# R42 Institute Fellowship

Quantum Computing - August 16, 2024

# Quantum Computing: Staff

## Fellows

**Chandra Govindarajan** (Baypoint, CA): Rising Junior at the University of California, Merced studying Computer Science and Engineering - Worked on Quantum Professor

**Arthur Van Meerbeeck** (Belgium): Computer Science Engineer, KULeuven - Worked on the Quantum Professor and algorithm

**William Colglazier** (Austin, TX): Data Engineering + Applied Math @ Texas A&M - Worked on Quantum Algorithm

**Varun Karlekar** (West Lafayette, IN): Senior at Purdue University - majoring in Computer Science - Worked on Quantum Professor

## Mentor

**Adrian Cable** (Montara, CA): Diffractive Optics (PhD, Cambridge), MIT Media Lab (novel electronics materials), 90+ patent applications, 100+ published papers, multiple 'deep tech' VC-funded start-ups & exits as CEO/CTO/founder

# Storyline

Quantum Computing (QC) is said to be the **next major advancement** following AI

But **progress is slow** due to insufficient education and available information

That’s why we developed a **large language model (LLM) acting as a Quantum professor**

We used this LLM to **create and run an algorithm on real quantum hardware** for a real life application

# Storyline

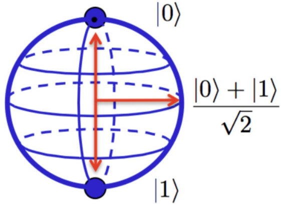
- Quantum Computing (QC) will be the **next major advancement** after AI, radically changing the kinds of problems computers can solve
- But **progress is slow** due limited up-to-date available public information, steep learning curve, and poor educational resources
- So, we developed a **large language model (LLM) ‘Quantum Professor’** to assist in education, research and code generation
- We used this LLM to become ‘QC experts’ in a few weeks, and **create and run an algorithm on real quantum hardware** for a real life application

# Content

1. What is Quantum Computing
2. How to accelerate quantum computing development
3. Quantum Professor, including a quick demo
4. Portfolio optimization using quantum computing
5. Future work

# Quantum computers use qubits to achieve exponential speed-up, driving new technological advancements

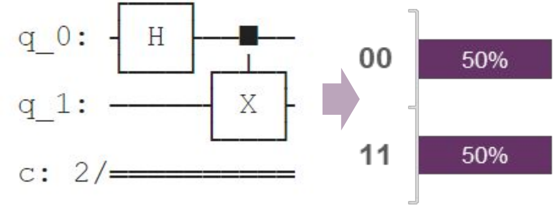
A quantum computer **uses qubits** instead of normal bits



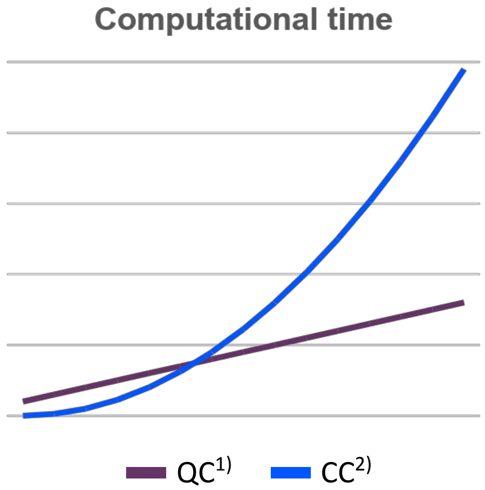
Qubits exhibit two key properties that distinguish them from classical bits**:**

- **Superposition**
- **Entanglement**

Quantum computers use **gates and circuits** to do operations on qubits



Resulting in **exponentially faster computational** capabilities



By leveraging superposition and entanglement **QC can process multiple possibilities simultaneously**, whereas classical computers (CC) process just one possibility at a time

Which can drive **new technological advancements**

- **Cryptography**: can break traditional encryption methods, or enable quantum encryption
- **Drug Discovery**: Simulating molecular structures to discover new drugs
- **Optimization Problems**: Improving logistics, traffic flow, resource management, etc.
- **Financial Modeling**: Better predictions and risk assessments in finance
- **Artificial Intelligence**: Enhancing ML algorithms and data processing capabilities

---

1) QC = Quantum Computer; 2) CC = Classical Computer

# Quantum computers leverage superposition and entanglement of qubits to achieve exponential speed-up

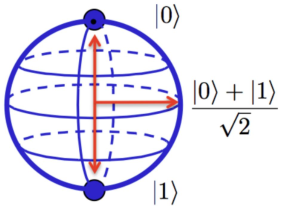
A quantum computer **uses qubits** instead of normal bits

**Bits:**

● 0

● 1

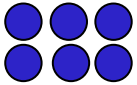
**Qubit:**



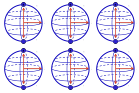
Qubits exhibit two key properties that distinguish them from classical bits:

- **Superposition**
- **Entanglement**

This allows QCs to **process many inputs simultaneously**

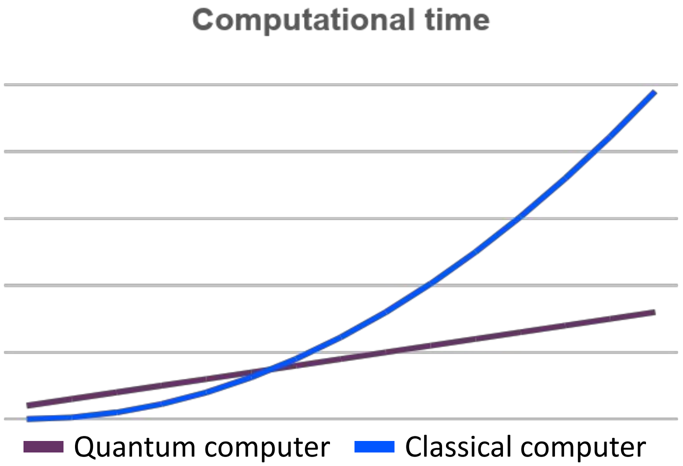


- n bits contain **n bits of information**



- n qubits contain **$2^n$ bits of information**

Resulting in **exponentially faster computational** capabilities



- **Quantum computers can process enormous datasets simultaneously**, classical computers must compute every step individually
- **Quantum computers are probabilistic**, classical computers are deterministic

