Pete the Baker

```
def cakes(recipe, available):
    lowest = float("inf")
    common_ingredients = set(recipe.keys()) & set(available.keys
    for k in recipe.keys():
        if k not in available.keys():
            return 0

    for key in common_ingredients:
        avail = available[key]
        reci = recipe[key]
        cake_made = avail // reci
        if cake_made < lowest:
            lowest = cake_made
    return lowest
```

My initial code, it is very slow as it uses two nested for statements, one to check for mis matching keys and one to find common ingredients.

This is probably an O(n) algorithm

```
def cakes(recipe, available):
    list = []
    for ingredient in recipe:
        if ingredient in available:
            list.append(available[ingredient]/recipe[ingredient]
        else:
            return 0
    return min(list)
```

More neater, this creates a list, and checks if those ingredients exist in the available, if so then it peforms a division between available and recipe

```
def cakes(recipe, available):
    return min(available.get(k, 0)/recipe[k] for k in recipe)
```

ideally this is what I should use, the use of List comprehension, by forming a list, getting keys and if its not there, then returning a 0