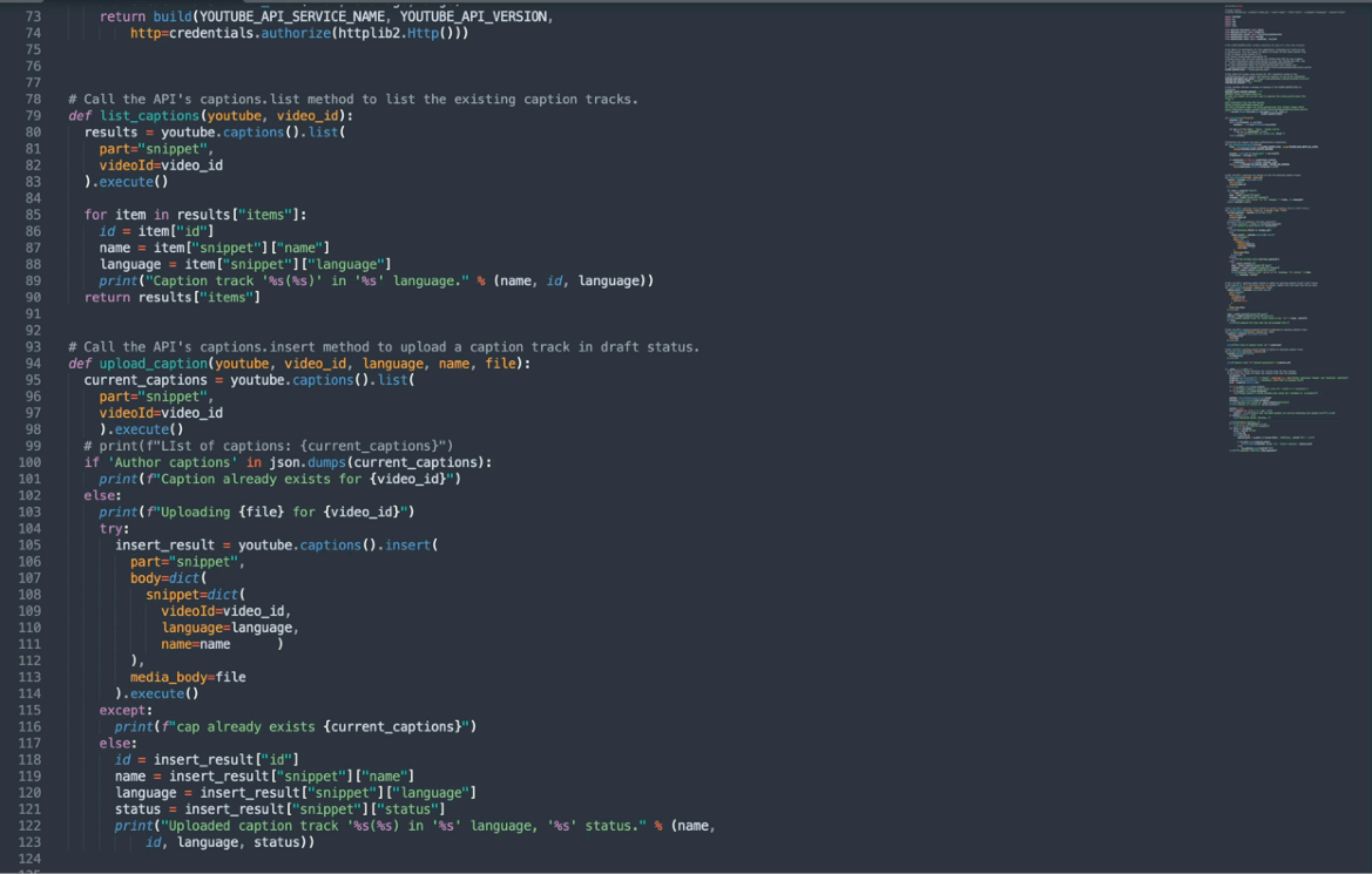
Peripheral awareness

Atención periférica

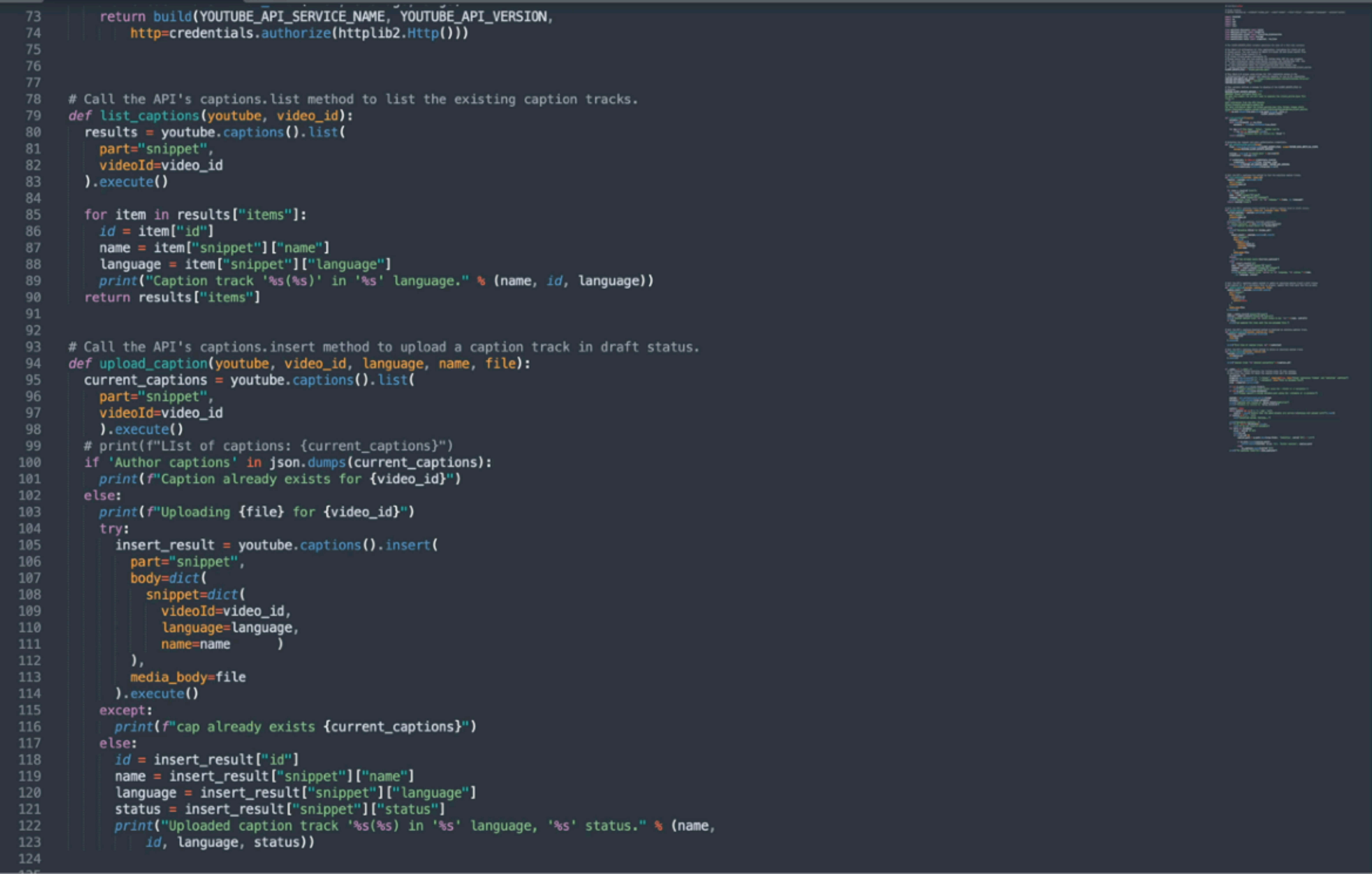
- Mientras tenemos el foco en algo puntual, seguimos prestando atención en segundo plano a lo que pasa en nuestra periferia.

- Cómo diseñamos para “la periferia” del foco de atención del usuario?