# This exponential speed-up can drive technological advancements in various areas

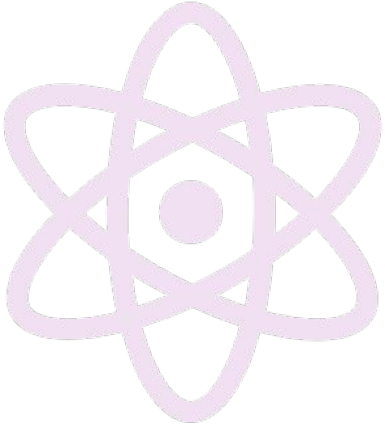
**Financial Modeling**
Predicting and assessing risk and returns in finance faster

**Cryptography**
Breaking traditional encryption methods, or enabling quantum encryption

**Drug Discovery**
Simulating molecular structures to discover new drugs efficiently

**Optimization Problems**
Improving logistics, traffic flow, resource mgmt., etc.

**Artificial Intelligence**
Enhancing ML algorithms and data processing capabilities



# Qiskit, IBM's quantum computing software, is used to develop and execute quantum algorithms

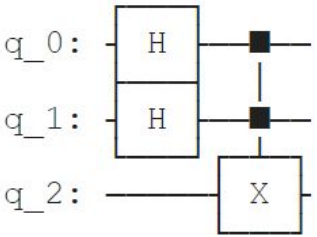
## Qiskit

An **open-source quantum computing framework** developed by IBM that enables users to create, manipulate, and execute quantum programs on simulators, and then on real quantum processors.

## Usage

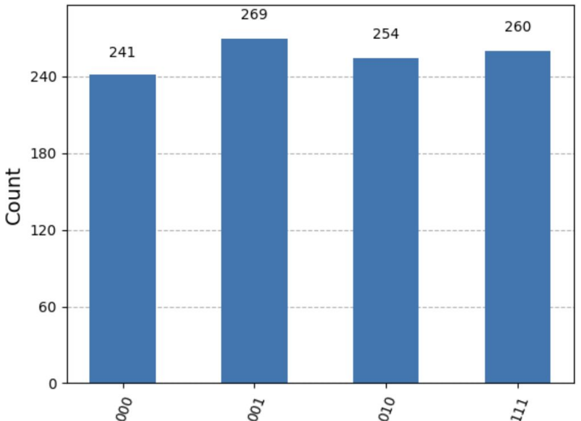
Used to create **quantum circuits**

```
circuit = QuantumCircuit(3,3)
circuit.h(0)  # hadamard
circuit.h(1)  # hadamard
circuit.ccx(0,1,2)
# controlled controlled not
```



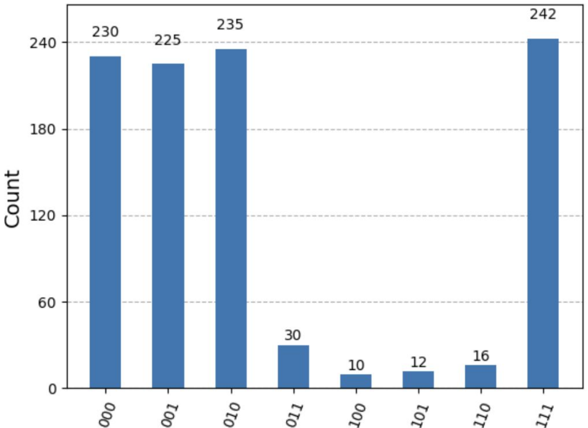
and to **simulate the results** of the circuit

```
backend = Aer.get_backend("qasm_simulator")
nc = transpile(circuit,backend)
job = backend.run(nc)
result = job.result()
```



Or **run it on a real quantum computer**

```
service = QiskitRuntimeService(
  channel="ibm_quantum",token=api_key)
backend = service.backend(
  name = "ibm_sherbrooke")
```



# The development of quantum computing faces several challenges, but solutions are already emerging

| Challenges | Solutions |
|---|---|
| **Limited resources and tutorials** on quantum coding | Train an **LLM with extensive quantum computing resources** - this is the challenge we've focused on |
| **Scaling up the number of qubits** while maintaining stability | Research and develop **new materials and architectures** and implement **modular quantum computing systems** |
| Quantum states are highly sensitive, causing **information loss and errors** | Implement **advanced error correction techniques** and develop more **robust qubit designs** |
| Quantum computers are **costly and resource-intensive** to build and maintain | Develop **cost-effective technologies and share resources** through cloud-based platforms |

# ChatGPT's limitations in generating quantum code inclined for a fine-tuned GPT-4o model with the latest QC resources

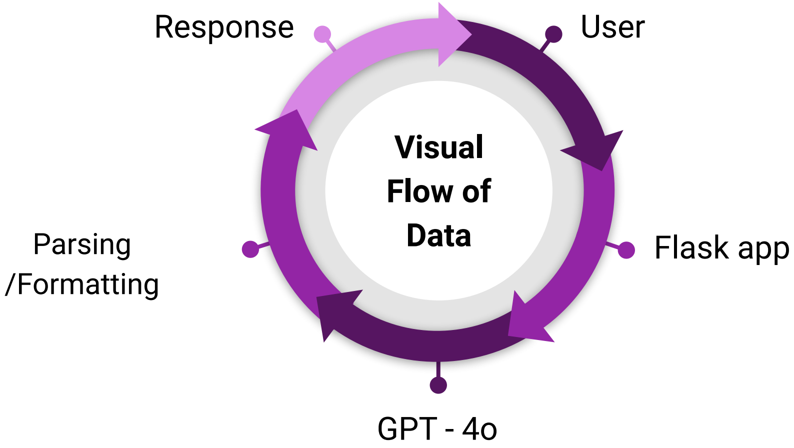
## Problems

- ChatGPT is trained on out-of-date material and generates poor code
- Model has a ‘neutral persona’

## Our Solution

- Fine-tuned GPT-4o model with up-to-date material including:
  - Latest IBM Qiskit API documentation
  - Relevant, current material on quantum computing R&D
- Quantum Professor persona

