PENNYLANE (XANADU) TUTORIAL

Aluno: Pedro Souza
Professor: Wilson Rabelo

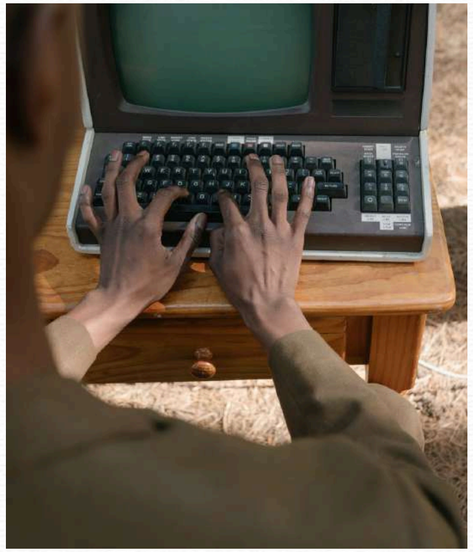
UFPA - 2026

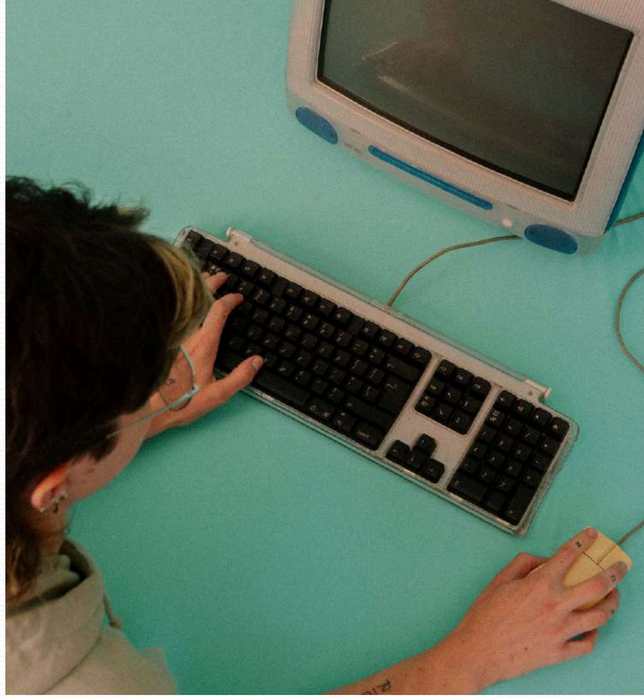
Índice

Introdução

Introdução ao Pennylane (Xanadu)

- É uma biblioteca open-source de Machine Learning Quântico (QML) da empresa canadense Xanadu.
- Integra ML clássico com computação quântica.
- Funciona com PyTorch, TensorFlow, JAX e NumPy (bibliotecas Python de ML)

