# 24. Working with Datasets

Topics

How to read data from a file
How to define a useful class for working with that data.
How to graphically display facts about that data using numpy and pyplot.

# Our Plan

We will cover these topics in the context of a single problem…

# The Problem

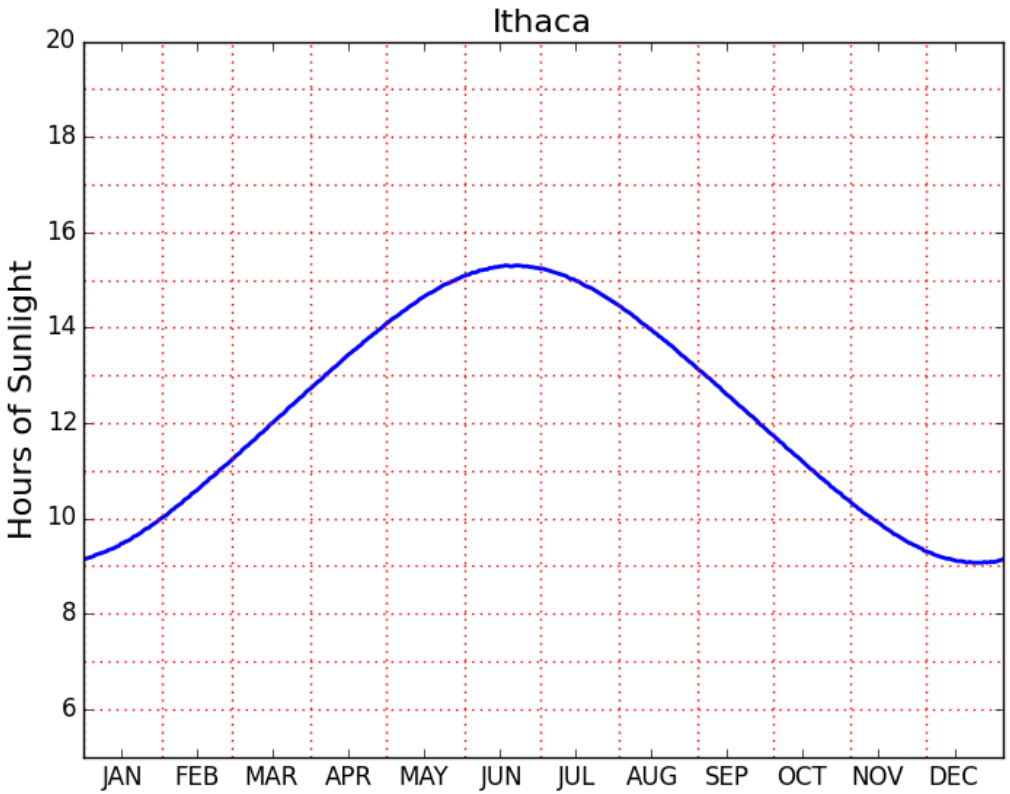
For various cities around the world, we would like to examine the "Sun Up" time throughout the year.

How does it vary from day to day?

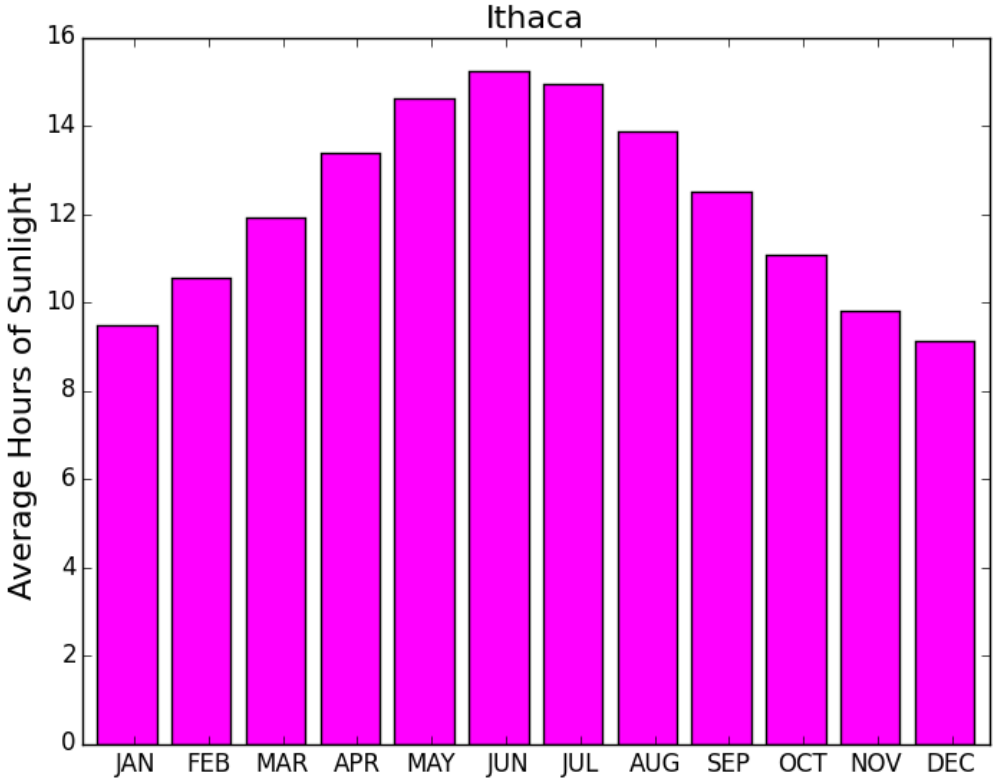
What are the monthly averages?