# Specifications



Visual Flow of Data

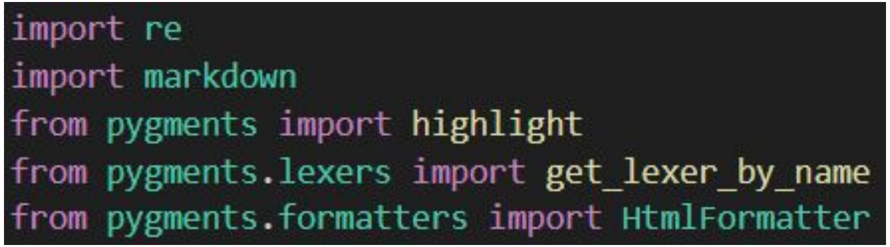
User

Flask app

GPT - 4o

Parsing /Formatting

Response



```
import re
import markdown
from pygments import highlight
from pygments.lexers import get_lexer_by_name
from pygments.formatters import HtmlFormatter
```

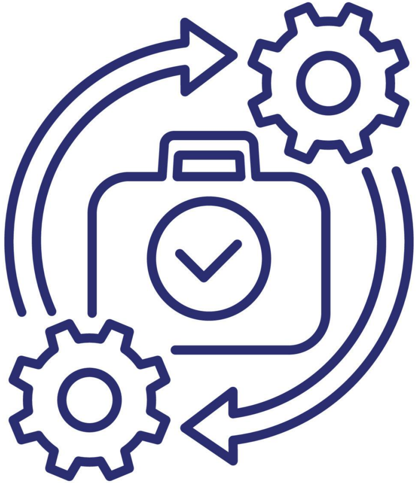
- OpenAI’s GPT-4o
- Leveraged Assistants API for a (quantum professor) persona - allows it to be fine-tuned
- Fine-tuned with:
  - Recent Quantum Computing Research
  - Up-to-date Qiskit documentation
- Flask used for the web app
  - Various Python Libraries for formatting

# Demo of the quantum professor

DEMO

# Quantum Portfolio Optimizer: what and why

- Portfolio optimization is essential.
- Rebalancing adjusts stocks based on new data.
- Quantum computing speeds up the process, so we can do it more often, with larger portfolios.
- Leads to higher returns, reduced risk, and tailored investments at faster rate.



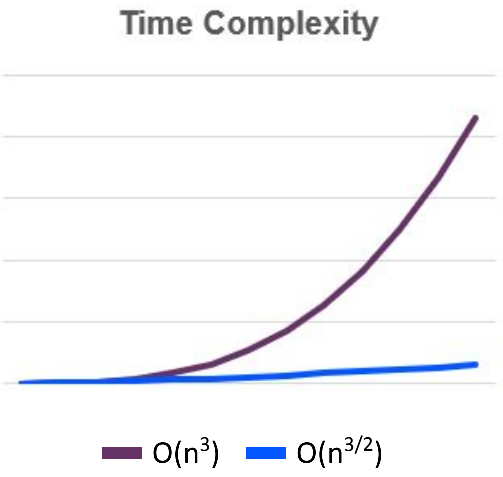
# Quantum computers have the potential to achieve quadratic speed-up in portfolio optimization algorithms

## Definition

**Process of selecting an optimal asset distribution**, out of a set of considered assets, according to some objective (E.g., maximizing return, minimizing risk, etc.)

## Time complexity

Most **classical** portfolio optimization algorithms have **$O(n^3)$** time complexity



**Quantum computers have 2 options** to solve portfolio optimization

**Variational algorithms**

- Heuristic algorithms (search algorithms that simplify the classical method to approximate the exact solution).
- Time improvement cannot be definitively proven

**Using Grover's search**

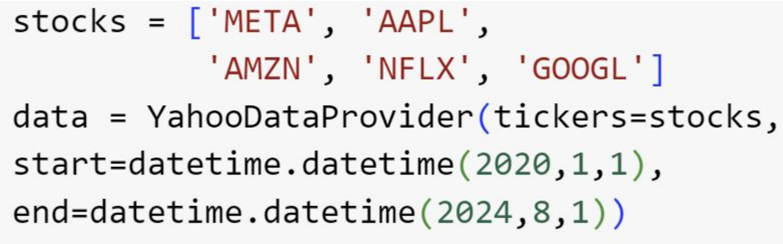
- Finds a specific item within an unsorted database
- Can achieve **quadratic speed-up $O(n^{3/2})$**

An **implementation of Grover's search for portfolio optimization does not exist yet**

# (pension) Funds can use the algorithm to rebalance and personalize their portfolios for optimal returns

## Data

- Data sourced from **Yahoo Finance**
- Selection from **all available stocks**
- **Customize the analysis period**
- **Daily updates available** for the latest data
- Demonstrate **toy model** using 5 stocks, so we can run on real quantum hardware today



```
stocks = ['META', 'AAPL',
          'AMZN', 'NFLX', 'GOOGL']
data = YahooDataProvider(tickers=stocks,
start=datetime.datetime(2020,1,1),
end=datetime.datetime(2024,8,1))
```

## Benefits

Because of the *advantages* of the quantum portfolio algorithm:

- **Quadratic time improvements**
- **Access to the most recent stock data**

(pension) funds can use this algorithm to :

- **Rebalance the portfolios** more frequently, ensuring the **highest returns**
- Provide **highly personalized portfolios**

# The quantum professor generates code for the portfolio optimization algorithm with specific parameters

## Specifications

**Stocks:**
META, AAPL, AMZN, NFLX, GOOGL

**Timeline:**
From 2020 until Now

**Budget:**
2 Stocks

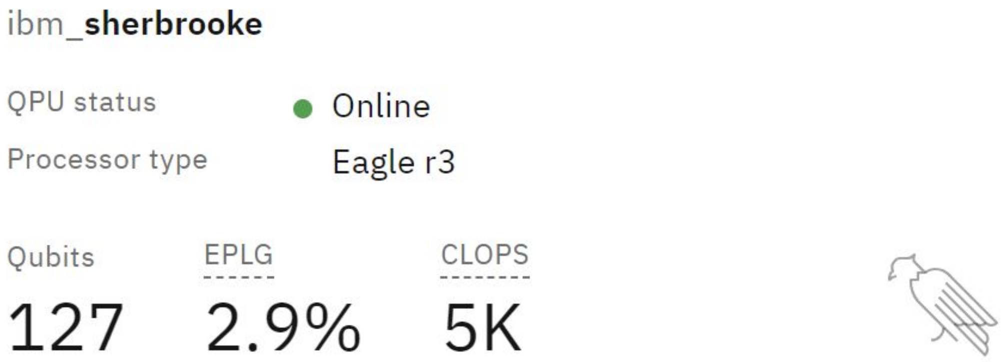
**Risk Appetite:**
0.7

**Goal:**
Selection of the **best 2 stocks that maximize returns** while **adhering to the specified risk parameter**

## Code generation using the Quantum Professor

- **Generate functional Qiskit code** for the algorithm - live demo!
- **Compute the result with quantum hardware** provided by IBM

