```
#cfb
configuration = cfbd.Configuration()
configuration.api_key['Authorization'] = '5Hs/5kCfcbyDhcVBb9
configuration.api_key_prefix['Authorization'] = 'Bearer'
playersAPI = cfbd.PlayersApi(cfbd.ApiClient(configuration))
teamsAPI = cfbd.TeamsApi(cfbd.ApiClient(configuration))
#data
winners = ["Trent Dilfer", "Tom Brady", "Brad Johnson", "Tom
losers = ["Kerry Collins", "Kurt Warner", "Rich Gannon", "Ja
years = np.arange(2001, 2023)
#plt
plt.style.use('dark_background')
```

Establishing CFBD data source, as well as Superbowl winning and losing quarterbacks.

```
#College Data
coll = []

url = requests.get('https://www.pro-football-reference.com/years/2005/draft.htm')
parse = BeautifulSoup(url.text, "lxml").prettify()
start = parse.find("<table")
end = parse.find("</table")
college2005 = pd.read_html(parse[start:end+8])[0]
college2005.insert(0, 'season', 2005)
coll.append(college2005)
```

Collect draft data from Pro Football Reference (PFR). This process is repeated for every year 2005 – 2022, then all collected data is combined into one dataframe.

```
df = college
df.loc[df['college'] == 'Ala-Birmingham', 'college'] = 'UAB'
df.loc[df['college'] == 'Alabama St.', 'college'] = 'Alabama State'
df.loc[df['college'] == 'Appalachian St.', 'college'] = 'Appalachian State'
df.loc[df['college'] == 'Arizona St.', 'college'] = 'Arizona State'
df.loc[df['college'] == 'Arkansas St.', 'college'] = 'Arkansas State'
df.loc[df['college'] == 'Ball St.', 'college'] = 'Ball State'
df.loc[df['college'] == 'Boise St.', 'college'] = 'Boise State'
df.loc[df['college'] == 'Boston Col.', 'college'] = 'Boston College'
df.loc[df['college'] == 'Central Florida', 'college'] = 'UCF'
```

College names are inconsistent between CFBD and PFR. 66 school names needed to be corrected.

```
Receptions
receptions = []
receptionsHolder = playersAPI.get_player_season_stats(year = 2004, category = 'receiving')
rec = pd.DataFrame([Player.to_dict() for Player in receptionsHolder])
rec['season'] = 2005
receptionsHolder = rec.to_dict('records')
receptions.append(receptionsHolder)
```

College reception data is collected using CFBD. The same process id used to collect the data in the other necessary years.

```
recs = receptions[receptions['stat_type'] == 'REC']
receptionsSums = recs.groupby(['team', 'season']).sum().reset_index()
receptionsSums.columns = ['team', 'season', 'rnk', 'player_id', 'team_total']
receptionsSums.drop(['rnk', 'player_id'], axis = 1, inplace = True)
receptionsSums = receptionsSums.merge(recs.reset_index(drop=True), how = 'left', on = ['season', 'team']).sort_values(by=['team', 'season', 'stat'])
receptionsSums.drop(['rnk', 'player_id', 'category', 'stat_type'], axis = 1, inplace = True)
receptionsSums.drop_duplicates(subset = ['team', 'season', 'team_total'], keep = 'last', inplace = True)
receptionsSums.columns = ['team', 'season', 'team_total', 'name', 'conference', 'max']
```

Data is trimmed to show only what is necessary. Data is then gouped by team and season, and sorted by receptions. More columns and rows are dropped such that each teams' season leader in receptions are the only records remaining.

```
maxper = df.merge(receptionsSums.reset_index(drop=True), how = 'left', left_on = ['season', 'college'], right_on = ['season', 'team'])
maxper['percent'] = round(((maxper['max'] / maxper['team_total']) * 100), 5)
maxper['percent'] = maxper['percent'].fillna(0)
maxper = maxper[maxper['percent'] != 0]

a1 = maxper[maxper['wcav'] < 20].reset_index()
a2 = maxper[(maxper['wcav'] >= 20) & (maxper['wcav'] < 40)].reset_index()
a3 = maxper[(maxper['wcav'] >= 40) & (maxper['wcav'] < 60)].reset_index()
a4 = maxper[(maxper['wcav'] >= 60) & (maxper['wcav'] < 80)].reset_index()
a5 = maxper[(maxper['wcav'] >= 80) & (maxper['wcav'] < 100)].reset_index()
a6 = maxper[(maxper['wcav'] >= 100) & (maxper['wcav'] < 120)].reset_index()
a7 = maxper[maxper['wcav'] >= 120].reset_index()
ranges = np.array([a1['percent'], a2['percent'], a3['percent'], a4['percent'], a5['percent'], a6['percent'], a7['percent']])
```