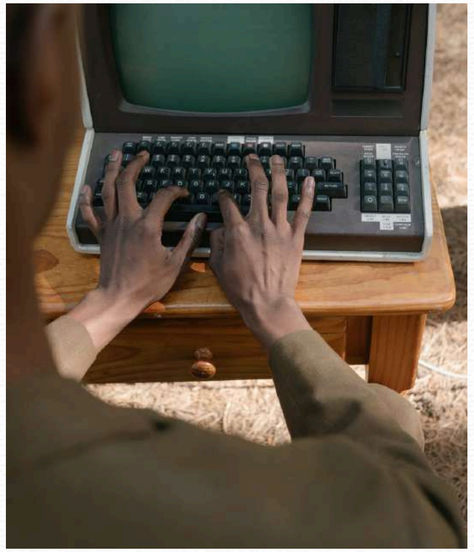


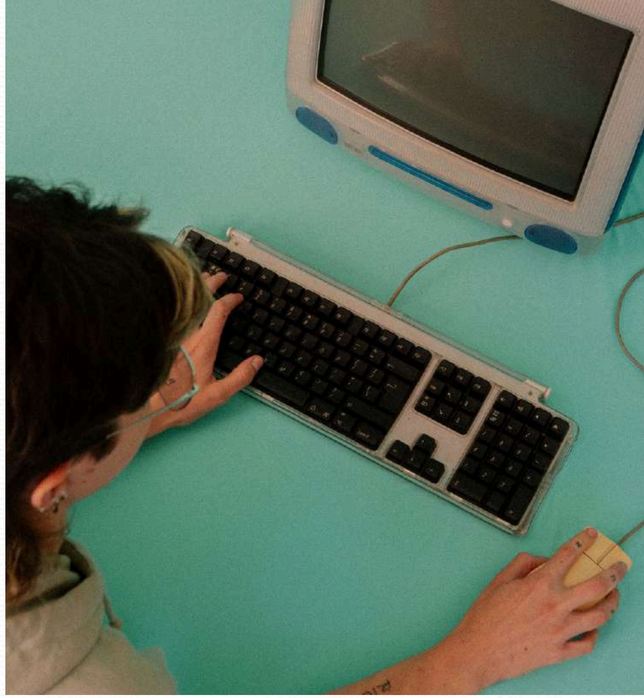


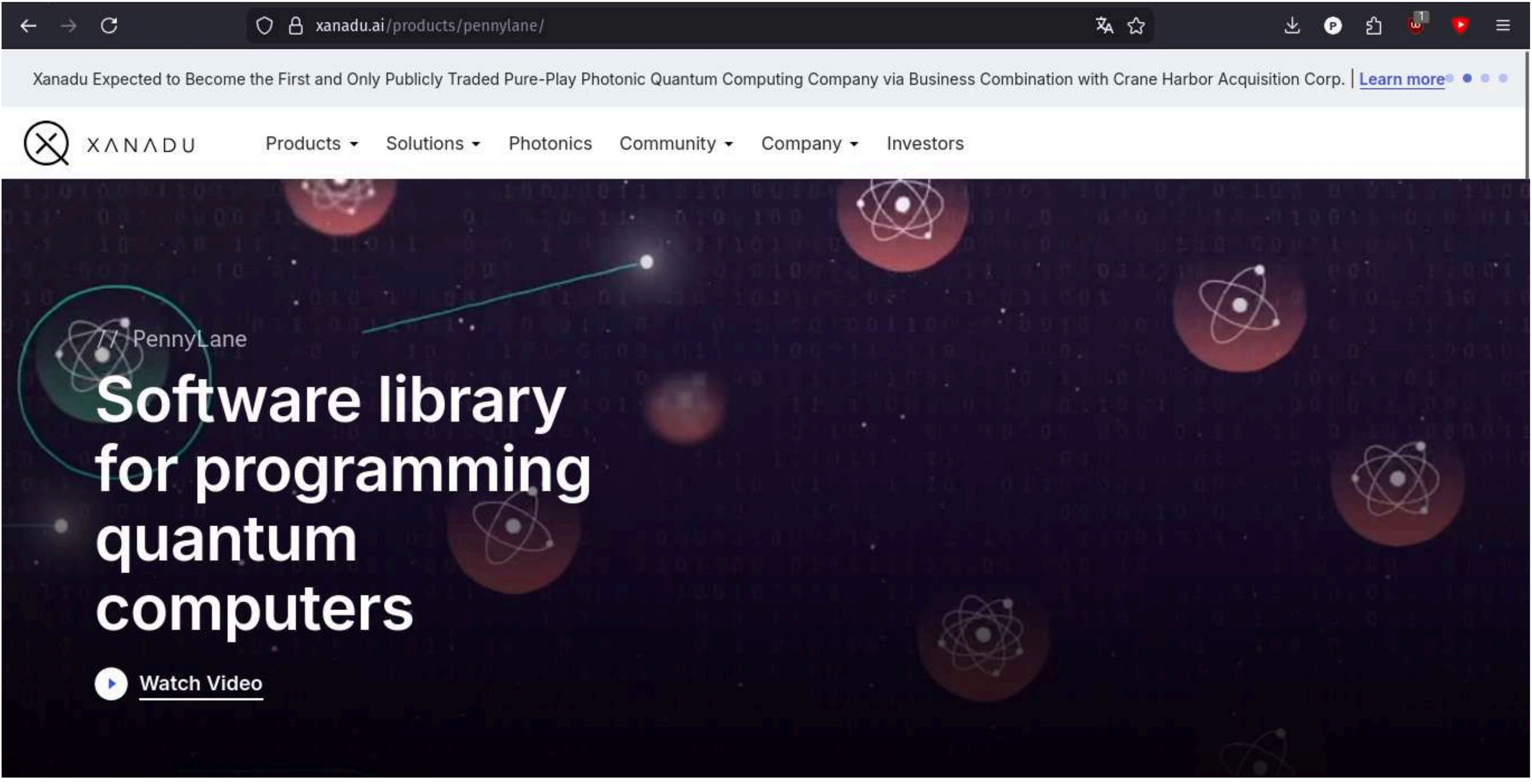
Introdução

Introdução ao Pennylane (Xanadu)

- Write Once, Run Anywhere:
  - Possibilidade de alterar simuladores e hardware mudando apenas uma única linha de código, sem afetar o restante do programa.
- Simuladores e hardware no mesmo ecossistema:
  - Acesso a simuladores de alta performance.
  - Transição simples de simulação para hardware real.






