Final Project: FIRE INCIDENT REPORTING

Group Member:

- Jinhui Hong
- Zongru Wang

According to the government website, a fire department in the United States responds to a fire somewhere in the nation every 24 seconds. Since fire department gets limited resources and it is important for them to handle every case. Our question would be what are the most frequent locations, reasons for fire reports for the entire 2018 years in the Boston area? How can we help the fire department to plan the deployment of limited resources?

We will work on the `fire incident report` within Boston areas in 2018 that to be shared with state and federal governments for reporting.

The reason why we care about this data is that those data allow us to identify trends, quantify activities, determine causes, plan the deployment of limited resources, and help in the reduction of loss of life and property caused by fires. What we want to figure out is finding the trends of the most frequent fire report's location, season, and time so that we can help the fire department plan the deployment of limited resources. In addition, because the data that we want to analyze includes the reason/description of the fire incident, we can figure out what happened most in different area, months, and time so that we can plan the deployment of resources for some specific incident. We will also analyze what is the emergency and nonemergency activities of fire departments while allowing the fire service to tell its story in an objective manner through its data.

We will use their data and `matplotlib` to reproduce key figures and results from their analysis on the `jupyter notebook` .

Their dataset is available from the [Analyze Boston (https://data.boston.gov/dataset/fire-incident-reporting)](https://data.boston.gov/dataset/fire-incident-reporting).

Setting up a Jupyter Notebook

In [270]:
```
# import the thing we needs to use for analizing the data
from statistics import mean, stdev
from math import sqrt
import matplotlib.pyplot as plt
import csv
%matplotlib inline
```

In [271]:
```
# our functions - we implemented those on the lab
# the data is not clean, so when we load the data, we need to remove the st
def load_data(filename):
    with open(filename, "r") as fin:
        dr = csv.DictReader(fin, quotechar='"', fieldnames=None, delimiter=
        l = [row for row in dr]
        for row in l:
            for key in row:
                row[key] = row[key].strip()
    return l

def sum_incident_as_season(season):
    return dict(Counter(season[0])+Counter(season[1])+Counter(season[2]))
```

(1.1) Begin by loading the data

In [272]:
```
# load the data
# dataset of incident-type-code, and property-use-code
incident_type_code = load_data("data/incident-type-code-list.csv")
property_use_code = load_data("data/property-use-code-list.csv")
# 12 months(one year) fire incident data for 2018
january = load_data("data/january.2018-bostonfireincidentopendata.csv")
february = load_data("data/february.2018-bostonfireincidentopendata.csv")
march = load_data("data/march.2018-bostonfireincidentopendata.csv")
april = load_data("data/april.2018-bostonfireincidentopendata.csv")
may = load_data("data/may.2018-bostonfireincidentopendata.csv")
june = load_data("data/june.2018-bostonfireincidentopendata.csv")
july = load_data("data/july.2018-bostonfireincidentopendata.csv")
august = load_data("data/august.2018-bostonfireincidentopendata.csv")
september = load_data("data/september.2018-bostonfireincidentopendata.csv")
october = load_data("data/october.2018-bostonfireincidentopendata.csv")
november = load_data("data/november.2018-bostonfireincidentopendata.csv")
december = load_data("data/december.2018-bostonfireincidentopendata.csv")
year = [january, february, march, april, may, june, july, august, september
```

(1.2) Clean the data

In [273]:
```
# we want a dictionary which every item's key is code, and value is descrip
incident_type_code = {line['code']:line['descript'] for line in incident_ty
# we also want to see what does the code represent, take in list of code, r
def transCode(code):
    return [incident_type_code[c] for c in code]

# transfer the specific incident code to general type of description
def getIncidentTypes(month_incident):
    group_code = {'Fire':0, 'Explosion':0, 'Medical/Emergency':0, 'Industri
                  'Public service':0, 'Mistake Information':0,'False alarm'
                  'Natural disaster':0, 'Special incident':0,'Undetermind i
    for code in month_incident:
        if code[0] == '1':
            group_code['Fire'] += month_incident[code]
        if code[0] == '2':
            group_code['Explosion'] += month_incident[code]
        if code[0] == '3':
            group_code['Medical/Emergency'] += month_incident[code]
        if code[0] == '4':
            group_code['Industrial accident'] += month_incident[code]
        if code[0] == '5':
            group_code['Public service'] += month_incident[code]
        if code[0] == '6':
            group_code['Mistake Information'] += month_incident[code]
        if code[0] == '7':
            group_code['False alarm'] += month_incident[code]
        if code[0] == '8':
            group_code['Natural disaster'] += month_incident[code]
        if code[0] == '9':
            group_code['Special incident'] += month_incident[code]
        if code == 'UUU':
            group_code['Undetermind incident'] += month_incident[code]
    return group_code
```