Sun Up Time = Sunset Time – Sunrise Time

# How Does Sun-Up Time Vary Day-to-Day?



# How Does Sun-Up Time Vary Month-To-Month?



# The Task Before Us…

1. Find a website where the data can be found.
2. Get that data into a file on our computer.
3. Understand how the data is laid out in the file.
4. Write python code that gets that data (or some aspect of it) into your python environment.

# Where Do We Get the Data?

Lots of choices. Google "Sunset Sunrise times"

We will use the U.S. Naval Observatory data service:

Visit:

**http://www.usno.navy.mil/**

# From the Website…

**Astronomical Applications**

**Data Services**

Sun and Moon rise and set times, Moon phases, eclipses, seasons, positions of solar system objects, and other data

Complete Sun and Moon Data for One Day
Sun or Moon Rise/Set Table for One Year
Phases of the Moon
more...

# We Downloaded Rise/Set Data For a Number of Cities

| | | | | |
|---|---|---|---|---|
| Anaheim | Anchorage | Arlington | Athens | Atlanta |
| Baltimore | Bangkok | Beijing | Berlin | Bogata |
| Boston | BuenosAires | Cairo | Chicago | Cincinnati |
| Cleveland | Denver | Detroit | Honolulu | Houston |
| Ithaca | Johannesburg | KansasCity | Lagos | London |
| LosAngeles | MexicoCity | Miami | Milwaukee | Minneapolis |
| Moscow | NewDelhi | NewYork | Oakland | Paris |
| Philadelphia | Phoenix | Pittsburgh | RiodeJaneiro | Rome |
| SanFrancisco | Seattle | Seoul | Sydney | Tampa |
| Teheran | Tokyo | Toronto | Washington | Wellington |

# One .dat File Per City

RiseSetData

```
Anaheim.dat
Anchorage.dat
Arlington.dat

        :

Toronto.dat
Washington.dat
Wellington.dat
```

We put all these files in a directory called RiseSetData

.dat and .txt files are common ways to house simple data. Don't worry about the difference.

# .txt and .dat Files have Lines

MyFile.dat

```
abcd
123 abc d fdd
xyz
3.14159    2.12345
```

There is an easy way to read the data in such a file line-by-line

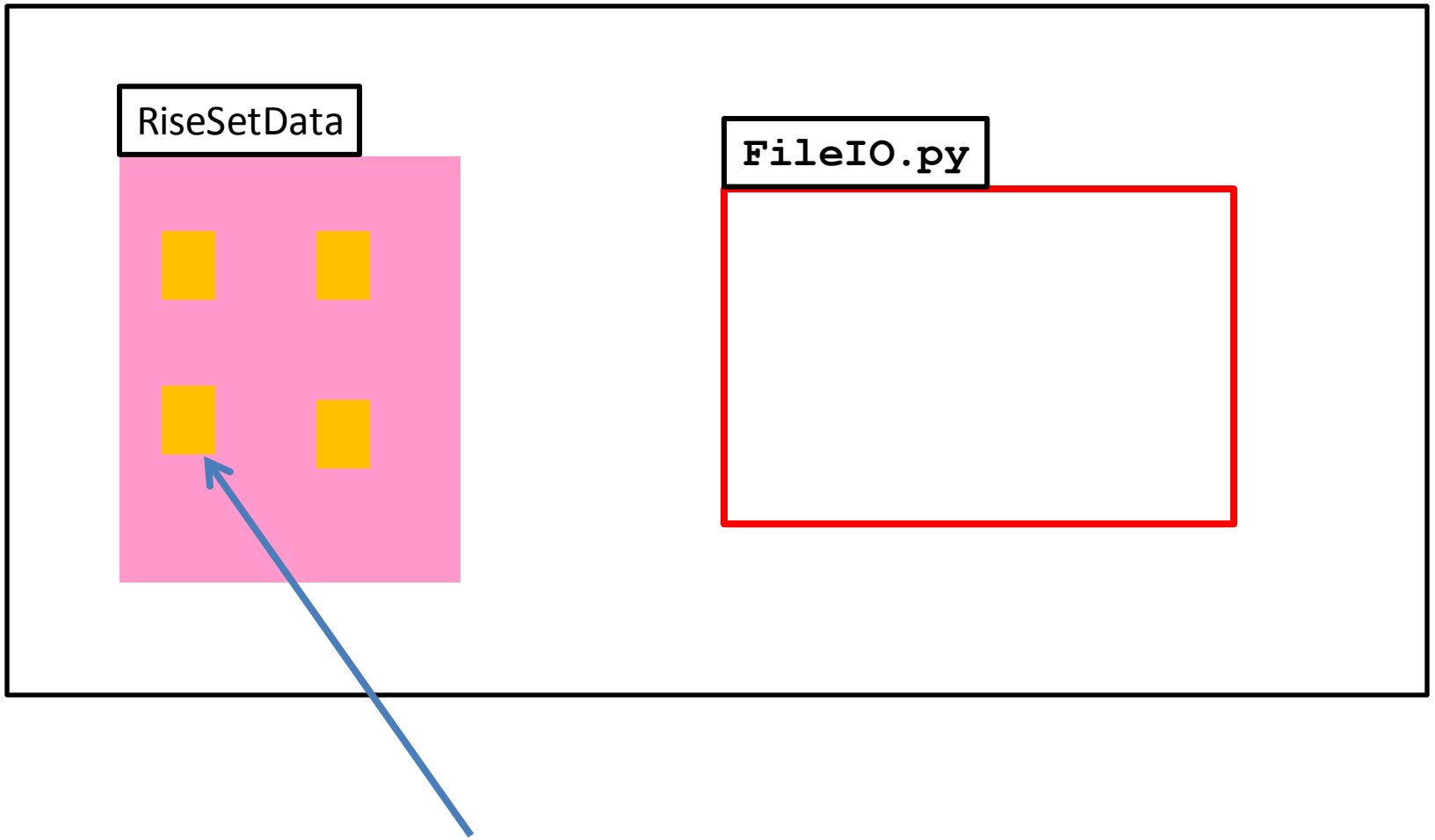
# Let's Read and Print the Data in `Ithaca.dat`

