C++ Vectors

Vectors are the C++ data type that most closely aligns with Python's list data type. Vectors are arrays that can be dynamically resized as a program runs; they can grow and shrink as necessary. Like C++ arrays, however, they do not use bounds checking by default. That is, if you make a vector of ten values and try to access vector[20], there's no guarantee what will happen.

You should use C++ arrays for constant arrays that are defined directly in your code, or for arrays whose size can be determined at compile-time. Use vectors otherwise.

	C++ arrays	**C++ vectors**	**Python lists**
Can be resized	No	Yes	Yes
Bounds-checking	No	No (optional)	Yes
Must know size at compile-time	Yes	No	No
Can be re-assigned all at once	No	Yes	Yes
Can be passed to functions	Yes	Yes	Yes
Can be returned from functions	Yes, but not straightforward	Yes	Yes
Literals available	Only for initialization, e.g., `int array[3] = {1, 2, 3};`	No	Yes (e.g., `[1, 2, 3]`)

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

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 item);`
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).
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	`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();`