In [1]:
```
                                                # Design spiral star
import turtle as t
a1=t.Turtle()
a1.pencolor('blue')
for i in range (50):
    a1.forward(i * 10)
    a1.left(200)
t.done()
```

In [ ]:
```
# Design spiral square
import turtle as t
a1 = t.Turtle()
a1.pencolor('blue')
for i in range (200):
    a1.forward(i)
    a1.left(100)
a1.done()
```

In [ ]:
```
# Design Hexagon
from turtle import *
pencolor('brown')
colors = ['blue','green','orange','black','red','purple']
for i in range(360):
    pencolor(colors[i%6])
    width(i/100 + 1)
    forward(i)
    left(59)
```

In [2]:
```
# Design Circle
from turtle import *
pensize(20)
pencolor('orange')
fillcolor('blue')
begin_fill()
circle(100)
end_fill()
```

In [ ]:
```
# Design Tangent Circle
from turtle import *
pensize(5)
pencolor('red')
for i in range(15):
    circle(10*i)
```

In [ ]:
```
# Design spiral Circle
from turtle import *
pensize(3)
pencolor('purple')
for i in range(100):
    circle(10+i,45)
```

In [ ]:
```
# Design Circle
from turtle import *
pensize(3)
colors = ['blue','green','red','yellow','orange','violet']
for i in range(30):
    pencolor(colors[i%6])
    circle(10*i)
    up()
    sety((10*i)*(-1))
    down()
```

In [1]:
```
# Design Olympics Logo
from turtle import *
pensize(6) # Set the pensize to 6 pixels
r1 = ["blue","black","red"]
for i in range(3):
    penup()
    pencolor(r1[i%3])
    goto(i*100,0)
    pendown()
    circle(45)
r2 = ["","violet","","green"]
for i in range(1,4,2):
    penup()
    pencolor(r2[i])
    goto(i*55,-55)
    pendown()
    circle(50)
```

In [ ]:
```
# Design Audi logo
from turtle import *
pensize(10)
colors = ['blue','green','red','yellow']
for i in range(4):
    pencolor(colors[i%4])
    penup()
    goto(i*70,0)
    begin_fill()
    pendown()
    circle(50)
    end_fill()
```

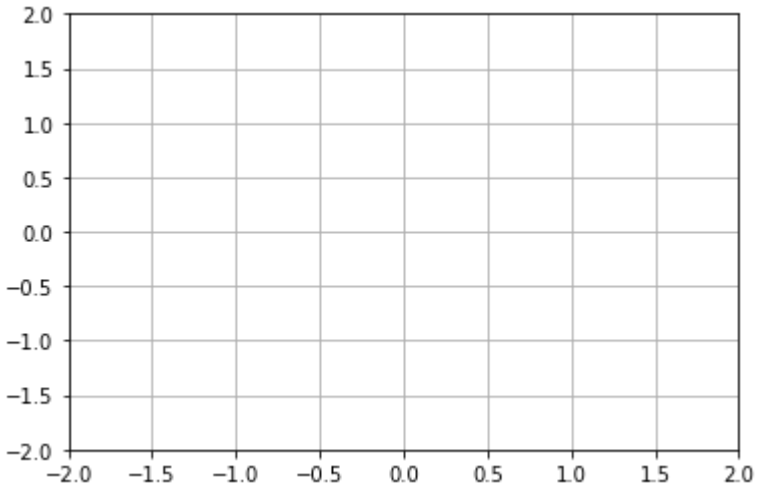
In [11]:
```
# Design Audi logo (example)
from turtle import *
from random import *
pensize(3)
colors = ['blue','green','red','yellow','pink','cyan']
for i in range(4):
    fillcolor(choice(colors))
    pencolor(colors[i%4])
    penup()
    goto(i*70,0)
    begin_fill()
    pendown()
    circle(50)
    end_fill()
```

