In [1]:
```
import yt_dlp
import json
import os
from datetime import datetime, timedelta
from urllib.error import HTTPError
```

This file will download all videos and live streams in the past 24 hours (skipping over any previously downloaded by using the json files within each folder to track the ID's of previously downloaded videos)

In [2]:
```
# All channel names (weird name for buccaneers because of they way it is set up)
channel_names = ["49ers", "AtlantaFalcons", "azcardinals", "BaltimoreRavens", "Bengals"
                "buffalobills", "CarolinaPanthers", "chargers", "ChicagoBears", "colts"
                "detroitlionsnfl", "eagles", "HoustonTexans", "jaguars", "KansasCityChi
                "NewOrleansSaints","NewYorkGiants", "nyjets", "packers", "patriots", "r
                "Titans", "vikings"]

# Path to the directory containing ffmpeg and ffprobe executables (if not in PATH)
ffmpeg_location = 'c:/users/12505/anaconda3/lib/site-packages/ffmpeg/bin'
#Limiting amount of videos to fetch. No need to fetch all the videos
max_videos_to_fetch = 10
```

In [3]:
```
# load the already downloaded videos that have their ids stored in the json file
def load_downloaded_videos(record_file):
    if os.path.exists(record_file):
        with open(record_file, 'r') as f:
            return json.load(f)
    return []
```

In [4]:
```
# will save the ids of any newly downloaded videos within the json file
def save_downloaded_videos(record_file, downloaded_videos):
    with open(record_file, 'w') as f:
        json.dump(downloaded_videos, f)
```

In [5]:
```
# convert the video to audio format using some options ydl offers
def download_audio(url, output_path):
    ydl_opts = {
        'outtmpl': output_path,
        'format': 'bestaudio/best',
        'postprocessors': [{
            'key': 'FFmpegExtractAudio',
            'preferredcodec': 'mp3',
            'preferredquality': '192',
        }],
        'ffmpeg_location': ffmpeg_location,  # Specify the path to ffmpeg and ffprobe i
    }
    try:
        with yt_dlp.YoutubeDL(ydl_opts) as ydl:
            ydl.download([url])
    except Exception as e:
        print(f"Error downloading {url}: {e}")
```

In [6]:
```
# will get the latest videos uploaded to a channel and only select those within a 24 hou
# if the video has already been downloaded the function will skip it.
def get_latest_videos(channel_url, downloaded_videos):
    ydl_opts = {
        'quiet': True,
        'extract_flat': 'in_playlist',
        'playlistend': max_videos_to_fetch,
        'skip_download': True,
    }
    with yt_dlp.YoutubeDL(ydl_opts) as ydl:
        result = ydl.extract_info(channel_url, download=False)
    one_day_ago = datetime.now() - timedelta(days=1)
    print(one_day_ago)
    new_videos = []
    print(f"Checking {len(result['entries'])} videos from the channel...{channel_url}")

    for entry in result['entries']:
        video_id = entry['id']
        video_url = f"https://www.youtube.com/watch?v={video_id}"
        try:
            # Get detailed info for each video to check upload date
            with yt_dlp.YoutubeDL({'quiet': True}) as ydl:
                video_info = ydl.extract_info(video_url, download=False)
            upload_date = datetime.strptime(video_info['upload_date'], '%Y%m%d')

            if upload_date > one_day_ago and video_id not in downloaded_videos:
                print(f"Adding video {video_url}, uploaded on {upload_date}")
                new_videos.append(video_url)
            else:
                print(f"Skipping video {video_url}, uploaded on {upload_date}")
                break
        except Exception as e:
            print(f"Error processing video {video_url}: {e}")

    return new_videos
```

