

Assignment 02: From bits to qubits

Objectives

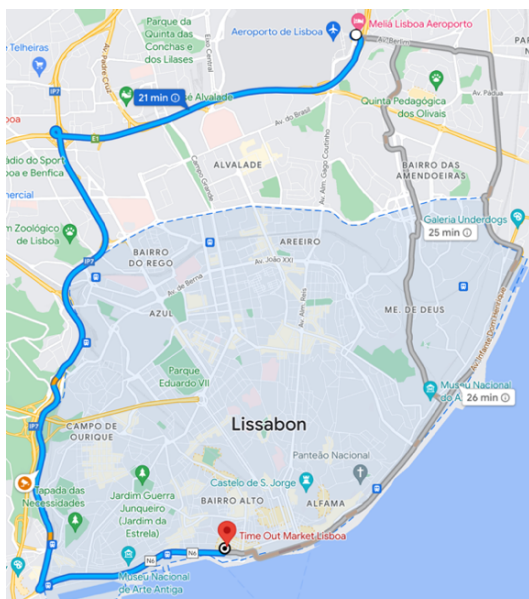
- You will learn why Quantum Computers are faster than Classical Computers
- You will understand the basic operations of the binary system
- You will be introduced to the differences between bits and qubits
- You will get to know the principles of programming in Python

Requirements

- Notebook or Desktop Computer with access to the internet
- Access to IBM Quantum Lab
- Optional: Calculator

Solution Steps

After you have provided initial insights in the first assignment, the client is interested in gaining more information about the navigation case. He would like to learn more about the differences between Classical and Quantum Computers and why Quantum Computers are able to faster identify optimal routes (lowest traffic + shortest route = lowest CO2 emissions).



To make the way how both types of computers solve this kind of optimization problem more tangible, think about a weekend trip to Lisbon.

Route Optimization

You and your friend Taylor arrive at the airport and rent a car. As you are hungry you want to visit the Time Out Market before going to the Airbnb. Your navigation system suggests several potential routes. How was the computer able to make those suggestions?

The task of route optimization can be simplified as follows: Let us say there are N potential routes to get from the airport to the Time Out Market and you want to select the one with the lowest CO2 emissions.

Option 1: You can try out all of the potential routes, record the time needed, calculate the CO2 emissions and rank the routes. This process will take N hours plus the time to record your results.

Option 2: The public knows which routes are currently busy and which routes have low traffic and hence lead to lower CO2 emissions. You can ask the public to vote for the best route and select the route with the highest votes. This process is likely to take about \sqrt{N} hours plus post-voting time.

Option 2 uses a **community interaction** and eliminates the need for one-to-one interactions, and thereby, **reduces the required time** from an order of N to an order of \sqrt{N} . This time reduction is significant for large N .

The bits of a **Classical Computer** behave in a deterministic and individualistic way. There is no “public of bits” knowing about other bits. As a result, sorting and other operations in a Classical Computer have to rely on something similar to **option 1**.

In contrast, the qubits of a **Quantum Computer** behave in a probabilistic and interacting way. Due to wave nature of qubits, **all qubits know about all other qubits**, and thus, there is a “**public of qubits**” **knowing about each other**. As a result, sorting and other operations in a Quantum Computer can rely on something similar to **option 2**.

Thus, the “community spirit” of qubits of a Quantum Computer reduces the time of sorting and other operations from an order of N to an order of \sqrt{N} and hence enables us to identify the optimal route more efficiently.

Before being able to explain our client how a Quantum Computer works, we first need to get familiar with the binary system as this is the foundation of classical computation.

1. The binary system – 0s and 1s

The first thing we need to know about the binary system is the idea of bits. Bits are the smallest amount of information a computer is able to process. They are supposed to be the simplest alphabet in the world. The so-called “binary code” states that 0s and 1s are the foundational language of computers. Those two numbers can be used to represent any information and can be imagined as on/off switch of a transistor [1].

```
01001100 01100101 01110100 00100111 01110011 00100000 01100110 01101001 01100111 01101000
01110100 00100000 01100011 01101100 01101001 01101101 01100001 01110100 01100101 00100000
01100011 01101000 01100001 01101110 01100111 01100101 00100000 01110111 01101001 01110100
01101000 00100000 01010001 01110101 01100001 01101110 01110100 01110101 01101101 00100000
01000011 01101111 01101101 01110000 01110101 01110100 01101001 01101110 01100111
```

The string¹ of bits might not look like much information to you, but this binary code spells out “Let’s fight climate change with Quantum Computing”.

The **binary number system** uses the number two to create any other number with a combination of 0s and 1s.