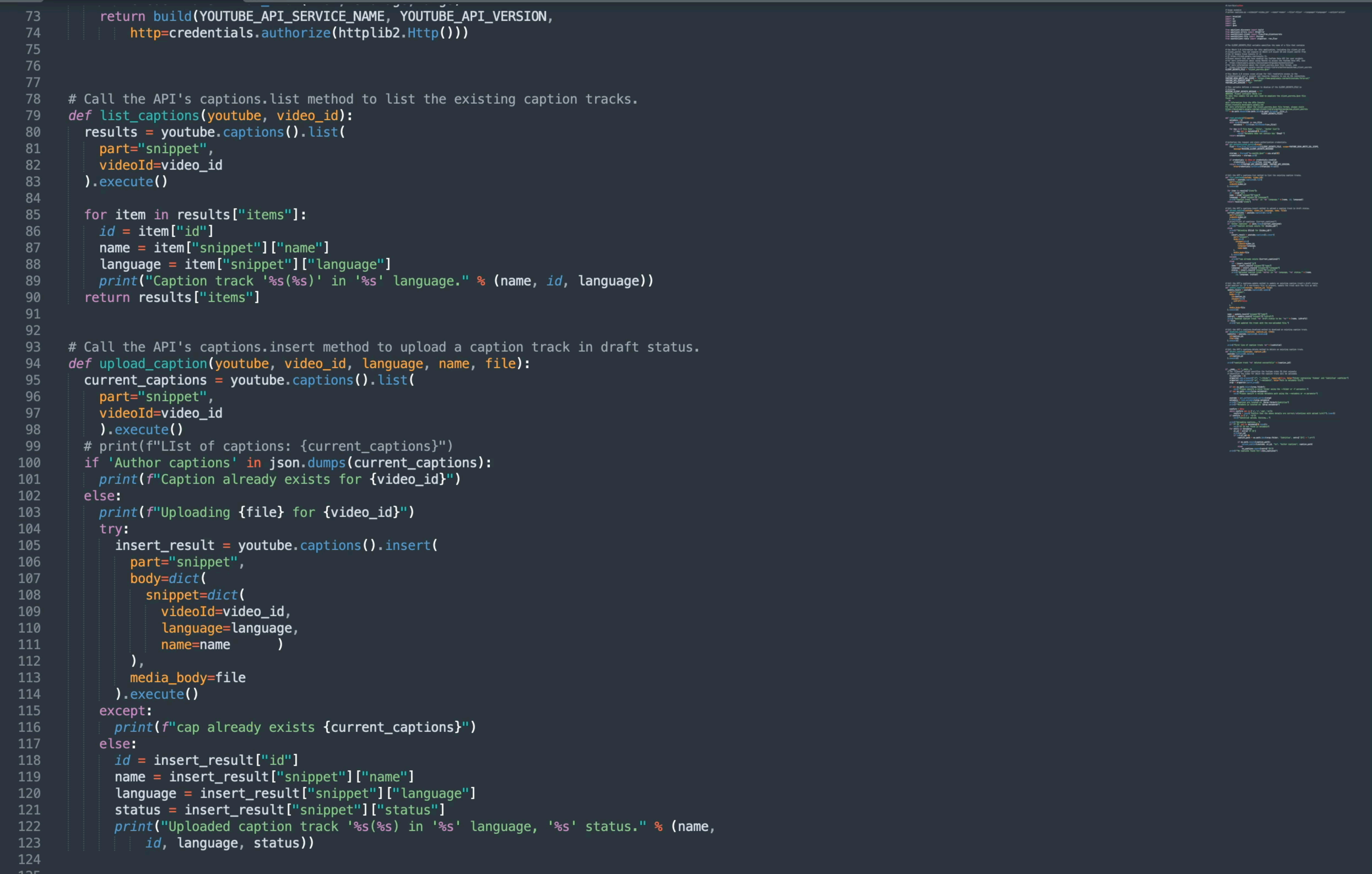
Sublime



```
    return build(YOUTUBE_API_SERVICE_NAME, YOUTUBE_API_VERSION,
        http=credentials.authorize(httplib2.Http()))



# Call the API's captions.list method to list the existing caption tracks.
def list_captions(youtube, video_id):
  results = youtube.captions().list(
    part="snippet",
    videoId=video_id
  ).execute()

  for item in results["items"]:
    id = item["id"]
    name = item["snippet"]["name"]
    language = item["snippet"]["language"]
    print("Caption track '%s(%s)' in '%s' language." % (name, id, language))
  return results["items"]


# Call the API's captions.insert method to upload a caption track in draft status.
def upload_caption(youtube, video_id, language, name, file):
  current_captions = youtube.captions().list(
    part="snippet",
    videoId=video_id
    ).execute()
  # print(f"LIst of captions: {current_captions}")
  if 'Author captions' in json.dumps(current_captions):
    print(f"Caption already exists for {video_id}")
  else:
    print(f"Uploading {file} for {video_id}")
    try:
      insert_result = youtube.captions().insert(
        part="snippet",
        body=dict(
          snippet=dict(
            videoId=video_id,
            language=language,
            name=name      )
        ),
        media_body=file
      ).execute()
    except:
      print(f"cap already exists {current_captions}")
    else:
      id = insert_result["id"]
      name = insert_result["snippet"]["name"]
      language = insert_result["snippet"]["language"]
      status = insert_result["snippet"]["status"]
      print("Uploaded caption track '%s(%s) in '%s' language, '%s' status." % (name,
          id, language, status))

