C++ Vectors

Vectors are the C++ data type that most closely aligns with Python's list data type. Some common list operations in Python are not directly available for vectors, such as printing an entire vector all at once, and using slices or negative indices. Python lists always do *bounds-checking*, meaning you can never assign to a list index that doesn't exist. C++ will let you assign to a vector index that doesn't exist (which you should avoid doing!).

Vector operations (*type* stands for a data type, like `int` or `double`):

Create an empty vector	`vector<type> v;`
Create a vector of a certain size	`vector<type> v(int size);`
Create a vector of a certain size, filled with a certain item	`vector<type> v(int size, type value);`
Initialize a vector	`vector<type> v = {val1, val2, val3...};`
Access or change an item *Use `v.at(int position)` instead of brackets if you want bounds-checking.*	Use brackets just like Python lists or C++ arrays wherever you'd use a regular variable or literal (i.e., with `cin`, `cout`, assignment, passing arguments, returning values, etc). No Python-like slicing operations or negative indices.
Re-assign entire vector	`v = v2;` (resizes `v` to have the same length as `v2`, then copies all of v2's items into v), or `v = {val1, val2, val3...};`
Return first item in vector	`v.front()`
Return last item in vector	`v.back()`
Resize a vector	`v.resize(int newsize);`
Resize a vector and fill with an item	`v.resize(int newsize, type item);`
Return a vector's current size	`v.size()`
Test whether a vector is empty	`v.empty()`
Insert an item at the end (increases size by 1)	`v.push_back(type item);`
Insert an item at position `n` in the vector, shifting items in positions `n` to `v.size() - 1` to the right.	`v.insert(v.begin() + n, type item);`
Insert an item at a position calculated from the end of the vector.	`v.insert(v.end() - n, type item);`
Remove an item at the end (decreases size by 1)	`v.pop_back();`
Remove an item from position `n`, shifting items in positions `n + 1` to `v.size() - 1` to the left.	`v.erase(v.begin() + n);` *[can use `v.end()` as well to erase from end]*
Clear the vector (remove all items and resize to 0)	`v.clear();`

Iterating through a vector by using indices:

```
vector<int> vec = {1, 1, 2, 3, 5, 8};

for (int x = 0; x < v.size(); x++)
{
  cout << vec[x] << endl;
}
```

Iterating through a vector without using indices:

```
vector<int> vec = {1, 1, 2, 3, 5, 8};

for (int item : vec)
{
  cout << item << endl;
}
```