,1]

Iteration 2:

1. Initialization:  
   remaining\_objects = [0,1,2,3]  
   valuations = [[11, 11, 22, 33, 0], [11, 22, 44, 55, 0], [11, 33, 22, 11, 0]]  
   players\_chosen\_objects = [0,0,1]  
   rights = [1,2,4]
2. Computing: player’s right / number of objects he chosen + y   
    - player 0: 1 / (0 + 0.5) = 2  
    - player 1: 2 / (0 + 0.5) = 4   
    - player 2: 4 / (1 + 0.5) = 2.667
3. The player with highest portion is the one that will choose now:  
    His valuations: [11,22,44,55,0] so object 3 is the most object he wants.  
    Now we will remove object 3: remaining\_objects = [0,1,2]  
   Update the valuations: [[11, 11, 22, 0, 0], [11, 22, 44, 0, 0], [11, 33, 22, 0, 0]]  
   Also update the number of objects player 1 took: players\_chosen\_objects = [0,1,1]

Iteration 3:

1. Initialization:  
   remaining\_objects = [0,1,2]  
   valuations = [[11, 11, 22, 0, 0], [11, 22, 44, 0, 0], [11, 33, 22, 0, 0]]  
   players\_chosen\_objects = [0,1,1]  
   rights = [1,2,4]
2. Computing: player’s right / number of objects he chosen + y   
    - player 0: 1 / (0 + 0.5) = 2  
    - player 1: 2 / (1 + 0.5) = 1.333   
    - player 2: 4 / (1 + 0.5) = 2.667
3. The player with highest portion is the one that will choose now:  
   His valuations: [11, 33, 22, 0, 0] so object 1 is the most object he wants.  
   Now we will remove object 1: remaining\_objects = [0,2]  
   Update the valuations: [[11, 0, 22, 0, 0], [11, 0, 44, 0, 0], [11, 0, 22, 0, 0]]  
   Also update the number of objects player 2 took: players\_chosen\_objects = [0,1,2]

Iteration 4:

1. Initialization:  
   remaining\_objects = [0,2]  
   valuations = [[11, 0, 22, 0, 0], [11, 0, 44, 0, 0], [11, 0, 22, 0, 0]]  
   players\_chosen\_objects = [0,1,2]  
   rights = [1,2,4]
2. Computing: player’s right / number of objects he chosen + y   
   - player 0: 1 / (0 + 0.5) = 2  
   - player 1: 2 / (1 + 0.5) = 1.333   
   - player 2: 4 / (2 + 0.5) = 1.6
3. The player with highest portion is the one that will choose now:  
   His valuations: [11, 0, 22, 0, 0] so object 2 is the most object he wants.  
   Now we will remove object 2: remaining\_objects = [0]  
   Update the valuations: [[11, 0, 0, 0, 0], [11, 0, 0, 0, 0], [11, 0, 0, 0, 0]]  
   Also update the number of objects player 0 took: players\_chosen\_objects = [1,1,2]

Iteration 5:

1. Initialization:  
   remaining\_objects = [0]  
   valuations = [[11, 0, 0, 0, 0], [11, 0, 0, 0, 0], [11, 0, 0, 0, 0]]  
   players\_chosen\_objects = [1,1,2]  
   rights = [1,2,4]
2. Computing: player’s right / number of objects he chosen + y   
   - player 0: 1 / (1 + 0.5) = 0.666  
   - player 1: 2 / (1 + 0.5) = 1.333   
   - player 2: 4 / (2 + 0.5) = 1.6
3. The player with highest portion is the one that will choose now:  
   His valuations: [11, 0, 0, 0, 0] so object 0 is the most object he wants.  
   Now we will remove object 0: remaining\_objects = []  
   Update the valuations: [[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]  
   Also update the number of objects player 2 took: players\_chosen\_objects = [1,1,3]

Once there are no remaining objects the algorithm stops.   
Results:   
Player 0 takes:

* item 2 with value 22

Player 1 takes:

* item 3 with value 55

Player 2 takes:

* item 4 with value 66
* item 1 with value 33
* item 0 with value 11

Example 2:

rights = [1,1,1]

valuations = [[10, 10, 10], [10, 10, 10], [10, 10, 10]]

y = 1

Iteration 1:

1. Initialization:  
   players\_chosen\_objects = [0,0,0]   
   remaining\_objects = [0,1,2]
2. Computing: player’s right / number of objects he chosen + y   
   - player 0: 1 / (0 + 1) = 1   
   - player 1: 1 / (0 + 1) = 1  
   - player 2: 1 / (0 + 1) = 1
3. The player with highest portion is the one that will choose now:  
   His valuations: [10, 10, 10] so object 0 is the most object he wants.  
   Now we will remove object 0: remaining\_objects = [1,2]  
   Update the valuations: [[0, 10, 10], [0, 10, 10], [0, 10, 10]]  
   Also update the number of objects player 0 took: players\_chosen\_objects = [1,0,0]

Iteration 2:

