In [1]:

```python
import random
```

In [2]:

```python
def majority_vote(votes):
    """This function creates a dictionary of counts and returns the key that has the highest counts
    If more than one key has the hightest counts, it picks one at random"""

    vote_counts = {}
    for vote in votes:
        if vote in vote_counts:
            vote_counts[vote] += 1
        else:
            vote_counts[vote] = 1

    winners = []
    max_count = max(vote_counts.values())

    for vote, count in vote_counts.items():
        if count == max_count:
            winners.append(vote)

    return random.choice(winners)
```

In [3]:

```python
votes = [1,1,1,2,2,3,3]
majority_vote(votes)
```

Out[3]:

```
1
```

In [4]:

```python
votes2 = ["M", "H", "H", "H", "M", "A"]
```

In [5]:

```python
majority_vote(votes2)
```

Out[5]:

```
'H'
```

A faster way to determine key associated with the majority of votes is to use mode from scipy module. However, it only works for list of numbers. It doens't work for list of strings.

In [6]:

```python
import scipy.stats as ss
```

In [7]:

```python
def majority_vote_short(votes):
    """ Quickly finds the majority of a list of numbers. It doesn't work for list of strings"""

    mode, count = ss.mstats.mode(votes)
    return mode
```

In [8]:

```python
majority_vote_short(votes)
```

Out[8]:

```
array([1.])
```

```
In [9]:
```

```
majority_vote_short(votes2)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-9-73b9d44fc226> in <module>
----> 1 majority_vote_short(votes2)

<ipython-input-7-274e6bb8a797> in majority_vote_short(votes)
      2     """ Quickly finds the majority of a list of numbers. It doesn't work for list of strings
"""
      3
----> 4     mode, count = ss.mstats.mode(votes)
      5     return mode

C:\Users\maria\Anaconda3\lib\site-packages\scipy\stats\mstats_basic.py in mode(a, axis)
    312             output = (ma.array(output[0]), ma.array(output[1]))
    313         else:
--> 314             output = ma.apply_along_axis(_mode1D, axis, a)
    315             newshape = list(a.shape)
    316             newshape[axis] = 1

C:\Users\maria\Anaconda3\lib\site-packages\numpy\ma\extras.py in apply_along_axis(func1d, axis, arr,
*args, **kwargs)
    393         i.put(indlist, ind)
    394         j = i.copy()
--> 395         res = func1d(arr[tuple(i.tolist())], *args, **kwargs)
    396         #  if res is a number, then we have a smaller output array
    397         asscalar = np.isscalar(res)

C:\Users\maria\Anaconda3\lib\site-packages\scipy\stats\mstats_basic.py in _mode1D(a)
    300
    301     def _mode1D(a):
--> 302         (rep,cnt) = find_repeats(a)
    303         if not cnt.ndim:
    304             return (0, 0)

C:\Users\maria\Anaconda3\lib\site-packages\scipy\stats\mstats_basic.py in find_repeats(arr)
    149     # Make sure we get a copy. ma.compressed promises a "new array", but can
    150     # actually return a reference.
--> 151     compr = np.asarray(ma.compressed(arr), dtype=np.float64)
    152     try:
    153         need_copy = np.may_share_memory(compr, arr)

C:\Users\maria\Anaconda3\lib\site-packages\numpy\core\numeric.py in asarray(a, dtype, order)
    536
    537     """
--> 538     return array(a, dtype, copy=False, order=order)
    539
    540

ValueError: could not convert string to float: 'M'
```

```
In [ ]:
```