## Compute resource:



ibm_**sherbrooke**

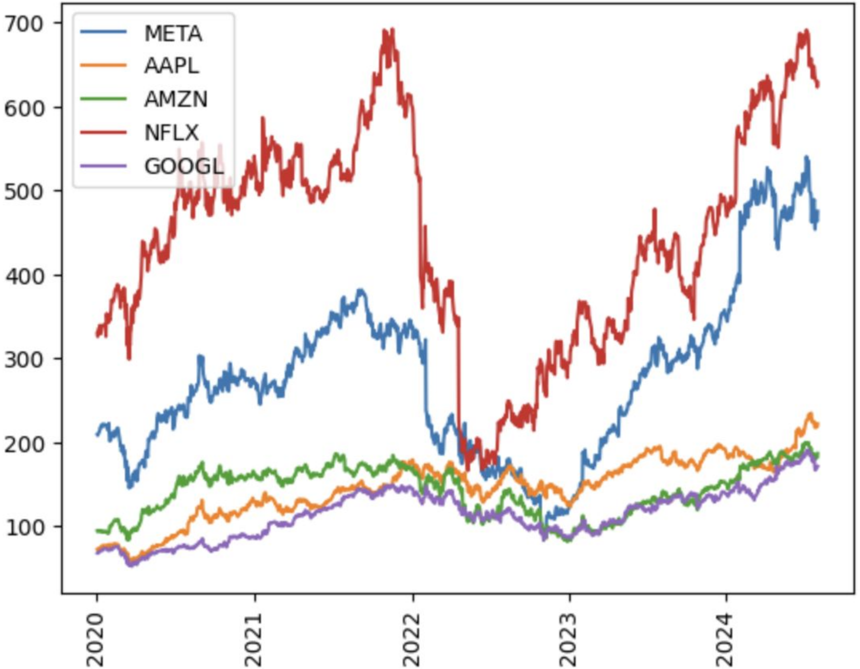
QPU status ● Online

Processor type Eagle r3

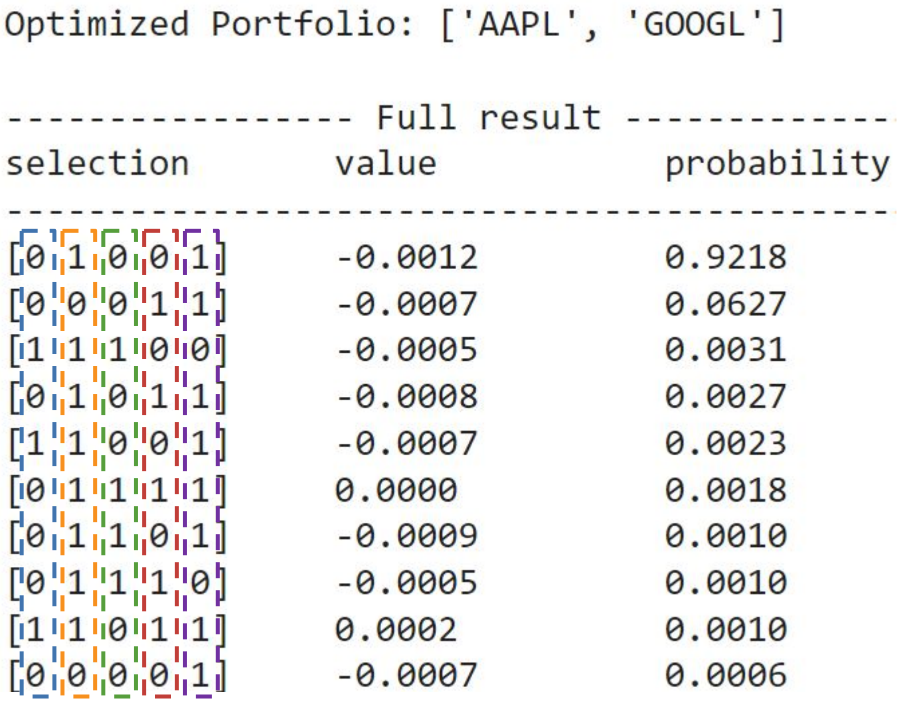
| Qubits | EPLG | CLOPS |
|---|---|---|
| 127 | 2.9% | 5K |

# AAPL and GOOGL form the optimal pair, with a 92% probability of selection on a real quantum computer

## Data



## Solution



```
Optimized Portfolio: ['AAPL', 'GOOGL']


------------------ Full result -------------
selection          value            probability
-----------------------------------------------
[0 1 0 0 1]        -0.0012          0.9218
[0 0 0 1 1]        -0.0007          0.0627
[1 1 1 0 0]        -0.0005          0.0031
[0 1 0 1 1]        -0.0008          0.0027
[1 1 0 0 1]        -0.0007          0.0023
[0 1 1 1 1]        0.0000           0.0018
[0 1 1 0 1]        -0.0009          0.0010
[0 1 1 1 0]        -0.0005          0.0010
[1 1 0 1 1]        0.0002           0.0010
[0 0 0 0 1]        -0.0007          0.0006
```

# What we're working on next

- Quantum Professor will be available to the public in the next couple of weeks (link will be sent to all R42 program attendees)
- Modelling on portfolio optimization: quantify how continuous optimization and individualized portfolios, made possible by quantum computing, lead to improved returns / lower risk
- Leverage Quantum Professor to generate algorithms/code for other important problems that are computationally hard on classical computers (cryptography, search, etc.)

