Course Introduction

CMSC 202 - Computer Science II

Instructor & Lecture Section

- Dr. Christopher Marron
  - Section 01: MoWe, 2:30 — 3:45 pm, LH1
  - Office: ITE 359
  - Email: cmarron@umbc.edu
- Office Hours:
  - Mo 1300 – 1400
  - Tu 1430 – 1530
  - We 1300 – 1400

What is CMSC 202?

- An introduction to
  - ***Object-oriented programming*** (OOP) and ***object-oriented design*** (OOD)
  - Basic ***software engineering*** techniques
- Emphasis on *proper program design* and *maintainability*
- Tools
  - C++ programming language, GCC (Gnu Compiler)
  - Linux (GL system)

Course Web Site and Blackboard

Links to syllabus, schedule, projects, and labs:
http://www.csee.umbc.edu/courses/undergraduate/202/fall15_marron/

All grades will be posted on Blackboard.

1/12/15 4

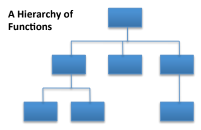
Review of the Syllabus

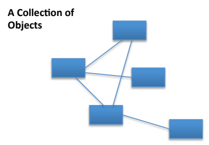
1/12/15 4

Procedural vs. OO Programming

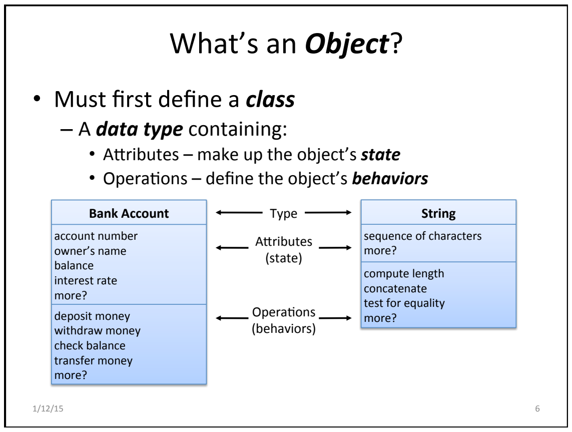
Procedural
- Modular units: functions
- Program structure: hierarchical
- Data and operations are *not* bound to each other
- Examples:
  - C, Pascal, Basic, Python

Object-Oriented (OO)
- Modular units: objects
- Program structure: a graph
- Data and operations are bound to each other
- Examples:
  - C++, Java, Python (huh?!)





1/12/15 5

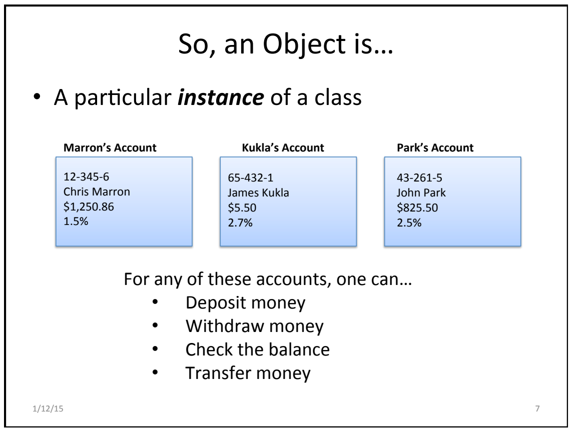


What's an ***Object***?

- Must first define a ***class***
  - A ***data type*** containing:
    - Attributes – make up the object's ***state***
    - Operations – define the object's ***behaviors***

Bank Account	← Type →	**String**
account number owner's name balance interest rate more?	← Attributes (state) →	sequence of characters more?
deposit money withdraw money check balance transfer money more?	← Operations (behaviors) →	compute length concatenate test for equality more?