As eight bits equal one byte, we can express numbers until 255 by using the number 2 with eight exponents ranging from 0 to 7. Important to note is that we read strings of bits from right to left. [2]

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
↓	↓	↓	↓	↓	↓	↓	↓
128	64	32	16	8	4	2	1

¹ In computer science sequences of characters are called strings

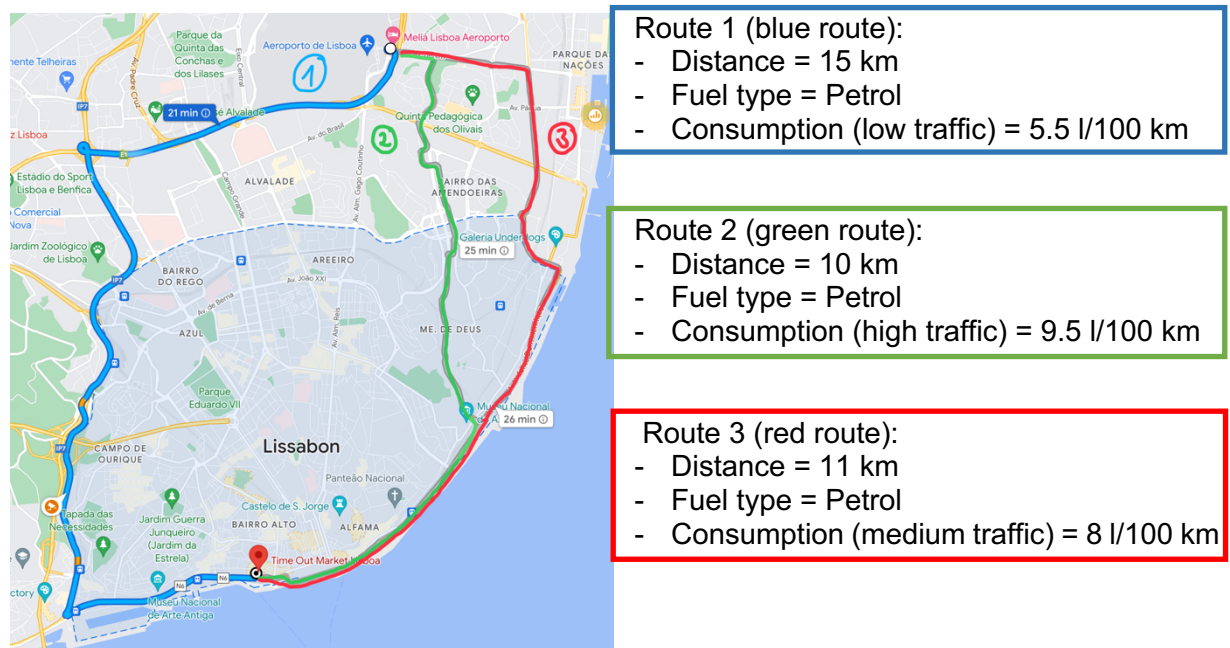
For getting a better understanding about the binary number system please get familiar with the following examples:

Decimal (Input)	Transition	Binary (Output)
128	$(1 * 2^7) + (0 * 2^6) + (0 * 2^5) + (0 * 2^4) + (0 * 2^3) + (0 * 2^2) + (0 * 2^1) + (0 * 2^0)$	10000000
1	$(0 * 2^7) + (0 * 2^6) + (0 * 2^5) + (0 * 2^4) + (0 * 2^3) + (0 * 2^2) + (0 * 2^1) + (1 * 2^0)$	00000001
213	$(1 * 2^7) + (1 * 2^6) + (0 * 2^5) + (0 * 2^4) + (0 * 2^3) + (1 * 2^2) + (0 * 2^1) + (1 * 2^0)$	11000101
255	$(1 * 2^7) + (1 * 2^6) + (1 * 2^5) + (1 * 2^4) + (1 * 2^3) + (1 * 2^2) + (1 * 2^1) + (1 * 2^0)$	11111111

Now it's your turn!

Task 1: Calculating CO2 savings with route optimization (12 min.)

1. Calculate the CO2 emissions for all of the suggested routes with the following tool:
https://co2.myclimate.org/en/car_calculators/new



2. Calculate the CO2 savings in kilogram when comparing the routes with the highest and lowest CO2 emissions.

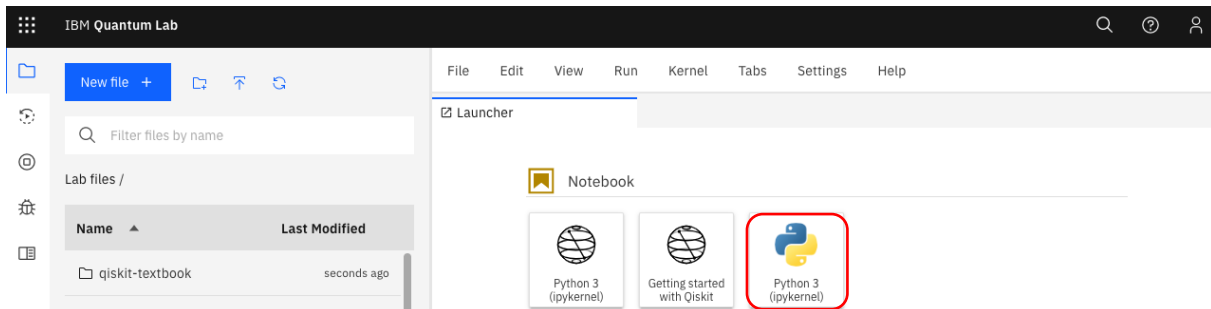
3. Imagine 163 people use the same navigation system per day and choose the route with the lowest CO2 emissions. Calculate the CO2 savings we achieved through route optimization per day in kilograms.