```



```
    return build(YOUTUBE_API_SERVICE_NAME, YOUTUBE_API_VERSION,
        http=credentials.authorize(httplib2.Http()))



# Call the API's captions.list method to list the existing caption tracks.
def list_captions(youtube, video_id):
  results = youtube.captions().list(
    part="snippet",
    videoId=video_id
  ).execute()

  for item in results["items"]:
    id = item["id"]
    name = item["snippet"]["name"]
    language = item["snippet"]["language"]
    print("Caption track '%s(%s)' in '%s' language." % (name, id, language))
  return results["items"]


# Call the API's captions.insert method to upload a caption track in draft status.
def upload_caption(youtube, video_id, language, name, file):
  current_captions = youtube.captions().list(
    part="snippet",
    videoId=video_id
    ).execute()
  # print(f"LIst of captions: {current_captions}")
  if 'Author captions' in json.dumps(current_captions):
    print(f"Caption already exists for {video_id}")
  else:
    print(f"Uploading {file} for {video_id}")
    try:
      insert_result = youtube.captions().insert(
        part="snippet",
        body=dict(
          snippet=dict(
            videoId=video_id,
            language=language,
            name=name      )
        ),
        media_body=file
      ).execute()
    except:
      print(f"cap already exists {current_captions}")
    else:
      id = insert_result["id"]
      name = insert_result["snippet"]["name"]
      language = insert_result["snippet"]["language"]
      status = insert_result["snippet"]["status"]
      print("Uploaded caption track '%s(%s) in '%s' language, '%s' status." % (name,
          id, language, status))