(1.3) get the number of incident happend for each month

In [274]:
```
def get_incident_num(month):
    incident_type_code_copy = {key:0 for key in incident_type_code}
    for line in month:
        # there are some data that is not in incident_type_code, we does nc
        if line['Incident Type'] in incident_type_code_copy.keys():
            incident_type_code_copy[line['Incident Type']]+=1
    return incident_type_code_copy

january_incident = getIncidentTypes(get_incident_num(january))
february_incident = getIncidentTypes(get_incident_num(february))
march_incident = getIncidentTypes(get_incident_num(march))
april_incident = getIncidentTypes(get_incident_num(april))
may_incident = getIncidentTypes(get_incident_num(may))
june_incident = getIncidentTypes(get_incident_num(june))
july_incident = getIncidentTypes(get_incident_num(july))
august_incident = getIncidentTypes(get_incident_num(august))
september_incident = getIncidentTypes(get_incident_num(september))
october_incident = getIncidentTypes(get_incident_num(october))
november_incident = getIncidentTypes(get_incident_num(november))
december_incident = getIncidentTypes(get_incident_num(december))
total_year_incident = [january_incident,february_incident,
                       march_incident,april_incident,
                       may_incident,june_incident,
                       july_incident,august_incident,september_incident,
                       october_incident, november_incident, december_incide
```

We also want to group month by different seasons

- Spring runs from March 1 to May 31;
- Summer runs from June 1 to August 31;
- Fall (autumn) runs from September 1 to November 30;
- Winter runs from December 1 to February 28 (February 29 in a leap year).

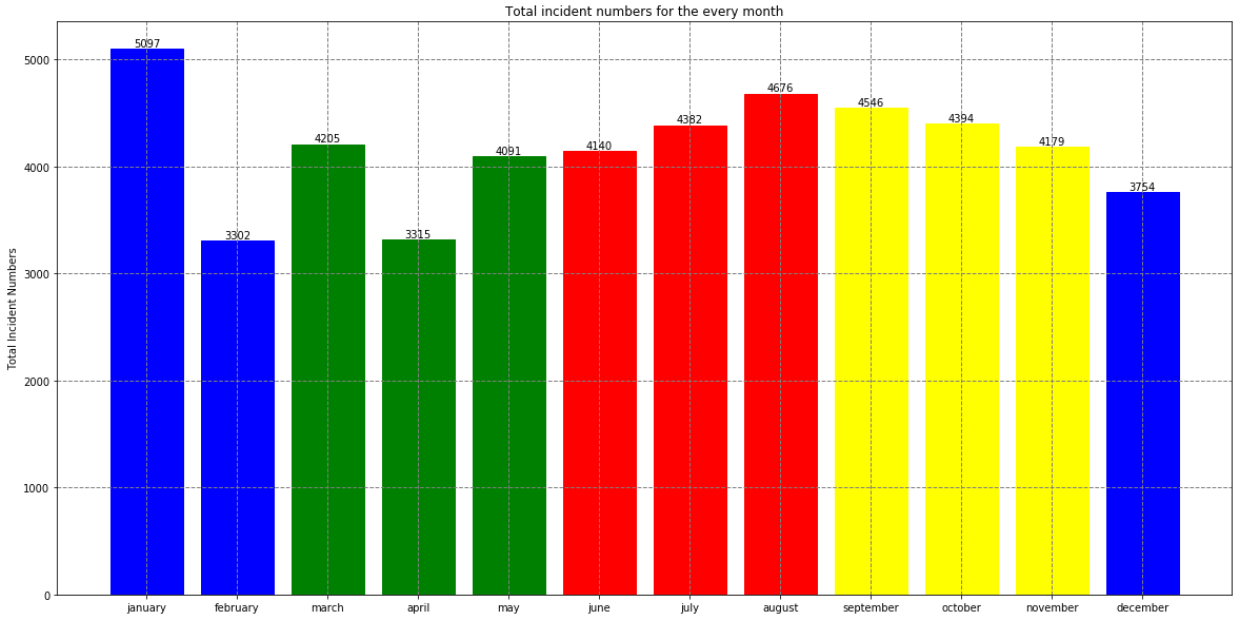
We will use different color to represent different seasons:

Season	Color
Spring	green
Summer	red
Fall	yellow
WInter	blue

In [275]:
```
spring = [february, march, april]
summer = [may, june, july]
fall = [august, september, october]
winter = [november, december, january]

spring_incident = [february_incident, march_incident, april_incident]
summer_incident = [may_incident, june_incident, july_incident]
fall_incident = [august_incident, september_incident, october_incident]
winter_incident = [november_incident, december_incident, january_incident]

spring_incident_dict = sum_incident_as_season(spring_incident)
summer_incident_dict = sum_incident_as_season(summer_incident)
fall_incident_dict = sum_incident_as_season(fall_incident)
winter_incident_dict = sum_incident_as_season(winter_incident)

# sum the incident for the given list seaon
def incident_sum(season):
    result = 0
    for i in season:
        result += len(i)
    return result


# we want to analyze which are top n most frequent incidents in the given m
def getMostFre(month_incident, n):
    month_incident_copy = month_incident.copy()
    top_n_incident_code = []
    while n>0:
        max_value = max(month_incident_copy.values())
        max_key = -1
        for k, v in month_incident_copy.items():
            if v == max_value:
                max_key = k
                top_n_incident_code.append(max_key)
        month_incident_copy.pop(max_key)
        n-=1
    return top_n_incident_code


spring_top_5_name = getMostFre(spring_incident_dict, 5)
summer_top_5_name = getMostFre(summer_incident_dict, 5)
fall_top_5_name = getMostFre(fall_incident_dict, 5)
winter_top_5_name = getMostFre(winter_incident_dict, 5)

spring_top_5_value = [spring_incident_dict[name] for name in spring_top_5_n
summer_top_5_value = [summer_incident_dict[name] for name in summer_top_5_n
fall_top_5_value = [fall_incident_dict[name] for name in fall_top_5_name]
winter_top_5_value = [winter_incident_dict[name] for name in winter_top_5_n
```

