```
            conn = sqlite3.connect('memory.db')
            conn.execute("INSERT INTO image_uploads (filename, url, upload_date)
            VALUES (?, ?, ?)", (image_listing[i], result['link'], datetime.
            strftime(datetime.now(), '%d-%m-%Y')))
            conn.commit()
            conn.close()

            print result['link']
            tts('Your image has been uploaded')

def show_all_uploads():
    conn = sqlite3.connect('memory.db')

    cursor = conn.execute("SELECT * FROM image_uploads")

    for row in cursor:
        print(row[0] + ': (' + row[1] + ') on ' + row[2])

    tts('Requested data has been printed on your terminal')

    conn.close()
```

That's a lot of code! Let's go through it line by line. You need the `sqlite3` and `datetime` modules to store the URLs where the images are uploaded and to get the date when the image was uploaded, respectively.

You import `ImgurClient` from the Imgur client to help you upload the images to Imgur. The `img_list_gen()` function takes `images_path` as an argument and searches for all the PNGs, GIFs, JPGs, and TIFFs, in the current folder and its subfolders. This code is essentially the same as the `mp3gen()` function you built in Chapter 5. It returns a list of all the files found with the extensions specified.

Next comes the `image_uploader()` function, which takes `speech_text`, `client_id`, `client_secret`, and `images_path` as arguments. Just as in the `define_subject` module, it splits the words, removes the keyword *upload*, and extracts the name of the image to be uploaded. You then pass `client_id` and `client_secret` to `ImgurClient` to authenticate.

Now you search for the specified image file among the list of images. If a match is found, you upload it to Imgur using the `client.upload_from_path()` function. This function takes the path of the image to be uploaded as an argument. You save the output of this function to a variable named `result` and save the image details to the database.

You use the SQL statement to insert the image values into the corresponding columns of the table. Note that `result['link']` stores the link where the image was uploaded on Imgur. The result dictionary also stores a host of other information you may wish to save in the database or look at. You then commit the changes made to the database and close the connection to it. You also display the link where the image has been uploaded on the terminal, and Melissa gives the message that the image has been uploaded.

The last function is the `show_all_uploads()` function, which retrieves all the stored entries in the `image_uploads` table and displays the stored information in a formatted fashion on the terminal. Melissa gives a message that the requested data has been