In [1]:
```
# Manuel Duran
# Project 3
# Tennis Betting Analysis
```

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
matplotlib.style.use('seaborn-dark-palette')
import os
import seaborn as sns
from subprocess import check_output
```

In [4]:
```
Tennis_Bet_DF = pd.read_csv('FinalTennisDataSet.csv', encoding = 'latin1')
Tennis_Bet_DF.WRank = pd.to_numeric(Tennis_Bet_DF.WRank, errors = 'coerce')
Tennis_Bet_DF.LRank = pd.to_numeric(Tennis_Bet_DF.LRank, errors = 'coerce')
```

In [22]:
```
Tennis_Bet_DF.head()
```

Out[22]:

	ATP	Location	Tournament	Date	Series	Court	Surface	Round	Best of	Wi
0	1	Adelaide	Australian Hardcourt Championships	3/01/2000	International	Outdoor	Hard	1st Round	3	Dos
1	1	Adelaide	Australian Hardcourt Championships	3/01/2000	International	Outdoor	Hard	1st Round	3	Enqv
2	1	Adelaide	Australian Hardcourt Championships	3/01/2000	International	Outdoor	Hard	1st Round	3	Es
3	1	Adelaide	Australian Hardcourt Championships	3/01/2000	International	Outdoor	Hard	1st Round	3	Fe
4	1	Adelaide	Australian Hardcourt Championships	3/01/2000	International	Outdoor	Hard	1st Round	3	From

5 rows × 59 columns

In [24]:
```
Tennis_Bet_DF.describe()
```

Out[24]:

	ATP	Best of	WRank	LRank	W1	L1	4
count	46652.000000	46652.000000	46636.000000	46554.000000	46423.000000	46423.000000	
mean	32.931000	3.373746	59.056180	94.083086	5.792667	4.043211	
std	17.953268	0.780315	73.300861	143.836733	1.239261	1.847833	
min	1.000000	-1.000000	1.000000	1.000000	0.000000	0.000000	
25%	19.000000	3.000000	17.000000	35.000000	6.000000	3.000000	
50%	33.000000	3.000000	41.000000	66.000000	6.000000	4.000000	
75%	49.000000	3.000000	77.000000	105.000000	6.000000	6.000000	
max	69.000000	5.000000	1890.000000	7380.000000	7.000000	7.000000	

8 rows × 49 columns

In [5]:
```
Tennis_Bet_DF['Diff'] =  Tennis_Bet_DF.LRank - Tennis_Bet_DF.WRank

Tennis_Bet_DF['Round_10'] = 10*round(np.true_divide(Tennis_Bet_DF.Diff,10))
Tennis_Bet_DF['Round_20'] = 20*round(np.true_divide(Tennis_Bet_DF.Diff,20))

Tennis_Bet_DF['Total Sets'] = Tennis_Bet_DF.Wsets + Tennis_Bet_DF.Lsets

Tennis_Bet_DF.W3 = Tennis_Bet_DF.W3.fillna(0)
Tennis_Bet_DF.W4 = Tennis_Bet_DF.W4.fillna(0)
Tennis_Bet_DF.W5 = Tennis_Bet_DF.W5.fillna(0)
Tennis_Bet_DF.L3 = Tennis_Bet_DF.L3.fillna(0)
Tennis_Bet_DF.L4 = Tennis_Bet_DF.L4.fillna(0)
Tennis_Bet_DF.L5 = Tennis_Bet_DF.L5.fillna(0)