In [7]:
```
# runs and downloads videos appropriately
def main():
    for channel_name in channel_names:
        output_dir = f'videos/{channel_name}/%(title)s.%(ext)s'

        # Path to save the record of downloaded video IDs
        record_file = f'videos/{channel_name}/downloaded_videos.json'

        # Directory to save downloaded audio files
        output_dir = f'videos/{channel_name}/%(title)s.%(ext)s'

        # Channel URL
        if(channel_name == "channel/UC0Wwu7r1ybaaR09ANhudTzA" or channel_name == "detro
            print("here")
            channel_url = f'https://www.youtube.com/{channel_name}/videos'
        else:
            print("yo")
            channel_url = f'https://www.youtube.com/c/{channel_name}/videos'

        # To Get Live Streams
        if(channel_name == "channel/UC0Wwu7r1ybaaR09ANhudTzA" or channel_name == "detro
            print("here1")
            live_url = f'https://www.youtube.com/{channel_name}/streams'
        else:
            print("yo1")
            live_url = f'https://www.youtube.com/c/{channel_name}/streams'
        downloaded_videos = load_downloaded_videos(record_file)
        latest_videos = get_latest_videos(channel_url, downloaded_videos)
        latest_live = get_latest_videos(live_url, downloaded_videos)

        try:
            for video_url in latest_videos: #get latest videos
                print(f"Downloading audio for {video_url}")
                download_audio(video_url, output_dir)
                video_id = video_url.split('=')[1]
                downloaded_videos.append(video_id)

            for video_url in latest_live: #get latest live streams
                print(f"Downloading audio for {video_url}")
                download_audio(video_url, output_dir)
                video_id = video_url.split('=')[1]
                downloaded_videos.append(video_id)

            save_downloaded_videos(record_file, downloaded_videos)
        except Exception as e:
            print(f"Error processing video {video_url}: {e}")
            continue


if __name__ == "__main__":
    main()
    print("Done!")
```

```
yo
yo1
2024-05-28 15:09:03.067596
Checking 10 videos from the channel...https://www.youtube.com/c/49ers/videos
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-7-3cd6064beefb> in <module>
     50
     51 if __name__ == "__main__":
---> 52     main()
     53     print("Done!")

<ipython-input-7-3cd6064beefb> in main()
     27
     28         downloaded_videos = load_downloaded_videos(record_file)
---> 29         latest_videos = get_latest_videos(channel_url, downloaded_videos)
     30         latest_live = get_latest_videos(live_url, downloaded_videos)
     31

<ipython-input-6-6427473ac176> in get_latest_videos(channel_url, downloaded_videos)
     21             # Get detailed info for each video to check upload date
     22             with yt_dlp.YoutubeDL({'quiet': True}) as ydl:
---> 23                 video_info = ydl.extract_info(video_url, download=False)
     24             upload_date = datetime.strptime(video_info['upload_date'], '%Y%m%d')
     25

~\anaconda3\lib\site-packages\yt_dlp\YoutubeDL.py in extract_info(self, url, download, i
e_key, extra_info, process, force_generic_extractor)
   1593                     raise ExistingVideoReached()
   1594                 break
-> 1595             return self.__extract_info(url, self.get_info_extractor(key), downlo
ad, extra_info, process)
   1596         else:
   1597             extractors_restricted = self.params.get('allowed_extractors') not in
(None, ['default'])
```

```
~\anaconda3\lib\site-packages\yt_dlp\YoutubeDL.py in wrapper(self, *args, **kwargs)
   1604             while True:
   1605                 try:
-> 1606                     return func(self, *args, **kwargs)
   1607                 except (DownloadCancelled, LazyList.IndexError, PagedList.IndexE
rror):
   1608                     raise

~\anaconda3\lib\site-packages\yt_dlp\YoutubeDL.py in __extract_info(self, url, ie, downl
oad, extra_info, process)
   1739 
   1740         try:
-> 1741             ie_result = ie.extract(url)
   1742         except UserNotLive as e:
   1743             if process:

~\anaconda3\lib\site-packages\yt_dlp\extractor\common.py in extract(self, url)
    732                     self.to_screen('Extracting URL: %s' % (
    733                         url if self.get_param('verbose') else truncate_string(ur
l, 100, 20)))
