What each function in the code does:

1. **random_bits(n: int) -> List[int]**
   a. Creates a list of n classical bits that Alice will try to securely send to Bob

2. **random_bases(n: int) -> List[int]**
   a. Returns a list of n integers where 0 represents the Z (computational) basis and 1 represents the X (Hadamard) basis
   b. Alice uses this for encoding while Bob and Eve use this for measurement
   c. Random bases ensure that an eavesdropper can't know in advance which measurement will reveal the correct bit

3. **prepare_circuits_for_alice(bits: List[int], bases: List[int]) -> List[QuantumCircuit]**
   a. Iterates through each bit-basis pair and creates a single-qubit Qiskit quantum circuit
   b. The function returns a list of these circuits which represent the qubits Alice will transmit to Bob

4. **measure_in_basis_circuits(bases: List[int], incoming_circuits: List[QuantumCircuit]) -> List[QuantumCircuit]**
   a. Takes circuits that represent qubits that have already been transmitted and copies them
   b. For each qubit, if the chosen measurement basis is X, apply a Hadamard to rotate the X basis back to the Z basis so that a computational measurement gives the right outcome
   c. Finally, it calls the measure(0,0) operation so the qubit will collapse to a classical bit on execution

5. **run_circuits_get_bits(circuits: List[QuantumCircuit], shots: int = 1) -> List[int]**
   a. Runs circuits on AerSimulator (with shots = 1) so that each circuit produces one outcome
   b. For each of those circuits it calls result.get_counts(i) to get a dictionary, extracts the key, and converts to an integer
   c. The returned list acts as Bob's or Eve's measured classical bits

6. **eve_intercept_resend(alice_circuits: List[QuantumCircuit], eve_bases: List[int]) -> List[QuantumCircuit]**
   a. Eve measures Alice's qubits with her own random basis choices, then runs them to get her measurement outcomes
   b. Using those outcomes, she prepares a new circuit which is sent towards Bob
   c. Since Eve will guess the wrong basis, her measurement disturbs the qubits, which introduces detectable errors.

7. **sift_key(alice_bits: List[int], alice_bases: List[int], bob_bits: List[int], bob_bases: List[int]) -> Tuple[List[int], List[int], List[int]]**
   a. Compare Alice's and Bob's basis choices in the list.
   b. Keep the ones where the bases match, then return Alice's sift key bits, Bob's sifted key bits, and the list of indices that were retained.

How QKD works and why it is secure:

1. **There are two types of basis that are used**
    a. Z-Basis with states $|0\rangle$ and $|1\rangle$
        i. Measuring in the Z basis tells you directly whether the qubit is $|0\rangle$ and $|1\rangle$
    b. X (Hadamard) Basis with states $|+\rangle$ and $|-\rangle$ which are defined as $(|0\rangle \pm |1\rangle) / \sqrt{2}$
        i. Measuring in the X basis utilizes a Hadamard gate and then a normal Z measurement
        ii. A measurement in X reveals whether the qubit is in superposition leaning toward $|+\rangle$ or $|-\rangle$

2. **How this can be used**
    a. When you measure using the wrong basis (i.e. measuring in Z when the state is $|+\rangle$) will result in a 50/50 probability of the opposite basis
        i. i.e. measuring in X basis when the state is $|0\rangle$ will result in a 50/50 chance of getting $|+\rangle$ or $|-\rangle$
    b. This incompatibility between measurements of the Z and X basis will "disturb" the qubit that Bob receives
        i. i.e Alice send $|+\rangle$ which Eve guess is the Z basis and measures as though it were the Z basis. This measurement destroys the superposition and it becomes either $|0\rangle$ or $|1\rangle$ with 50% probability, and the qubit is stuck in that state. Then, when Bob tries to measure that qubit in the X-basis (the correct basis), he will get a random outcome. This randomness will show up as error when Alice and Bob compare results and be detectable as interference by Eve.
        ii. This process is irreversible meaning Eve cannot prevent this interference from altering the qubit
        iii. The no-cloning theorem prevents Even from making a copy of the qubit to measure later in the right basis

3. **Purpose and implementation of error correction**
    a. Errors can occur in a multitude of ways
        i. Imperfection in equipment
        ii. Channel noise
        iii. Eavesdropping
    b. Error correction helps ensure that
        i. Alice and Bob end up with identical keys
        ii. If Alice and Bob have too many mismatched keys, the keys would be useless in cryptography
    c. Only parity informations in communicated NOT the whole key itself
        i. Parity information is essentially a 1-bit summary of whether the number of 1s in a block of bits is even or odd
        ii. Alice and Bob use this information to detect where their sifted keys differ, then correct the errors while leaking as little information as possible

        iii. Eve may learn whether the block had an even or odd number of errors, but not the actual values of the bits

4. **Cascade**
    a. Cascade is one of the most popular error correction protocols for QKD
    b. Step-by-step outline
        i. Alice and Bob divide their keys into blocks of certain lengths and then compute parity (even or odd number of 1s) and then exchange these bits publicly
        ii. If a block has a mismatch, Alice and Bob perform a binary search:
            1. Split the block in half, compare parities again
            2. Keep splitting until they identify the exact bit where they differOnce found Bob flips that bit to match Alice
        iii. After fixing one block, it may change the parity of blocks in earlier passes. So, Cascade makes multiple passes with larger block sizes.
    c. With enough passes, Cascade can correct all errors with high probability
    d. The use of only parity information means that it leaks very little usable information to Eve

5. **Privacy Amplification**
    a. After applying error correction our sifted keys should now be identical, but Eve may have learned some partial information
    b. Privacy amplification aims to ensure that whatever Eve knows becomes useless
    c. How it works
        i. Alice and Bob take their corrected key and apply a hash function to it, which reduces it to a shorter key
        ii. For example, if Eve knows 100 bits of a 10,000-bit key and we compress it into 5,000 bits using a hash, the 100 known bits no longer map nearly into the new, shortened key
        iii. To Eve, this new key looks random, and the shortened key is secure and shared between Alice and Bob