Task8 唐露函

8.1：各地延误

In [48]:
```
import pymysql # 连接 MySQL 所用包
import pandas as pd # 数据框所用包
# 建立和一个 MySQL 数据库的连接，需要指明 MySQL 数据库相关的信息
conn = pymysql.connect(
host='localhost',
user='root',
passwd='tang3186490', # xxx 为安装时设置的密码
port=3306,
db='tanglh'
)
# 产生一个游标，游标用来执行具体的 sql 命令
cur = conn.cursor()
sql1 = 'select distinct Dest from flight_info' # SQL 语句
cur.execute(sql1) # 执行输入的 SQL 命令
res= cur.fetchall() # 取回查询结果
```

In [52]:
```
#print(res)
```

In [*]:
```
output = pd.DataFrame() # 新建空数据框
for i in range(len(res)):
    sql2="select avg(DepTime-CRSDepTime), min(DepTime-CRSDepTime), max(DepTime-CRSDep
    cur.execute(sql2)
    res2 = cur.fetchall()
    new = pd.DataFrame(res2) # 将运行结果转换为数据框
    new['Dest'] = res[i] # 新建列，存入出发地
    # 重命名各列
    new.columns = ['Avg_DepDelay', 'Min_DepDelay', 'Max_DepDelay', 'Dest']
    # 对列进行重新排序
    colorder = [ 'Dest','Avg_DepDelay', 'Min_DepDelay', 'Max_DepDelay']
    new = new[colorder]
    # 将该次查询结果存入数据框 output 中
    output = pd.concat([output,new],ignore_index=True)
# 将输出数据框按 Avg_DepDelay 降序排列
outputdesc = output.sort_values(by='Avg_DepDelay', ascending=False)
# 输出数据框前 10 行
print(outputdesc.head(10))
```

	Dest	Avg_ArrDelay	Min_ArrDelay	Max_ArrDelay
214	OTH	35.8343	-2098	566
123	GST	28.3976	-21	257
54	CEC	27.9705	-1803	512
278	STX	27.1435	-70	569
283	TEX	23.8627	-32	595
58	CIC	22.5541	-2129	448
53	CDV	21.4463	-82	504
255	SBP	20.2134	-2357	2399
152	JNU	19.4915	-2141	2351
231	PSE	19.4878	-72	686

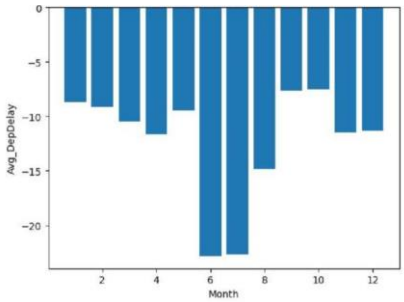
8.2 个月平均延误

In [20]:
```
import pymysql # pymysql 包用来连接 MySQL
import pandas as pd # pandas 包用来储存和计算所需的数据
import math # math 包用来完成一些计算
```

In [36]:
```
# 建立和一个 MySQL 数据库的连接，需要指明 MySQL 数据库相关的信息
conn = pymysql.connect(
    host='localhost',
    user='root',
    passwd='tang3186490',
    port=3306,
    db='tanglh'
)
cur = conn.cursor()  # 产生一个游标，游标用来执行具体的 sql 命令
```

In [39]:
```
for i in range(1,13):
    sql3 = "select avg(DepTime-CRSDepTime) from flight_time where Id in (select Id fr
    cur.execute(sql3)
    res = cur.fetchall()
    new = pd.DataFrame(res3) # 将运行结果转换为数据框
    new['Month'] = i # 新建列，存入出发地
    # 重命名各列
    new.columns = ['Avg_DepDelay', 'Month']
    # 对列进行重新排序
    colorder = ['Month', 'Avg_DepDelay']
    new = new[colorder]
    # 将该次查询结果存入数据框 output 中
    output = output.append(new, ignore_index=True)
#输出
import matplotlib.pyplot as plt
print(output)
plt.bar(output["Month"],output["Avg_ArrDelay"])
plt.xlabel("Month")
plt.ylabel("Avg+DepDelay")
```

