Task 05-01

- Write a Python program called **hamming_weight.py** that implements a function to calculate the population count of a given integer
- Calculate and display the Hamming Weight of 95,601
- Compare the result of your function to a similar capability already available in Python
- Upload your solution to the BNL QIS101 SharePoint site

Task 05-02

- Create a Python program called **prime_racer4.py** based upon prime_racer3.py that further decreases its runtime
- Various hints:
  - Avoid trial dividing every odd number from
  - Instead, only need to test if $sample \% p==0 \; where \; p \in primes$
  - Precompute an array of primes
  - Trial divide every sample using only this array of primes
- Upload your solution to the BNL QIS101 SharePoint site

prime_racer3.py

```
Number of primes found: 785
Total run time (sec): 0.031
```

prime_racer4.py *(DaveB)*

```
Number of primes found: 785
Total run time (sec): 0.016
```

Task 05-03

- Complete the existing program called **connect_four.py** to determine and display the name of the winner (Player 1 or Player 2) of a given Connect Four board
- The current code contains three lists (board1, board2, board3), with each 2D list encoding a board configuration
  - 0 = The space is empty (neither player has a checker in that space)
  - 1 = The space is occupied by a checker owned by Player 1
  - 2 = The space is occupied by a checker owned by Player 2
- Follow the standard rules for Connect Four to determine the winner (if there is a winner for that board)
- Upload your solution to the BNL QIS101 SharePoint site