1/12/15 6



So, an Object is…

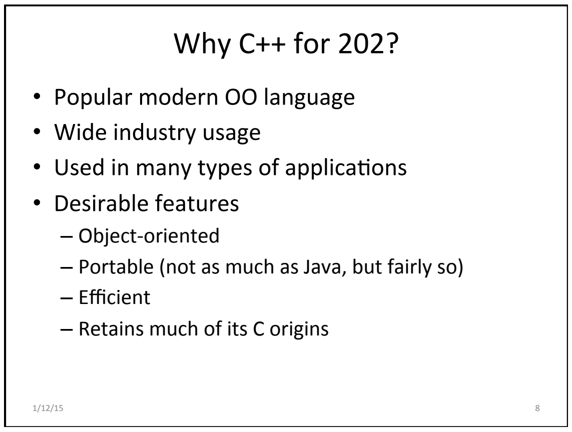
- A particular ***instance*** of a class

Marron's Account	**Kukla's Account**	**Park's Account**
12-345-6 Chris Marron $1,250.86 1.5%	65-432-1 James Kukla $5.50 2.7%	43-261-5 John Park $825.50 2.5%

For any of these accounts, one can…

- Deposit money
- Withdraw money
- Check the balance
- Transfer money

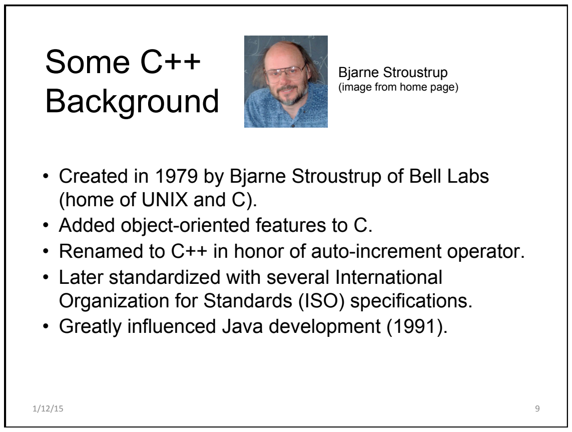
1/12/15 7



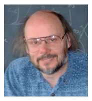
Why C++ for 202?

- Popular modern OO language
- Wide industry usage
- Used in many types of applications
- Desirable features
  - Object-oriented
  - Portable (not as much as Java, but fairly so)
  - Efficient
  - Retains much of its C origins

1/12/15 8

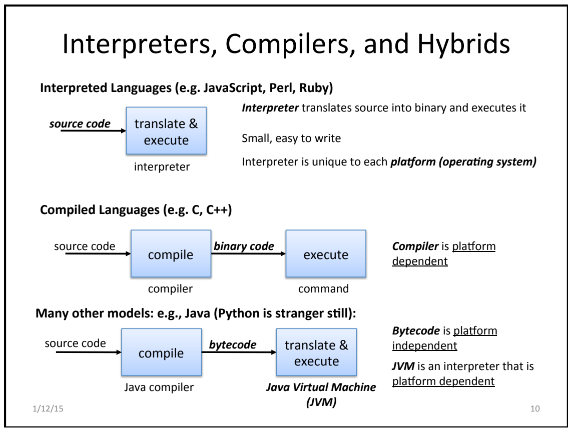


Some C++ Background

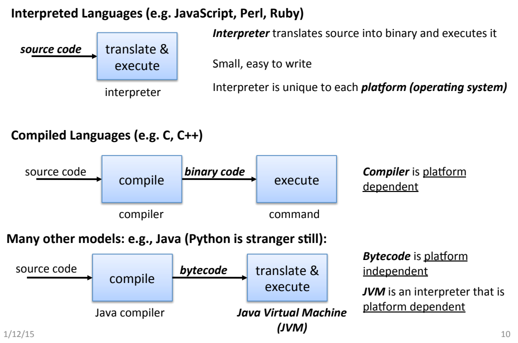

Bjarne Stroustrup
(image from home page)

- Created in 1979 by Bjarne Stroustrup of Bell Labs (home of UNIX and C).
- Added object-oriented features to C.
- Renamed to C++ in honor of auto-increment operator.
- Later standardized with several International Organization for Standards (ISO) specifications.
- Greatly influenced Java development (1991).