Tennis_Bet_DF['Sets Diff'] = Tennis_Bet_DF.W1+Tennis_Bet_DF.W2+Tennis_Bet_DF.
Final_DF = Tennis_Bet_DF
```

In [25]:
```
Final_DF.head()
```

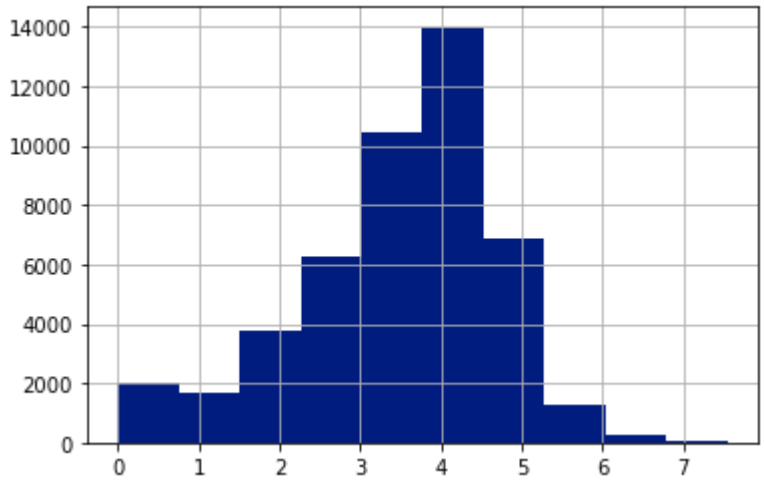
Out[25]:

	ATP	Location	Tournament	Date	Series	Court	Surface	Round	Best of	Wi
0	1	Adelaide	Australian Hardcourt Championships	3/01/2000	International	Outdoor	Hard	1st Round	3	Dos
1	1	Adelaide	Australian Hardcourt Championships	3/01/2000	International	Outdoor	Hard	1st Round	3	Enqv