1. Initialization:  
   remaining\_objects = [1,2]  
   valuations = [[0, 10, 10], [0, 10, 10], [0, 10, 10]]  
   players\_chosen\_objects = [1,0,0]  
   rights = [1,1,1]
2. Computing: player’s right / number of objects he chosen + y:  
   - player 0: 1 / (1+1) = 0.5  
   - player 1: 1 / (0+1) = 1   
   - player 2: 1 / (0+1) = 1
3. The player with highest portion is the one that will choose now:  
   His valuations: [0, 10, 10] so object 1 is the most object he wants.  
   Now we will remove object 1: remaining\_objects = [2]  
   Update the valuations: [[0, 0, 10], [0, 0, 10], [0, 0, 10]]  
   Also update the number of objects player 1 took: players\_chosen\_objects = [1,1,0]

Iteration 3:

1. Initialization:  
   remaining\_objects = [2]  
   valuations = [[0, 0, 10], [0, 0, 10], [0, 0, 10]]  
   players\_chosen\_objects = [1,1,0]  
   rights = [1,1,1]
2. Computing: player’s right / number of objects he chosen + y:  
   - player 0: 1 / (1+1) = 0.5  
   - player 1: 1 / (1+1) = 0.5   
   - player 2: 1 / (0+1) = 1
3. The player with highest portion is the one that will choose now:  
   His valuations: [0, 0, 10] so object 0 is the most object he wants.  
   Now we will remove object 0: remaining\_objects = []  
   Update the valuations: [[0, 0, 0], [0, 0, 0], [0, 0, 0]]  
   Also update the number of objects player 2 took: players\_chosen\_objects = [1,1,1]

Once there are no remaining objects the algorithm stops.   
Results:   
Player 0 takes:

* item 0 with value 10

Player 1 takes:

* item 1 with value 10

Player 2 takes:

* item 2 with value 10

Example 3:

rights = [1,2,3]

valuations = [[10, 10, 10, 10, 10], [10, 10, 10, 10, 10], [10, 10, 10, 10, 10]]

y = 0.5

Iteration 1:

1. Initialization:  
   players\_chosen\_objects = [0,0,0]   
   remaining\_objects = [0,1,2,3,4]
2. Computing: player’s right / number of objects he chosen + y:  
   - player 0: 1 / (0 + 0.5) = 2  
   - player 1: 2 / (0 + 0.5) = 4  
   - player 2: 3 / (0 + 0.5) = 6
3. The player with highest portion is the one that will choose now:  
   His valuations: [10, 10, 10, 10, 10] so object 0 is the most object he wants.  
   Now we will remove object 0: remaining\_objects = [1,2,3,4]  
   Update the valuations: [[0, 10, 10, 10, 10], [0, 10, 10, 10, 10]], [0, 10, 10, 10, 10]]]  
   Also update the number of objects player 2 took: players\_chosen\_objects = [0,0,1]

Iteration 2:

1. Initialization:  
   remaining\_objects = [1,2,3,4]  
   valuations = [[0, 10, 10, 10, 10], [0, 10, 10, 10, 10]], [0, 10, 10, 10, 10]]]  
   players\_chosen\_objects = [0,0,1]  
   rights = [1,2,3]
2. Computing: player’s right / number of objects he chosen + y:  
   - player 0: 1 / (0 + 0.5) = 2  
   - player 1: 2 / (0 + 0.5) = 4   
   - player 2: 3 / (1 + 0.5) = 2
3. The player with highest portion is the one that will choose now:  
   His valuations: [0, 10, 10, 10, 10] so object 1 is the most object he wants.  
   Now we will remove object 0: remaining\_objects = [2,3,4]  
   Update the valuations: [[0, 0, 10, 10, 10], [0, 0, 10, 10, 10]], [0, 0, 10, 10, 10]]]  
   Also update the number of objects player 1 took: players\_chosen\_objects = [0,1,1]

Iteration 3:

1. Initialization:  
   remaining\_objects = [2,3,4]  
   valuations = [[0, 0, 10, 10, 10], [0, 0, 10, 10, 10]], [0, 0, 10, 10, 10]]]  
   players\_chosen\_objects = [0,1,1]  
   rights = [1,2,3]
2. Computing: player’s right / number of objects he chosen + y:  
   - player 0: 1 / (0 + 0.5) = 2   
   - player 1: 2 / (1 + 0.5) = 1.333   
   - player 2: 3 / (1 + 0.5) = 2
3. The player with highest portion is the one that will choose now:  
   His valuations: [0, 0, 10, 10, 10] so object 2 is the most object he wants.  
   Now we will remove object 2: remaining\_objects = [3,4]  
   Update the valuations: [[0, 0, 0, 10, 10], [0, 0, 0, 10, 10]], [0, 0, 0, 10, 10]]]  
   Also update the number of objects player 0 took: players\_chosen\_objects = [1,1,1]

Iteration 4:

