1. **Differenza tra il protocollo HTTP 1.0 e HTTP 1.1**
   HTTP 1.0 crea una nuova connessione per ogni richiesta dal client al server, mentre HTTP 1.1 introduce il concetto di "keep-alive". Questo significa che la connessione può rimanere aperta e essere riutilizzata per più richieste, migliorando la velocità di caricamento delle pagine.
2. **I messaggi HTTP: Richiesta e Risposta**
   In HTTP, una **richiesta** parte dal client (di solito un browser) e chiede al server una risorsa, come una pagina web. Una **risposta** è il messaggio che il server invia indietro, fornendo i dati richiesti o segnalando eventuali errori. Entrambi i messaggi sono composti da tre parti: una riga iniziale (che specifica il tipo di richiesta o risposta), delle intestazioni (header) che contengono informazioni aggiuntive, e il corpo del messaggio (body), che può includere i dati della risorsa richiesta, come un file HTML.
3. **HTTP keep-alive**
   Nel messaggio di esempio, la riga **"Connection: keep-alive"** indica che il browser vuole mantenere la connessione aperta anche dopo aver ricevuto una risposta. Questo riduce il tempo necessario per richieste successive. Il parametro **`id=1389`** dopo **`/percorsi-studio`** è un modo per passare informazioni aggiuntive al server, spesso usato per identificare specifici dati (in questo caso, potrebbe riferirsi a un percorso di studio con quell'ID).
4. **Status Code di un messaggio di risposta**
   Gli **status code** sono numeri che indicano l'esito di una richiesta. Ad esempio, un **200 OK** significa che la richiesta è andata a buon fine, mentre un **404 Not Found** indica che la risorsa non è stata trovata sul server. Ce ne sono vari, ognuno con un significato preciso che descrive cosa è successo con la richiesta.
5. **Metodi HTTP: GET e POST**
   - **GET** richiede dati dal server, e i parametri sono visibili nell'URL (es: `?id=1389`).
   - **POST** invia dati al server, ma questi non compaiono nell'URL. In PHP, i dati inviati tramite GET e POST possono essere gestiti rispettivamente con "`$_GET`" e "`$_POST`".
6. **Codifica URL**
   Alcuni caratteri non possono essere inclusi direttamente in un URL (come spazi o simboli speciali), quindi vengono codificati. Ad esempio, uno spazio viene trasformato in "**`%20`**" , e un simbolo come" **`&`** " diventa "**`%26`**". Questa codifica serve per assicurarsi che i dati nell'URL vengano interpretati correttamente.
7. **Altri metodi di richiesta HTTP**
   - **PUT** carica o aggiorna una risorsa sul server.
   - **DELETE** elimina una risorsa.
   - **TRACE** mostra il percorso che una richiesta ha seguito.
   - **CONNECT** stabilisce un tunnel per una connessione sicura.