Operators Caveat

```
print("Hi"*4)
```

Works. Why?

Functions

The C++ Way:

```
double ConvertFahrenheitToCelsius (double fahrenheit)
{
    //  Conversion function
    return (fahrenheit - 32) * (5. / 9.);
}

int main()
{
    double fahrenheit;
    std::cout << "Insert degrees fahrenheit: ";
    std::cin >> fahrenheit;
    std::cout << fahrenheit << " in celsius is " <<
    ConvertFahrenheitToCelsius(fahrenheit);
}
```

The Python Way

```
def ConvertFahrenheitToCelsius(fahrenheit):
        return (fahrenheit - 32) * (5 / 9)



fahrenheit = float(input("Insert degrees fahrenheit: "))
print(fahrenheit, "in celsius is",
convertFahrenheitToCelsius(fahrenheit))
```

The Python Way

```
fahrenheit = float(input("Insert degrees fahrenheit: "))
print(fahrenheit, "in celsius is",
ConvertFahrenheitToCelsius(fahrenheit), ".")

def ConvertFahrenheitToCelsius(fahrenheit):
return (fahrenheit - 32) * (5 / 9)
```

Doesn’t work, even though the book says it should. Good rule of thumb is to make sure you define the function before you use it.

Scope

C++

```
// Global Scope
int X = 99;
void funct(int Y){
    // Local scope
    Z = X + Y;
}
int main()
{
    funct(1);
    // result = 100
}
```

Python

```
# Global scope
X = 99

def func(Y):
    # Local scope
    Z = X + Y
    return Z

func(1)
# result = 100
```

Modules

C++

MyModule.hpp

```
// A function that does something
void moduleFunction (int x);
```

MyModule.cpp

```
#include "MyModule.hpp"

void moduleFunction (int x)
{
    // Do something...
}
```

MyProgram.cpp

```
#include "MyModule.hpp"

int main()
{
    // Call a function from my module
    moduleFunction (123);
}
```

Python

Terminal Help main.py - Lesson 3 - Visu

printName.py

```
1   def printMyName():
2       print("Lindsay", "Spencer")
```

main.py

```
1   import printName
2
3   printName.printMyName()
```

Flow Control & Loops

While Loop

C++

```cpp
int main()
{
    int i = 1;
    while (i < 6) {
        std::cout << i;
        i++;
    }
}
```

Python

```python
i = 1
while i < 6:
    print(i)
    i += 1
```

The `continue` Statement

```
i = 0
while i < 6:
    i += 1
    if i == 3:
        continue
    print(i)
```

With the continue statement we can stop the current iteration, and continue with the next

The `break` Statement

```
i = 1
while i < 6:
    print(i)
    if i == 3:
    break
    i += 1
```

With the break statement we can stop the loop even if the while condition is true.

*https://www.w3schools.com/python/python_while_loops.asp

For Loop

- We'll get to it later . . .