`FileIO.py`

```
FileName = 'RiseSetData/Ithaca.dat'
f = file(FileName, 'r')
for s in f:
    print s
f.close()
```

`RiseSetData` and `FileIO.py` must be in the same folder.

# Module and Data



# Reading Files: The Rules

```
FileName = 'RiseSetData/Ithaca.dat'
f = file(FileName, 'r')
for s in f:
    print s
f.close()
```

'r' means open the file for reading

# Reading Files: The Rules

```
FileName = 'RiseSetData/Ithaca.dat'
f = file(FileName, 'r')
for s in f:
    print s
f.close()
```

f refers to a file object

# Reading Files: The Rules

```
FileName = 'RiseSetData/Ithaca.dat'
f = file(FileName, 'r')
for s in f:
    print s
f.close()
```

This will print every line in the file.

# Reading Files: The Rules

```
FileName = 'RiseSetData/Ithaca.dat'
f = file(FileName, 'r')
for s in f:
    print s
f.close()
```

The file should be closed after it is completely read.

What do the lines in Ithaca.dat look like?

# There Are 33 Lines

```
Ithaca
W07629N4226
1   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
2   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
3   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S


28  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
29  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
30  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
31  R S  R S  R S  R S  R S  R S  R S
```

The provider of the file typically tells you how the data is structured

# From the Naval Observatory Website

The first line names the city and the second line encodes its latitude and longitude, e.g.,

```
Ithaca
W07629N4226
```

and …

The rise and set times are then specified day-by-day with the data for each month housed in a pair of columns.

In particular, columns 2k and 2k+1 have the rise and set times for month k (Jan=1, Feb = 2, Mar = 3, etc.)

Column 1 specifies day-of-the-month, 1 through 31. Blanks are used for nonexistent dates (e.g., April 31).

# The Data for a Particular City is Housed in a 33-line .dat file

```
Ithaca
W07629N4226
 1   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
 2   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
 3   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S


28   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
29   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
30   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
31   R S  R S  R S  R S  R S  R S  R S
```

Line 1 has the name of the city

# The Data for a Particular City is Housed in a 33-line .dat file

```
Ithaca
W07629N4226
 1   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
 2   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
 3   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S


28   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
29   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
30   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
31   R S  R S  R S  R S  R S  R S  R S
```

Line 2 encodes its longitude and latitude

# Helper Function: LongLat

A latlong string has length 11

W08140N4129

```
def LongLat(s):
    Long = float(s[1:4])+float(s[4:6])/60
    if s[0]=='E':
        Long = -Long
    Lat = float(s[7:9])+float(s[9:11])/60
    if s[6]=='S':
        Lat = -Lat
    return (Lat,Long)
```

# The Data for a Particular City is Housed in a 33-line .dat file

```
Ithaca
W07629N4226
 1   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
 2   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
 3   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S


28   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
29   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
30   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
31   R S  R S  R S  R S  R S  R S  R S
```

The remaining lines house the rise-set data.
Each R and S is a length-4 string: '0736'

# Helper Function: `ConvertTime`

```
def ConvertTime(s):
    x = float(s[:2])+float(s[2:])/60
    return x
```

In comes a length-4 string and back comes a float that encodes the time in hours

'0736' ----> 7 + 36/60 hours ----> 7.6

# The Data for a Particular City is Housed in a 33-line .dat file

```
Ithaca
W07629N4226
 1   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
 2   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
 3   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S


28   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
29   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
30   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
31   R S  R S  R S  R S  R S  R S  R S
```

Day -Number followed by 12 rise-set pairs, one pair for each month

# The Data for a Particular City is Housed in a 33-line .dat file

```
Ithaca
W07629N4226
1   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
2   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
3   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S


28  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
29  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
30  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
31  R S  R S  R S  R S  R S  R S  R S
```

Day -Number followed by 11 rise-set pairs, one pair for each month except February

# The Data for a Particular City is Housed in a 33-line .dat file

```
Ithaca
W07629N4226
1   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
2   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
3   R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S


28  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
29  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
30  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S  R S
31  R S  R S  R S  R S  R S  R S  R S
```

Day -Number followed by 7 rise-set pairs, one pair for each 31-day moth

# Recall the Motivating Problem

For various cities around the world, we would like to examine the "Sun Up" time throughout the year.

How does it vary from day to day?

What are the monthly averages?

Let's define a class that makes this easy.

