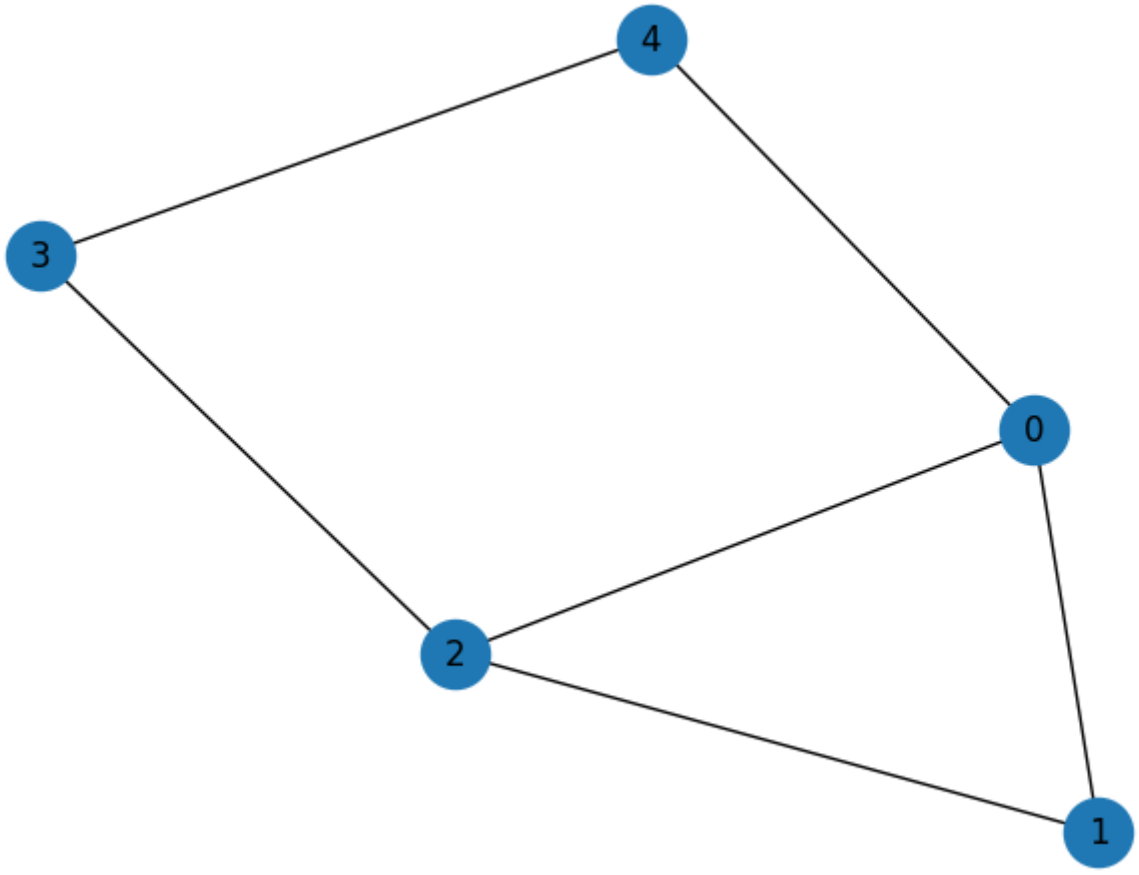
In [1]:
```
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister
from qiskit.visualization import *
from qiskit.quantum_info import Statevector
from qiskit.transpiler.preset_passmanagers import generate_preset_pass_manager
from qiskit_ibm_runtime import Sampler
from numpy import pi
```

In [2]:
```
from qiskit_ibm_runtime import QiskitRuntimeService

service = QiskitRuntimeService(channel="ibm_cloud",
                               token="",
                               instance="firsthwinstance")
```