4. Please write your result by using the same logic as depicted in the table above.

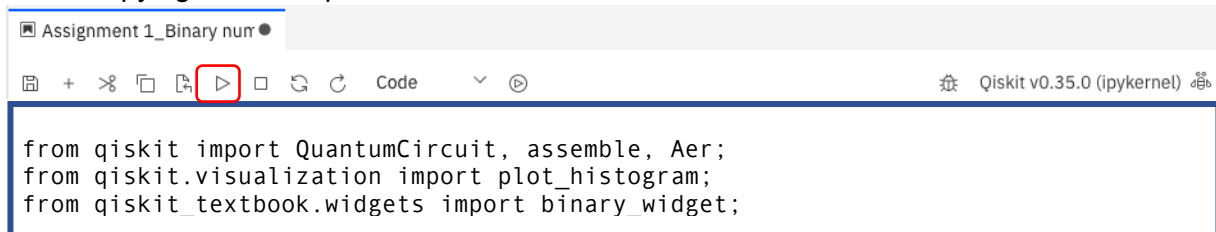
Decimal (Input)	Transition	Binary (Output)
_____	$(_ * 2^7) + (_ * 2^6) + (_ * 2^5) + (_ * 2^4) + (_ * 2^3) + (_ * 2^2) + (_ * 2^1) + (_ * 2^0)$	_____

5. To check whether your result is right, please go to IBM Quantum Lab (<https://lab.quantum-computing.ibm.com>) and follow the next steps:

5.1) Open a new Python 3 (ipykernel) Notebook.



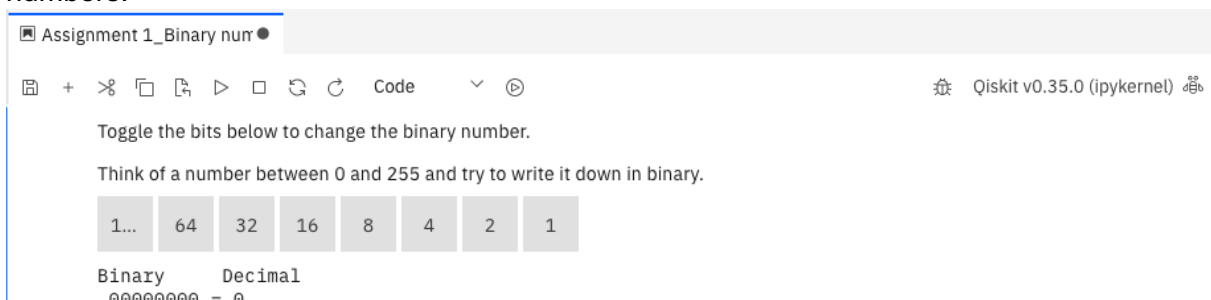
5.2) Please copy and paste the following code to access the required toolboxes. You can execute the code by pressing “shift + enter” or the “Run” button. When copying the code, please make sure that each line is below the other as in the code box.



5.3) Press run or “shift + enter” to execute the following code.



5.4) With the output of step 5.3) you can test your results and experiment with creating other numbers.



To advance our computation skills, let's experiment with alphanumeric characters and understand how to write words or sentences by using the binary system. The so-called American Standard Code for Information Interchange (ASCII) defines how various signs are coded in binary.

Task 2: Write the name of the route with the shortest distance in binary (10 min.)

Please use the shortened ASCII table in **Appendix 1 at the end of this assignment** to write the name of the route with the shortest distance in binary:

Your suggestion: _____

You can check whether your suggestion is correct by executing the following codes which convert binary strings to alphanumeric characters. Please use the notebook in IBM Quantum Lab (<https://lab.quantum-computing.ibm.com>) you have already created for experimenting with the numeric binary system.