# The Class Daylight

5 Attributes

```
Name :     name of the city [str]
Lat:       latitude in degrees  [float]
Long:      longitude in degrees [float]
RiseTime:  rise time in hours
                 [length-365 numpy array]
SetTime:   set time in hours
                   [length-365 numpy array]
```

# What the Constructor Will Do

It will have one argument: the name of a city.

It will then read the .dat file associated with that city and proceed to set up the 5 attributes.

# The Constructor

```
def Daylight(self,CityName):

    FileName = 'RiseSetData/'+ CityName + '.dat'

    f = file(FileName, 'r')


                :
```

Notice how the complete file name is computed from the city name

# The Constructor

```
lineNum = 0
for s in f:
    parts = s.split()
    lineNum+=1
    if lineNum==1:
        self.City = parts[0]
    elif lineNum==2:
       (self.Lat,self.Long) = LatLong(parts[0])
    else:
```

Code that builds the RiseTime and SetTime arrays

```
f.close()
```

Recall how split works…

# s.split

```
s = '1  0535   0816  0542  0713'
x = s.split()
print x
['1','0535','0816','0542','0713']
```

# The Method SunUp

```
def SunUp(self):
    """ Returns a length-365 numpy array
    that indicates the sun up time each
    day of the year.

    PreC: self is a Daylight object
    """
    return self.SetTime - self.RiseTime
```

Subtract the rise time values from the set time values. With numpy, can subtract arrays

# Now Let's Display Some Ithaca Rise/Set Data

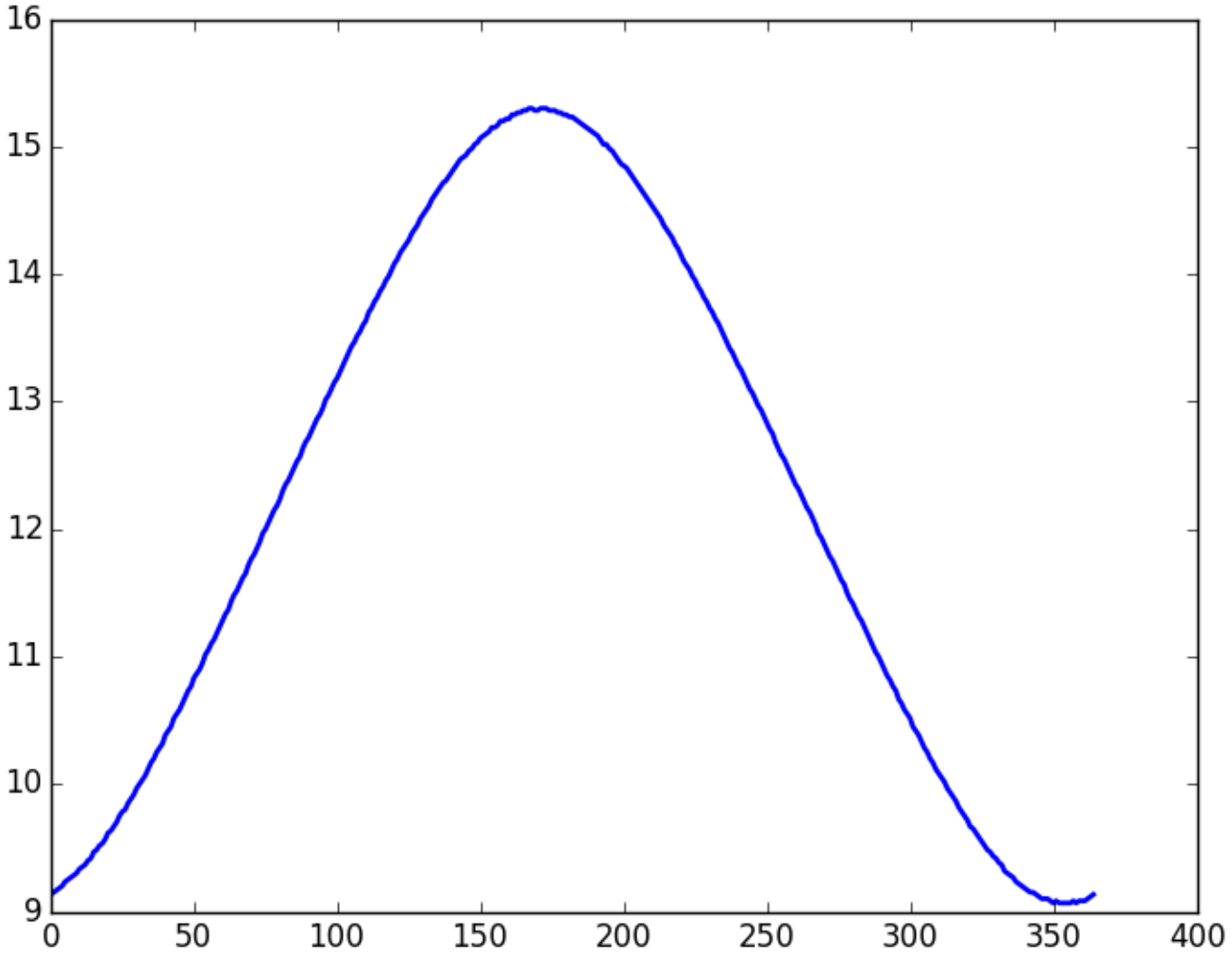
```
from numpy import *
From pylab import *
```