In [3]:
```
from qiskit_ibm_runtime import QiskitRuntimeService
QiskitRuntimeService.save_account(channel="ibm_cloud",
                                  token="",
                                  instance="firsthwinstance", name="Renzo Diomedi",
                                  overwrite=True)
```

In [4]:

```
import rustworkx as rx
from rustworkx.visualization import mpl_draw as draw_graph
import numpy as np

n = 5

graph = rx.PyGraph()
graph.add_nodes_from(np.arange(0, n, 1))
edge_list = [(0, 1, 1.0), (0, 2, 1.0), (0, 4, 1.0), (1, 2, 1.0), (2, 3, 1.0), (3, 4, 1.0)]
graph.add_edges_from(edge_list)
draw_graph(graph, node_size=600, with_labels=True)
```


4
3
0
2
1

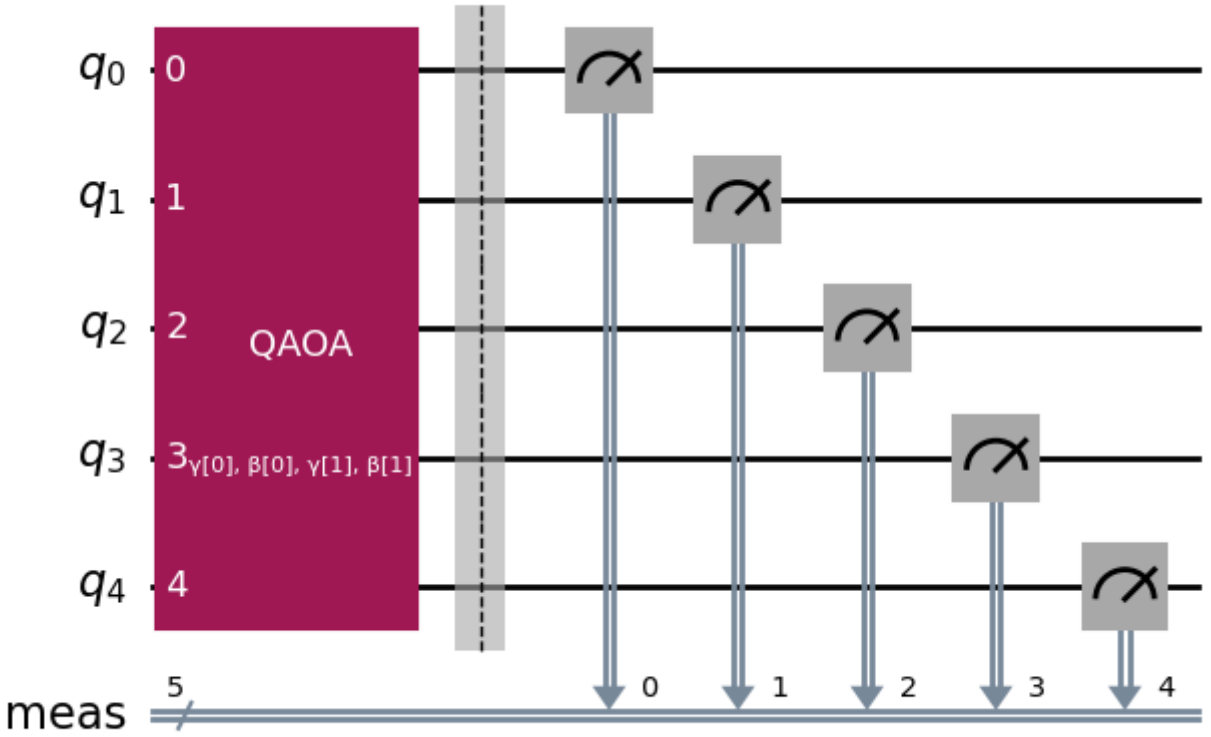
In [5]:

```python
from qiskit.quantum_info import SparsePauliOp
def build_max_cut_paulis(graph: rx.PyGraph) -> list[tuple[str, float]]:
    """Convert the graph to Pauli list.

    This function does the inverse of `build_max_cut_graph`
    """
    pauli_list = []
    for edge in list(graph.edge_list()):
        paulis = ["I"] * len(graph)
        paulis[edge[0]], paulis[edge[1]] = "Z", "Z"

        weight = graph.get_edge_data(edge[0], edge[1])

        pauli_list.append(("".join(paulis)[::-1], weight))

    return pauli_list


max_cut_paulis = build_max_cut_paulis(graph)

cost_hamiltonian = SparsePauliOp.from_list(max_cut_paulis)
print("Cost Function Hamiltonian:", cost_hamiltonian)
```

```
Cost Function Hamiltonian: SparsePauliOp(['IIIZZ', 'IIZIZ', 'ZIIIZ', 'IIZZI', 'IZZII', 'ZZIII'],
              coeffs=[1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j, 1.+0.j])
```

