Workshop: Introduction to Qiskit

First Version: 19-20-2021

Author: Luis Daniel Benavides Navarro

Qiskit

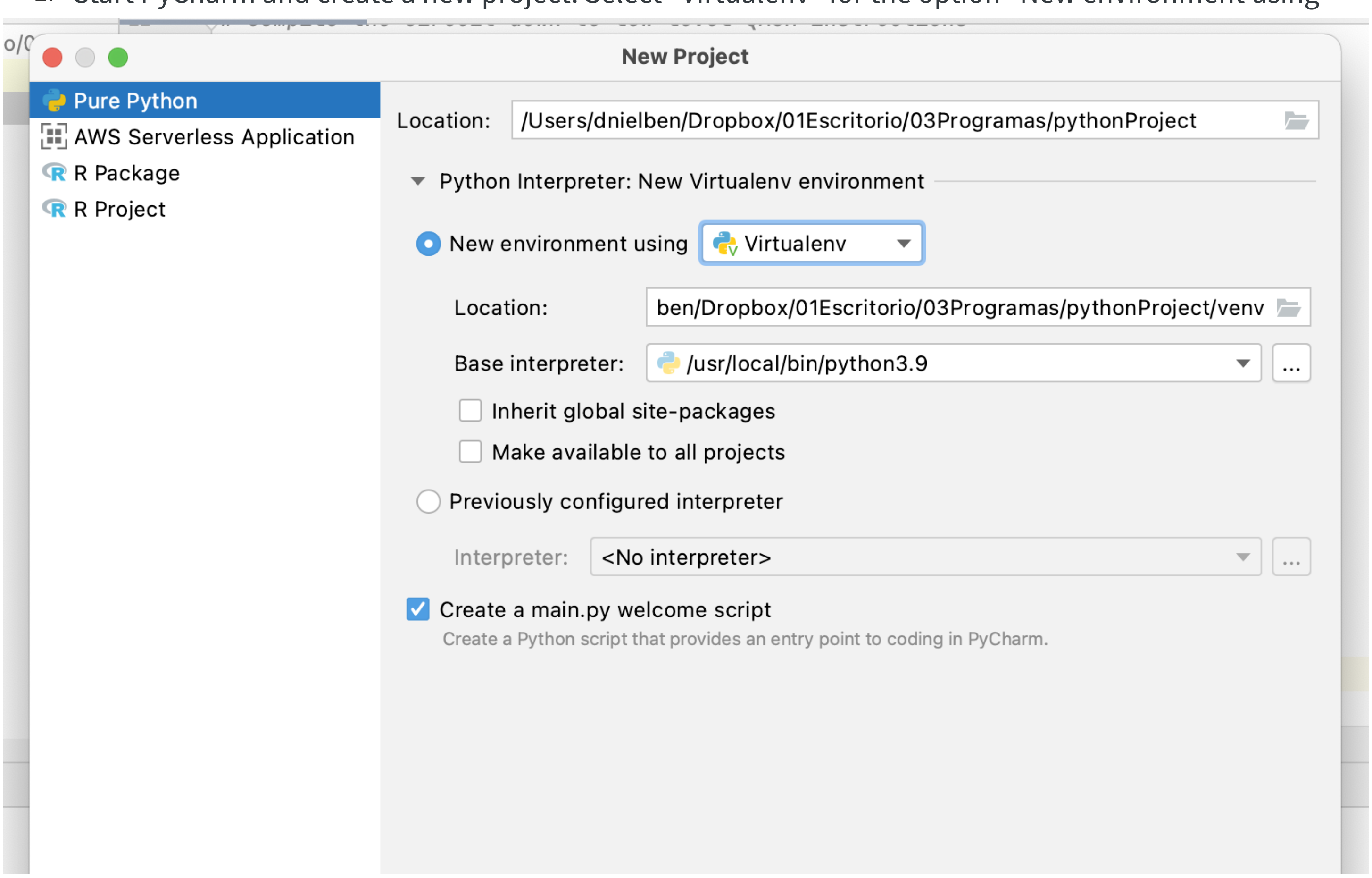
"Qiskit [kiss-kit] is an open source SDK for working with quantum computers at the level of pulses, circuits and application modules" (https://qiskit.org/). Programmers can write full local programs that can use IBM's quantum infrastructure. It allows, for example, to execute a quantum program in IBM's infrastructure from the local machine.