--> 734                     ie_result = self._real_extract(url)
    735                     if ie_result is None:
    736                         return None

~\anaconda3\lib\site-packages\yt_dlp\extractor\youtube.py in _real_extract(self, url)
   4159 
   4160         live_broadcast_details, live_status, streaming_data, formats, automatic_
captions = \
-> 4161             self._list_formats(video_id, microformats, video_details, player_res
ponses, player_url, duration)
   4162         if live_status == 'post_live':
   4163             self.write_debug(f'{video_id}: Video is in Post-Live Manifestless mo
de')

~\anaconda3\lib\site-packages\yt_dlp\extractor\youtube.py in _list_formats(self, video_i
d, microformats, video_details, player_responses, player_url, duration)
   4062                         else None)
   4063         streaming_data = traverse_obj(player_responses, (..., 'streamingData'))
-> 4064         *formats, subtitles = self._extract_formats_and_subtitles(streaming_dat
a, video_id, player_url, live_status, duration)
   4065         if all(f.get('has_drm') for f in formats):
   4066             # If there are no formats that definitely don't have DRM, all have D
RM

~\anaconda3\lib\site-packages\yt_dlp\extractor\youtube.py in _extract_formats_and_subtit
les(self, streaming_data, video_id, player_url, live_status, duration)
   3819                     decrypt_nsig = self._cached(self._decrypt_nsig, 'nsig', quer
y['n'][0])
   3820                     fmt_url = update_url_query(fmt_url, {
-> 3821                         'n': decrypt_nsig(query['n'][0], video_id, player_url)
   3822                     })
   3823                 except ExtractorError as e:

~\anaconda3\lib\site-packages\yt_dlp\extractor\youtube.py in inner(*args, **kwargs)
   3066             if cache_id not in self._player_cache:
   3067                 try:
-> 3068                     self._player_cache[cache_id] = func(*args, **kwargs)
   3069                 except ExtractorError as e:
   3070                     self._player_cache[cache_id] = e

~\anaconda3\lib\site-packages\yt_dlp\extractor\youtube.py in _decrypt_nsig(self, s, vide
o_id, player_url)
   3101         try:
   3102             extract_nsig = self._cached(self._extract_n_function_from_code, 'nsi
g func', player_url)
```

```
-> 3103             ret = extract_nsig(jsi, func_code)(s)
   3104         except JSInterpreter.Exception as e:
   3105             try:

~\anaconda3\lib\site-packages\yt_dlp\extractor\youtube.py in extract_nsig(s)
   3162         def extract_nsig(s):
   3163             try:
-> 3164                 ret = func([s])
   3165             except JSInterpreter.Exception:
   3166                 raise

~\anaconda3\lib\site-packages\yt_dlp\jsinterp.py in resf(args, kwargs, allow_recursion)
    848             global_stack[0].update(kwargs)
    849             var_stack = LocalNameSpace(*global_stack)
--> 850             ret, should_abort = self.interpret_statement(code.replace('\n', '
'), var_stack, allow_recursion - 1)
    851             if should_abort:
    852                 return ret

~\anaconda3\lib\site-packages\yt_dlp\jsinterp.py in interpret_statement(self, stmt, loca
l_vars, allow_recursion, *args, **kwargs)
    183                 cls.write(stmt, level=allow_recursion)
    184             try:
--> 185                 ret, should_ret = f(self, stmt, local_vars, allow_recursion, *ar
gs, **kwargs)
    186             except Exception as e:
    187                 if cls.ENABLED:

~\anaconda3\lib\site-packages\yt_dlp\jsinterp.py in interpret_statement(self, stmt, loca
l_vars, allow_recursion)
    438             err = None
    439             try:
--> 440                 ret, should_abort = self.interpret_statement(try_expr, local_var
s, allow_recursion)
    441                 if should_abort:
    442                     return ret, True

~\anaconda3\lib\site-packages\yt_dlp\jsinterp.py in interpret_statement(self, stmt, loca
l_vars, allow_recursion, *args, **kwargs)
    183                 cls.write(stmt, level=allow_recursion)
    184             try:
--> 185                 ret, should_ret = f(self, stmt, local_vars, allow_recursion, *ar
gs, **kwargs)
    186             except Exception as e:
    187                 if cls.ENABLED:

~\anaconda3\lib\site-packages\yt_dlp\jsinterp.py in interpret_statement(self, stmt, loca
l_vars, allow_recursion)
    459             if m:
    460                 sub_expr, expr = self._separate_at_paren(expr[m.end() - 1:])
--> 461                 ret, should_abort = self.interpret_statement(sub_expr, local_var
s, allow_recursion)
    462                 if should_abort:
    463                     return ret, True

~\anaconda3\lib\site-packages\yt_dlp\jsinterp.py in interpret_statement(self, stmt, loca
l_vars, allow_recursion, *args, **kwargs)
    183                 cls.write(stmt, level=allow_recursion)
    184             try:
--> 185                 ret, should_ret = f(self, stmt, local_vars, allow_recursion, *ar
gs, **kwargs)
    186             except Exception as e:
    187                 if cls.ENABLED:

~\anaconda3\lib\site-packages\yt_dlp\jsinterp.py in interpret_statement(self, stmt, loca
l_vars, allow_recursion)
    530         if len(sub_expressions) > 1:
    531             for sub_expr in sub_expressions:
--> 532                 ret, should_abort = self.interpret_statement(sub_expr, local_var
s, allow_recursion)
    533                 if should_abort:
    534                     return ret, True

~\anaconda3\lib\site-packages\yt_dlp\jsinterp.py in interpret_statement(self, stmt, loca
l_vars, allow_recursion, *args, **kwargs)
    183                 cls.write(stmt, level=allow_recursion)
    184             try:
--> 185                 ret, should_ret = f(self, stmt, local_vars, allow_recursion, *ar
gs, **kwargs)
    186             except Exception as e:
    187                 if cls.ENABLED:

~\anaconda3\lib\site-packages\yt_dlp\jsinterp.py in interpret_statement(self, stmt, loca
l_vars, allow_recursion)
    619                 continue
    620             left_val = self.interpret_expression(op.join(separated), local_vars,
allow_recursion)
--> 621             return self._operator(op, left_val, right_expr, expr, local_vars, al
low_recursion), should_return
    622 
    623         if m and m.group('attribute'):

~\anaconda3\lib\site-packages\yt_dlp\jsinterp.py in _operator(self, op, left_val, right_
expr, expr, local_vars, allow_recursion)
    300             right_expr = _js_ternary(left_val, *self._separate(right_expr, ':',
1))
    301 
--> 302         right_val = self.interpret_expression(right_expr, local_vars, allow_recu
rsion)
    303         if not _OPERATORS.get(op):
    304             return right_val

~\anaconda3\lib\site-packages\yt_dlp\jsinterp.py in interpret_expression(self, expr, loc
al_vars, allow_recursion)
    770 
    771     def interpret_expression(self, expr, local_vars, allow_recursion):
--> 772         ret, should_return = self.interpret_statement(expr, local_vars, allow_re
cursion)
    773         if should_return:
    774             raise self.Exception('Cannot return from an expression', expr)

~\anaconda3\lib\site-packages\yt_dlp\jsinterp.py in interpret_statement(self, stmt, loca
l_vars, allow_recursion, *args, **kwargs)
    183                 cls.write(stmt, level=allow_recursion)
    184             try:
--> 185                 ret, should_ret = f(self, stmt, local_vars, allow_recursion, *ar
gs, **kwargs)
    186             except Exception as e:
    187                 if cls.ENABLED:

~\anaconda3\lib\site-packages\yt_dlp\jsinterp.py in interpret_statement(self, stmt, loca
l_vars, allow_recursion)
    400         if expr.startswith('('):
    401             inner, outer = self._separate_at_paren(expr)
--> 402             inner, should_abort = self.interpret_statement(inner, local_vars, al
low_recursion)
    403             if not outer or should_abort:
    404                 return inner, should_abort or should_return

~\anaconda3\lib\site-packages\yt_dlp\jsinterp.py in interpret_statement(self, stmt, loca
l_vars, allow_recursion, *args, **kwargs)
    183                 cls.write(stmt, level=allow_recursion)
    184             try:
--> 185                 ret, should_ret = f(self, stmt, local_vars, allow_recursion, *ar
gs, **kwargs)
    186             except Exception as e:
    187                 if cls.ENABLED:

~\anaconda3\lib\site-packages\yt_dlp\jsinterp.py in interpret_statement(self, stmt, loca
l_vars, allow_recursion)
    530         if len(sub_expressions) > 1:
    531             for sub_expr in sub_expressions:
--> 532                 ret, should_abort = self.interpret_statement(sub_expr, local_var
s, allow_recursion)
    533                 if should_abort:
    534                     return ret, True