In [ ]:
```

```

In [16]:
```
"""
============================
The double pendulum problem
============================

This animation illustrates the double pendulum problem.
"""

# Double pendulum formula translated from the C code at
# http://www.physics.usyd.edu.au/~wheat/dpend_html/solve_dpend.c

from numpy import sin, cos
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integrate
import matplotlib.animation as animation

G = 9.8  # acceleration due to gravity, in m/s^2
L1 = 1.0  # length of pendulum 1 in m
L2 = 1.0  # length of pendulum 2 in m
M1 = 1.0  # mass of pendulum 1 in kg
M2 = 1.0  # mass of pendulum 2 in kg


def derivs(state, t):

    dydx = np.zeros_like(state)
    dydx[0] = state[1]

    del_ = state[2] - state[0]
    den1 = (M1 + M2)*L1 - M2*L1*cos(del_)*cos(del_)
    dydx[1] = (M2*L1*state[1]*state[1]*sin(del_)*cos(del_) +
               M2*G*sin(state[2])*cos(del_) +
               M2*L2*state[3]*state[3]*sin(del_) -
               (M1 + M2)*G*sin(state[0]))/den1

    dydx[2] = state[3]

    den2 = (L2/L1)*den1
    dydx[3] = (-M2*L2*state[3]*state[3]*sin(del_)*cos(del_) +
               (M1 + M2)*G*sin(state[0])*cos(del_) -
               (M1 + M2)*L1*state[1]*state[1]*sin(del_) -
```

```
                (M1 + M2)*G*sin(state[2]))/den2

    return dydx

# create a time array from 0..100 sampled at 0.05 second steps
dt = 0.05
t = np.arange(0.0, 20, dt)

# th1 and th2 are the initial angles (degrees)
# w10 and w20 are the initial angular velocities (degrees per second)
th1 = 120.0
w1 = 0.0
th2 = -10.0
w2 = 0.0

# initial state
state = np.radians([th1, w1, th2, w2])

# integrate your ODE using scipy.integrate.
y = integrate.odeint(derivs, state, t)

x1 = L1*sin(y[:, 0])
y1 = -L1*cos(y[:, 0])

x2 = L2*sin(y[:, 2]) + x1
y2 = -L2*cos(y[:, 2]) + y1

fig = plt.figure()
ax = fig.add_subplot(111, autoscale_on=False, xlim=(-2, 2), ylim=(-2, 2))
ax.grid()

line, = ax.plot([], [], 'o-', lw=2)
time_template = 'time = %.1fs'
time_text = ax.text(0.05, 0.9, '', transform=ax.transAxes)


def init():
    line.set_data([], [])
    time_text.set_text('')
    return line, time_text


def animate(i):
```

```
    thisx = [0, x1[i], x2[i]]
    thisy = [0, y1[i], y2[i]]

    line.set_data(thisx, thisy)
    time_text.set_text(time_template % (i*dt))
    return line, time_text

ani = animation.FuncAnimation(fig, animate, np.arange(1, len(y)),
                              interval=25, blit=True, init_func=init)

# ani.save('double_pendulum.mp4', fps=15)
plt.show()
```



In [22]:
```
from turtle import *
from random import randint


def create_rectangle(turtle, color, x, y, width, height):
    turtle.penup()
    turtle.color(color)
    turtle.fillcolor(color)
    turtle.goto(x, y)
    turtle.pendown()
    turtle.begin_fill()

    turtle.forward(width)
    turtle.left(90)
    turtle.forward(height)
    turtle.left(90)
    turtle.forward(width)
    turtle.left(90)
    turtle.forward(height)
    turtle.left(90)

    # fill the above shape
    turtle.end_fill()
    # Reset the orientation of the turtle
    turtle.setheading(0)


def create_circle(turtle, x, y, radius, color):
    oogway.penup()
    oogway.color(color)
    oogway.fillcolor(color)
    oogway.goto(x, y)
    oogway.pendown()
    oogway.begin_fill()
    oogway.circle(radius)
    oogway.end_fill()


BG_COLOR = "black"

# "Yesterday is history, tomorrow is a mystery, but today is a gift. That is why it is called the present."
#                                                                 — Oogway to Po under the peach tree, Kung Fu Pand
```

```
a
oogway = Turtle()
# set turtle speed
oogway.speed(2)
screen = oogway.getscreen()
# set background color
screen.bgcolor(BG_COLOR)
# set tile of screen
screen.title("Merry Christmas")
# maximize the screen
screen.setup(width=1.0, height=1.0)

y = -100
# create tree trunk
create_rectangle(oogway, "red", -15, y-60, 30, 60)

# create tree
width = 240
oogway.speed(10)
while width > 10:
    width = width - 10
    height = 10
    x = 0 - width/2
    create_rectangle(oogway, "green", x, y, width, height)
    y = y + height

# create a star a top of tree
oogway.speed(1)
oogway.penup()
oogway.color('yellow')
oogway.goto(-20, y+10)
oogway.begin_fill()
oogway.pendown()
for i in range(5):
    oogway.forward(40)
    oogway.right(144)
oogway.end_fill()

tree_height = y + 40

# create moon in sky
# create a full circle
create_circle(oogway, 230, 180, 60, "white")
```

```
# overlap with full circle of BG color to make a crescent shape
create_circle(oogway, 220, 180, 60, BG_COLOR)

# now add few stars in sky
oogway.speed(10)
number_of_stars = randint(20,30)
# print(number_of_stars)
for _ in range(0,number_of_stars):
    x_star = randint(-(screen.window_width()//2),screen.window_width()//2)
    y_star = randint(tree_height, screen.window_height()//2)
    size = randint(5,20)
    oogway.penup()
    oogway.color('white')
    oogway.goto(x_star, y_star)
    oogway.begin_fill()
    oogway.pendown()
    for i in range(5):
        oogway.forward(size)
        oogway.right(144)
    oogway.end_fill()

# print greeting message
oogway.speed(1)
oogway.penup()
msg = "Merry Christmas"
oogway.goto(0, -200)  # y is in minus because tree trunk was below x axis
oogway.color("white")
oogway.pendown()
oogway.write(msg, move=False, align="center", font=("Arial", 15, "bold"))

oogway.hideturtle()
screen.mainloop()
```