We use numpy for arrays and pylab for plotting

# A Simple Plot

```
A = Daylight('Ithaca')
D = A.SunUp()
plot(D)
show()
```

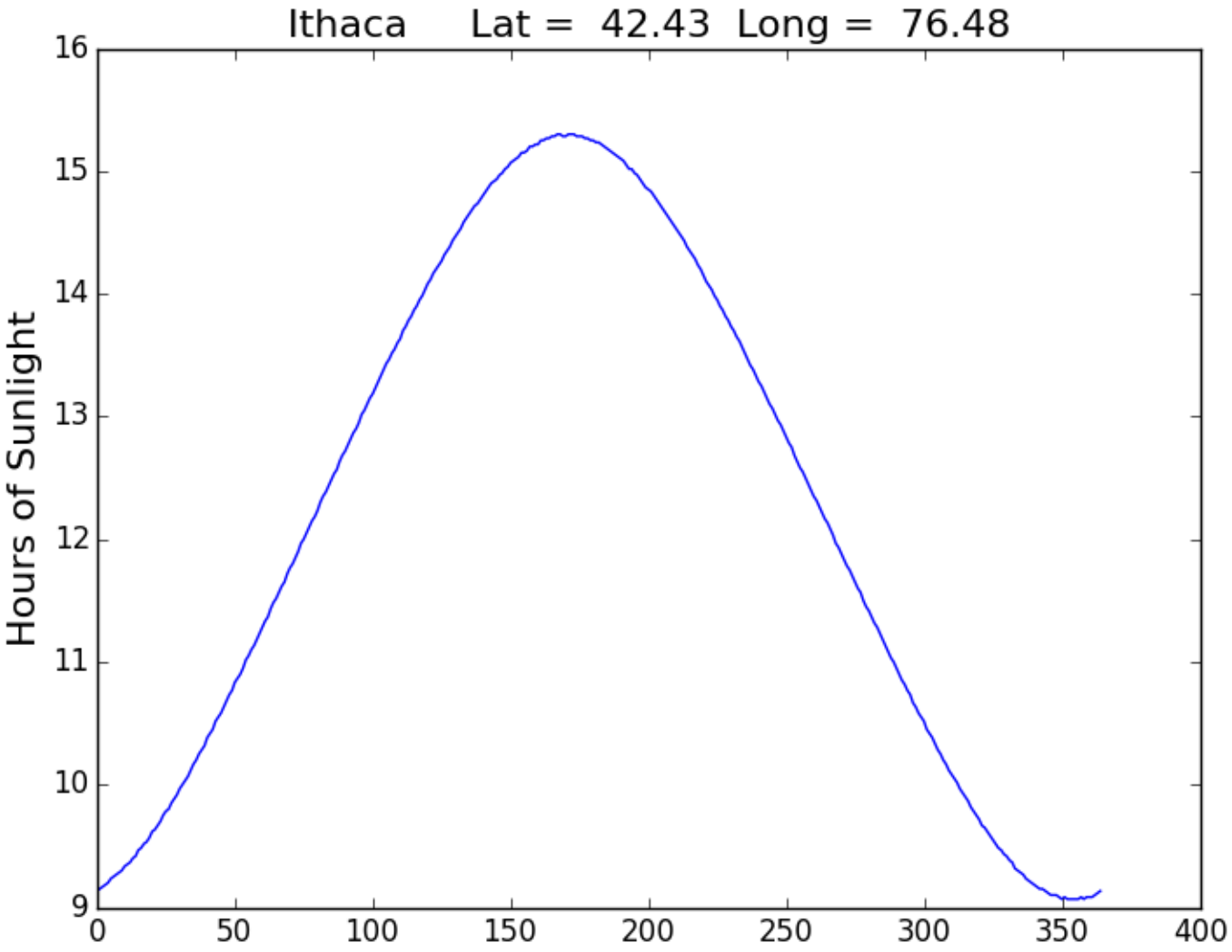
This is how you display the values in a numpy array like D.




How about a title and a labeling of the y-axis?

