COMP2012H Honors Object-Oriented Programming and Data Structures
Syntax Comparison between Python and C++: Basics and Program Flow Control

The purpose of this set of notes is to help you quickly transfer your basic knowledge of Python to that of C++. Please note that it is not a complete summary of our lecture notes. For all the C++ features discussed in COMP2012H, you have to carefully study the lecture notes on our course website.

In Python | **In C++**

Hello World Program

In Python

```
"""
 File: hello_world.py
 A common program used to demo a new language
"""
print("Hello World!")
```

In C++

```
/*
 * File: hello_world.cpp
 * A common program used to demo a new language
 */
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello world" << endl;
    return 0;
}
```

Note: Every C++ program must have exactly one main() function which is the entry point of the program.

Executing a Python program

1. execute the program:
   python hello_world.py

Executing a C++ program

1. compile the program:
   g++ -o hello_world.out hello_world.cpp
2. execute the program:
   hello_world.out

Basic Output

In Python

To print the word "abc" with a newline character:

```
print("abc")
```

Or,

```
print("abc", end = "\n")
```

In C++

To print the word "abc" with a newline character:

```
cout << "abc" << endl;
```

where `endl` means "end of the line".
Or ,

```
cout << "abc\n";
```

Comments

In Python

- for one or more lines of comments:

  `""" ... """`

- for one line of comment only:

  `# ...`

In C++

- for one or more lines of comments:

  `/* ... */`

- for one line of comment only:

  `// ...`

Including a module/library

In Python

```
import random
```

In C++

```
#include <iostream>
```

Statements

In Python

- A statement is a line of code.
- Only extra blanks and tabs are ignored.
- If the line of the statement is too long, one may break it into several lines using "\".

For example:

```
print("Hello", \
" world")
print("!")
```

In C++

- Each statement ends in a semicolon ";"
- Extra blanks, tabs, lines are ignored.
- More than one statement can be on one line.
- A statement may be spread over several lines.

For example:

```
cout << "Hello" <<
" world";
cout << "!" << endl;
```

Variables

In Python

- Basic Data Types:
  - Integer:
    Examples of values: `0`, `1`, `100`, `-101`, ...
  - Floating point:
    Examples of values: `0.5`, `-123.908232`
  - String:
    Examples of values: `"A"`, `'abc'`, `"comp 2012H"`, ...
  - Boolean:
    Examples of values: `True`, `False`
- Variables need not be declared and their data types are inferred from the assignments.
  For examples:

```
num1 = 100 # integer data type
num2 = 0.05 # float data type
```

In C++

- Basic Data Types:
  - Integer: `short`, `int`, `long`, `long long`, etc.
    Examples of values: `0`, `1`, `100`, `-101`, ...
  - Floating point: `float`, `double`, `long double`, etc.
    Examples of values: `0.5`, `-123.908232`
  - Character: `char`
    Examples of values: `'A'`, `'a'`, `'B'`, `'b'`, ...
  - Boolean: `bool`
    Examples of values: `true`, `false`
- Variables have to be declared and defined.
  For examples:

```
int num1;
num1 = 100;
double num2 = 0.05;
```

if Statement

In Python

```
if (<bool-expr>) :
   <stmt>
```

In C++

```
if (<bool-expr>) <stmt>
```

In Python

```
if (<bool-expr>) :
   <stmt(s)>
```

In C++

```
if (<bool-expr>) { <stmt(s)> }
```

In Python

```
if (<bool-expr>) :
   <stmt>
else :
   <stmt>
```

In C++

```
if (<bool-expr>) <stmt> else <stmt>
```

In Python

```
if (<bool-expr>) :
   <stmt(s)>
else :
   <stmt(s)>
```

In C++

```
if (<bool-expr>) { <stmt(s)> } else { <stmt(s)> }
```

In Python

```
if (<bool-expr>) :
   <stmt(s)>
elif (<bool-expr>) :
   <stmt(s)>
```

In C++

```
if (<bool-expr>)
{
  <stmt(s)>
} else if (<bool-expr>) {
   <stmt(s)>
}
```

In Python

```
if (<bool-expr>) :
   <stmt(s)>
elif (<bool-expr>) :
   <stmt(s)>
else :
   <stmt(s)>
```

In C++

```
if (<bool-expr>)
{
   <stmt(s)>
} else if (<bool-expr>) {
   <stmt(s)>
} else {
   <stmt(s)>
}
```

Note: Blocks are identified by having the same indentation.
For example:

```
x = -5
if x > 0 :
   print("x is positive", end="")
   if x % 2 :
      print(" and odd.")
   else :
      print(" and even.")
elif (x < 0) and (x % 2) :
   print("x is negative and odd.")
elif (x < 0) and (not (x % 2)) :
   print("x is negative and even.")
else :
   print("x is zero.")
```