2	1	Adelaide	Australian Hardcourt Championships	3/01/2000	International	Outdoor	Hard	1st Round	3	Es
3	1	Adelaide	Australian Hardcourt Championships	3/01/2000	International	Outdoor	Hard	1st Round	3	Fe
4	1	Adelaide	Australian Hardcourt Championships	3/01/2000	International	Outdoor	Hard	1st Round	3	From

5 rows × 59 columns

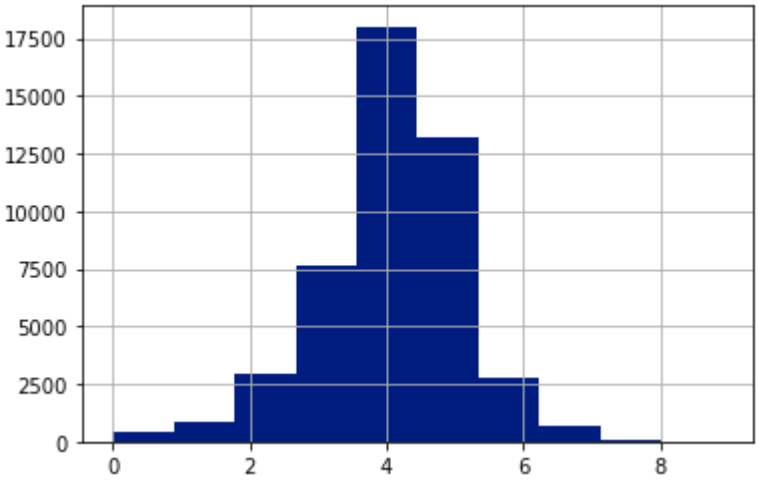
In [28]:
```
np.log(Final_DF['WRank']).hist()
```

Out[28]: <AxesSubplot:>



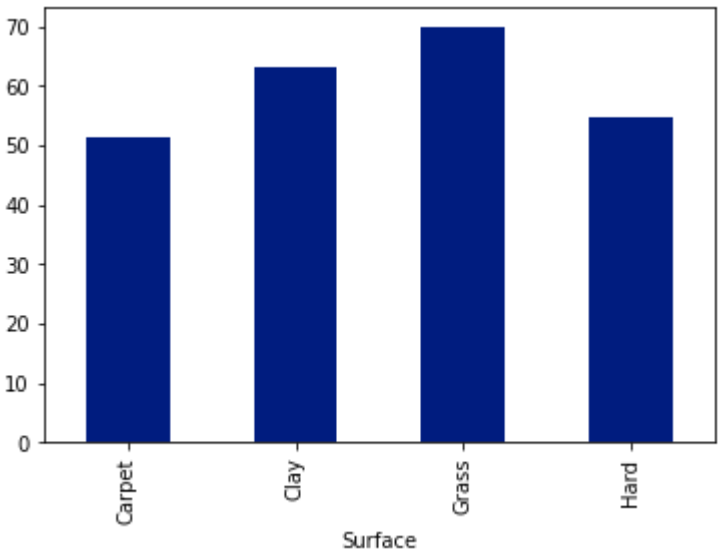
In [29]:
```
np.log(Final_DF['LRank']).hist()
```

Out[29]: <AxesSubplot:>