1. Initialization:  
   remaining\_objects = [3,4]  
   valuations = [[0, 0, 0, 10, 10], [0, 0, 0, 10, 10]], [0, 0, 0, 10, 10]]]  
   players\_chosen\_objects = [1,1,1]  
   rights = [1,2,3]
2. Computing: player’s right / number of objects he chosen + y:  
   - player 0: 1 / (1 + 0.5) = 0.666   
   - player 1: 2 / (1 + 0.5) = 1.333   
   - player 2: 3 / (1 + 0.5) = 2
3. The player with highest portion is the one that will choose now:  
   His valuations: [0, 0, 0, 10, 10] so object 3 is the most object he wants.  
   Now we will remove object 3: remaining\_objects = [4]  
   Update the valuations: [[0, 0, 0, 0, 10], [0, 0, 0, 0, 10]], [0, 0, 0, 0, 10]]]  
   Also update the number of objects player 2 took: players\_chosen\_objects = [1,1,2 ]

Iteration 5:

1. Initialization:  
   remaining\_objects = [4]  
   valuations = [[0, 0, 0, 0, 10], [0, 0, 0, 0, 10]], [0, 0, 0, 0, 10]]]  
   players\_chosen\_objects = [1,1,2]  
   rights = [1,2,3]
2. Computing: player’s right / number of objects he chosen + y:  
   - player 0: 1 / (1 + 0.5) = 0.666   
   - player 1: 2 / (1 + 0.5) = 1.333   
   - player 2: 3 / (2 + 0.5) = 1.2
3. The player with highest portion is the one that will choose now:  
   His valuations: [0, 0, 0, 0, 10] so object 4 is the most object he wants.  
   Now we will remove object 4: remaining\_objects = []  
   Update the valuations: [[0, 0, 0, 0, 0], [0, 0, 0, 0, 0]], [0, 0, 0, 0, 0]]]  
   Also update the number of objects player 1 took: players\_chosen\_objects = [1,2,2 ]

Once there are no remaining objects the algorithm stops.   
Results:   
Player 0 takes:

* item 2 with value 10

Player 1 takes:

* item 1 with value 10
* item 4 with value 10

Player 2 takes:

* item 0 with value 10
* item 3 with value 10

Iteration 3:  
 1. Initialization:  
 remaining\_objects = [0,1,2]  
 valuations = [[11, 11, 22, 0, 0], [11, 22, 44, 0, 0], [11, 33, 22, 0, 0]]  
 players\_chosen\_objects = [0,1,1]  
 rights = [1,2,4]

2. Computing: player’s right / number of objects he chosen + y   
 - player 0: 1 / (0 + 0.5) = 2  
 - player 1: 2 / (1 + 0.5) = 1.333   
 - player 2: 4 / (1 + 0.5) = 2.667   
  
 3. The player with highest portion is the one that will choose now:  
 His valuations: [11, 33, 22, 0, 0] so object 1 is the most object he wants.  
 Now we will remove object 1: remaining\_objects = [0,2]  
 Update the valuations: [[11, 0, 22, 0, 0], [11, 0, 44, 0, 0], [11, 0, 22, 0, 0]]  
 Also update the number of objects player 2 took: players\_chosen\_objects = [0,1,2]

Iteration 4:  
 1. Initialization:  
 remaining\_objects = [0,2]  
 valuations = [[11, 0, 22, 0, 0], [11, 0, 44, 0, 0], [11, 0, 22, 0, 0]]  
 players\_chosen\_objects = [0,1,2]  
 rights = [1,2,4]

2. Computing: player’s right / number of objects he chosen + y   
 - player 0: 1 / (0 + 0.5) = 2  
 - player 1: 2 / (1 + 0.5) = 1.333   
 - player 2: 4 / (2 + 0.5) = 1.6   
  
 3. The player with highest portion is the one that will choose now:  
 His valuations: [11, 0, 22, 0, 0] so object 2 is the most object he wants.  
 Now we will remove object 2: remaining\_objects = [0]  
 Update the valuations: [[11, 0, 0, 0, 0], [11, 0, 0, 0, 0], [11, 0, 0, 0, 0]]  
 Also update the number of objects player 2 took: players\_chosen\_objects = [1,1,2]

Iteration 5:  
 1. Initialization:  
 remaining\_objects = [0]  
 valuations = [[11, 0, 0, 0, 0], [11, 0, 0, 0, 0], [11, 0, 0, 0, 0]]  
 players\_chosen\_objects = [1,1,2]  
 rights = [1,2,4]

2. Computing: player’s right / number of objects he chosen + y   
 - player 0: 1 / (1 + 0.5) = 0.666  
 - player 1: 2 / (1 + 0.5) = 1.333   
 - player 2: 4 / (2 + 0.5) = 1.6   
  
 3. The player with highest portion is the one that will choose now:  
 His valuations: [11, 0, 0, 0, 0] so object 0 is the most object he wants.  
 Now we will remove object 0: remaining\_objects = []  
 Update the valuations: [[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]  
 Also update the number of objects player 2 took: players\_chosen\_objects = [1,1,3]

Once there are no remaining objects the algorithm stops.   
Results:

Player 0 takes:

* item 2 with value 22

Player 1 takes:

* item 3 with value 55

Player 2 takes:

* item 4 with value 66
* item 1 with value 33
* item 0 with value 11