	Month	Avg_ArrDelay
0	1	-8.6849
1	2	-9.1064
2	3	-10.5158
3	4	-11.6839
4	5	-9.4574
5	6	-22.8242
6	7	-22.7129
7	8	-14.8095
8	9	-7.6420
9	10	-7.4839
10	11	-11.4847
11	12	-11.3430



8.3 到达延误时间的影响因素

```
import pymysql # pymysql 包用来连接 MySQL
import pandas as pd # pandas 包用来储存和计算所需的数据
import math # math 包用来完成一些计算
```

```
# 建立和一个 MySQL 数据库的连接，需要指明 MySQL 数据库相关的信息
conn = pymysql.connect(
    host='localhost',
    user='root',
    passwd='tang3186490',
    port=3306,
    db='tanglh'
)
cur = conn.cursor()  # 产生一个游标，游标用来执行具体的 sql 命令
```

```
# 是否可以停止查询
is_over = False
# 给所要计算的值赋值 0
Distance_sum = 0 # 距离总和
Delay_sum = 0 # 到达延误时间总和
Distance_squared_sum = 0 #距离平方的总和
Delay_squared_sum = 0 # 到达延误时间平方的总和
Cross_sum = 0 # 交叉项总和
# 用 N 记录一共有多少条数据
N = 0
# 用 pos 记录查询的偏移位置
pos = 0
# 使用 while 循环来实现查询命令，直到查询完所有的数据
while is_over==False:
# 查询 flight_info 的 id, Distance, Cancelled
    sql = "select id, Distance, Cancelled from flight_info limit 500000 offset %s"%(p
    cur.execute(sql)
    res1 = cur.fetchall()
    new = pd.DataFrame(res1) # 获取 sql 查询结果
    if new.empty: # 如果本次查询为空，停止查询
        is_over = True
    else:
```

```
        new.columns = ['id', 'Cancelled'] # 修改 new 的列名
 # 筛选出 Cancelled=0 的数据，记为 new1
        new1 = new[new['Cancelled']==0]
# 处理数值过大的问题
# new1['Distance'] = new1['Distance']*0.001
 # 记录目前查询到的数据数量
        N += len(new1)
 # 选出航班 id
        IDlist = new1['id'].tolist()
 # 对于本次选出的 id,从 flight_time 查询到达的实际时间、计划时间
        sql2 = "select ArrTime, CRSArrTime from flight_time where Id in %s"
 # 执行查询 ArrTime, CRSArrTime 的 sql 语句 sql2，并储存到 new2 中
        cur.execute(sql2,(IDlist,))
        res2 = cur.fetchall()
        new2 = pd.DataFrame(res2)
        new2.columns = ['ArrTime', 'CRSArrTime']
# 计算所需的五个数
# 求出距离的总和
        Distance_sum += new1['Distance'].sum()
# 求出延误时间总和
        Delay_sum += (new2['ArrTime']-new2['CRSArrTime']).sum()
# 求出距离平方的总和
        Distance_squared_sum += (new1['Distance']**2).sum()
# 求出延误时间平方的总和
        Delay_squared_sum +=((new2['ArrTime']-new2['CRSArrTime'])**2).sum()
 # 求出交叉项总和
        Cross_sum += (new1['Distance']*(new2['ArrTime']-new2['CRSArrTime'])).sum()
        pos = pos + 500000 # 查询完就更新偏移位置
        new = pd.DataFrame()
```

```
# 用 r1 表示 r 的分子（只是方便说明，大家可以直接计算，不用拆开）
r1 = N*Cross_sum - Distance_sum*Delay_sum
# 用 r2 表示 r 的分母
r2 = math.sqrt(N*Distance_squared_sum-Distance_sum**2)*\
 math.sqrt(N*Delay_squared_sum-Delay_sum**2)
# 计算 r
r = r1/r2
# 显示 r，打印前六位小数
print(round(r,6))
```

Output:0.1110523