In [13]:
```python
# 4. (Call center) Complete the code in the iPython notebook:
def empirical_pmf(months, max_count, time_ini, time_end, verbose):
    # initializing count result
    result = np.zeros(max_count)

    # loop months and count for each months
    for month in months:
        # counts value for certain month
        counts = compute_counts(time_ini, time_end, month, verbose)

        # Feb, Apr, June, Sep, Nov has days less than 31.
        # slice counts to exact days
        if month_number[month] == 2:
            counts = counts[:28]
        elif month_number[month] == 4 or 6 or 9 or 11:
            counts = counts[:30]

        # for every day, if given day is not weekend, then add to count
        # add counts to certain result index
        for i in range(len(counts)):
            if is_weekend(month_number[month], i+1) == False:
                result[int(counts[i])] += 1

    # get final pmf
    pmf = result / sum(result)

    # return its value
    return pmf
```

```python
# import math to use factorial function
import math

def poisson_model(months, max_count,time_ini,time_end,verbose):

    # initialize two ndarrays with max_count, also initialize total calls and number of days
    pmf = np.zeros(max_count)
    result = np.zeros(max_count)
    totalCalls = 0
    numDays = 0

    # loop months and count for each months
    for month in months:
        # counts value for certain month
        counts = compute_counts(time_ini, time_end, month, verbose)

        # Feb, Apr, June, Sep, Nov has days less than 31.
        # slice counts to exact days
        if month_number[month] == 2:
            counts = counts[:28]
        elif month_number[month] == 4 or 6 or 9 or 11:
            counts = counts[:30]

        # for every day, if given day is not weekend, then count
        # add total counts to totalcalls and add 1 for each success if statement
        for i in range(len(counts)):
            if is_weekend(month_number[month], i+1) == False:
                totalCalls += counts[i]
                numDays += 1

    # lambda should be mean value, so divide total calls by number of days
    param = totalCalls / numDays

    # loop x from 0 to length of pmf, and append possion values to pmf[x]
    for x in range(len(pmf)):
        pmf[x] = (np.exp(-param) * (param ** x)) / math.factorial(x)

    # return pmf
    return pmf
```