This Python program demonstrates the **Diffie-Hellman key exchange algorithm**, which is a way for two people (commonly called Alice and Bob) to securely agree on a **shared secret key** over an insecure channel (like the internet), without sending the key itself.

Here's a step-by-step explanation of what each part of the code does:

◆ Step 1: Input Public Parameters

```
p = int(input("Enter a large prime number (p): "))
g = int(input("Enter a primitive root modulo p (g): "))
```

- `p`: A large **prime number**, which is publicly known.
- `g`: A **primitive root modulo `p`**, also public. It's a number with special mathematical properties that make it suitable for key exchange.
- These two numbers (`p` and `g`) are agreed upon by both parties before the exchange begins.

◆ Step 2: Input Private Keys

```
a = int(input("User A, enter your private key (a): "))
b = int(input("User B, enter your private key (b): "))
```

- `a` and `b` are **private keys**, chosen separately by Alice and Bob.
- These values are kept **secret** and never shared.

◆ Step 3: Generate Public Keys

```
A = pow(g, a, p)  # Alice's public key
B = pow(g, b, p)  # Bob's public key
```

- Each person calculates a **public key** using their private key:
  - Alice computes `A = g^a mod p`
  - Bob computes `B = g^b mod p`
- These public keys **can be shared openly** with each other.

◆ Step 4: Compute Shared Secret Key

```
shared_secret_alice = pow(B, a, p)
shared_secret_bob = pow(A, b, p)
```

- After exchanging public keys:
  - Alice computes the shared secret as `B^a mod p`
  - Bob computes the shared secret as `A^b mod p`
- Because of the math behind modular exponentiation:
  - `B^a mod p = A^b mod p = shared secret`
  - So both parties end up with the **same secret key**, without having sent it over the network.

◆ Step 5: Confirm Shared Secret Matches

```
if shared_secret_alice == shared_secret_bob:
    print(f"✅ Success! Shared secret key established: {shared_secret_alice}")
```

- This checks if both parties have calculated the **same shared key**.
- If they match, the key exchange was successful.

✅ Summary:

This code simulates how two users can **safely agree on a shared key** using the **Diffie-Hellman algorithm**, even if someone is listening to the communication.

Would you like a version of this that includes automatic random key generation instead of manual input?