1) Within this step we define a variable called “a_binary_string”. Please insert your suggested binary code for the name of the route with the shortest distance between the quotation marks in the following code:



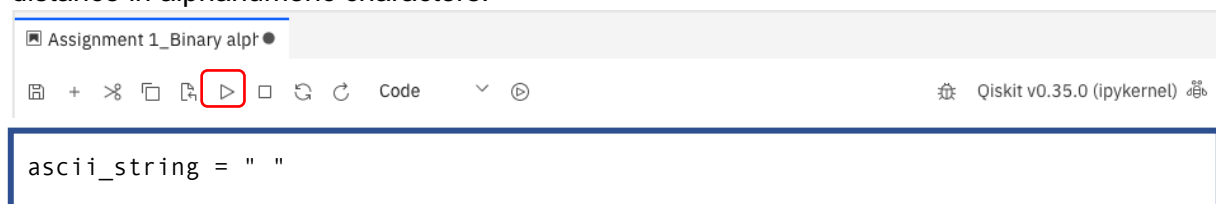
```
a_binary_string = "please insert your suggestion here"
```

2) Execute the following code, which splits the string at the whitespaces.



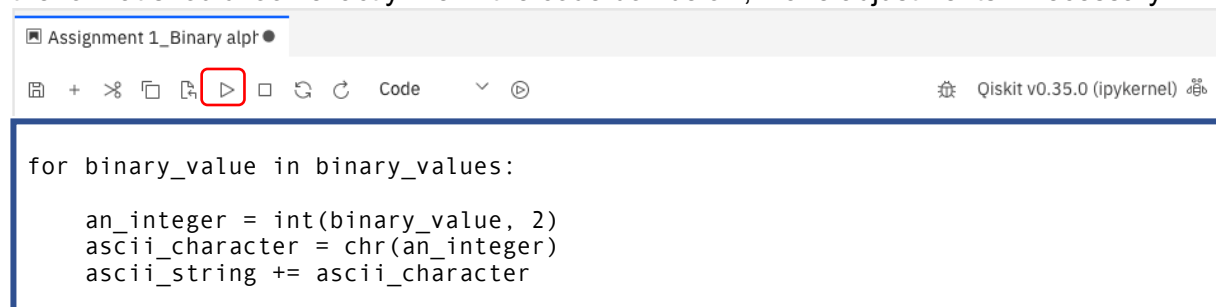
```
binary_values = a_binary_string.split()
```

3) Initialize the variable which is going to be assigned to the name of the route with the shortest distance in alphanumeric characters.



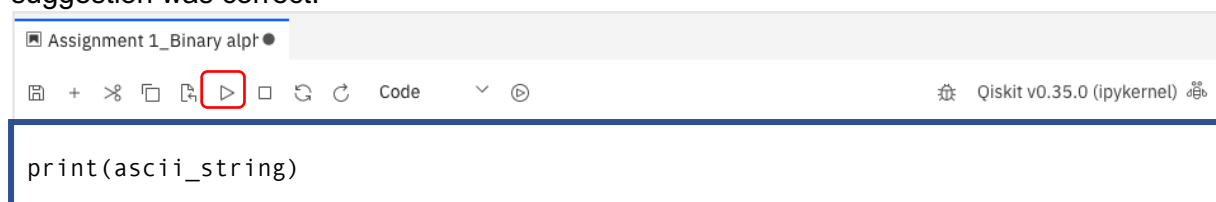
```
ascii_string = " "
```

4) Run the code which converts the sequence of binary strings to the base 2 decimal integer which is further converted to ASCII characters. Please make sure that after copying the code the format should look exactly like in the code box below; make adjustments if necessary.



```
for binary_value in binary_values:
    an_integer = int(binary_value, 2)
    ascii_character = chr(an_integer)
    ascii_string += ascii_character
```

5) Display the result of the previous code by using the command “print” for the variable created in step 3. The output of this step equals the name of the shortest route, indicating our suggestion was correct.



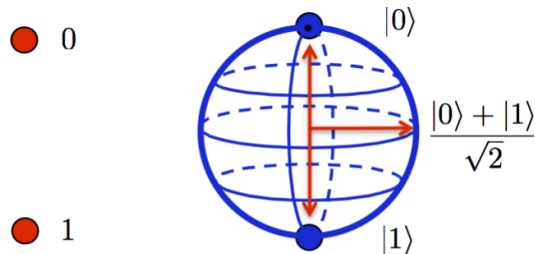
```
print(ascii_string)
```

2. Atoms of computation: bits and qubits

2a) Difference in computation power

Short recap: Classical Computers are based on bits which refer to binary values and can therefore be either 0 or 1.

Quantum Computers on the other hand are based on qubits: the quantum version of a bit. Qubits can also be 0 and 1 or a combination of both values. This means that a qubit can be, with a probability of e.g. 70% in the state 1 and with a 30% chance in the state 0. In Quantum Computing this combination is called superposition [3].



Classical Bit

Qubit

[4]

To imagine the impact of this characteristics on the computation power, let's compare the number of states that can be created by two bits vs. two qubits.