In [ ]:

```
### Idiomatic Python
 - Transformation Code into Beautiful,Idiomatic Python
 - Replaces traditional index manipulation with python Code looping idioms
```

In [24]:
```
# Looping over a range of number
for i in [0,1,2,3,4,5]:
    print(i ** 2,end=' ')

print()
#pythonic-way
for i in range(6):
    print(i ** 2,end=" ")
```

```
0 1 4 9 16 25
0 1 4 9 16 25
```

In [25]:
```
# Looping over the collection(List)
li=[1,2,3,4,5]
for i in range(len(li)):
    print(li[i],end=' ')
print()
#pythonic-way
for i in li:
    print(i,end=' ')
```

```
1 2 3 4 5
1 2 3 4 5
```

In [26]:
```
# looping from backwards
li = [1,2,3,4,5]
for i in range(len(li)-1,-1,-1):
    print(li[i],end=' ')
print()
#pythonic-way
for i in reversed(li):
    print(i,end=' ')
```

```
5 4 3 2 1
5 4 3 2 1
```

In [ ]:
```
# Looping over the collection with Index
li=[]
```

Lambda Functions

- anonymous functions means that a function is without a name.
- any lambda function on python will be defined with lambda.
- syntax : lambda arg : expression

In [1]:
```
def square(n):
    return n*n
square(10)
```

Out[1]:
```
100
```

In [2]:
```
a = lambda x : x * x
print (a(10))
```

```
100
```

Use of Lambda with filter

In [3]:
```
def filterList(li):
    a = []
    for i in li:
        if i % 2 == 0:
            a.append(i)
    return a
li = [1,2,3,4,5,6,7,8]
filterList(li)
```

Out[3]:
```
[2, 4, 6, 8]
```

In [4]:
```
li = [1,2,3,4,5,6,7,8]
lam_li = list(filter(lambda x :(x%2==0),li))
print(lam_li)
```

```
[2, 4, 6, 8]
```

Lambda with map()

- map() function in python takes ina function and list as argument.

In [6]:
```
def squareList(li):
    a= []
    for i in li:
        a.append(i*2)
    return a
li = [1,2,3,4,5,6,7,8]
squareList(li)
```

Out[6]:
```
[2, 4, 6, 8, 10, 12, 14, 16]
```

In [7]:
```
# Lambda with map()
li = [1,2,3,4,5,6,7,8]
map_list=list(map(lambda x:x*2,li))
print(map_list)
```

```
[2, 4, 6, 8, 10, 12, 14, 16]
```

Lambda with reduce()

In [8]:
```
def sumlist(li):
    s = 0
    for i in li:
        s += i
    return s
li = [1,2,3,4,5,6,7,8]
sumlist(li)
```

Out[8]:
```
36
```

In [9]:
```
from functools import reduce
li = [1,2,3,4,5,6,7,8]
s = reduce((lambda x,y:x+y),li)
print(s)
```

```
36
```

standard package

Pandas,Numpy and Matplotlib

In [ ]:
```
### Pandas
### use cases
- data cleaning
- data transformation
- data analysis

### notations
-series
  - a series is a one dimensional object (which generating of output)
-dataframes
  - a dataframe is a two dimensional object
```