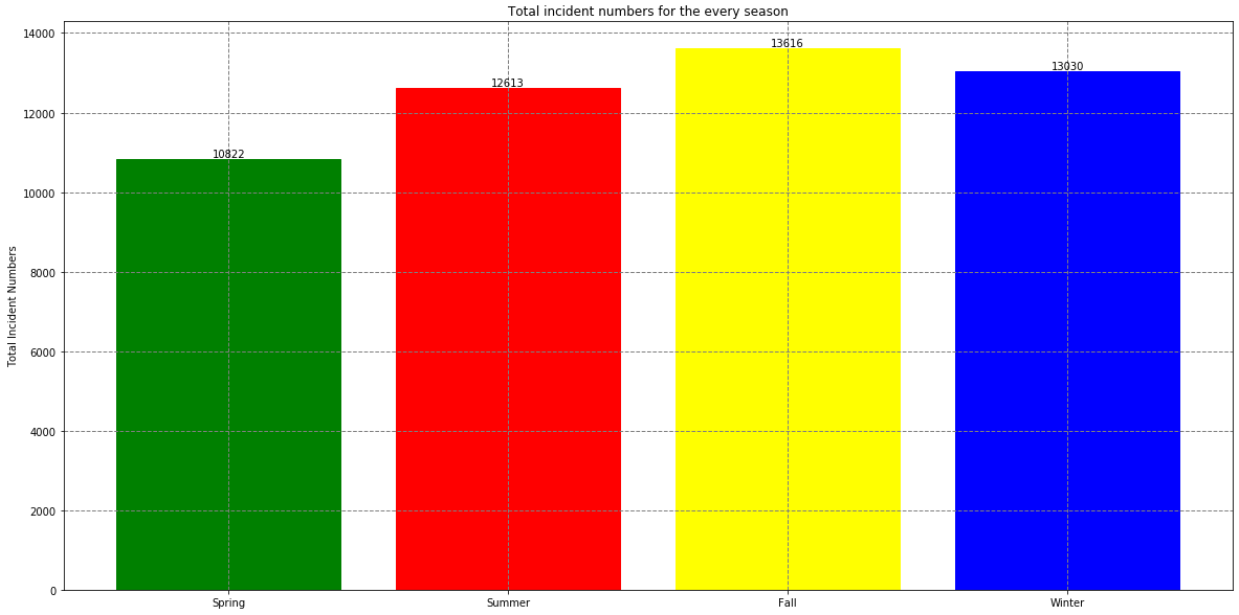
In [276]:
```
# funtion to display value on the top
def setTop(axs, y):
    rects = axs.patches
    for rect, label in zip(rects, y):
        height = rect.get_height()
        axs.text(rect.get_x() + rect.get_width() / 2, height + 5, label,
                ha='center', va='bottom')
```

In [277]:
```
# we will draw the number of incidents each month
# create the plot
X = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
# the most frequently incident for the every month(discription)
# the value corresponding to the number of incidents each month
Y = [len(i) for i in year]
fig, ax = plt.subplots(figsize=(20, 10))
bar = ax.bar(X, Y)
bar[0].set_color('blue')
bar[1].set_color('blue')
bar[2].set_color('green')
bar[3].set_color('green')
bar[4].set_color('green')
bar[5].set_color('red')
bar[6].set_color('red')
bar[7].set_color('red')
bar[8].set_color('yellow')
bar[9].set_color('yellow')
bar[10].set_color('yellow')
bar[11].set_color('blue')

ax.set_xticks(X)
ax.set_xticklabels(
    [f"january",
     f"february",
     f"march",
     f"april",
     f"may",
     f"june",
     f"july",
     f"august",
     f"september",
     f"october",
     f"november",
     f"december"])
ax.set_ylabel("Total Incident Numbers")
ax.set_title("Total incident numbers for the every month");
ax.grid(color='grey', linestyle='--', linewidth=1)

setTop(ax, Y)
```



We can see that the most frequent month is January and March is least frequent month Since most frequent month is January, we want to check if the reason to cuase it most frequent is weather, so we want to check the season and see the general data because single month cannot represent the season and season is more representative

