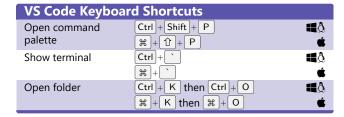
Q# Tutorial Quick Reference

Working with Tutorials

Command Line Basics Change directory cd dirname Go to home cd ~ Go up one directory cd .. Make new directory mkdir dirname Open current code . directory in VS Code

Building Project and Running Tests	
Change directory to project directory	cd project-dir
Build solution	dotnet build
Run all unit tests	dotnet test



Documentation	
Quantum	https://docs.microsoft.com/
Development Kit	quantum
Q# Language	https://docs.microsoft.com/
Reference	en-us/quantum/quantum-qr-intro
Q# Library	https://docs.microsoft.com/
Reference	en-us/qsharp/api/

Q# Language

Primitive Types	
64-bit integers	Int
Double-precision floats	Double
Booleans	Bool
	true or false
Qubits	Qubit
Pauli operators	Pauli
	PauliI, PauliX, PauliY, or PauliZ
Measurement	Result
results	Zero or One
Sequences of	Range
integers	e.g.: 110 or 510
Strings	String

Q# Language (continued)

Derived Types	
Arrays	elementType[]
Tuples	(type0, type1,)
Functions	<pre>inputType -> outputType e.g.: ArcTan2 : (Double, Double) -> Double</pre>
Operations	<pre>inputType => outputType : variants e.g.: H : (Qubit => () : Adjoint, Controlled)</pre>

Calling w/	Functors
Adjoint call	<pre>(Adjoint Name)(operationInputs)</pre>
Controlled call	<pre>(Controlled Name)(controlQubits, operationInputs)</pre>

Variables	
Declare immutable	let name = value
Declare mutable	mutable <i>name</i> = <i>value</i>
Update mutable	set name = value

Arrays	
Allocation	<pre>mutable name = new Type[length]</pre>
Length	Length(<i>name</i>)
<i>i</i> th element (0-based indexing)	name[i]
Array literal	[value0; value1;] e.g.:[true; false; true]
Slicing (subarray)	<pre>let name = name[startend]</pre>

```
Control Flow

For loop for (ind in range) { ... }
e.g.: for (i in 0..N-1) { ... }

Repeat-until-
success loop until condition
fixup { ... }

Conditional if cond1 { ... }
statement elif cond2 { ... }
else { ... }
```

Qubits and Operations on Qubits

```
Standard Library Functions and Operations
Pauli operations
                    I : (Qubit => ())
                    X : (Qubit => ())
                    Y : (Oubit => ())
                    Z : (Qubit => ())
Hadamard
                    H : (Qubit => ())
Controlled-NOT
                    CNOT : ((control : Qubit,
                    target : Qubit) => ())
Measure one qubit
                   M : Qubit => Result
in Pauli Z basis
Perform joint mea-
                    Measure : (Pauli[], Qubit[]) =>
surement of qubits
in given Pauli bases
                   R : (Pauli, Double, Qubit) => ()
Rotate about given
Pauli axis
Rotate about Pauli
                    Rx : (Double, Qubit) => ()
X, Y, Z axes
                    Ry : (Double, Qubit) => ()
                    Rz : (Double, Qubit) => ()
Reset qubit to |0\rangle
                    Reset : Qubit => ()
                    ResetAll : Qubit[] => ()
Reset qubits to
|0..0\rangle
Apply a gate to each
                    ApplyToEach : ((Qubit => ()),
aubit
                    Qubit[]) => ()
Display a log mes-
                   Message : String -> ()
sage
```