# Contact Us!

## Quantum Computing

**Chandra Govindarajan**:
Email: cgovindarajan19@gmail.com
LinkedIn: https://www.linkedin.com/in/chandragovind/

**Varun Karlekar**:
Email: varun.m.karlekar@gmail.com
LinkedIn: https://www.linkedin.com/in/varun-manish-karlekar/

**William Colglazier**:
Email: williamcolglazier@tamu.edu
LinkedIn: https://www.linkedin.com/in/william-colglazier-9b70bb27a

**Arthur Van Meerbeeck**:
Email: arthurvanmeerbeeck@gmail.com
LinkedIn: https://www.linkedin.com/in/arthur-van-meerbeeck-05052b171/

**Adrian Cable**:
Email: adrian.cable@gmail.com
LinkedIn: https://www.linkedin.com/in/adrian-cable-91730221/

# Questions

# The development of quantum computing faces several challenges, but solutions are already emerging

| Challenges | Solutions |
|---|---|
| **Quantum Coding and Information Availability**:<br>Limited availability of comprehensive resources and documentation especially on quantum coding | **Train an LLM with extensive quantum computing resources** to make quantum information more accessible and improve quantum coding education |
| **Scalability:**<br>Scaling up the number of qubits while maintaining stability and entanglement | Research and develop **new materials and architectures** and implement modular quantum computing systems |
| **Quantum Decoherence:**<br>Quantum states are highly sensitive to their environment, leading to loss of information and errors | Implement **advanced error correction techniques** and develop more **robust qubit designs** |
| **High Cost and Resource Demand:**<br>Building and maintaining quantum computers is extremely expensive and resource-intensive | Develop **cost-effective technologies and share resources** through cloud-based quantum computing platforms |

# The quantum professor generates code for the portfolio optimization algorithm with specific parameters

## Specifications

**Stocks:**
META, AAPL, AMZN, NFLX, GOOGL

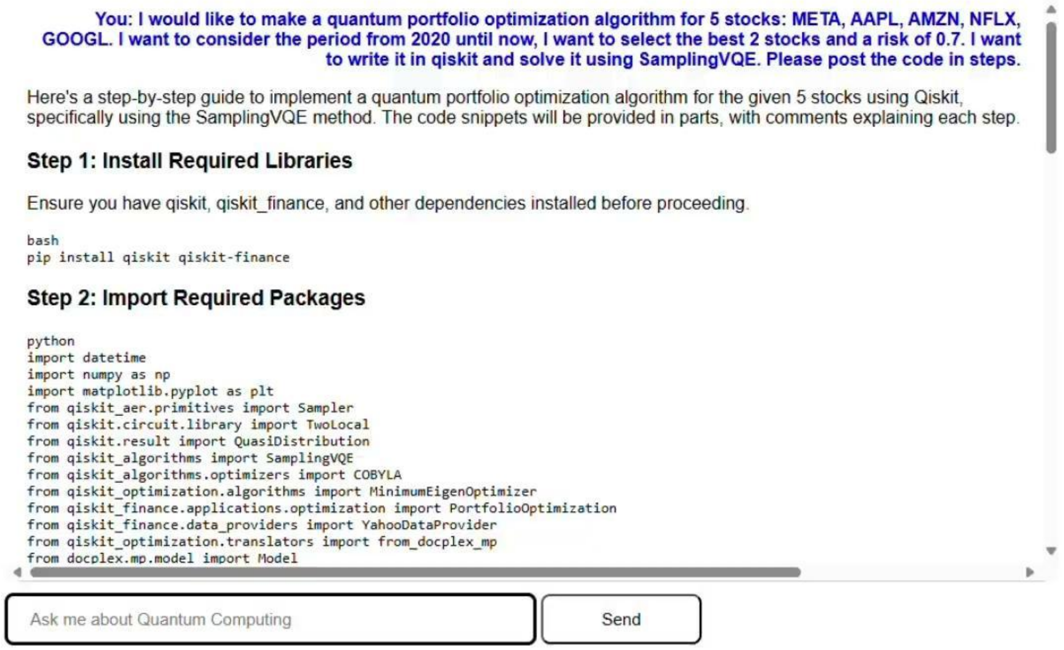
**Timeline:**
From 2020 until Now

**Budget:**
2 Stocks

**Risk Appetite:**
0.7

**Goal:**
Selection of the **best 2 stocks that maximize returns** while **adhering to the specified risk parameter**

## Code generation using the Quantum Professor



You: I would like to make a quantum portfolio optimization algorithm for 5 stocks: META, AAPL, AMZN, NFLX, GOOGL. I want to consider the period from 2020 until now, I want to select the best 2 stocks and a risk of 0.7. I want to write it in qiskit and solve it using SamplingVQE. Please post the code in steps.

Here's a step-by-step guide to implement a quantum portfolio optimization algorithm for the given 5 stocks using Qiskit, specifically using the SamplingVQE method. The code snippets will be provided in parts, with comments explaining each step.

**Step 1: Install Required Libraries**

Ensure you have qiskit, qiskit_finance, and other dependencies installed before proceeding.

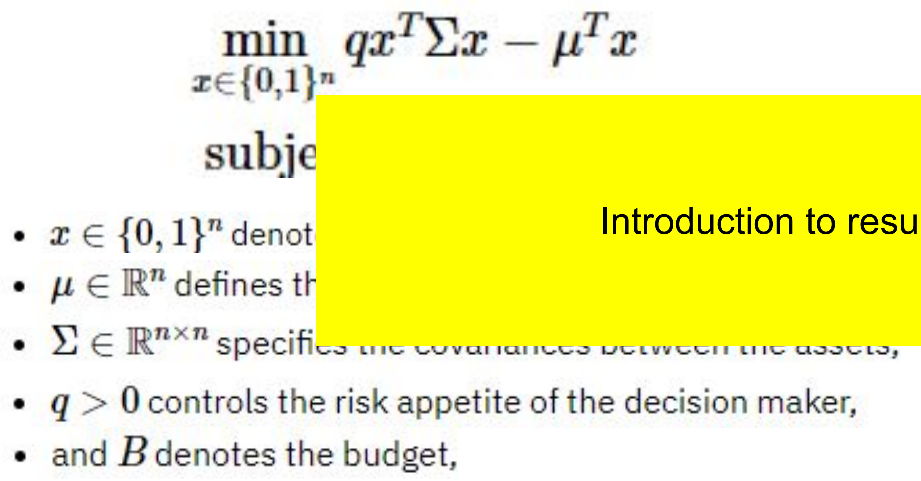
```
bash
pip install qiskit qiskit-finance
```