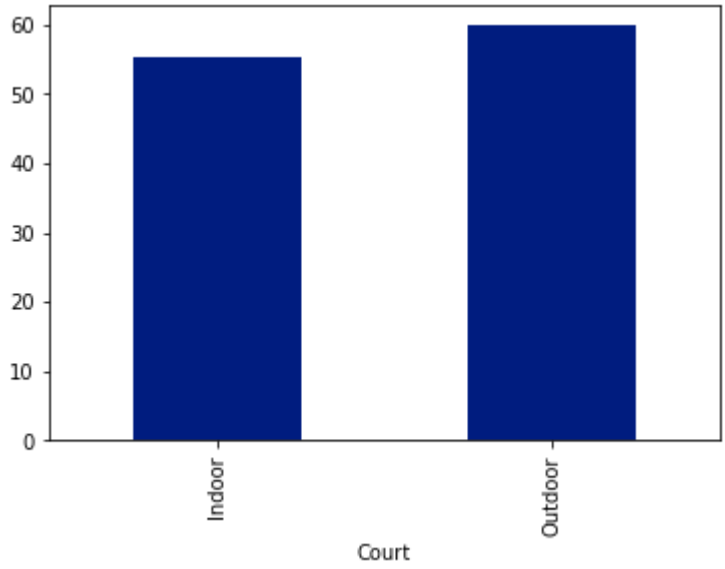
In [33]:
```
Final_DF.groupby("Surface").mean()["WRank"].T.plot(kind="bar")
```

Out[33]: <AxesSubplot:xlabel='Surface'>



In [34]:
```
Final_DF.groupby("Court").mean()["WRank"].T.plot(kind="bar")
```

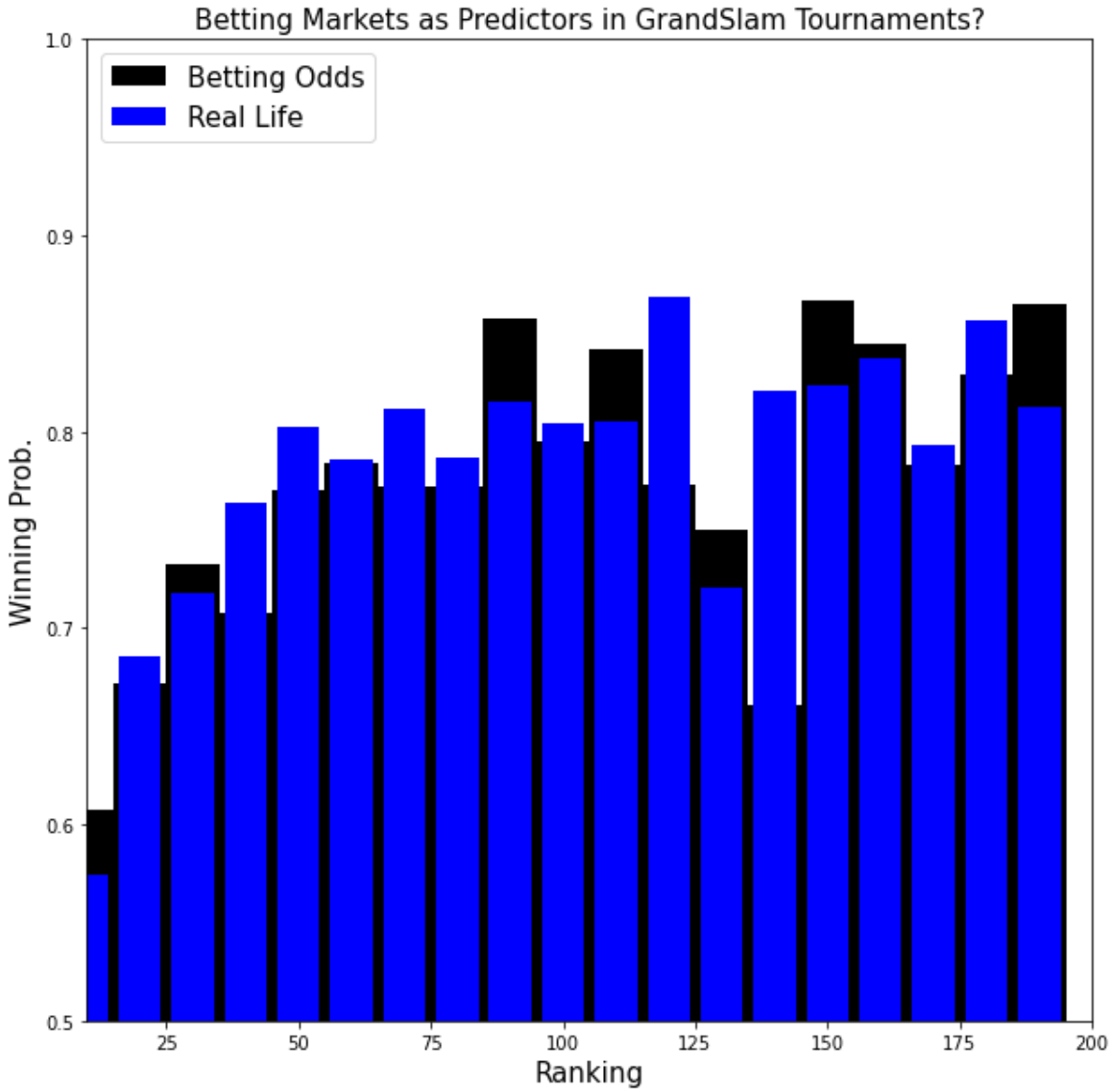
Out[34]: <AxesSubplot:xlabel='Court'>



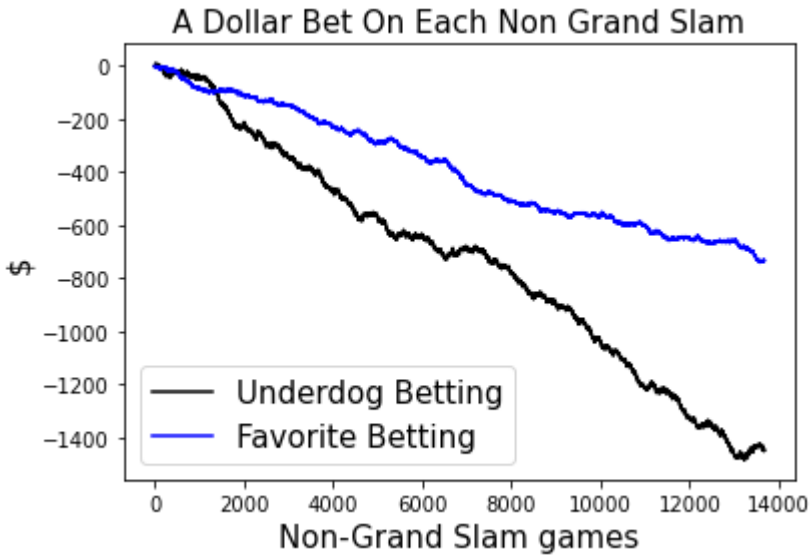
In [6]:
```
Non_GrandSlam_DF = Final_DF[~(Final_DF.Series == 'Grand Slam')]
GrandSlam_DF = Final_DF[Final_DF.Series == 'Grand Slam']
```

In [14]:

```
bins = np.arange(10,200,10)
Gs_prob = []

for bi in bins:

    pos = bi
    neg = -pos

    pos_wins = len(GrandSlam_DF[GrandSlam_DF.Round_10 == pos])
    neg_wins = len(GrandSlam_DF[GrandSlam_DF.Round_10 == neg])
    Gs_prob.append(np.true_divide(pos_wins,pos_wins + neg_wins))


prob = []

for bi in bins:

    W = np.true_divide(1,np.mean(GrandSlam_DF.AvgW[GrandSlam_DF.Round_10 == b
    L = np.true_divide(1,np.mean(GrandSlam_DF.AvgL[GrandSlam_DF.Round_10 == b
    ratio = np.true_divide(1,L + W)
    part_ratio = (ratio - 1)/2 + 1
    prob.append(W/part_ratio)

plt.figure(figsize = (10,10))
plt.bar(bins,prob, width = 10, color = 'black')
plt.bar(bins,Gs_prob, width = 8, color = 'blue')
plt.xlabel('Ranking', fontsize = 15)
plt.ylabel('Winning Prob.', fontsize = 15)
plt.xlim([10,200])
plt.ylim([0.5,1])
plt.title('Betting Markets as Predictors in GrandSlam Tournaments?', fontsize
plt.legend(['Betting Odds','Real Life'], loc = 2, fontsize = 15)
plt.show()
```


Betting Markets as Predictors in GrandSlam Tournaments?
Betting Odds
Real Life
Winning Prob.
Ranking
1.0
0.9
0.8
0.7
0.6
0.5
25
50
75
100
125
150
175
200

In [20]:
```
Non_GrandSlam_DF = Non_GrandSlam_DF[~np.isnan(Non_GrandSlam_DF.AvgW)]
mo_over = 0
mo_under = 0
tracking_over = []
tracking_under = []

for row in Non_GrandSlam_DF.iterrows():

    if row[1].Diff>0:
        mo_over = mo_over + row[1].AvgW - 1
        mo_under = mo_under - 1

    else:
        mo_over = mo_over - 1
        mo_under = mo_under + row[1].AvgW - 1

    tracking_over.append(mo_over)
    tracking_under.append(mo_under)

    if np.isnan(mo_over):
        break

    if np.isnan(mo_under):
        break


plt.figure()
plt.plot(tracking_under,'black')
plt.plot(tracking_over,'blue')
plt.xlabel('Non-Grand Slam games', fontsize = 15)
plt.ylabel('$', fontsize = 15)
plt.title('A Dollar Bet On Each Non Grand Slam', fontsize = 15)
plt.legend(['Underdog Betting', 'Favorite Betting'], loc = 3, fontsize = 15)
plt.show()
```



In [21]:

```
GrandSlam_DF = GrandSlam_DF[~np.isnan(GrandSlam_DF.AvgW)]
mo_over = 0
mo_under = 0
tracking_over = []
tracking_under = []

for row in GrandSlam_DF.iterrows():

    if row[1].Diff>0:
        mo_over = mo_over + row[1].AvgW - 1
        mo_under = mo_under - 1

    else:
        mo_over = mo_over - 1
        mo_under = mo_under + row[1].AvgW - 1

    tracking_over.append(mo_over)
    tracking_under.append(mo_under)

    if np.isnan(mo_over):
        break

    if np.isnan(mo_under):
        break


plt.figure()
plt.plot(tracking_under,'black')
plt.plot(tracking_over,'blue')
plt.xlabel('Grand Slam games',fontsize = 15)
plt.ylabel('Money Balance [$]', fontsize = 15)
plt.title('A Dollar Bet On Each Grand Slame')
plt.legend(['Underdog Betting', 'Favorite Betting'], loc = 3, fontsize = 15)
plt.show()
```