# A Simple Plot

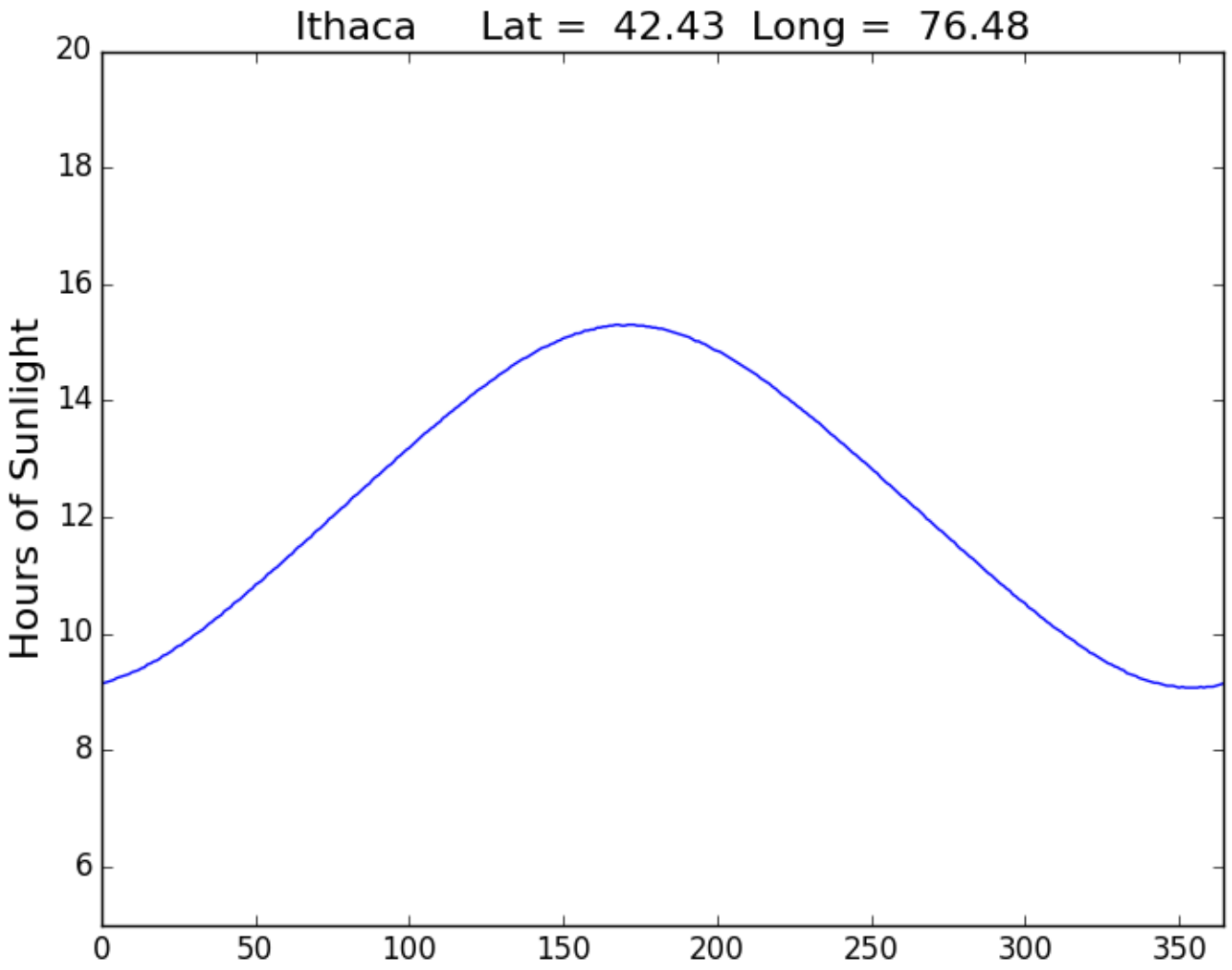
```
A = Daylight('Ithaca')
D = A.SunUp()
plot(D)

titlestr = '%s  Lat = %6.2f  Long = %6.2f' % (A.City,A.Lat,A.Long)
title(titlestr,fontsize=16)
ylabel('Hours of Sunlight',fontsize=16)
show()
```



