Ecrire un programme élémentaire avec qiskit

(valide à v2.1)

In [ ]:
```
# valider que qiskit est installé à la version voulue

import qiskit
print(qiskit.__version__)
```

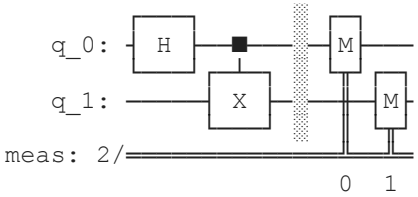
In [2]:
```
# Une syntaxe, la plus simple possible
# importer la classe QuantumCircuit pour pouvoir l'instancier (qc)
# ici avec 2 qubits
from qiskit import QuantumCircuit

qc = QuantumCircuit(2)

# la construction du circuit consiste à ajouter des portes quantiques
# qui sont des méthodes le l'objet qc de la classe QuantumCircuit
# ici on ajoute une porte Hadamard sur le qubit 0
# et une porte CNOT (CX) entre le qubit 0 et le qubit 1
# on demande l'ajout des opérations de mesure vers un registre classique
# et on "dessine" le circuit pour vérifier qu'il est correct

qc.h(0)
qc.cx(0, 1)
qc.measure_all()
qc.draw()
```

Out[2]:



In [3]:
```
# on utilise le sampler de l'API de Qiskit pour exécuter le circuit
# il s'agit de cumuler les résultats d'un certain nombre de shots
# ici on demande 100 shots

# exerices, modifier le circuit pour obtenir :
#     seulement le résutlat 11
```

```
#     seulement le résultat 01 (remarque : endian)
#     une superposition des quatre résultats possibles

from qiskit.primitives import StatevectorSampler
sampler = StatevectorSampler()
job = sampler.run([qc], shots=100)
result = job.result()
print(result)
print(f" > Counts: {result[0].data['meas'].get_counts()}")
d = result[0].data['meas'].get_counts()
d
```

```
PrimitiveResult([SamplerPubResult(data=DataBin(meas=BitArray(<shape=(), num_shots=100, num_bits=2>)), metadata={'shots': 100, 'cir
cuit_metadata': {}})], metadata={'version': 2})
 > Counts: {'11': 61, '00': 39}
```

Out[3]: {'11': 61, '00': 39}

JM.Torres

20250721