In [2]:
```
import pandas as pd
internal1 = {'s1': 35,'s2':35,'s3':35}
internal2 = {'s1':35,'s2':35,'s3':35}
internal1 = pd.Series(internal1)
internal2 = pd.Series(internal2)
print(internal1)
print(internal2)
```

```
s1    35
s2    35
s3    35
dtype: int64
s1    35
s2    35
s3    35
dtype: int64
```

In [3]:
```
final = {'Internal1':internal1,'Internal2':internal2}
final = pd.DataFrame(final)
print(final)
```

```
    Internal1  Internal2
s1         35         35
s2         35         35
s3         35         35
```

In [4]:
```
final.columns # returns the name of columns
```

Out[4]:
```
Index(['Internal1', 'Internal2'], dtype='object')
```

In [5]:
```
final.values # returns all row values in two dimensional array
```

Out[5]:
```
array([[35, 35],
       [35, 35],
       [35, 35]], dtype=int64)
```

In [6]:
```
final.values[1] # 1 represents the row index
```

Out[6]:
```
array([35, 35], dtype=int64)
```

In [9]:
```
final.values[2][0] = 350 # update the data frame values
```

In [10]:
```
final.values
```

Out[10]:
```
array([[ 35,  35],
       [ 35,  35],
       [350,  35]], dtype=int64)
```

In [11]:
```
for row in final.values:
    print('Internal1 - ',row[0],'Internal2 - ',row[1])
```

```
Internal1 -  35 Internal2 -  35
Internal1 -  35 Internal2 -  35
Internal1 -  350 Internal2 -  35
```

In [13]:
```
final.loc['s4'] = [20,18] # insert the new records in the data frame
final
```

Out[13]:

	Internal1	Internal2
s1	35	35
s2	35	35
s3	350	35
s4	20	18

In [14]:
```
final.values[2] = [155,255]
final
```

Out[14]:

	Internal1	Internal2
s1	35	35
s2	35	35
s3	155	255
s4	20	18

In [15]:
```
pwd
```

Out[15]:
```
'C:\\Users\\HP\\Desktop\\psap'
```

In [26]:
```
filePath = 'data files/Income.csv'
def readCsvData(filePath):
    return pd.read_csv(filePath)
readCsvData(filePath)
```

Out[26]:

	GEOID	State	2005	2006	2007	2008	2009	2010	2011	2012	2013
0	04000US01	Alabama	37150	37952	42212	44476	39980	40933	42590	43464	41381
1	04000US02	Alaska	55891	56418	62993	63989	61604	57848	57431	63648	61137
2	04000US04	Arizona	45245	46657	62993	46914	45739	46896	48621	47044	50602
3	04000US05	Arkansas	36658	37057	40795	39586	36538	38587	41302	39018	39919
4	04000US06	California	51755	55319	55734	57014	56134	54283	53367	57020	57528

In [25]:
```
filePath = 'data files/Income.csv'
def readCsvData(filePath):
    return pd.read_csv(filePath)
df=readCsvData(filePath)
```

In [22]:
```
# Extract income of all states in year 2013
# Alabama : 41381
# ................
for row in df.values:
    print(row[1], ' : ',row[-4])
```

```
Alabama  :  40933
Alaska  :  57848
Arizona  :  46896
Arkansas  :  38587
California  :  54283
```

In [21]:
```
print(df.values)
```

```
[['04000US01' 'Alabama' 37150 37952 42212 44476 39980 40933 42590 43464
  41381]
 ['04000US02' 'Alaska' 55891 56418 62993 63989 61604 57848 57431 63648
  61137]
 ['04000US04' 'Arizona' 45245 46657 62993 46914 45739 46896 48621 47044
  50602]
 ['04000US05' 'Arkansas' 36658 37057 40795 39586 36538 38587 41302 39018
  39919]
 ['04000US06' 'California' 51755 55319 55734 57014 56134 54283 53367
  57020 57528]]
```

In [24]:
```
# Average income of California
def avgofState():
    s = 0
    for i in range(2,11):
        s += df.values[4][i]
    return s // len(df.values[4][2:])
avgofState()
```

Out[24]: 55350

In [27]:
```
# Function which only displays the column names in the list
def columnnames(df):
    li = []
    columns=df.columns
    for i in columns:
        li.append(i)
    return li
columnnames(df)
```