With two bits we can create the following combinations of states:

```
[ '0', '0' ]
[ '0', '1' ]
[ '1', '0' ]
[ '1', '1' ]
```

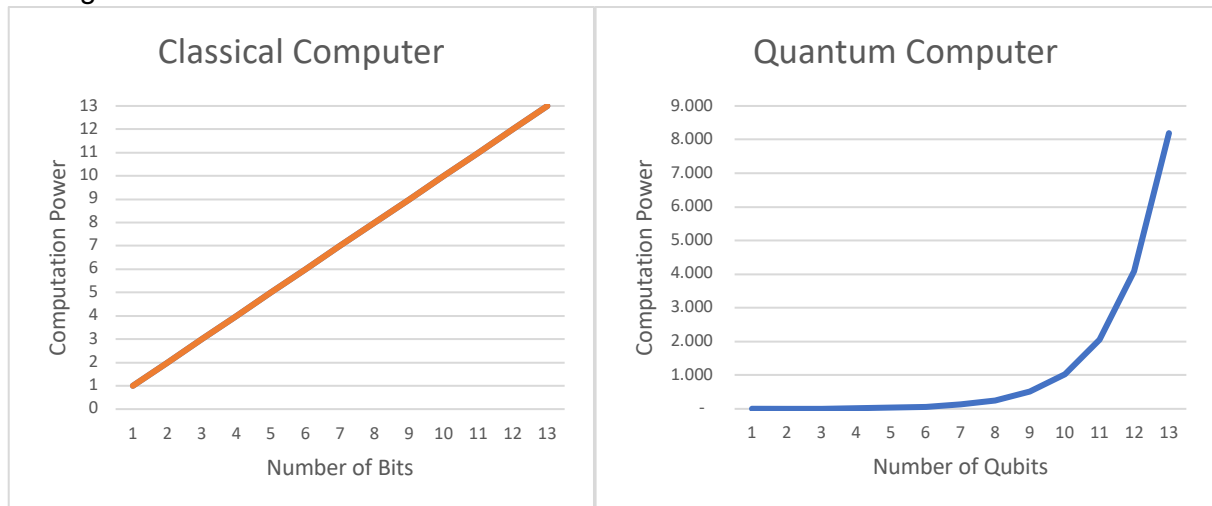
As a classical bit can only have the state 0 or 1 at a time, two bits can only be in one of the four combinations at a single point in time. For example, if bit 1 has the state 0 and bit 2 has the state 1, we can observe the second of the combinations above ['0', '1']. This indicates that we can create n states with n bits.

Two qubits however, can take all of those four combinations at once, meaning that we can create 2^n states with n qubits. We can observe this phenomenon due to superposition, as this allows the qubit to be in the following states at one time: $\alpha |00\rangle + \beta |01\rangle + \gamma |10\rangle + \delta |11\rangle$. $\alpha, \beta, \gamma, \delta$ represent the probabilities in which of the states the qubits currently are. [5]

# of qubits	Transition	# of bits	RAM
1	2^1	2	2 bits
2	2^2	4	4 bits
3	2^3	8	1 byte
4	2^4	16	2 bytes
5	2^5	32	4 bytes
6	2^6	64	8 bytes
7	2^7	128	16 bytes
8	2^8	256	32 bytes
9	2^9	512	64 bytes
10	2^{10}	1,024	128 bytes
11	2^{11}	2,048	256 bytes
12	2^{12}	4,096	512 bytes
13	2^{13}	8,192	1 kB
256	2^{256}	$1.16 * 10^{77}$	$1.32 * 10^{64}$ TB

This table shows for example that 13 qubits can store 1,024 bytes, while 13 bits can only store 1.5 bytes [8]. The last row of this table shows the computation power of the most powerful Quantum Computer of IBM. In May 2022 the Quantum Computer had 256 qubits which is equal to the power of 1.16×10^{77} bits [6]. As this number is hard to imagine, note that one billion is expressed as 10^9 and the number of atoms in the universe is between 10^{84} and 10^{89} [7].

We can also visualize the growth of computation power by adding a bit vs. a qubit by the means of diagrams.



While the computational power of Classical Computers increases linearly with adding one bit, you can see, that the ability of each qubit to be in two states at the same time exponentially boosts the capacity of Quantum Computers to process large computations without space and memory constraints that limit Classical Computers.