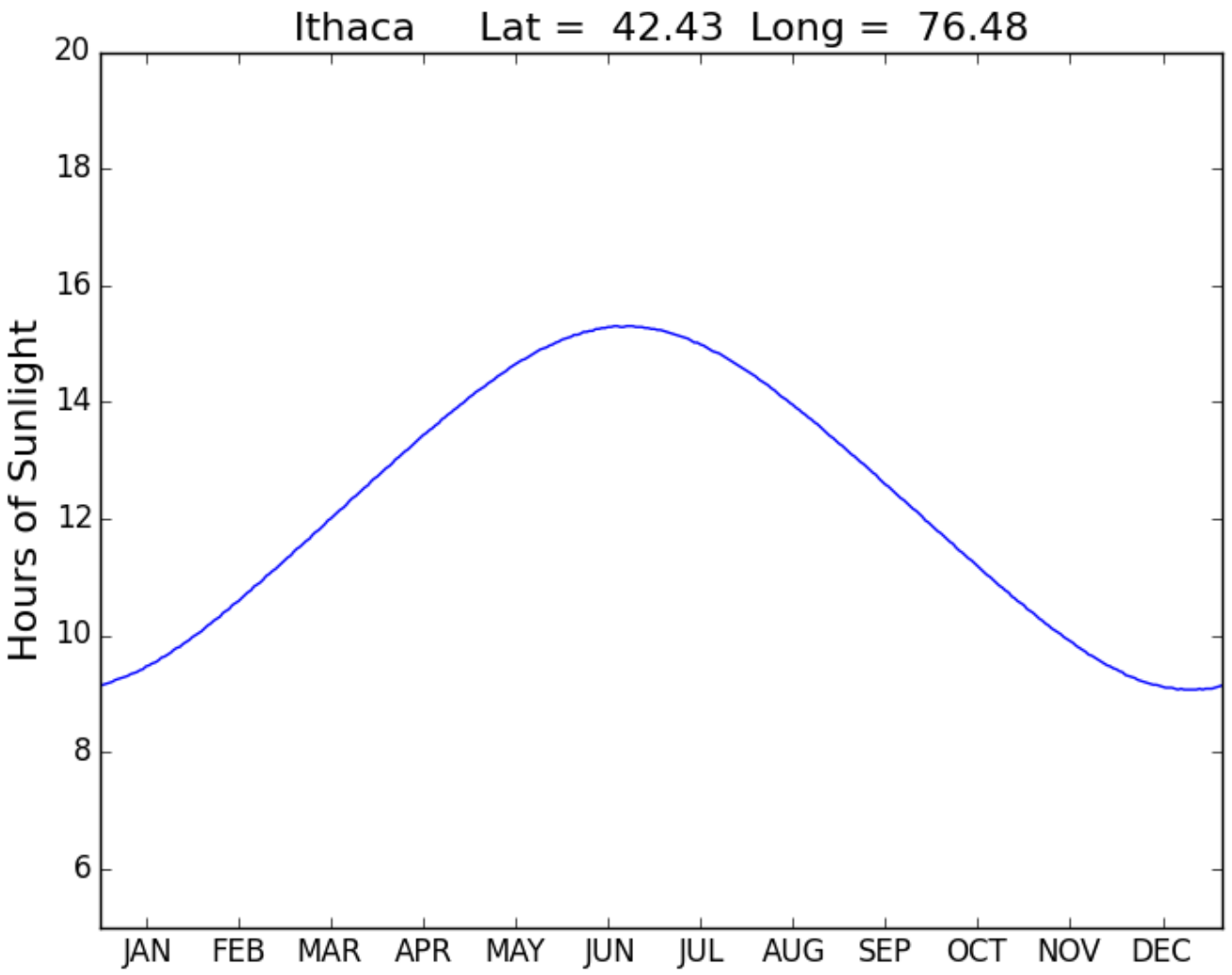

Modify the x range and the y range

# A Simple Plot

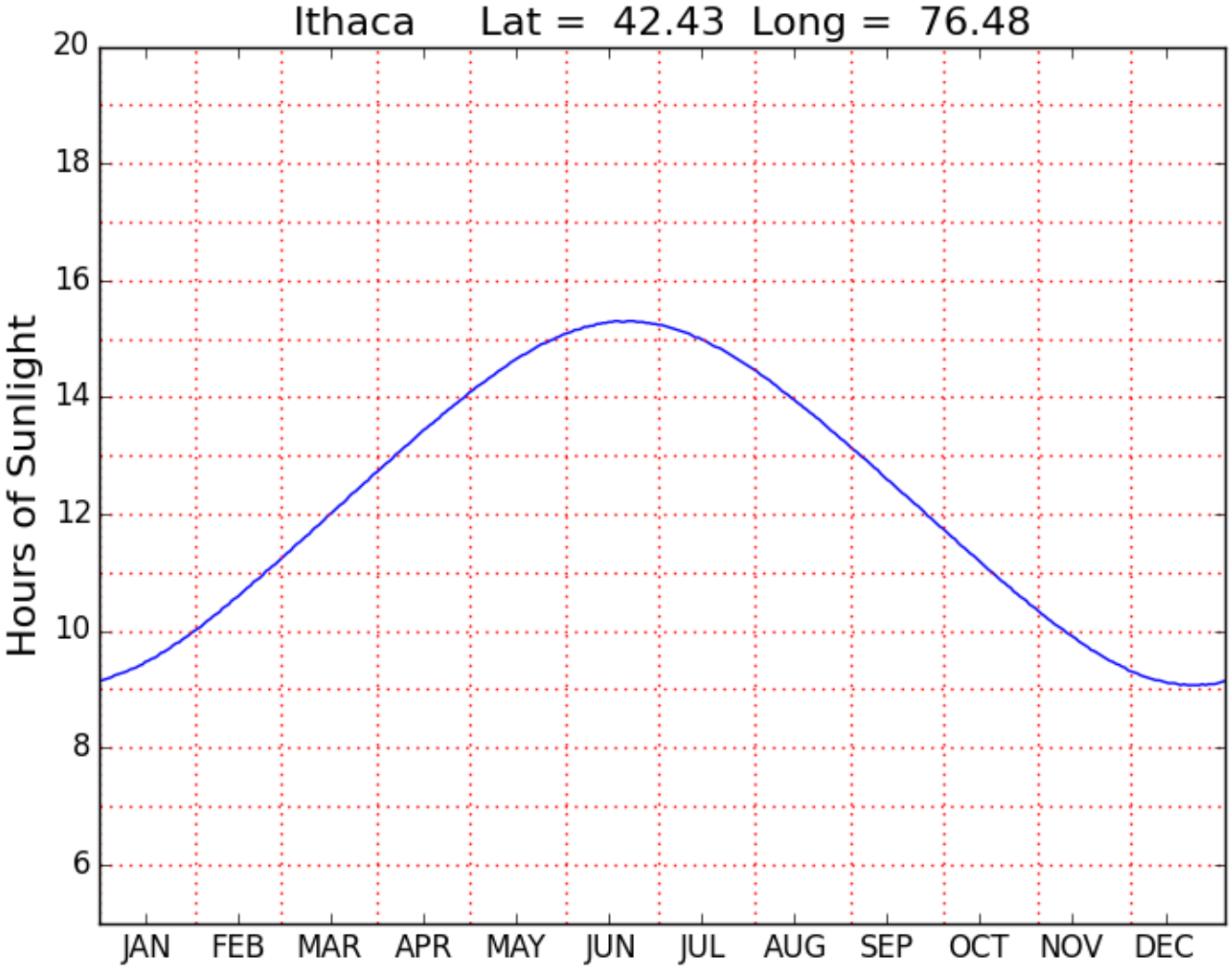
```
A = Daylight('Ithaca')
D = A.SunUp()
plot(D)

titlestr = '%s  Lat = %6.2f  Long = %6.2f' % (A.City,A.Lat,A.Long)
title(titlestr,fontsize=16)
ylabel('Hours of Sunlight',fontsize=16)

xlim(0,364)
ylim(5,20)

show()
```