Out[27]:
```
['GEOID',
 'State',
 '2005',
 '2006',
 '2007',
 '2008',
 '2009',
 '2010',
 '2011',
 '2012',
 '2013']
```

In [4]:
```
import pandas as pd
filePath = 'data files/RegularSeasonCompactResults.csv'
def readCsvData(filePath):
    return pd.read_csv(filePath)
readCsvData(filePath)
```

Out[4]:

	Season	Daynum	Wteam	Wscore	Lteam	Lscore	Wloc	Numot
0	1985	20	1228	81	1328	64	N	0
1	1985	25	1106	77	1354	70	H	0
2	1985	25	1112	63	1223	56	H	0
3	1985	25	1165	70	1432	54	H	0
4	1985	25	1192	86	1447	74	H	0
5	1985	25	1218	79	1337	78	H	0
6	1985	25	1228	64	1226	44	N	0
7	1985	25	1242	58	1268	56	N	0
8	1985	25	1260	98	1133	80	H	0
9	1985	25	1305	97	1424	89	H	0
10	1985	25	1307	103	1288	71	H	0
11	1985	25	1344	75	1438	71	N	0
12	1985	25	1374	91	1411	72	H	0
13	1985	25	1412	70	1397	65	N	0
14	1985	25	1417	87	1225	58	H	0
15	1985	26	1116	65	1368	62	H	0
16	1985	26	1120	92	1391	50	H	0
17	1985	26	1135	65	1306	60	A	0
18	1985	26	1143	58	1388	53	H	0
19	1985	26	1153	50	1184	48	H	0
20	1985	26	1165	47	1159	40	A	0
21	1985	26	1171	55	1216	52	H	0
22	1985	26	1173	76	1134	56	H	0
23	1985	26	1177	59	1296	58	H	0
24	1985	26	1193	79	1265	76	A	0
25	1985	26	1196	106	1416	55	H	0

	Season	Daynum	Wteam	Wscore	Lteam	Lscore	Wloc	Numot
26	1985	26	1206	95	1137	77	H	0
27	1985	26	1210	79	1149	66	H	0
28	1985	26	1211	64	1102	59	A	0
29	1985	26	1234	76	1114	47	H	0
...	...	...	...	...	...	...	...	...
145259	2016	130	1452	69	1328	67	N	0
145260	2016	131	1114	72	1418	65	N	0
145261	2016	131	1138	64	1103	61	N	0
145262	2016	131	1163	77	1396	62	N	0
145263	2016	131	1167	57	1308	54	N	0
145264	2016	131	1201	68	1361	63	N	0
145265	2016	131	1214	81	1354	69	N	0
145266	2016	131	1218	64	1253	60	N	0
145267	2016	131	1242	81	1452	71	N	0
145268	2016	131	1246	93	1208	80	N	0
145269	2016	131	1272	74	1408	54	N	0
145270	2016	131	1277	64	1268	61	N	0
145271	2016	131	1292	55	1330	53	N	0
145272	2016	131	1314	61	1438	57	N	0
145273	2016	131	1332	88	1428	57	N	0
145274	2016	131	1345	76	1276	59	N	0
145275	2016	131	1371	69	1437	67	N	0
145276	2016	131	1372	82	1394	60	N	0
145277	2016	131	1380	54	1238	53	N	0
145278	2016	131	1386	82	1173	79	N	0
145279	2016	131	1392	80	1436	74	H	0
145280	2016	131	1401	71	1261	38	N	0

	Season	Daynum	Wteam	Wscore	Lteam	Lscore	Wloc	Numot
145281	2016	131	1419	82	1426	71	N	0
145282	2016	131	1433	76	1172	54	N	0
145283	2016	131	1451	62	1285	59	N	0
145284	2016	132	1114	70	1419	50	N	0
145285	2016	132	1163	72	1272	58	N	0
145286	2016	132	1246	82	1401	77	N	1
145287	2016	132	1277	66	1345	62	N	0
145288	2016	132	1386	87	1433	74	N	0

145289 rows × 8 columns

In [6]:
```
import pandas as pd
filePath = 'data files/RegularSeasonCompactResults.csv'
def readCsvData(filePath):
    return pd.read_csv(filePath)
df=readCsvData(filePath)
```

In [7]:
```
df.shape
```

Out[7]: (145289, 8)