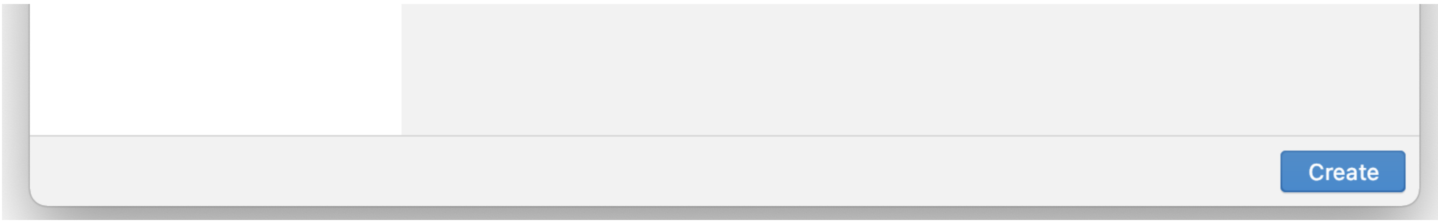
It combines development tools with an API to manipulate and use IBM's quantum infrastructure.

Installation

To execute this installation we recommend to install PyCharm. PyCharm provides a standard python environment supporting virtual environments natively. Even if you are an expert in python we recommend to use virtual environments to install the Quiskit.

1. Start PyCharm and create a new project. Select "Virtualenv" for the option "New environment using"





2. Install Qiskit

```
pip install qiskit
```

3. Install the visualization support for Qiskit

```
pip install qiskit[visualization]
```

4. If you are using zsh, for example in MacOS, you can write instead:

```
pip install 'qiskit[visualization]'
```

Try your first program

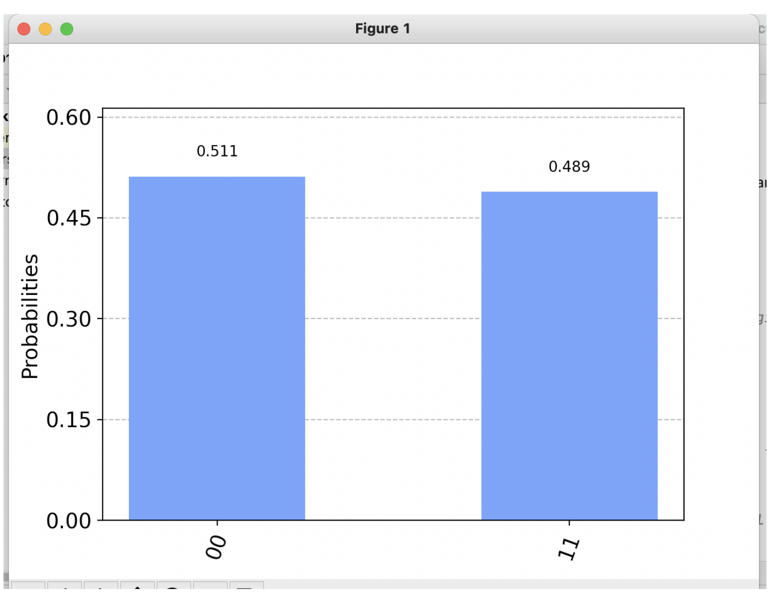
```
import numpy as np
from qiskit import QuantumCircuit, transpile
from qiskit import Aer
from qiskit.visualization import plot_histogram
import matplotlib.pyplot as plt

# Use Aer's qasm_simulator
simulator = Aer.get_backend('qasm_simulator')

# Create a Quantum Circuit acting on the q register
circuit = QuantumCircuit(2, 2)

# Add a H gate on qubit 0
circuit.h(0)

# Add a CX (CNOT) gate on control qubit 0 and target qubit 1
circuit.cx(0, 1)

# Map the quantum measurement to the classical bits
circuit.measure([0,1], [0,1])

# compile the circuit down to low-level QASM instructions
# supported by the backend (not needed for simple circuits)
compiled_circuit = transpile(circuit, simulator)
```

```
# Execute the circuit on the qasm simulator
job = simulator.run(compiled_circuit, shots=1000)

# Grab results from the job
result = job.result()

# Returns counts
counts = result.get_counts(circuit)
print("\nTotal count for 00 and 11 are:",counts)

# Draw the circuit
print(circuit)

# Plot a histogram
plot_histogram(counts)
plt.show()
```

You should see a text output as the following:

```
Total count for 00 and 11 are: {'00': 476, '11': 524}
     ┌───┐     ┌─┐   
q_0: ┤ H ├──■──┤M├───
     └───┘┌─┴─┐└╥┘┌─┐
q_1: ─────┤ X ├─╫─┤M├
          └───┘ ║ └╥┘
c: 2/═══════════╩══╩═
                0  1 

Process finished with exit code
```

You should also get the following window:



x= y=0.3687

Ejercicio

1. 1. Analyze and simulate the following circuits/states. Be careful with the output mapping so that your analysis results match those of the program.
    1. $|01\rangle$
    2. $|100\rangle$
    3. $(H \otimes I)|00\rangle$
2. Find the quantum circuits that implement the reversible gates Uf for the four functions f: {0,1} ⇢ {0,1}. Simulate them with programs.
3. 3. Implement Deutsch's algorithm and verify that it works to detect which of the functions in point 2 are balanced and constant.