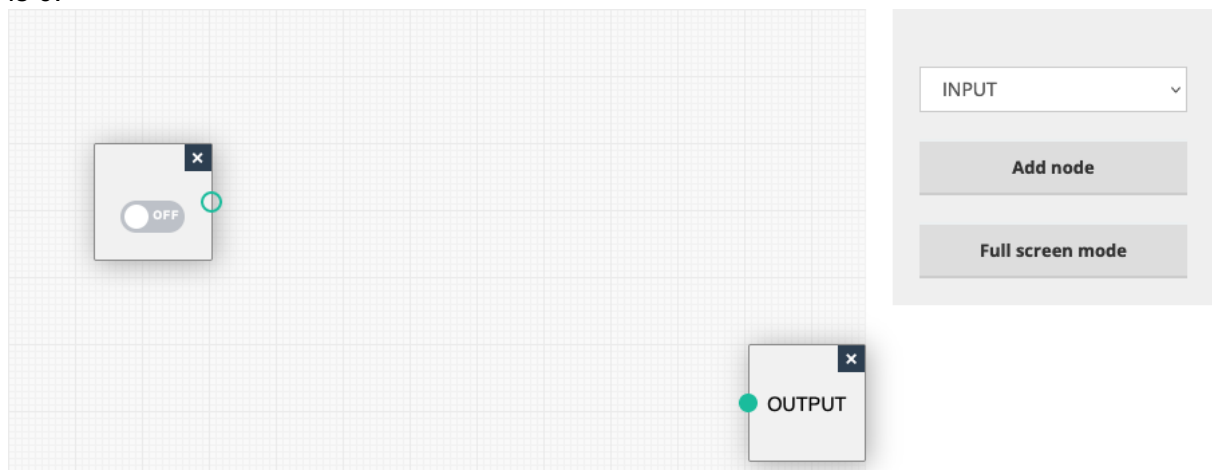
2b) Manipulation of Bits

Another difference between Classical Computers and Quantum Computers is the way bits/qubits are manipulated. To be able to compute, a Classical Computer uses logical gates (AND, OR, NOT) to manipulate bits. Quantum Computers, on the other hand use quantum logic gates to manipulate qubits and thus create computation power.

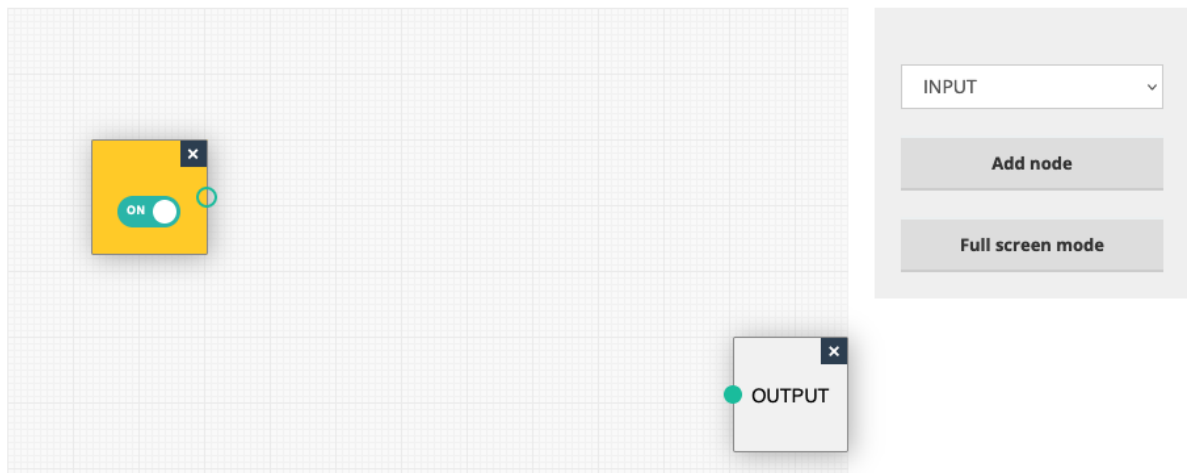
Task 3: Performing a NOT gate on classical bits (10 min.)

To get to know the basic manipulations of **bits** please visit the following website: <https://academo.org/demos/logic-gate-simulator/>.

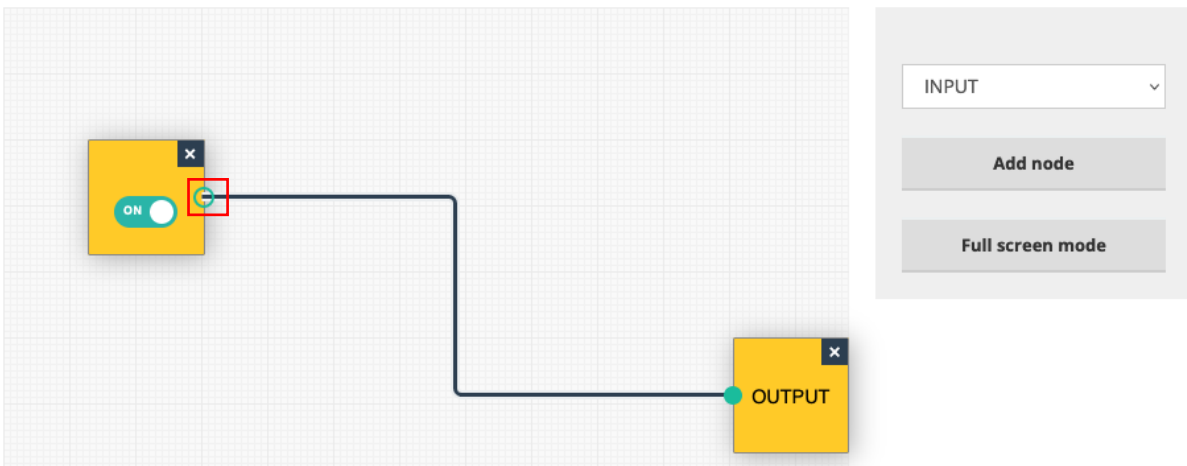
1. In the basic scenario you have an On/Off button (input node) and output node. In this simulator the state 1 is illustrated with the switch showing on, while off indicates that the state is 0.



