Python to C++

1. Main Program

Python: Program code is indented after colon “:”

```python
def main():
    body of program
```

C++: Has more setup overhead.

Requires prefixed `#include <iostream>` and `using namespace std;` directives; code is enclosed between {} braces

```cpp
#include <iostream>
using namespace std;
void main()
{
    body of program
}
```

Note: all statements in C++ terminate with a semicolon “;”

2. Variables

Python: variables are declared implicitly with the type (int, float, string) determined by the initial value assigned

```python
number = 17
x = 3.14159
str = "Hello World"
```

C++: All variable must be explicitly declared before they are used; initializations are optional.

```cpp
int number = 17;
float x = 3.14159;
string str = "Hello World";
```

Note Python uses the *sticky note model* for assigning values to variable; C++ uses the *post office box model*.

3. Assignment Statements

For both Python and C++ syntax and semantics are the same except that statements in C++ must terminate with a semi-colon “;”. Unlike Python, C++ does not support simultaneous assignment statements.

4. Input/Output

Python: `input` and `print` statements; `input` functions like a simultaneous assignment statement with a built in prompt. If multiple values are read, they must be separated by comma

```python
a,b = input("Enter two integers ")
print x, y
```
Note that print always advances to the next line; to not advance to the next line, end with a comma

```plaintext
print x, y,
```

C++ uses `cin >>` and `cout <<` to read and write.

```plaintext
cout << "Enter two integers ";
cin >> a >> b;
```

Note that any prompt must be done with `cout <<`. If multiple values are read in, unlike Python they must be separated by whitespace characters; that is blanks, tabs or newline characters. A comma is not whitespace.

```plaintext
cout << x << y;
```

Unlike Python, C++ does not automatically advance to the next line. To do so you must explicitly output a newline character, denoted by ‘\n’

```plaintext
cout << x << y << '\n';
```

Alternately you can use `endl`

```plaintext
cout << x << y << endl;
```

Note: The syntax of I/O for C++ is derived from the semantics of I/O. In C++ I/O is seen as a stream of text characters either directed to the console output device (abbreviated cout) or taken from the console input device (abbreviated cin). The `<< and `>>` are called insertion and extraction operators. So `cout << number;` means insert the value of number into the output stream to the console output device and `cin >> number;` means extract from the input stream the value for number and store in number. This is why we say `cout << x << y;` and `cin >> a >> b;` using the `<< (insertion) and `>> (extraction) operators between values/variables instead of using commas.

### 5 if else statements

Python:

```plaintext
if number == 0:
    <if true clause>
else:
    <else false clause>
```

Note the bodies of the `<if true clause>` and `<else false clause>` are indented

C++

```plaintext
if (number == 0)
{
    <if true clause>
}
else
{
    <else false clause>
}
```

Note the bodies of the `<if true clause>` and the `<else false clause>` are enclosed between within `{ }` braces. Also the condition being tested, `(number == 0)`, must be enclosed by parentheses.
Multi-way branching: Python uses the `elif` statement (a contraction of else if); C++ uses `else if`.

6. Counting Loops

Python for loop

```python
for i in range(10):
    <body of loop>
```

```python
for i in [2, 3, 5, 7, 11]:
    <body of loop>
```

In Python the loop index `i` iterates over the sequence taking on in turn each value in the sequence and executing the body of the loop. The general form of the Python for loop is

```python
for <var> in <sequence>:
    <body of loop>
```

C++ for loop

```cpp
for (int i = 0; i < 10; i++)
{
    <body of loop>
}
```

In C++ the loop index `i` is initialized (`i = 0`), tested (`i < 10`) and if the test is true the body of the loop is executed and then the loop index is incremented (`i++` or `i = i + 1`). This is equivalent to the first Python for loop above. Unfortunately the second Python for loop cannot be implemented in C++ using a for loop. On the other hand there are some things a C++ form loop can do that a Python for can’t. The general form is

```cpp
for (initial step; terminal condition; update action)
{
    <body of loop>
}
```

7. While Loops

Like if else statements while loops in Python and C++ are similar in syntax and semantics, the main difference being that Python uses a colon and indents the body of the loop while C++ requires parentheses around the test condition and encloses the body of the loop within `{}` braces.

Python while loop

```python
while a % b != 0:
    b = b+1
```

C++ while loop

```cpp
while (a % b != 0)
{
    b = b+1;
}
```
8. Grouping Statements – indenting vs {

Python uses indentation to group statements that are subordinate to an if, elif, or else statement or make up the body of a loop. C++ uses pairs of braces {} to do the same. In C++ a set of statements enclosed by {} is called a compound statement.