~\anaconda3\lib\site-packages\yt_dlp\jsinterp.py in interpret_statement(self, stmt, loca
l_vars, allow_recursion, *args, **kwargs)
    183                 cls.write(stmt, level=allow_recursion)
    184             try:
--> 185                 ret, should_ret = f(self, stmt, local_vars, allow_recursion, *ar
gs, **kwargs)
    186             except Exception as e:
    187                 if cls.ENABLED:

~\anaconda3\lib\site-packages\yt_dlp\jsinterp.py in interpret_statement(self, stmt, loca
l_vars, allow_recursion)
    761                            for v in self._separate(m.group('args'))]
    762             if fname in local_vars:
--> 763                 return local_vars[fname](argvals, allow_recursion=allow_recursio
n), should_return
    764             elif fname not in self._functions:
    765                 self._functions[fname] = self.extract_function(fname)

~\anaconda3\lib\site-packages\yt_dlp\utils\_utils.py in __call__(self, *args, **kwargs)
   5006
   5007     def __call__(self, *args, **kwargs):
-> 5008         return self.func(*args, **kwargs)
   5009
   5010     @classmethod

~\anaconda3\lib\site-packages\yt_dlp\jsinterp.py in resf(args, kwargs, allow_recursion)
    848             global_stack[0].update(kwargs)
    849             var_stack = LocalNameSpace(*global_stack)
--> 850             ret, should_abort = self.interpret_statement(code.replace('\n', '
'), var_stack, allow_recursion - 1)
    851             if should_abort:
    852                 return ret

~\anaconda3\lib\site-packages\yt_dlp\jsinterp.py in interpret_statement(self, stmt, loca
l_vars, allow_recursion, *args, **kwargs)
    183                 cls.write(stmt, level=allow_recursion)
    184             try:
--> 185                 ret, should_ret = f(self, stmt, local_vars, allow_recursion, *ar
gs, **kwargs)
    186             except Exception as e:
    187                 if cls.ENABLED:

~\anaconda3\lib\site-packages\yt_dlp\jsinterp.py in interpret_statement(self, stmt, loca
l_vars, allow_recursion)
    488                     break
    489                 try:
--> 490                     ret, should_abort = self.interpret_statement(body, local_var
s, allow_recursion)
```

```
    491                         if should_abort:
    492                             return ret, True

~\anaconda3\lib\site-packages\yt_dlp\jsinterp.py in interpret_statement(self, stmt, loca
l_vars, allow_recursion, *args, **kwargs)
    183                 cls.write(stmt, level=allow_recursion)
    184             try:
--> 185                 ret, should_ret = f(self, stmt, local_vars, allow_recursion, *ar
gs, **kwargs)
    186             except Exception as e:
    187                 if cls.ENABLED:

~\anaconda3\lib\site-packages\yt_dlp\jsinterp.py in interpret_statement(self, stmt, loca
l_vars, allow_recursion)
    606 
    607         for op in _OPERATORS:
--> 608             separated = list(self._separate(expr, op))
    609             right_expr = separated.pop()
    610             while True:

~\anaconda3\lib\site-packages\yt_dlp\jsinterp.py in _separate(expr, delim, max_split)
    266             in_unary_op = (not in_quote and not in_regex_char_group
    267                            and after_op not in (True, False) and char in '-+')
--> 268             after_op = char if (not in_quote and char in OP_CHARS) else (char.is
space() and after_op)
    269 
    270             if char != delim[pos] or any(counters.values()) or in_quote or in_un
ary_op:

KeyboardInterrupt:
```

In [ ]:

```
# def download_audio(url, output_path):
#     ydl_opts = {
#         'outtmpl': output_path,
#         'format': 'bestaudio/best',
#         'postprocessors': [{
#             'key': 'FFmpegExtractAudio',
#             'preferredcodec': 'mp3',
#             'preferredquality': '192',
#         }],
#         'ffmpeg_location' : ffmpeg_location,
#     }
#     try:
#         with yt_dlp.YoutubeDL(ydl_opts) as ydl:
#             ydl.download([url])
#     except Exception as e:
#         print(f"Error downloading {url}: {e}")
```

In [ ]:

```
# ffmpeg_location = 'c:/users/12505/anaconda3/lib/site-packages/ffmpeg/bin'
# # List of YouTube URLs to download
# urls = [
#     'https://www.youtube.com/watch?v=r7jgEE6t1u4&ab_channel=NewOrleansSaints',
# ]

# # Directory to save downloaded videos
# output_dir = '/videos/%(title)s.%(ext)s'

# for url in urls:
#     download_audio(url, output_dir)
```