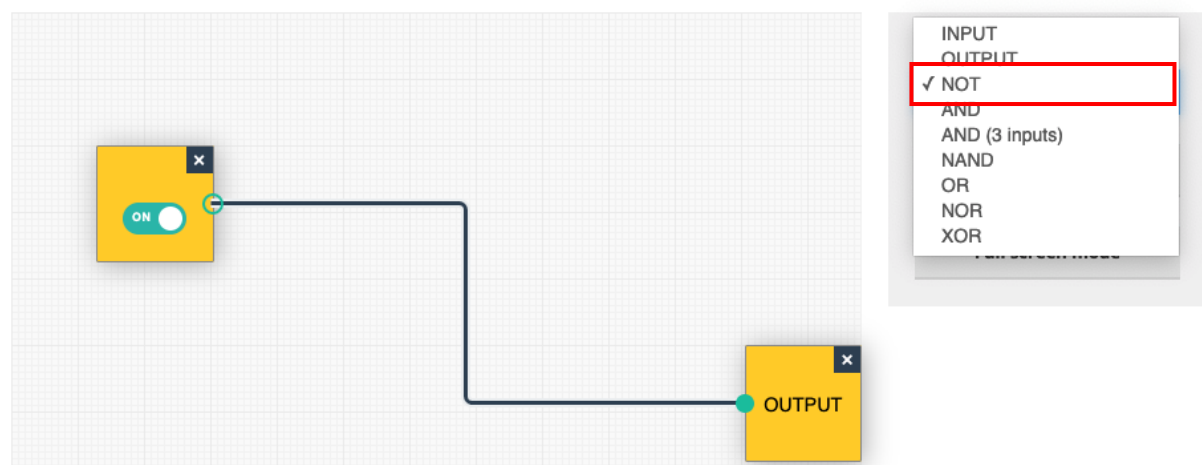
2. You will notice that switching the button to on will turn the light of the input node on. The output node is not affected as both nodes are not connected to each other.



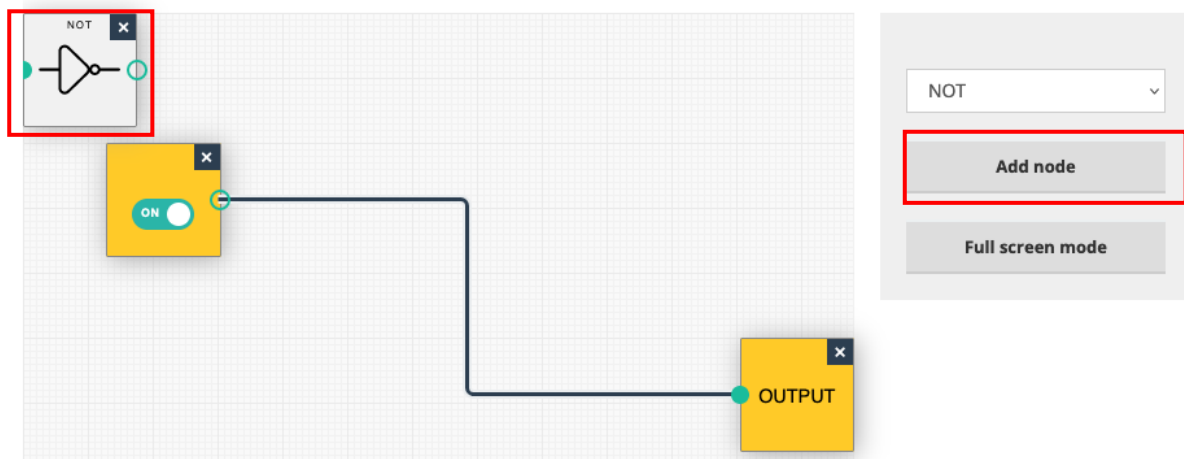
3. You can connect both nodes by dragging the circle of the input node to the output node. You will notice that the light of the output node also turns on.