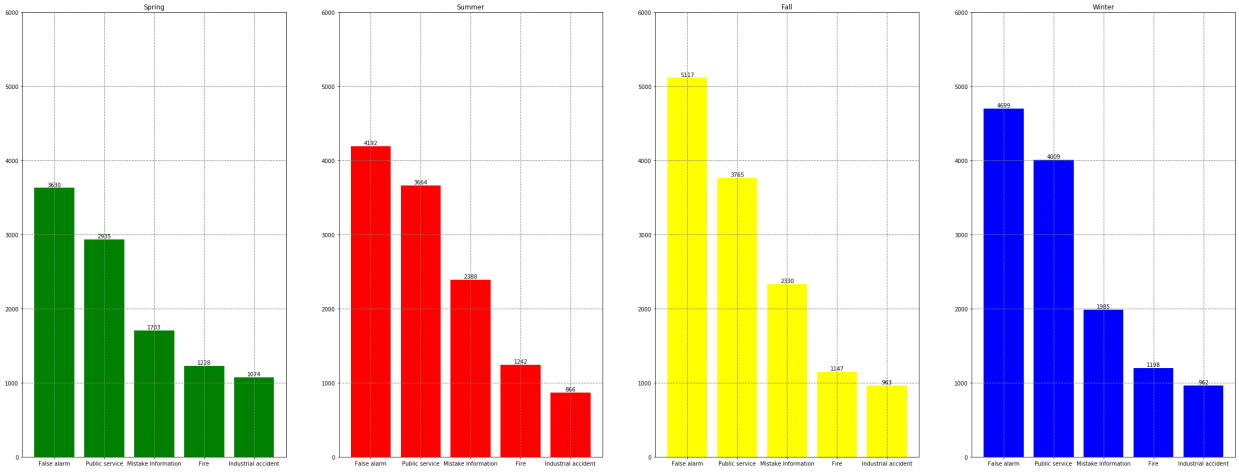
In [278]:
```
# create the plot
# we will draw the number of incidents each season
X = [1, 2, 3, 4]
Y = [incident_sum(spring), incident_sum(summer), incident_sum(fall), incide

fig, ax = plt.subplots(figsize=(20, 10))
bar = ax.bar(X, Y)
bar[0].set_color('green')
bar[1].set_color('red')
bar[2].set_color('yellow')
bar[3].set_color('blue')

ax.set_xticks(X)
ax.set_xticklabels(['Spring', 'Summer', 'Fall', 'Winter'])
ax.set_ylabel("Total Incident Numbers")
ax.set_title("Total incident numbers for the every season");
ax.grid(color='grey', linestyle='--', linewidth=1)
setTop(ax, Y)
```



- From the graph, we can see that Fall is the season which has most frequent incidents
- We want to know what exactly causes this, in another word, what are the top number incidents happend for fall and other season, so we can further analyze

In [279]:
```
# we will draw the number of incidents each season
X = [1, 2, 3, 4, 5]
spring_top_5_name = getMostFre(spring_incident_dict, 5)
summer_top_5_name = getMostFre(summer_incident_dict, 5)
fall_top_5_name = getMostFre(fall_incident_dict, 5)
winter_top_5_name = getMostFre(winter_incident_dict, 5)

spring_top_5_value = [spring_incident_dict[name] for name in spring_top_5_n
summer_top_5_value = [summer_incident_dict[name] for name in summer_top_5_n
fall_top_5_value = [fall_incident_dict[name] for name in fall_top_5_name]
winter_top_5_value = [winter_incident_dict[name] for name in winter_top_5_n

# create the plot
fig, (ax1, ax2, ax3, ax4) = plt.subplots(ncols=4, figsize=(40, 15))

# define the plots as before, for all four 'axes'
ax1.bar(X, spring_top_5_value, color='green')
ax2.bar(X, summer_top_5_value, color='red')
ax3.bar(X, fall_top_5_value, color='yellow')
ax4.bar(X, winter_top_5_value, color='blue')

# annotate the plots as before, for all three 'axes'
ax1.set_xticks(X)
ax1.set_xticklabels(spring_top_5_name)
ax1.set_title("Spring");
ax1.grid(color='grey', linestyle='--', linewidth=1)
ax1.set_ylim(0, 6000)

ax2.set_xticks(X)
ax2.set_xticklabels(summer_top_5_name)
ax2.set_title("Summer");
ax2.grid(color='grey', linestyle='--', linewidth=1)
ax2.set_ylim(0, 6000)

ax3.set_xticks(X)
ax3.set_xticklabels(fall_top_5_name)
ax3.set_title("Fall");
ax3.grid(color='grey', linestyle='--', linewidth=1)
ax3.set_ylim(0, 6000)

ax4.set_xticks(X)
ax4.set_xticklabels(winter_top_5_name)
ax4.set_title("Winter");
ax4.grid(color='grey', linestyle='--', linewidth=1)
ax4.set_ylim(0, 6000)

plt.suptitle("Top 5 incident numbers for the every season in 2018", fontsiz
setTop(ax1, spring_top_5_value)
setTop(ax2, summer_top_5_value)
setTop(ax3, fall_top_5_value)
setTop(ax4, winter_top_5_value)
```