```

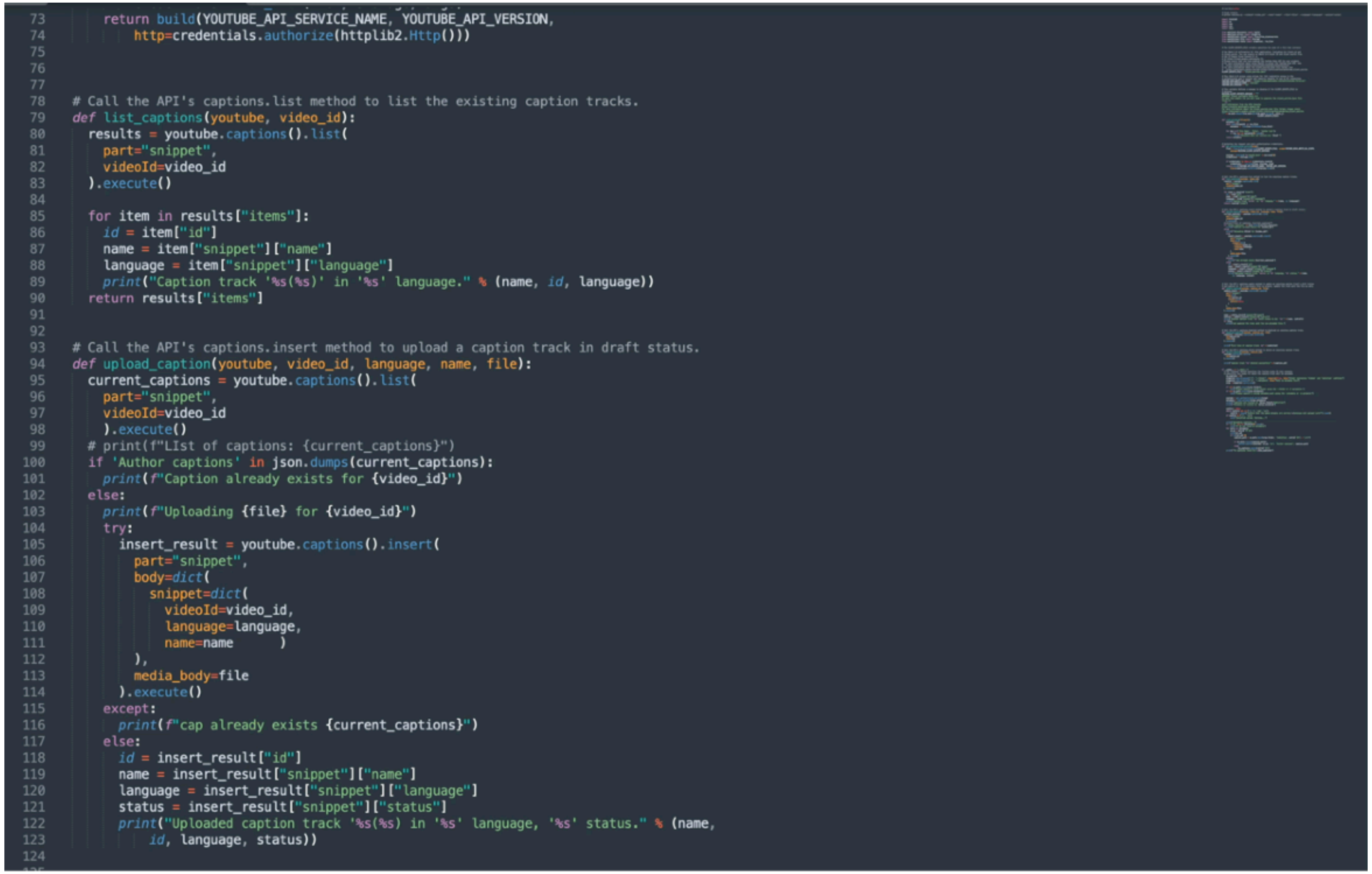


```
    return build(YOUTUBE_API_SERVICE_NAME, YOUTUBE_API_VERSION,
        http=credentials.authorize(httplib2.Http()))



# Call the API's captions.list method to list the existing caption tracks.
def list_captions(youtube, video_id):
  results = youtube.captions().list(
    part="snippet",
    videoId=video_id
  ).execute()

  for item in results["items"]:
    id = item["id"]
    name = item["snippet"]["name"]
    language = item["snippet"]["language"]
    print("Caption track '%s(%s)' in '%s' language." % (name, id, language))
  return results["items"]


# Call the API's captions.insert method to upload a caption track in draft status.
def upload_caption(youtube, video_id, language, name, file):
  current_captions = youtube.captions().list(
    part="snippet",
    videoId=video_id
    ).execute()
  # print(f"LIst of captions: {current_captions}")
  if 'Author captions' in json.dumps(current_captions):
    print(f"Caption already exists for {video_id}")
  else:
    print(f"Uploading {file} for {video_id}")
    try:
      insert_result = youtube.captions().insert(
        part="snippet",
        body=dict(
          snippet=dict(
            videoId=video_id,
            language=language,
            name=name       )
        ),
        media_body=file
      ).execute()
    except:
      print(f"cap already exists {current_captions}")
    else:
      id = insert_result["id"]
      name = insert_result["snippet"]["name"]
      language = insert_result["snippet"]["language"]
      status = insert_result["snippet"]["status"]
      print("Uploaded caption track '%s(%s) in '%s' language, '%s' status." % (name,
          id, language, status))