**Step 2: Import Required Packages**

```
python
import datetime
import numpy as np
import matplotlib.pyplot as plt
from qiskit_aer.primitives import Sampler
from qiskit.circuit.library import TwoLocal
from qiskit.result import QuasiDistribution
from qiskit_algorithms import SamplingVQE
from qiskit_algorithms.optimizers import COBYLA
from qiskit_optimization.algorithms import MinimumEigenOptimizer
from qiskit_finance.applications.optimization import PortfolioOptimization
from qiskit_finance.data_providers import YahooDataProvider
from qiskit_optimization.translators import from_docplex_mp
from docplex.mp.model import Model
```

# Modern Portfolio Theory (MPT), evaluates assets based on risk and return, making it suitable for quantum computers

## Problem statement



$$\min_{x\in\{0,1\}^n} \; q x^T \Sigma x - \mu^T x$$

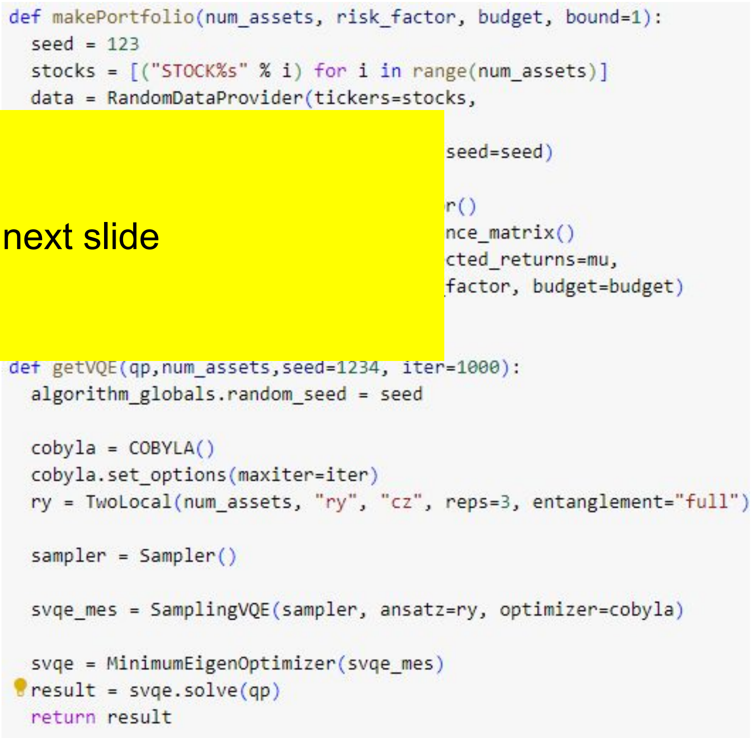
subje

- $x \in \{0,1\}^n$ denot
- $\mu \in \mathbb{R}^n$ defines th
- $\Sigma \in \mathbb{R}^{n \times n}$ specifies the covariances between the assets,
- $q > 0$ controls the risk appetite of the decision maker,
- and $B$ denotes the budget,

Introduction to results of next slide

The resulting **problem can be mapped to a Hamiltonian**[1] whose ground state corresponds to the optimal solution

## Quantum Code



```
def makePortfolio(num_assets, risk_factor, budget, bound=1):
  seed = 123
  stocks = [("STOCK%s" % i) for i in range(num_assets)]
  data = RandomDataProvider(tickers=stocks,

                                                       seed=seed)

                                                       r()
                                                       nce_matrix()
                                                       cted_returns=mu,
                                                       factor, budget=budget)

def getVQE(qp,num_assets,seed=1234, iter=1000):
  algorithm_globals.random_seed = seed

  cobyla = COBYLA()
  cobyla.set_options(maxiter=iter)
  ry = TwoLocal(num_assets, "ry", "cz", reps=3, entanglement="full")

  sampler = Sampler()

  svqe_mes = SamplingVQE(sampler, ansatz=ry, optimizer=cobyla)

  svqe = MinimumEigenOptimizer(svqe_mes)
  result = svqe.solve(qp)
  return result
```

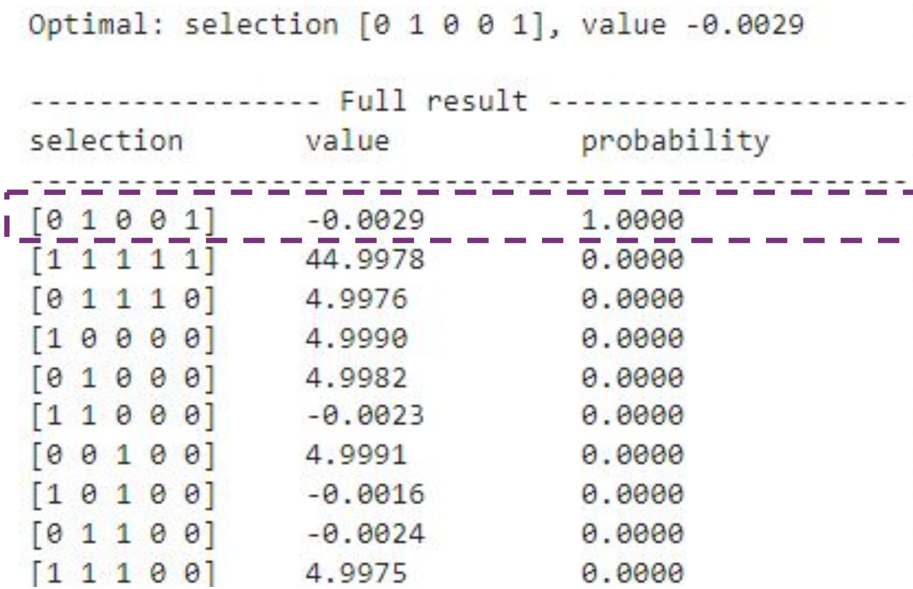
---

[1] In quantum mechanics, the Hamiltonian of a system is an operator corresponding to the total energy of that system