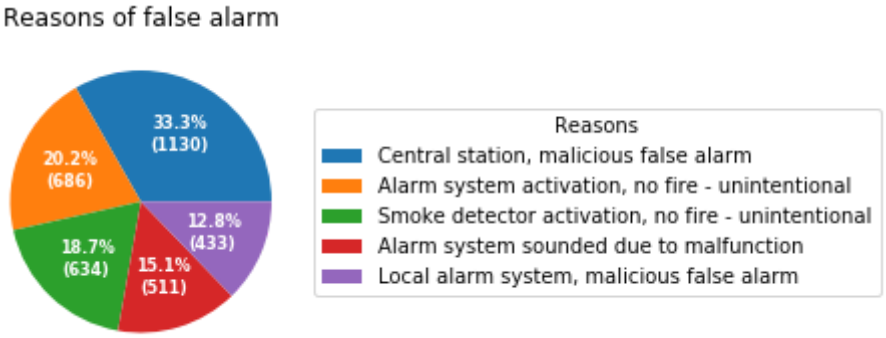
Top 5 incident numbers for the every season in 2018



We found an interesting result: we can see that top 5 incident types are exactly same for every season even the rank of those incidents are same for every season! We can see that the main incidents type that cause the Fall having most incidents is False alarm.

Then if we analysis the specifc type of incident of the False alarm, we can find the main reason causes the False alarm in fall. We decide to use pie graph to show the result.

In [288]:
```
# first, get the top 5 reason to cause false alarm in Fall
a8 = get_incident_num(august)
s9 = get_incident_num(september)
o10 = get_incident_num(october)
fall_code = [a8, s9, o10]
fall_total = sum_incident_as_season(fall_code)
false_alarm_reasons = {incident_type_code[k]:fall_total[k] for k in fall_tc
top5reason = getMostFre(false_alarm_reasons, 5)
top5reason_value = [false_alarm_reasons[i] for i in top5reason]


import numpy as np

fig, ax = plt.subplots(figsize=(6, 3), subplot_kw=dict(aspect="equal"))

data = [float(x) for x in top5reason_value]

def func(pct, allvals):
    absolute = int(pct/100.*np.sum(allvals))
    return "{:.1f}%\n({:d})".format(pct, absolute)


wedges, texts, autotexts = ax.pie(data, autopct=lambda pct: func(pct, data)

ax.legend(wedges, top5reason,
          title="Reasons",
          loc="center left",
          bbox_to_anchor=(1, 0, 0.5, 1))

plt.setp(autotexts, size=8, weight="bold")

ax.set_title("Reasons of false alarm")

plt.show()
```



Then we can get the top five types of false alarm that happens most in the Fall.

- The 33.3% of flase alarm comes from Central station, malicious false alarm, and 20.2% false clarm comes from unintenional Alarm system activation, and 18.7% comes from the unintentional smoke detector, both of them are unintentional.
- 15.1% of false alarms comes from alarms in malfuction, and 12.8% comes from the malicous false alarm in local alarm system.