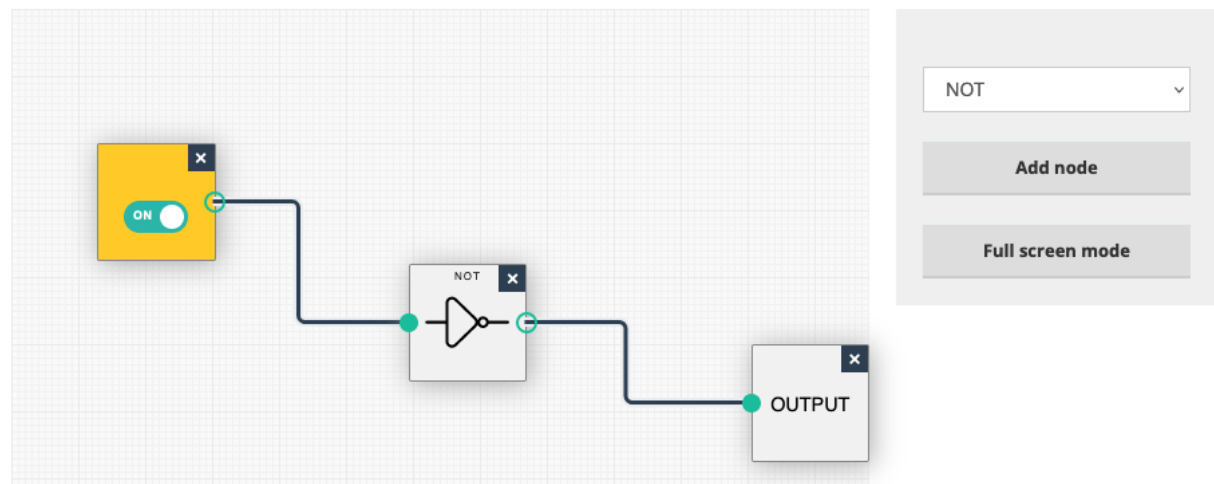
4. We can check what happens when we add a NOT-Gate by selecting "NOT" in the drop-down menu.



5. Click add node and the NOT-Gate will appear in the left upper corner of the browser.



6. Place the NOT-Gate between the On/Off switch and the Output and connect the nodes with each other.



7. Change the state of the switch several times and observe how the output changes. You will notice that flipping the value of a bit from 0 to 1 or vice versa can be achieved with performing a NOT-gate.

Also, qubits can be manipulated in a similar way. However, as those operations are a little bit more complex you will learn more about the different quantum gates in the next assignment.

2c) Probabilities vs. amplitudes

While Classical Computers follow the principles of classical probabilities, Quantum theory uses amplitudes that have to be converted into probabilities by squaring them. However, as this principle is not crucial for completing the next assignments you are provided with an optional assignment to better understand amplitudes in extension to this learning lab.

Useful Resources for Own Research

Learn more about the calculation with bits:

<https://qiskit.org/textbook/ch-states/atoms-computation.html>

Extended ASCII table:

<https://www.ibm.com/docs/en/aix/7.1?topic=adapters-ascii-decimal-hexadecimal-octal-binary-conversion-table>

Retrospective

Please answer the following questions:

1. Why is a Quantum Computer faster in route optimization compared to a Classical Computer?
2. What is a string?
3. How many bits form 1 byte?
4. How many distinct values can a byte have?
5. What is the difference in the values of bits and qubits?
6. How does the computation power grows when comparing Classical Computers with Quantum Computers?
7. What is the name of the gate to flip the value of a bit?

Sources

- [1] <https://www.bbc.co.uk/bitesize/guides/zwsbwmn/revision/1>
- [2] <https://web.stanford.edu/class/cs101/bits-bytes.html>
- [3] <https://www.educba.com/qubits-vs-bits/>
- [4] https://www.researchgate.net/figure/Figure-1-Classical-Bit-Vs-Qubit_fig2_308414229
- [5] <https://medium.com/@adubey40/classical-bit-vs-qubit-fa6c6c06e8f>
- [6] <https://research.ibm.com/blog/quantum-volume-256>
- [7] <https://www.swr.de/wissen/1000-antworten/wissenschaft-und-forschung/wie-viele-atome-gibt-es-im-universum-100.html>
- [8] <https://vincentlauzon.com/2018/03/21/quantum-computing-how-does-it-scale/>

Appendix 1

ASCII	Binary
space	100000
,	101100
.	101110
A	1000001
B	1000010
C	1000011
D	1000100
E	1000101
F	1000110
G	1000111
H	1001000
I	1001001
J	1001010
K	1001011
L	1001100
M	1001101
N	1001110
O	1001111
P	1010000
Q	1010001
R	1010010
S	1010011
T	1010100
U	1010101
V	1010110
W	1010111
X	1011000
Y	1011001
Z	1011010
[1011011
\	1011100
]	1011101
^	1011110
_	1011111
`	1100000
a	1100001
b	1100010
c	1100011
d	1100100
e	1100101
f	1100110

g	1100111
h	1101000
i	1101001
j	1101010
k	1101011
l	1101100
m	1101101
n	1101110
o	1101111
p	1110000
q	1110001
r	1110010
s	1110011
t	1110100
u	1110101
v	1110110
w	1110111
x	1111000
y	1111001
z	1111010