Note: Blocks are identified by pairs of braces ({}).
For example:

```
int x = -5;
if (x > 0)
{
   cout << "x is positive";
   if (x % 2)
      cout << " and odd." << endl;
   else
      cout << " and even." << endl;
} else if ((x < 0) && (x % 2)) {
   cout << "x is negative and odd." << endl;
} else if ((x < 0) && !(x % 2)) {
   cout << "x is negative and even." << endl;
} else {
   cout << "x is zero." << endl;
}
```

if-else Operator

In C++, there are if-else expressions. The syntax is:
`<condition> ? <result1> : <result2>`
It means that if `<condition>` is true, the expression's value will be `<result1>`, otherwise it will be `<result2>`.
For example:

```
int x = 2, y = 3;
int z = (x > y) ? x : y;
cout << z << endl;
// the output will be 3
```

while Loop

```
while (<bool-expr>) :
    <stmt(s)>
```

Note: Blocks are identified by having the same indentation.

For example:

```
i = 10
while i > 0:
   i = i - 2
   print(i)
```

```
while (<bool-expr>)
   <stmt>

while (<bool-expr>)
{
    <stmt(s)>
}

do (<bool-expr>)
    <stmt>

do
{
    <stmt(s)>
} while (<bool-expr>);
```

Note: Blocks are identified by pairs of braces ({}).

For example:

```
int i = 10;
while (i > 0)
{
   i -= 2;
   cout << i << endl;
}
```

for Loop

```
for <item> in <a list of item> :
   <stmt(s)>
```

For example:

```
for i in range(10):
   print(i)
```

```
for (<for-initialization>; <bool-exp>;
<post-processing>) { <stmt(s)> }
```

For example:

```
for (int i = 0; i < 10; i++)
    cout << i << endl;
```

break and continue

In a `for` loop, `break` means to stop the whole loop; while `continue` means to skip the current execution.

the same.

Functions

A Python function need not specify the parameter types and return types.
For example,

```
""" File: function_example.py
    A Python Program with two functions:
    PrintNum() and AddOne()
"""
def PrintNum(num):
   print("The number is", num)

def AddOne(num):
   return (num + 1)

PrintNum(10)
PrintNum(AddOne(10))
```

A C++ function has to specify the parameter types and return types.
For example,

```
/* File: function_example.cpp
   A C++ Program with two functions:
   PrintNum() and AddOne()
 */
#include <iostream>
using namespace std;

void PrintNum(int num)
{
   cout << "The number is " << num << endl;
}

int AddOne(int num)
{
   return (num + 1);
}

int main()
{
   PrintNum(10);
   PrintNum(AddOne(10));
   return 0;
}
```

Some Operators in Python and C++

		Python			C++		
		Symbol	Example	Output	Symbol	Example	Output
Arithmetic Operators	Addition	+	1 + 2	3	Same		
	Subtraction	-	1 - 2	-1	Same		
	Multiplication	*	1 * 2	2	Same		
	Division	/	1 / 2	0.5	/	1.0 / 2	0.5
	Integer Division	//	1 // 2	0	/	1 / 2	0
	Modulus (Remainder)	%	9 % 4	1	Same		
	Power	**	2 ** 3	8	Nil		
Assignment Operators	Assignment	=	x = y		Same		
	Addition Assignment	+=	x += y		Same		
	Subtraction Assignment	-=	x -= y		Same		
	Multiplication Assignment	*=	x *= y		Same		
	Division Assignment	/=	x /= y		Same		
Relational Operators	And	and	True and False	False	&&	true && false	false
	Or	or	True or False	True	||	true || false	true
	Not	not	not False	True	!	!false	true
Comparison Operators	Larger than	>	20 > 10	True	Same		
	Larger than or equal to	>=	20 >= 10	True	Same		
	Smaller than	<	20 < 10	False	Same		
	Smaller than or equal to	<=	20 <= 10	False	Same		
	Equal to	==	20 == 10	False	Same		
	Not equal to	!=	20 != 10	True	!=	20 != 10	true

Increment Operators	Post-increment	Nil			++	x = 1; y = 2;
						y = x++;
						cout << x <<
						" " << y;
							2 1

	Pre-increment	Nil			++	x = 1; y = 2;
						y = ++x;
						cout << x <<
						" " << y;
							2 2

Decrement Operators	Post-decrement	Nil			--	x = 1; y = 2;
						y = x--;
						cout << x <<
						" " << y;
							0 1

	Pre-decrement	Nil			--	x = 1; y = 2;
						y = --x;
						cout << x <<
						" " << y;
							0 0

References:

1. Cay Horstmann. (2012). C++ For Everyone. Second Edition. Wiley.