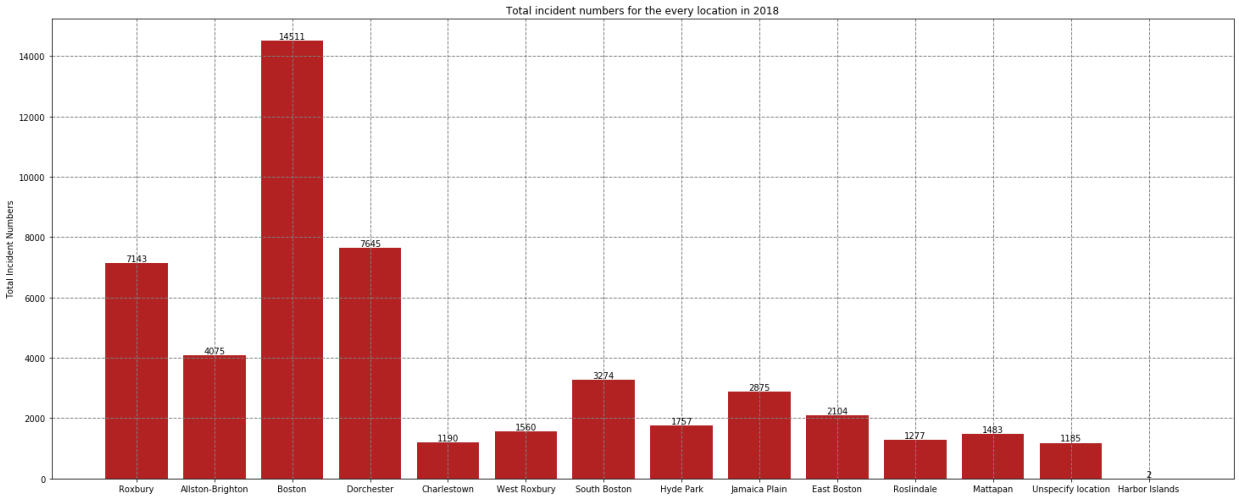
- The unintentional misalam takes 38.7% in the top 5 incident type in false alarm and malicious false alarm takes 46.1% of the top 5 incident type in false alarm.
- From that we can see the malicious triger the false alarm takes almost half of the top 5 incident type, which wastes lots of time and money from public resourcs.
- If we can get the most potential location of these malicious false alarm, and add some ways to monitor this location or lets local police force payt more attention on these area, the fire department can comfirm and cancel the false alarm faster or even don't have to go and check, then lots of public resource can be save and be used on more import part.

Now, we want to see where are the most frequent locations

(1.1) Get the number of incident happend for every neighbor

In [264]:
```
# get the number of incident for locations of given month
def get_neighborhood_num(month):
    # since a few number of the incident does not have location, so we repl
    neighborhoods = {line['Neighborhood']:0 for line in month}
    for line in month:
        neighborhoods[line['Neighborhood']]+=1
    neighborhoods['Unspecify location'] = neighborhoods['']
    neighborhoods.pop('')
    return neighborhoods

from collections import Counter
# sum 12 months incident
def sum_incident_location():
    # initilize the location incidents dict
    z = get_neighborhood_num(january)
    for month in year:
        z = dict(Counter(get_neighborhood_num(month))+Counter(z))
    # we want to minus get_neighborhood_num(january) since we added this tw
    z = dict(Counter(z)-Counter(get_neighborhood_num(january)))
    return z


dic = sum_incident_location()
# get list of locations and corresponding index value
dic_location = dic.keys()
dic_value = dic.values()
```

In [265]:
```
# we will draw the locations and their number of incidents each month
RED = "firebrick"

# locations
X = list(range(len(sum_incident_location())))
# the value corresponding to the number of incidents each month
Y = dic_value
# create the plot
fig, ax = plt.subplots(figsize=(25, 10))
ax.bar(X, Y, color=[RED])
ax.set_xticks(X)
ax.set_xticklabels(dic_location)
ax.set_ylabel("Total Incident Numbers")
ax.set_title("Total incident numbers for the every location in 2018");
ax.grid(color='grey', linestyle='--', linewidth=1)

rects = ax.patches
for rect, label in zip(rects, Y):
    height = rect.get_height()
    ax.text(rect.get_x() + rect.get_width() / 2, height + 5, label,
            ha='center', va='bottom')
```



From the graph:

- It looks like `Boston` is the area of most frequent incidents. `Harbor Island` has least amount of incidents which is only 2 in the entire year. In addition, `Roxbury` and `Dorchester` 's fire reports were less than Boston but still greater than most of other areas.

- We can conclude that `Boston` area's fire department should get most of resources and funds from govenment. Second is `Roxbury` and `Dorchester . Allston-Brighton` 's fire department should also has good amount of resources.
- If necessary, some area's fire department which are close to `Boston` could send their resources for helping Boston area's fire department
- From the false alarm type, we can assue that the central station in boston area will have lots of malicious false alarm, more monitors and police force needed to be placed near the central station area in boston. So, once the fire department receive the alarm, the can confirm and cancel the false alarm faster and don't have to go and check everytime. Which will save public resource and let fire department save more time and money on more important and enmergency issues.

In conclusion, we cleaned us our data by strip the space for each coloums and rows Since a few number of the incident does not have location, so we replace the name with Unspecify location