A Guide for Transitioning from Python to C++

Jillian Tang & Ethan Chi

C++ is a very different language from Python, but some simple principles can help guide you through the changes! Let's have a look.

Semicolons, Parentheses, and Braces? Oh my!

Curly Braces {}

- Curly braces (`{ }`) are almost exactly equivalent to Python indentation.
- Use them for **if/else/while** statements, function definitions

```
void function(int a, string b) {
      if (a == 1) {
            cout << b << endl;
      }
}
```

- Like in Python, we usually indent inside brackets.

Conditions

- An **if/else** statement or **while** loop requires **parentheses** around the condition:

```
if (a == 1) {
      cout << b << endl;
}
```

- **And** is represented by **&&**, **or** is represented by **||**, and negation is represented by **!**.

```
if ((a == 3 && b == 2) || c != 1) {
      cout << "either a equals 3 and b equals 2 or c does not equal 1";
}
```

- Elif is replaced by `else if`. (Actually, elif is a shortened version of else if.)

```
if (a == 1) {
      cout << "One" << endl;
} else if (a == 2) {
      cout << "Two" << endl;
}
```

Semicolons

- You need a semicolon at the end of every **non-control statement.**
  - Setting a variable:

    ```
    int i = 0;
    ```

- Calling a function
  ```
  doSomething(a, b);
  ```
- **Do not** put a semicolon at the end of a **control statement**.
  - If and else:
    ```
    if (i == 1) { // no semicolon here!
    	cout << “Test” << endl;
    } else {
    	cout << “else” << endl;
    }
    ```
  - While statements:
    ```
    while (a == 1) {
    	cout << a << endl;
    }
    ```
- **Do not** put a semicolon at the end of a statement beginning with a #.
  ```
  	#include “strlib.h”
  ```

Types

C++ is a **typed** language, which means that you sometimes need to explicitly say what type something is.

- A type is a fundamental kind of value. Examples include `int`, `string`, `char` (single character, not in Python), `double` (equivalent of Python float)
- You must explicitly state the type when declaring a variable, but not while using it after that.
  ```
  int i = 0;
  i = i + 2;   // no type needed here
  ```
- Function **parameters** must also have types; also, every function must include a **return type**. If the function doesn’t return anything, it has return type **void.** However, you don’t have to include the types when calling the function.
  ```
  int function(int a, string b) {
  	return a + 2;
  }
  void function2(int a, string b) {
  	cout << b << a + 2 << endl;
  }
  function(3, “test”);
  ```

Syntax Differences

Here are some common things that differ from C++ to Python.

- **Single-line comments** are made using **//**. **Multi-line comments** are made using **/*** to start and ***/** to end.

  ```
  /* This function takes in an integer and a string.
   * It returns the value of the integer plus 2.
   */
  int function(int a, string b) {
        return a + 2; // does the adding
  }
  ```
- To loop through a range of values, instead of using range as in Python, you define a variable and increment it through a for loop:

  ```
  for (int i = 0; i < 10; i++) {
        cout << i << “ ”;
  } // prints out 0 1 2 3 4 5 6 7 8 9
  ```
- To loop through a collection (like a list in Python), you use the for-each syntax:

  ```
  for (int i : numbers) {
        cout << i << “ “;
  } // prints out the contents of numbers
  ```
- To print out something, use `cout <<` followed by what you want to print (and << `endl` if you want a new line). You can directly pass in variables through this, but you must separate different values with <<.

  ```
  cout << “hello world” << endl;
  string name = “Mary”;
  int age = 20;
  cout << “My name is “ << name << “ and my age is “ << age << endl;
  ```
- The **++** operator is equivalent to += 1 (and similarly for --):

  ```
  int i = 3;
  i++;
  cout << i << endl; // prints out “4”
  ```
- If you call a function before its definition in your code, your C++ compiler will think it doesn’t exist. Use **forward declarations** (aka function prototypes) at the beginning of your code to let it know that these functions exist in your code.

  ```
  void function2(int a, string b);
  int function(int a, string b) {
        function2(a, b); // now you can call this before writing
        return a + 2;    // function2
  ```

```
}
void function2(int a, string b) {
      cout << b << a + 2 << endl;
}
```

Important Functions

A lot of important functions that you might use in Python aren't available in C++ by default. Instead, you have to #include the appropriate library at the top of your file. Here are some examples:

- **stringSplit** splits a string by a delimiter (like Python .split())
  ```
  #include "strlib.h" // put this at the top of your file!
  string data = "a,b,c";
  Vector<string> components = stringSplit(data, ",");
  ```
- **toLowerCase** converts a string to its lowercase version (see also toUpperCase) [from "strlib.h"]:
  ```
  string data = "ABC";
  cout << toLowerCase(data) << endl; // prints abc
  ```
- **getLine** gets a string from the user (see also getInteger) [from "simpio.h"]:
  ```
  string data = getLine("Please enter some text: ");
  ```
- **getYesOrNo** gets a boolean from the user [from "simpio.h"]:
  ```
  bool yesOrNo = getYesOrNo("Yes or no? ");
  ```
- **startsWith(**str, prefix**)** checks whether a string begins with the prefix (see also endsWith) [from "strlib.h"].
- **stringToInteger(**str**)** and **integerToString(**int**)** do the appropriate conversions, as do **stringToReal(**str**)** and **stringToReal(**d**)** [from "strlib.h"].
- **randomInteger(**low, high**)** returns a random integer between low and high, inclusive [from "random.h"].
- str**.find(**b**)** checks if the character or string b is found in str and returns its index if it's found, or string::npos otherwise.
  ```
  string a = "pineapple"
  cout << a.find("apple") << endl; // prints 4
  if (a.find('p') != string::npos) { // equivalent of 'if 'p' in a'
        cout << "p is in string a!" << endl;
  }
  ```

To view more about Stanford libraries, check out the documentation:
https://web.stanford.edu/dept/cs_edu/cppdoc/