# The VQE has over a 96% probability of selecting the optimal portfolio from 5 assets with a budget of 2

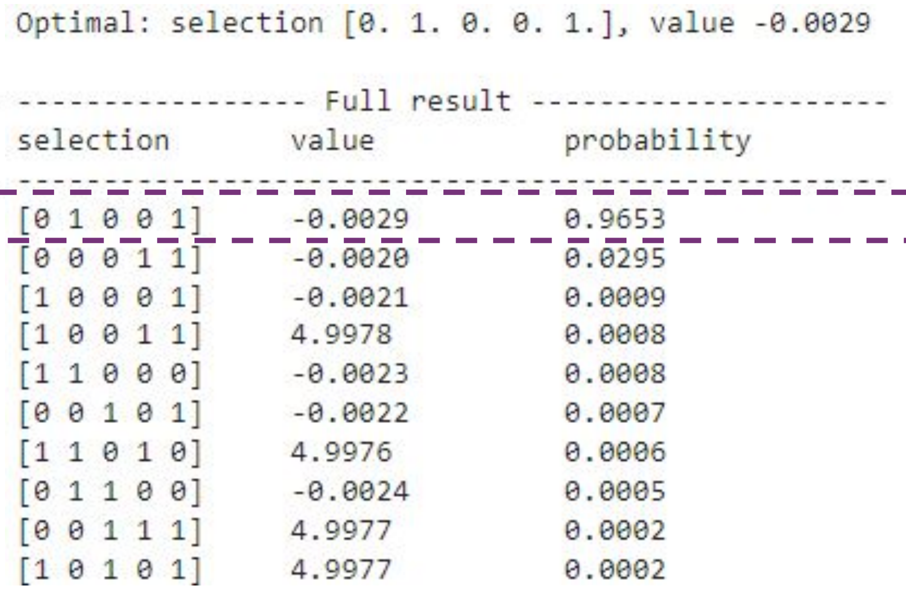
## Exact solution

## Variational Quantum Eigensolver (VQE)



```
Optimal: selection [0 1 0 0 1], value -0.0029

---------------- Full result --------------------
selection          value          probability
-------------------------------------------------
[0 1 0 0 1]      -0.0029          1.0000
[1 1 1 1 1]      44.9978          0.0000
[0 1 1 1 0]      4.9976           0.0000
[1 0 0 0 0]      4.9990           0.0000
[0 1 0 0 0]      4.9982           0.0000
[1 1 0 0 0]      -0.0023          0.0000
[0 0 1 0 0]      4.9991           0.0000
[1 0 1 0 0]      -0.0016          0.0000
[0 1 1 0 0]      -0.0024          0.0000
[1 1 1 0 0]      4.9975           0.0000
```



```
Optimal: selection [0. 1. 0. 0. 1.], value -0.0029

---------------- Full result --------------------
selection          value          probability
-------------------------------------------------
[0 1 0 0 1]      -0.0029          0.9653
[0 0 0 1 1]      -0.0020          0.0295
[1 0 0 0 1]      -0.0021          0.0009
[1 0 0 1 1]      4.9978           0.0008
[1 1 0 0 0]      -0.0023          0.0008
[0 0 1 0 1]      -0.0022          0.0007
[1 1 0 1 0]      4.9976           0.0006
[0 1 1 0 0]      -0.0024          0.0005
[0 0 1 1 1]      4.9977           0.0002
[1 0 1 0 1]      4.9977           0.0002
```

# Why?

### Why

- Wanted to make chatbot LLM that would give us information about Quantum Computing as well as generate code in Qiskit

### Which model
### Which finetuning data
### Other specifications

- OpenAI GPT 4o.
- Assistant API
- Finetuned with various PDFs on quantum computing, QC algorithms, and Qiskit code

### Improvements vs chatgpt

- More detailed vs ChatGPT
- Generates working Code in qiskit

### Some pics of GUI

- Will add at the end

# Limitations of LLMs on quantum coding

## Limitations | Severity

### 1. Library Version:

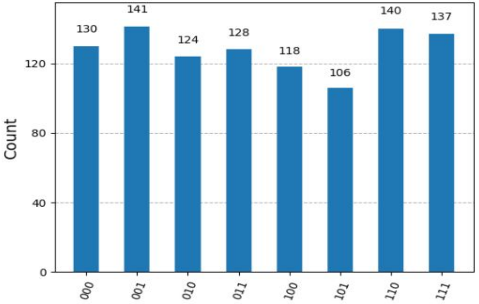
◔

**Limitation:** ChatGPT generates code for an older library version which no longer works on a modern set-up
**Solution:** Update the imports, as well as some functions (E.g., execute()) to run the simulation

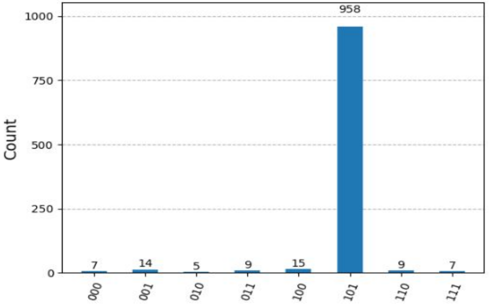
### 2. Output when simulating Grover's algorithm for 3 qubits with solution 101[1]:

●

**Limitation:**



**Solution:**



- ChatGPT cannot simulate a basic QC algorithm, requiring us to independently search and correct errors
- This raises concerns about its suitability for the project's 2nd phase, where we aim to develop an algorithm ourselves

### 3. Available coding resources to strengthen the LLM's coding capabilities:

◕

**Limitation:** There is limited amount of data and code examples available to train the model
**Solution:** Lowering the barrier of entry for developers that are not familiar with the field

- IBM already provides good coding tutorials
- Train LLM for quantum coding: better code fuels a better model in a virtuous cycle