Maxper reception shares are finally made into percentages. Quarterbacks who have a 0 WCAV (never played in the NFL) are removed. The remaining players are grouped by WCAV as a measurement of the career success.

```
colors =['orchid', 'deeppink', 'cyan', 'deepskyblue', 'coral', 'yellow', 'lime']
vioparts = plt.violinplot(ranges)
boxparts = plt.boxplot(ranges, showfliers = False)
for i in range(7):
    vioparts['bodies'][i].set_color(colors[i])
    vioparts['cbars'].set_visible(False)
    vioparts['cmins'].set_color('yellow')
    vioparts['cmins'].set_linewidth(2)
    vioparts['cmaxes'].set_color('yellow')
    vioparts['cmaxes'].set_linewidth(2)

plt.suptitle("Max Reciever Reception Share vs Weighted Career Approximate Value (PFR)")
plt.ylabel("Max Reciever Reception Share")
plt.xlabel("Weighted Career Approximate Value (Intervals of 20)")
plt.show()
```

The reception share data is plotted. The same process is repeated for yardage shares.

```
mn = list(round(item.get_ydata()[0], 1) for item in boxparts['caps'][::2])
mx = list(round(item.get_ydata()[0], 1) for item in boxparts['caps'][1::2])
q1 = list(round(min(item.get_ydata()), 1) for item in boxparts['boxes'])
q3 = list(round(max(item.get_ydata()), 1) for item in boxparts['boxes'])
print("Minimum values:\t\t", mn)
print("Maximum values:\t\t", mx)
print("Quartile 1 values:\t", q1)
print("Quartile 3 values:\t", q3)
iqr = []
for i in range(len(q1)):
    iqr.append(round((q3[i] - q1[i]), 1))
print("Interquartile ranges:\t", iqr)
medians = list(round(item.get_ydata()[0], 1) for item in boxparts['medians'])
print("Median values: ", medians)
```

This shows some basic statistical datapoints for the data.

```
qbs = df[df['pos'] == 'QB']
qbs['wcav'] = qbs['wcav'].fillna(0)
qbs['wcav'] = qbs['wcav'].astype(float)
wrs = df[df['pos'] == 'WR']
use = pd.DataFrame(columns = df.columns)
for i, r in qbs.iterrows():
  d = df[((df['season'] == r['season']) | (df['season'] == (r['season'] - 1)) | (df['season'] == (r['season'] + 1)))]
  d = d[(d['college'] == r['college']) & (d['pos'] == 'WR')]
  d['qb_val'] = r['wcav']
  d['qb_name'] = r['name']
  d['qb_round'] = r['round']
  use = pd.concat([use, d])

use['qb_val'] = use['qb_val'].astype(int)
use['pick'] = use['pick'].astype(int)
use = use.reset_index()
use.sort_values(['pick'], inplace = True)
use = use.reset_index()

best = use[(use['qb_val'] > 40)]
best.drop(['level_0'], axis = 1, inplace = True)
best = best.reset_index()

x1 = best[best['round'] == 1]['qb_val']
x2 = best[best['round'] == 2]['qb_val']
x3 = best[best['round'] == 3]['qb_val']
x4 = best[best['round'] == 4]['qb_val']
x5 = best[best['round'] == 5]['qb_val']
x6 = best[best['round'] == 6]['qb_val']
x7 = best[best['round'] == 7]['qb_val']
kwargs = dict(alpha=0.6, bins=6)
plt.hist(x1, **kwargs, color='lime', label='round 1')
plt.hist(x2, **kwargs, color='yellow', label='round 2')
plt.hist(x3, **kwargs, color='orangered', label='round 3')
plt.hist(x4, **kwargs, color='dodgerblue', label='round 4')
plt.hist(x5, **kwargs, color='cyan', label='round 5')
plt.hist(x6, **kwargs, color='deeppink', label='round 6')
plt.hist(x7, **kwargs, color='darkorchid', label='round 7')
plt.xlabel("QB Value in NFL")
plt.ylabel("Frequency")
plt.legend(title = "WR selected in:")
good = qbs[qbs['wcav'] > 40]
unused = good.shape[0] - len(set(best['qb_name']))
used = good.shape[0]
plt.suptitle("Value of QB vs College WR Round Drafted")
title = "{}/{} qualified QBs with no college WR drafted".format(unused, used)
plt.title(title, fontsize = 10)
```

The code for the histogram, showcasing WCAV of the QB vs the round their WR was selected.