for (int ind =0; ind < 16; ind++){
    // Do stuff
}
#include <iostream>
using namespace std;
int main(){
    double fahrenheit = -50.0;
    double celcius = 0;
    cout << "|" << " Farenheit " << "|";
    cout << " Celcius " << "|" << endl;
    for (int ind = 0; ind < 16; ind++){
        fahrenheit += 10;
        celcius = (fahrenheit-32.0)*(5.0/9.0);
        cout << "| " << fahrenheit << " | ";
        cout << celcius << " | " << endl;
    }
    return 0;
}
if (test1) {
   // if test1 is not 0
} else if (test2){
   // if test1 is 0 but test2 is not 0
} else {
   // test1 and test2 are both 0
}

VERY SIMILAR TO PYTHON
TESTS

NOT EQUAL
a != b

EQUAL
a == b

GREATER THAN
a > b

GREATER THAN OR EQUAL
a >= b

LESS THAN
a < b

LESS THAN OR EQUAL
a <= b

AND
a && b

OR
a || b

NOT
!a

VERY SIMILAR TO PYTHON
```cpp
#include <iostream>
#include <cmath>
using namespace std;

int main(){
   double discriminant = 0;
   double a, b, c;
   cout << "Please enter a, b, c coefficients of a quadratic: ";
   cin >> a;
   cin >> b;
   cin >> c;
   cout << endl;
   discriminant = pow(b, 2.0) - 4*a*c;
   if (discriminant > 0.0) {
      cout << "The equation has two roots. The roots are ";
      cout << "x = " << (-b + sqrt(discriminant))/(2*a) << " and ";
      cout << "x = " << (-b - sqrt(discriminant))/(2*a) << endl;
   } else if (discriminant < 0.0) {
      cout << "The equation has no real roots." << endl;
   } else {
      cout << "The equation has one root: ";
      cout << "x = " << -b/(2*a) << endl;
   }
   cout << endl;
   return 0;
}
```
string user_input="";
cout << "Please enter a temperature in Farenheit: ";
cin >> user_input;
#include <iostream>
#include <string>
#include <cstdlib>

using namespace std;

int main(){
   double fahrenheit = -50.0;
   double celcius = 0;
   string user_input="";
   cout << "Please enter a temperature in Farenheit: ";
   cin >> user_input;
   cout << endl;
   cout << "The temperature in Celcius is ";
   fahrenheit = atof(user_input.c_str());
   celcius = (fahrenheit-32.0)*(5.0/9.0);
   cout << celcius << "." << endl;
   return 0;
}"
string first_string = "A";
string second_string = "B";
cout << first_string + second_string << endl;

GIVES “AB”

SOME STRING METHODS:

size
clear
erase
append
compare
replace
pop_back
push_back
copy
insert
swap
find
find_first_not_of
rfind

SEE CPPREFERENCE UNDER BASIC STRING
#include <iomanip>

FORMAT STRINGS WITH IOMANIP LIBRARY
SEND FORMATTING TO COUT OBJECT

cout << setw(10) << right

THIS SETS WIDTH 10, RIGHT JUSTIFIED (FLUSH)

OTHER STRING FORMATTING

setbase
setfill
setprecision
setw
right

left
fixed
scientific
dec
hex

SEE CPPREFERENCE UNDER BASIC STRING
WHILE

```c
while(test1){
    // Runs as long as test1 not 0
}
```

ALMOST THE SAME AS PYTHON
#include <iostream>
#include <string>
#include <iomanip>
using namespace std;

int main(){
    double position = 100.0;
    double velocity = 0.0;
    double time_elapsed = 0.0;
    const double g = -9.78;
    const double time_interval = 0.25;

cout << setw(10) << right;
cout << "Time" << setw(1) << "|" << setw(10);
cout << "Velocity" << setw(1) << "|" << setw(10);
cout << "Position" << setw(1) << "|" << setw(10);
cout << endl;

while(position > 0.0){
    cout << setw(10) << setprecision(4);
    cout << time_elapsed << setw(1) << "|" << setw(10);
    cout << velocity << setw(1) << "|" << setw(10);
    cout << position << setw(1) << "|" << setw(10);
    cout << endl;

time_elapsed += time_interval;
velocity += time_interval * g;
position += time_interval * velocity;
}
return 0;}

Monday, December 6, 2010
DO WHILE

```python
do {
    // always runs once
    // then runs as long as test1 not 0
} while(test1);
```

DON'T HAVE THAT IN PYTHON

(Not a big loss)
CONTINUE/BREAK

```python
while(test1)
    // do some stuff
    break;  // leave the while loop
    // never gets done
}
// break takes you here

while(test1)
    // break takes you here
    // do some stuff
    continue;  // back to the top of the loop
    // because of continue we don’t get here
```

SAME SAME IN PYTHON

BREAK-CONTINUE WORKS IN FOR, WHILE AND DO-WHILE LOOPS
SWITCH (SPAWN OF SATAN)

```cpp
switch(signal){
    case 1:
        // when signal==1
        cout << "My brain hurts" << endl;
        break; // leave switch statement
    case 2:
        // when signal==2
        cout << "I can’t see anything" << endl;
        break; // leave switch statement
    default:
        cout << "No! Open the door and come in." << endl;
        break; // leave switch statement
}
// breaks land here
```