xanadu.ai/products/pennylane/
Xanadu Expected to Become the First and Only Publicly Traded Pure-Play Photonic Quantum Computing Company via Business Combination with Crane Harbor Acquisition Corp. | Learn more
XANADU
Products
Solutions
Photonics
Community
Company
Investors
// PennyLane
Software library for programming quantum computers
Watch Video

Instalação via VSCode

- Requisitos básicos:
  - Python 3.7 ou superior.
  - Ambiente virtual recomendado (venv ou conda).
- Na pasta do arquivo - é recomendado usar Jupyter notebook (.ipynb)- cria-se e ativa-se o ambiente virtual:

```
pedroraiol@pop-os:~$ python3 -m venv venv
pedroraiol@pop-os:~$ source venv/bin/activate
(venv) pedroraiol@pop-os:~$
```

- Instala-se o pennylane via pip install:

```
(venv) pedroraiol@pop-os:~$ pip install pennylane
Collecting pennylane
  Downloading pennylane-0.42.3-py3-none-any.whl (4.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.8/4.8 MB 1.2 MB/s eta 0:00:00
Collecting diastatic-malt
  Downloading diastatic malt-2.15.2-py3-none-any.whl (167 kB)
```

Instalação via VSCode

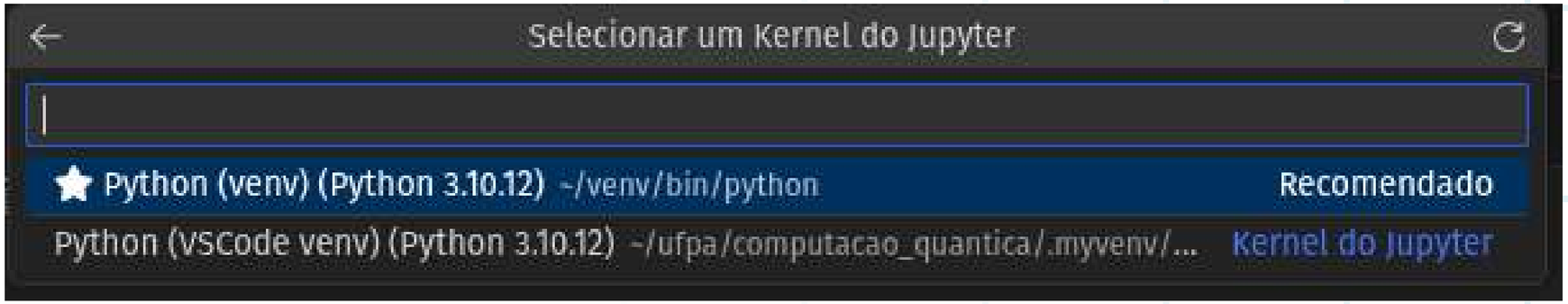
- Instala-se o jupyter dentro do venv via pip:

```
(venv) pedroraiol@pop-os:~$ pip install jupyter ipykernel
Collecting jupyter
  Downloading jupyter-1.1.1-py2.py3-none-any.whl (2.7 kB)
Collecting ipykernel
```

- Registrar o kernel do venv no jupyter:

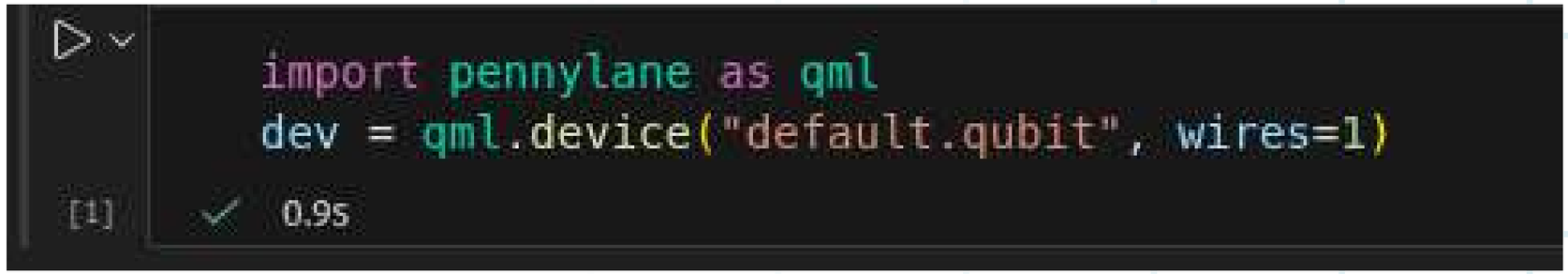
```
(venv) pedroraiol@pop-os:~$ python -m ipykernel install --user --name venv --display-name "Python (venv)"
Installed kernelspec venv in /home/pedroraiol/.local/share/jupyter/kernels/venv
```

- Seleciona-se o kernel Python (venv) criado no VSCode:



Importando o Pennylane

- Importa-se o pennylane no notebook jupyter e cria-se um simulador quântico de 1 qubit (wire) para testar se foi instalado corretamente:



- Instalação OK!