Label the x-axis with month names


Ithaca Lat = 42.43 Long = 76.48
Hours of Sunlight
20
18
16
14
12
10
8
6
JAN
FEB
MAR
APR
MAY
JUN
JUL
AUG
SEP
OCT
NOV
DEC

Add a Grid

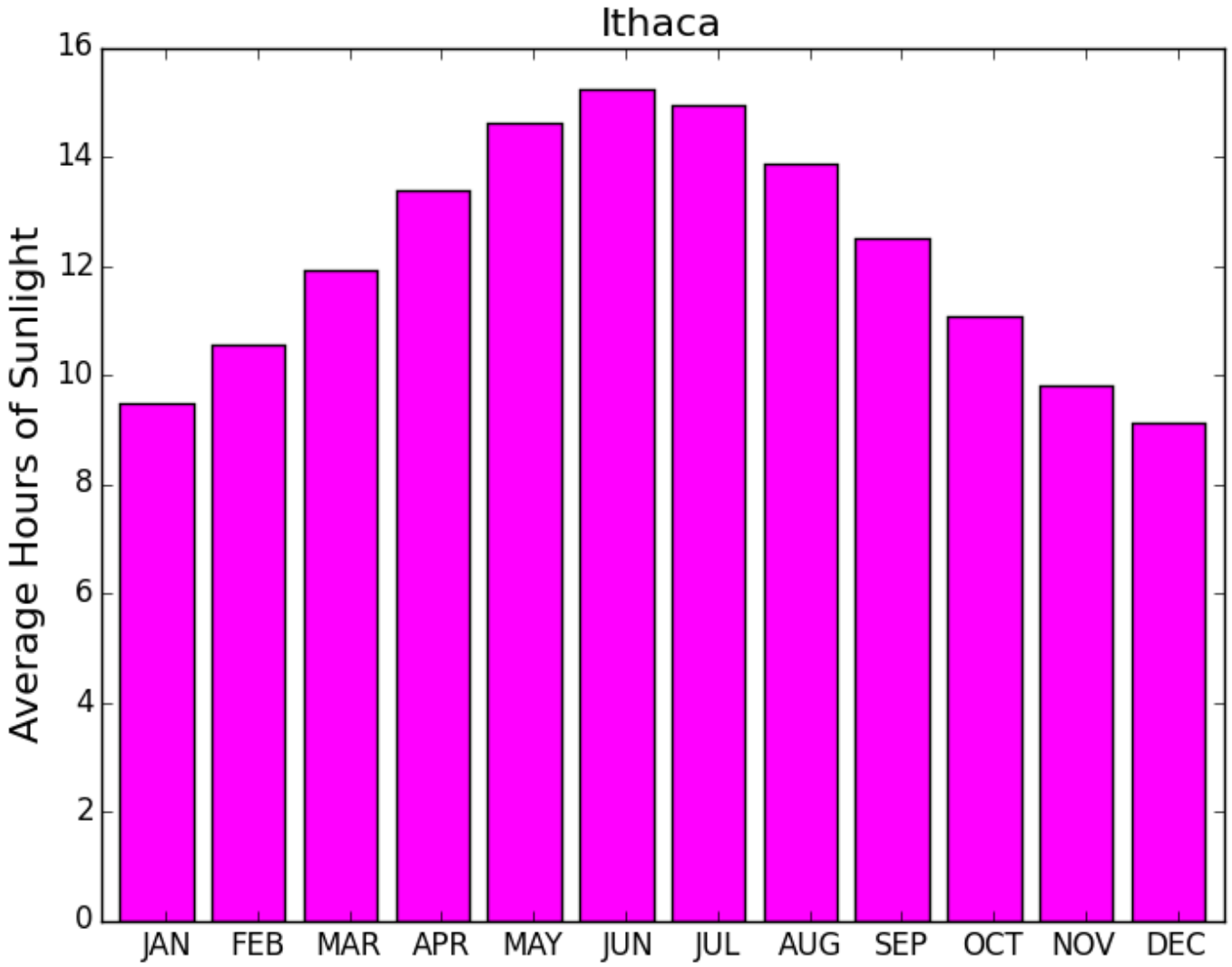

Ithaca Lat = 42.43 Long = 76.48
Hours of Sunlight
20
18
16
14
12
10
8
6
JAN
FEB
MAR
APR
MAY
JUN
JUL
AUG
SEP
OCT
NOV
DEC

# Monthly Averages

```
def MonthAves(self):
    x = zeros((12,1))
    D = self.SunUp()
    start  = [0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334]
    finish = [30, 58, 89, 119, 150, 180, 211, 242, 272, 303, 333,364]
    for k in range(12):
        z = D[start[k]:finish[k]]
        x[k] = sum(z)/len(z)
    return x
```

# A Bar Plot

```
A = Daylight('Ithaca')
M = A.MonthAves()

bar(range(12),M,facecolor='magenta')
xlim(-.2,12)
ylabel('Average Hours of Sunlight')
title(A.City,fontsize=16)
show()
```


Ithaca
Average Hours of Sunlight
0
2
4
6
8
10
12
14
16
JAN
FEB
MAR
APR
MAY
JUN
JUL
AUG
SEP
OCT
NOV
DEC