In [6]:
```
from qiskit.circuit.library import QAOAAnsatz

circuit = QAOAAnsatz(cost_operator=cost_hamiltonian, reps=2)
circuit.measure_all()

circuit.draw('mpl')
```

Out[6]:



In [7]:
```
circuit.parameters
```

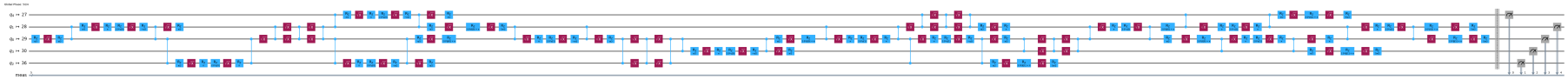
Out[7]:
```
ParameterView([ParameterVectorElement(β[0]), ParameterVectorElement(β[1]), ParameterVectorElement(γ[0]), P
arameterVectorElement(γ[1])])
```

In [8]:

```
from qiskit_ibm_runtime import QiskitRuntimeService
from qiskit.transpiler.preset_passmanagers import generate_preset_pass_manager


# QiskitRuntimeService.save_account(channel="ibm_quantum", token="<MY_IBM_QUANTUM_TOKEN>", overwrite=Tru
# service = QiskitRuntimeService(channel='ibm_quantum')
backend = service.least_busy(min_num_qubits=127)
print(backend)

# Create pass manager for transpilation
pm = generate_preset_pass_manager(optimization_level=3,
                                  backend=backend)

candidate_circuit = pm.run(circuit)
candidate_circuit.draw('mpl', fold=False, idle_wires=False)
```

```
<IBMBackend('ibm_torino')>
```

Out[8]:



In [9]:
```
from qiskit import QuantumCircuit
#from qiskit_ibm_runtime import QiskitRuntimeService, SamplerV2 as Sampler
candidate_circuit = QuantumCircuit(2)
candidate_circuit.measure_all()
# Create empty circuit
#empty_circuit = QuantumCircuit(2)
#empty_circuit.measure_all()

# You'll need to specify the credentials when initializing QiskitRuntimeService, if they were not previ
service = QiskitRuntimeService()
backend = service.least_busy(operational=True, simulator=False)

sampler = Sampler(backend)
job = sampler.run([candidate_circuit])
print(f"job id: {job.job_id()}")
result = job.result()
print(result)
```

```
job id: d0ctjqec80ps73dgmrt0
PrimitiveResult([SamplerPubResult(data=DataBin(meas=BitArray(<shape=(), num_shots=4096, num_bits=2>)), met
adata={'circuit_metadata': {}})], metadata={'execution': {'execution_spans': ExecutionSpans([DoubleSliceSp
an(<start='2025-05-06 09:49:09', stop='2025-05-06 09:49:12', size=4096>)])}, 'version': 2})
```

In [10]:
```
job = service.job('d0ctjqec80ps73dgmrt0')
job.status()
```

Out[10]:
```
'DONE'
```

In [ ]:
```

```