Hello World da CQ no pennylane (estados de Bell)

- Instala-se matplotlib no kernel criado:

```
import sys
!{sys.executable} -m pip install matplotlib
```

[2] ✓ 19.4s

  - Roda-se o seguinte código:

```python
import pennylane as qml
from pennylane import numpy as np
# criação do dispositivo quântico com 2 qubits
dev = qml.device("default.qubit", wires=2)

# Marca a função como QNode ligado ao dispositivo "dev"
# Um QNode é uma combinação de um circuito quântico e um dispositivo quântico que pode ser executado
@qml.qnode(dev)
def prepare_bell_state(index):
    qml.Hadamard(wires=0) # Hadamard no primeiro qubit cria superposição no |0⟩
    qml.CNOT(wires=[0, 1]) # CNOT entre o primeiro e o segundo qubit cria emaranhamento gerando o
estado |Φ+⟩
    if index == 0:
        pass  # |Φ+⟩ já está preparado
    elif index == 1:
        qml.PauliZ(wires=0)  # altera o estado para |Φ-⟩ aplicando Z no primeiro qubit
    elif index == 2:
        qml.PauliX(wires=1)  # altera o estado para |Ψ+⟩ aplicando X no segundo qubit
    elif index == 3:
        # altera o estado para |Ψ-⟩ aplicando X no segundo qubit e Z no primeiro qubit
        qml.PauliX(wires=1)
        qml.PauliZ(wires=0)
    # retorna o estado final dos qubits
    return qml.state()
# Loop para preparar e exibir os quatro estados de Bell
for i in range(4):
    bell_state = prepare_bell_state(i)
    # printa os estados de Bell
    print(f"Bell state |Φ_{i}⟩: {bell_state}")
    # desenhar o circuito quântico em ASCII e matplotlib
    print(qml.draw(prepare_bell_state)(i))
    qml.draw_mpl(prepare_bell_state)(i)
```

