Announcements

- HW 4 due next Friday by start of class

- Will need textbook in 1-2 weeks

(C email preferred.)

CS 180: Lecture 3
We'll use Iostream & OutputStream the most.

```
<table>
<thead>
<tr>
<th>Library</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;istream&gt;</code></td>
<td>Parent class for streams that can process input and output</td>
</tr>
<tr>
<td><code>&lt;ostream&gt;</code></td>
<td>Parent class for all output streams</td>
</tr>
<tr>
<td><code>&lt;stringstream&gt;</code></td>
<td>String stream for input and output</td>
</tr>
<tr>
<td><code>&lt;cout&gt;</code></td>
<td>Output stream</td>
</tr>
<tr>
<td><code>&lt;cin&gt;</code></td>
<td>Input/output file stream</td>
</tr>
</tbody>
</table>
```

C++ has several predefined, useful classes.
Use end instead of "in".

Can Court

> or <

Separate dishon of variables by

Notes: - QPS - Can + Can

Load's Standard Input/Output

Using namespace std;
ninclude <fstream>

Using Court + Can
Figure 7: Demonstration of console output in Python and C++. We assume that variables `first` and `last` have previously been defined as strings, and that `total` is an integer.

```cpp
C++

```
C++

```cpp
cout << team << " ranked " << rank << " out of " << total << " teams."

for each team do:
```

Python

```python
print("%s: %d of %d teams (%.2f%%) (team, rank, total)
```
Note: Precision stays set to 3.

```cpp
// pi << endl;
cout << "pi is " << fixed << setprecision(3) << pi << endl;
```

In C++:

```
pi is 3.141
```

Output:

```
pi is 3.141
```

Print:

```
pi is 3.141
```

Setting precision is harder:
C++: cin

Python: raw_input: Strings

Note (for getline): Strings a string

- input: a string

Strings up to the newline

cout. getline:

Getting (cin, person):

cout >> "What is your name?";

string person;

person = raw_input('What is your name?')
Note: Don't need to cast.

```cpp
int number;

cout << "Enter a number from 1 to 10: ";

cin >> number;

for (int i = 1; i <= 10; i++)
{
    number = i;
}
```

Python:

```python
cin = input("Enter a number from 1 to 10: ")

for i in range(1, 11):
    number = i
```

Cin: Other data types
A + b can have different types. Note: -a + b

Choosing multiple inputs:

Some other differences with cin:
A word of caution:

Ex:

```cpp
string person;
cout << "What is your name? ";
cin >> person;
```

If I type:

Erin Wolf Chambers - enter

value of person is "Erin" still waiting
empty string

food = ""/\n"/ pizza/"

age = 40

40 enters pizza store

getline (cin, food);

cout << "What would you like to eat? " << cin >> age;

if (age > 50)

cout << "How old are you? "

string food;

for age:

Another caution:
must be a C-style string.
To append:

`osstream datstream(scorestxt, ios::app);`

First, use "a" option in Python

If a file already exists, file will overwrite by default; opening ofstream overwrites output.
Much more confusing.

which allows both input & output.

This is also an "feature" object.

from

__feature__
Why?

Never:

Why?

Never:

Never:

Creating an instance of a class.

Creating an instance of a class.

Creating an instance of a class.
class Point

    double x;  // Constructor //
    double y;  // Constructor //

    explicit declaration of data members

    void setX(double val)
    { x = val; }
    void setY(double val)
    { y = val; }

    double getX() const
    { return x; }
    double getY() const
    { return y; }

    void setX(double val)
    { x = val; }
    void setY(double val)
    { y = val; }

    double getX() const
    { return x; }
    double getY() const
    { return y; }

    Point(): x(0), y(0)  // Constructor //

    public:

    double x;
    double y;

    private:

    class Point
- Empty body

(Declaration)
- Must initialize variables here
- Initialize list - x(0), y(0)
- Name of class
- No return value

Constructor:
- Constructor, not just used in constructor
- Data (public or private) is explicitly declared; not just used in constructor

Classes - Differences:
but body is still empty.

- Allows default parameters.

```
{ } // (Point(double_initialX=0.0, double_initialY=0.0) : x(initialX), y(initialY))
```

A more complicated constructor:
myPoint. setX(3);

Point myPoint = 3;
ERROR

in mean:

Access control - Public versus Private

Could use this if this - x if necessary.

Current object get understood to be attributes of
No self can just use x or y

Other things to note:
Data will give a compile error if a const member

forced by compiler.

void sex(double val);  
  
void sex(val);  

Return x;  

double get(); const

different?

Other things to note (cont):
Robust Point Class:

```java
public class Point {
    private double x; // Initial x
    private double y; // Initial y

    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }

    public double getX() {
        return x;
    }

    public double getY() {
        return y;
    }
}
```
difference

Robust Point Class Con:
* Two versions of using operators, will be x + y
- x + other, x \rightarrow allowed \text{ if \text{ inside \text{ the class}}}

Things to note:
```cpp
{ 
    std::cout << "x,y"; >> (int p.x) >> (int p.y); >> p.x; 
    std::cout >> p.y; 
    std::cout << get_p(p); 
}

// Invoke existing form with Point as left operand 

{ 
    return p * factor; // Free-standing operator definition, outside the formal Point class definition 
}; 

Point operator+(double factor, Point p); 

Additional Functions (not in class) 
```