BB84 Cheat Sheet

Essential Tools

Concept	Example

choices(...) function to randomly select any number of elements from a given list.	Python
	`from random import choices`
	
	`choices(choice_list, k = num_choices)`

Create a python dictionary.	Python `my_dictionary = {key1: value1, key2: value2, key3: value3, ...}`
Access an element of a dictionary.	Python `my_dictionary[key1]`

Use a for loop to execute the same code multiple times:
• index is the loop control variable, which takes on each value in the range, one at a time. This follows the same naming conventions as any other variable.
	Python
	`for index in range(number):`
	`  code to repeat`

• range(number) provides all the numbers from 0 up to, but not including, number **NOTE**: Don't forget the colon at the end.	

BB84 Protocol with Cirq

Concept	Example
Step #1: Generate the key.	Python `alice_key = choices([0, 1], k = num_bits`
Step #2: Alice picks bases.	Python `alice_bases = choices(['Z', 'X'], k = num_bits)`

Step #3: Alice creates qubits.	Python
	`alice_circuit = cirq.Circuit()`
	
	`for bit in range(num_bits):`
	
	`encode_value = alice_key[bit]`
	`encode_gate = encode_gates[encode_value`
	
	`basis_value = alice_bases[bit]`
	`basis_gate = basis_gates[basis_value]`
	
	`qubit = qubits[bit]`
	`alice_circuit.append(encode_gate(qubit)`
	`alice_circuit.append(basis_gate(qubit))`

Step #4: Alice sends qubits to Bob.	No code needed for this step.

Step #5: Bob picks bases.	Python
	`bob_bases = choices(['Z', 'X'], k = num_bits)`
	
	`bob_circuit = cirq.Circuit()`
	`for bit in range(num_bits):`
	`basis_value = bob_bases[bit]`
	`basis_gate = basis_gates[basis_value]`
	
	`qubit = qubits[bit]`
	`bob_circuit.append(basis_gate(qubit))`

Step #6: Bob measures qubits.	Python `bob_circuit.append(cirq.measure(qubits, key = 'bob key'))`

Step #7: Bob creates his key.	Python
	`bb84_circuit = alice_circuit + bob_circuit`
	
	`sim = cirq.Simulator()`
	`results = sim.run(bb84_circuit)`
	`bob_key = results.measurements['bob key'][0]`

Step #8: Alice and Bob compare bases.	Python
	`final_alice_key = []`
	`final_bob_key = []`
	
	`for bit in range(num_bits):`
	
	`if alice_bases[bit] == bob_bases[bit]:`
	`final_alice_key.append(alice_key[bit])`
	`final_bob_key.append(bob_key[bit])`

Step #9: Checking for an Eavesdropper.	Python
	`if final_alice_key[0] == final_bob_key[0]:`
	`  final_alice_key = final_alice_key[1:]`
	`  final_bob_key = final_bob_key[1:]`
	`  print('We can use our keys!')`
	
	`else:`
	`  print('Eve was listening, we need to use a different channel!')`