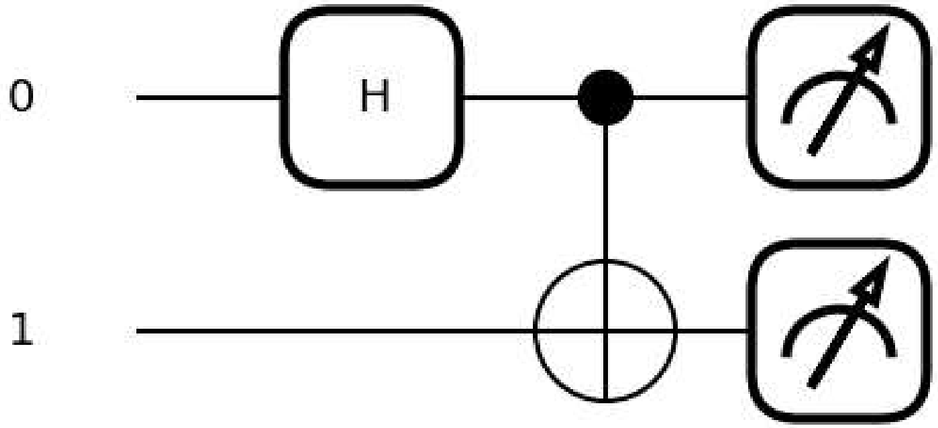
Hello World da CQ no pennylane

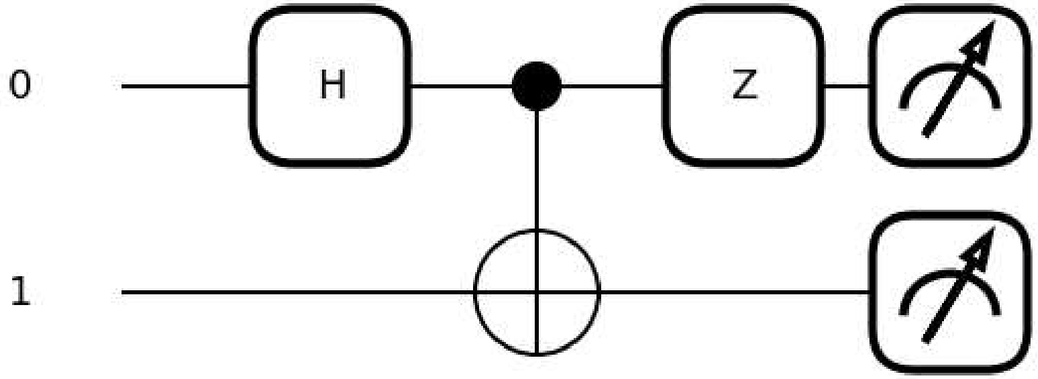
- Tem-se como saídas:

```
Bell state |Φ_0⟩: [0.70710678+0.j 0.        +0.j 0.        +0.j 0.70710678+0.j]
0: ─╭H─╭●─┤  State
1: ──────╰X─┤  State
Bell state |Φ_1⟩: [ 0.70710678+0.j  0.        +0.j -0.        +0.j -0.70710678+0.j]
0: ─╭H─╭●──Z─┤  State
1: ──────╰X────┤  State
Bell state |Φ_2⟩: [0.        +0.j 0.70710678+0.j 0.70710678+0.j 0.        +0.j]
0: ─╭H─╭●────┤  State
1: ──────╰X──X─┤  State
Bell state |Φ_3⟩: [ 0.        +0.j  0.70710678+0.j -0.70710678+0.j -0.        +0.j]
0: ─╭H─╭●──Z─┤  State
1: ──────╰X──X─┤  State
```

Hello World da CQ no pennylane

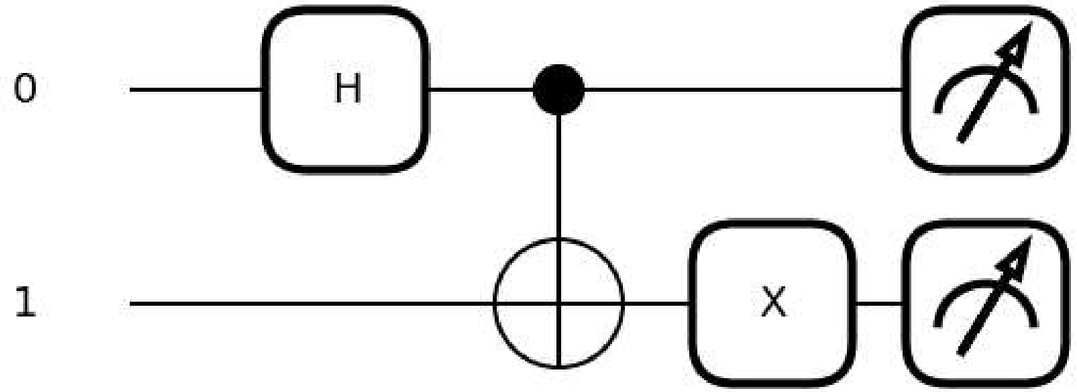
- Tem-se como saídas:

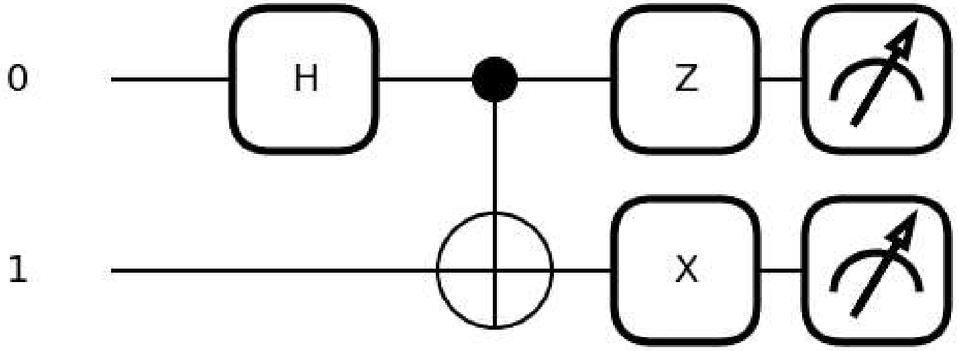




Hello World da CQ no pennylane

- Tem-se como saídas:





Comparação com outras bibliotecas

Biblioteca	Empresa	Foco
Pennylane	Xanadu	QML
Qiskit	IBM	Computação quântica em geral
QDK	Microsoft	Algoritmos Quânticos

- As bibliotecas se complementam.
- Pennylane foca em QML.
- Qiskit foca em hardware real.
- QDK foca em algoritmos e arquitetura.