```
def RunDownload(segments_filename, json_file_path):
# Main downloading function
  Segments <- PrepSegments(json_file_path):
    # parses JSON file
    Segments <- Segment(Start, End, Score, Id)
    return Segments

  FilenameMap <- GenListOfUrls(Segments):
    for Segment in Segments:
      split long Segment into fractions
      for [point_start, point_end] in fractions:
        urls, filenames <- betweenPoints(point_start, point_end)
        # generates three images from standing at point_start
        # and three images from standing at point_end
        return urls, filenames
    # build list of urls and filenames to download

  DownloadUrlFilenameMap(FilenameMap, Segments):
    for [url, filename, segment_id, nth_image] in FilenameMap:
      loaded, error <- url_retrieve_with_retry(url, filename)
      Segments[segment_id].HasLoadedImages[nth_image] = loaded
      Segments[segment_id].ErrorMessage[nth_image] = error

  SaveDataFile(Segments, segments_filename) # save pickled array
```

Listing 1: RunDownloader -> RunDownload

```
def RunCheck(segments_filename):
# Check downloaded segments and fix them
  Segments = LoadDataFile(segments_filename)
  if (HasErroneousData(Segments, ERROR)):
    Segments = FixDataFile_FailedDownloads(Segments, ERROR)
  SaveDataFile(Segments, segments_filename)

def FixDataFile_FailedDownloads(Segments):
  BrokenSegments = []
  for (i, Segment) in Segments:
    if Segment.ErrorMesages[i] == ERROR
      BrokenSegments.append(Segment)

  FilenameMapOfBroken <- GenListOfUrls(BrokenSegments)
  DownloadUrlFilenameMap(FilenameMapOfBroken, BrokenSegments)
  return Segments
```

Listing 2: RunDownloader -> RunCheck