DON'T HAVE THIS IN PYTHON
REASON: NASTY SOURCE OF BUGS
SWITCH FALL THROUGH

FALL THROUGH MEANS IT EXECUTES ALL CODE PAST FIRST MATCHING CASE EASY TO MAKE MISTAKES

JUST SAY "NO" TO SWITCH STATEMENTS

```cpp
switch(signal){
    case 1:
        // when signal==1
        // does code in 1,2 and default
        cout << "My brain hurts" << endl;
    case 2:
        // when signal==2
        // does code 2 and default
        cout << "I can’t see anything" << endl;
    default:
        cout << "No! Open the door and come in." << endl;
}
```

Monday, December 6, 2010
FUNCTION (PROCEDURE)

MUST DEFINE INPUT/OUTPUT TYPES
DEFINE BEFORE USEING

```cpp
#include <iostream>

using namespace std;

void foo(void){
    cout << "Hello there." << endl;
    return;
}

int main(void){
    foo();
    return 0;
}
```
#include <iostream>
using namespace std;
// declare function
void foo(void);
// function prototype
int main(void){
    // use function
    foo();
    return 0;
}
// define function
void foo(void){
    cout << "Hello there." << endl;
    return;
}

IF YOU DECLARE FUNCTIONS AT TOP (YOU SHOULD)
ORDER DOESN'T MATTER
LIKE IF__MAIN__ TRICK IN PYTHON
```cpp
#include <iostream>
using namespace std;

void f2c(double);

int main(void){
    f2c(-40);
    return 0;
}

void f2c(double fahrenheit){
    double celcius = (fahrenheit-32)*5.0/9.0;
    cout << fahrenheit << " Fahrenheit is ";
    cout << celcius << " Celcius." << endl;
    return;
}
```
```cpp
#include <iostream>
using namespace std;

double f2c(double);

int main(void){
    double fahrenheit = -40;
    cout << fahrenheit << " Fahrenheit is ";
    cout << f2c(fahrenheit) << " Celcius." << endl;
    return 0;
}

double f2c(double fahrenheit){
    return (fahrenheit-32)*5.0/9.0;
}
```
#include <iostream>
using namespace std;

void mess_you_up(string);

int main(void){
    string value = "the password is fibble snap"
    cout << "value = " << value << endl;
    cout << "calling (pass by value) mess_you_up(value)" << endl;
    mess_you_up(value);
    cout << "value = " << value << endl;
    return 0;
}

void mess_you_up(string value){
    // Only messing up a COPY ... not the original
    value = "Now value is MESSED UP GOOD"
    cout << "inside mess_you_up, value = " << value << endl;
    return;
}
#include <iostream>
using namespace std;

void swapper(int, int);

int main(void) {
    int x = 5, y = 7;
    cout << "x, y = " << x << ", " << y << endl;
    cout << "calling (pass by value) swapper(x, y)" << endl;
    swapper(x, y);
    cout << "x, y = " << x << ", " << y << endl;
    return 0;
}

void swapper(int x, int y) {
    int temp = y;
    y = x;
    x = temp;
    return;
}
Getting a Reference

```
int x=5 // a variable
x   // get the value (5)
&x // get a reference to the variable (like memory address)
```

What You See

Int

X

5

Behind the Scenes

4-bytes

0x7fff5fbfее0c
PASS BY REFERENCE

```cpp
#include <iostream>
using namespace std;

void swapper(int &, int &);

int main(void) {
    int x = 5, y = 7;
    cout << "x, y = " << x << ", " << y << endl;
    cout << "calling (pass by reference) swapper(x,y)" << endl;
    swapper(x, y);
    cout << "x, y = " << x << ", " << y << endl;
    return 0;
}

void swapper(int &x, int &y) {
    int temp = y;
    y = x;
    x = temp;
    return;
}
```

NOW IT WORKS
#include <iostream>
using namespace std;
void mess_you_up(string &);
int main(void){
    string value = "the password is fibble snap";
    cout << "value = " << value << endl;
    cout << "calling (pass by reference) mess_you_up(value)" << endl;
    mess_you_up(value);
    cout << "value = " << value << endl;
    return 0;
}

void mess_you_up(string &value){
    // messing up the original
    value = "Now value is MESSED UP GOOD";
    cout << "inside mess_you_up, value = " << value << endl;
    return;
}
```cpp
#include <iostream>
using namespace std;

void square_by_ref(const int &in_x, int &out_x) {
    out_x = in_x * in_x;
    return;
}

int main(void) {
    int in_x = 5, out_x = 0;
    cout << "in_x = " << in_x << "," << "out_x = " << out_x << endl;
    cout << "Calling square_by_ref " << endl;
    square_by_ref(in_x, out_x);
    cout << "in_x = " << in_x << "," << "out_x = " << out_x << endl;
}
```
THANKS