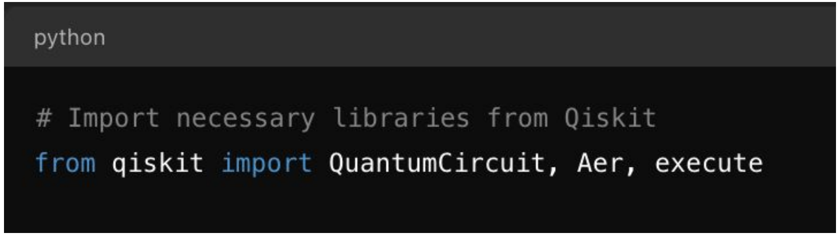
● High ○ Low

---

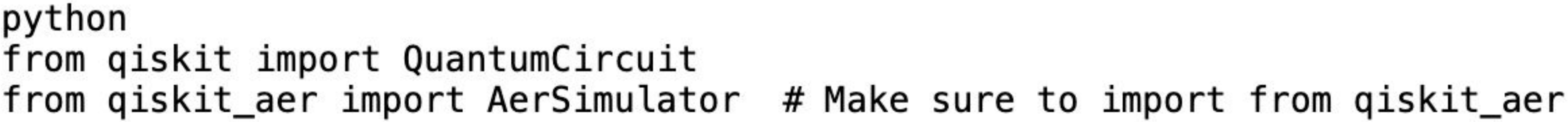
1) This setup is used as an example; the code does not work for other configurations either

# Why?

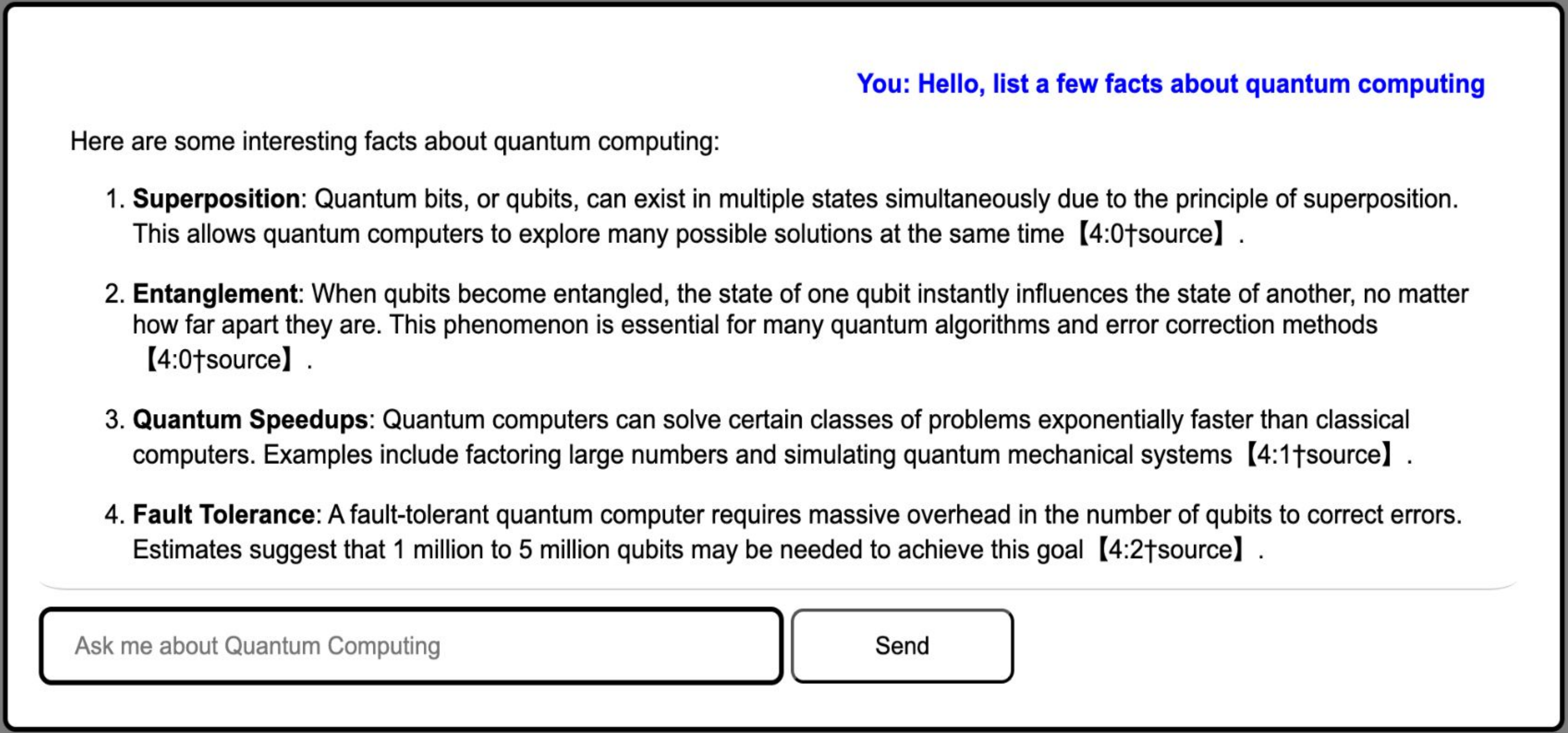
- Wanted an easy way to access information about quantum computing as well as a way to generate code in Qiskit
- ChatGPT knowledge base wasn't always up to date.
- Didn't generate working qiskit code
- The Quantum Professor LLM provides with detailed information about Quantum Computing that is given in an easy to understand manner



```python
# Import necessary libraries from Qiskit
from qiskit import QuantumCircuit, Aer, execute
```



```
python
from qiskit import QuantumCircuit
from qiskit_aer import AerSimulator  # Make sure to import from qiskit_aer
```



**You: Hello, list a few facts about quantum computing**

Here are some interesting facts about quantum computing:

1. **Superposition**: Quantum bits, or qubits, can exist in multiple states simultaneously due to the principle of superposition. This allows quantum computers to explore many possible solutions at the same time【4:0†source】.
2. **Entanglement**: When qubits become entangled, the state of one qubit instantly influences the state of another, no matter how far apart they are. This phenomenon is essential for many quantum algorithms and error correction methods【4:0†source】.
3. **Quantum Speedups**: Quantum computers can solve certain classes of problems exponentially faster than classical computers. Examples include factoring large numbers and simulating quantum mechanical systems【4:1†source】.
4. **Fault Tolerance**: A fault-tolerant quantum computer requires massive overhead in the number of qubits to correct errors. Estimates suggest that 1 million to 5 million qubits may be needed to achieve this goal【4:2†source】.