In [8]:
```
# To know only first few rows --- head()
df.head()
```

Out[8]:

	Season	Daynum	Wteam	Wscore	Lteam	Lscore	Wloc	Numot
0	1985	20	1228	81	1328	64	N	0
1	1985	25	1106	77	1354	70	H	0
2	1985	25	1112	63	1223	56	H	0
3	1985	25	1165	70	1432	54	H	0
4	1985	25	1192	86	1447	74	H	0

In [9]:
```
# convert all column names into list
df.columns.tolist()
```

Out[9]:
```
['Season', 'Daynum', 'Wteam', 'Wscore', 'Lteam', 'Lscore', 'Wloc', 'Numot']
```

In [10]:
```
df.describe()
```

Out[10]:

	Season	Daynum	Wteam	Wscore	Lteam	Lscore	Numot
count	145289.000000	145289.000000	145289.000000	145289.000000	145289.000000	145289.000000	145289.000000
mean	2001.574834	75.223816	1286.720646	76.600321	1282.864064	64.497009	0.044387
std	9.233342	33.287418	104.570275	12.173033	104.829234	11.380625	0.247819
min	1985.000000	0.000000	1101.000000	34.000000	1101.000000	20.000000	0.000000
25%	1994.000000	47.000000	1198.000000	68.000000	1191.000000	57.000000	0.000000
50%	2002.000000	78.000000	1284.000000	76.000000	1280.000000	64.000000	0.000000
75%	2010.000000	103.000000	1379.000000	84.000000	1375.000000	72.000000	0.000000
max	2016.000000	132.000000	1464.000000	186.000000	1464.000000	150.000000	6.000000

In [11]:
```
df.max()
```

Out[11]:
```
Season    2016
Daynum     132
Wteam     1464
Wscore     186
Lteam     1464
Lscore     150
Wloc         N
Numot        6
dtype: object
```

In [12]:
```
df['Lteam'].max() #One column value max value
```

Out[12]:
```
1464
```

In [13]:
```
df['Lteam'].min() #One column value min value
```

Out[13]:
```
1101
```

In [14]:
```
df['Lteam'].sum()
```

Out[14]:
```
186386037
```

In [15]:
```
df['Season'].value_counts()
```

Out[15]:
```
2016    5369
2014    5362
2015    5354
2013    5320
2010    5263
2012    5253
2009    5249
2011    5246
2008    5163
2007    5043
2006    4757
2005    4675
2003    4616
2004    4571
2002    4555
2000    4519
2001    4467
1999    4222
1998    4167
1997    4155
1992    4127
1991    4123
1996    4122
1995    4077
1994    4060
1990    4045
1989    4037
1993    3982
1988    3955
1987    3915
1986    3783
1985    3737
Name: Season, dtype: int64
```

In [16]:
```
df.loc[:3]
```

Out[16]:

	Season	Daynum	Wteam	Wscore	Lteam	Lscore	Wloc	Numot
0	1985	20	1228	81	1328	64	N	0
1	1985	25	1106	77	1354	70	H	0
2	1985	25	1112	63	1223	56	H	0
3	1985	25	1165	70	1432	54	H	0

In [17]:
```
df.iloc[:3]
```

Out[17]:

	Season	Daynum	Wteam	Wscore	Lteam	Lscore	Wloc	Numot
0	1985	20	1228	81	1328	64	N	0
1	1985	25	1106	77	1354	70	H	0
2	1985	25	1112	63	1223	56	H	0

In [18]:
```
df.sort_values('Lscore').head()
```

Out[18]:

	Season	Daynum	Wteam	Wscore	Lteam	Lscore	Wloc	Numot
100027	2008	66	1203	49	1387	20	H	0
49310	1997	66	1157	61	1204	21	H	0
89021	2006	44	1284	41	1343	21	A	0
85042	2005	66	1131	73	1216	22	H	0
103660	2009	26	1326	59	1359	22	H	0

In [19]:
```
df.sort_values('Lscore').tail()
```

Out[19]:

	Season	Daynum	Wteam	Wscore	Lteam	Lscore	Wloc	Numot
77873	2003	110	1383	142	1254	140	H	2
24970	1991	68	1258	186	1109	140	H	0
22074	1990	96	1261	148	1258	141	H	0
16853	1989	68	1258	162	1109	144	A	0
17867	1989	92	1258	181	1109	150	H	0

In [ ]:
```

```