```

Atención periférica

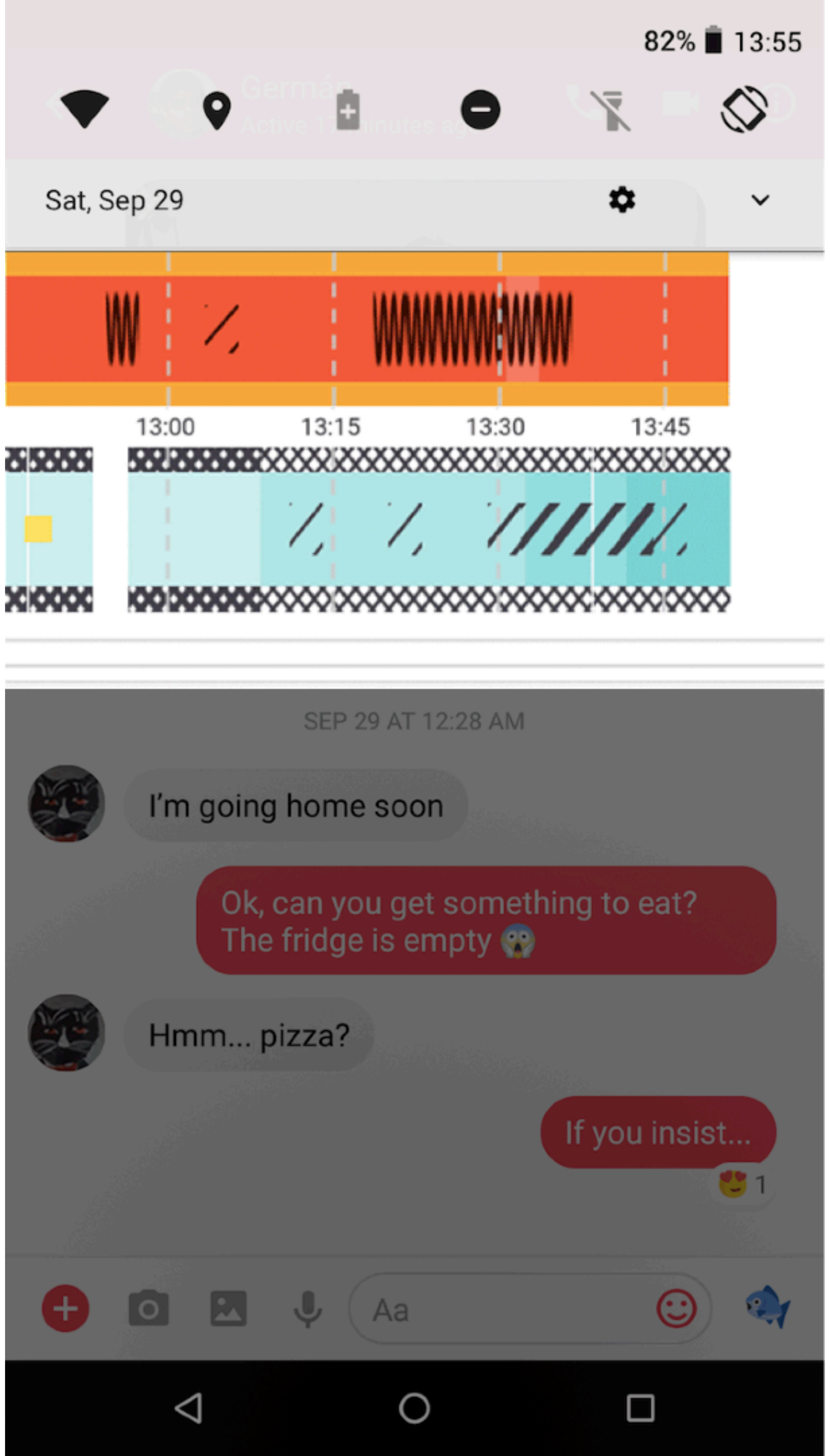
Peripheral awareness

- Mientras tenemos el foco en algo puntual, seguimos prestando atención en segundo plano a lo que pasa en nuestra periferia.
- Cómo diseñamos para “la periferia” del foco de atención del usuario?



```
    return build(YOUTUBE_API_SERVICE_NAME, YOUTUBE_API_VERSION,
        http=credentials.authorize(httplib2.Http()))



# Call the API's captions.list method to list the existing caption tracks.
def list_captions(youtube, video_id):
  results = youtube.captions().list(
    part="snippet",
    videoId=video_id
  ).execute()

  for item in results["items"]:
    id = item["id"]
    name = item["snippet"]["name"]
    language = item["snippet"]["language"]
    print("Caption track '%s(%s)' in '%s' language." % (name, id, language))
  return results["items"]


# Call the API's captions.insert method to upload a caption track in draft status.
def upload_caption(youtube, video_id, language, name, file):
  current_captions = youtube.captions().list(
    part="snippet",
    videoId=video_id
    ).execute()
  # print(f"LIst of captions: {current_captions}")
  if 'Author captions' in json.dumps(current_captions):
    print(f"Caption already exists for {video_id}")
  else:
    print(f"Uploading {file} for {video_id}")
    try:
      insert_result = youtube.captions().insert(
        part="snippet",
        body=dict(
          snippet=dict(
            videoId=video_id,
            language=language,
            name=name       )
        ),
        media_body=file
      ).execute()
    except:
      print(f"cap already exists {current_captions}")
    else:
      id = insert_result["id"]
      name = insert_result["snippet"]["name"]
      language = insert_result["snippet"]["language"]
      status = insert_result["snippet"]["status"]
      print("Uploaded caption track '%s(%s) in '%s' language, '%s' status." % (name,
          id, language, status))
```

Sublime



I'm going home soon

Ok, can you get something to eat? The fridge is empty 😱

Hmm... pizza?

If you insist...

Visualizaciones de “información contextual” para parejas

Lifelines

Griggio, Nouwens, McGrenere and Mackay. CHI 2019. https://doi.org/10.1145/3290605.3300853