1/12/15 9



Interpreters, Compilers, and Hybrids



Interpreted Languages (e.g. JavaScript, Perl, Ruby)

Interpreter translates source into binary and executes it

Small, easy to write

Interpreter is unique to each ***platform (operating system)***

Compiled Languages (e.g. C, C++)

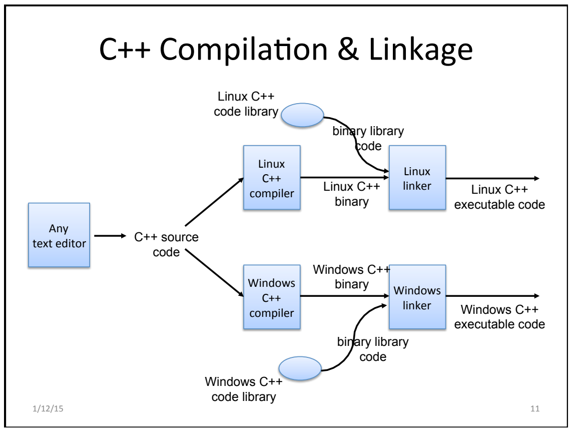
Compiler is platform dependent

Many other models: e.g., Java (Python is stranger still):

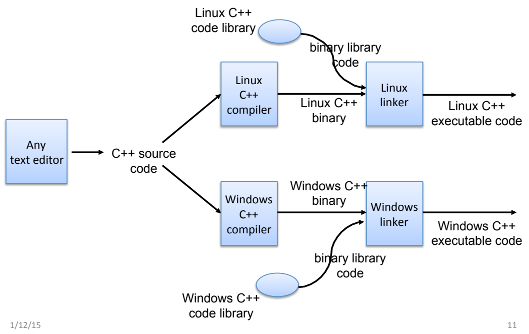
Bytecode is platform independent

JVM is an interpreter that is platform dependent

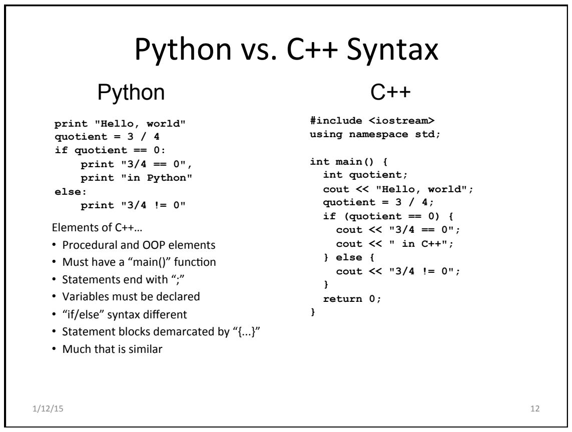
1/12/15 10



C++ Compilation & Linkage



1/12/15 11



Python vs. C++ Syntax

Python

```
print "Hello, world"
quotient = 3 / 4
if quotient == 0:
    print "3/4 == 0",
    print "in Python"
else:
    print "3/4 != 0"
```

Elements of C++…

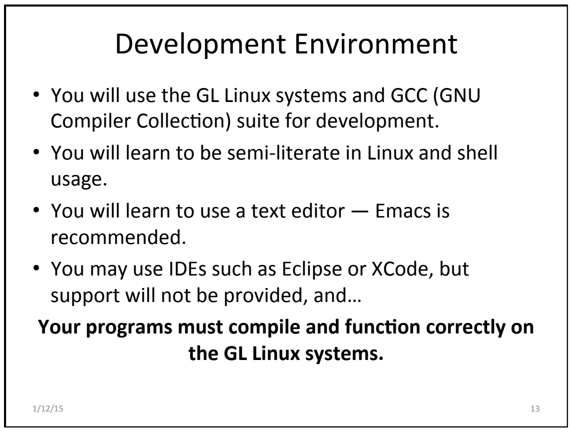
- Procedural and OOP elements
- Must have a "main()" function
- Statements end with ";"
- Variables must be declared
- "if/else" syntax different
- Statement blocks demarcated by "{...}"
- Much that is similar

C++

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

int main() {
  int quotient;
  cout << "Hello, world";
  quotient = 3 / 4;
  if (quotient == 0) {
    cout << "3/4 == 0";
    cout << " in C++";
  } else {
    cout << "3/4 != 0";
  }
  return 0;
}
```

1/12/15 12

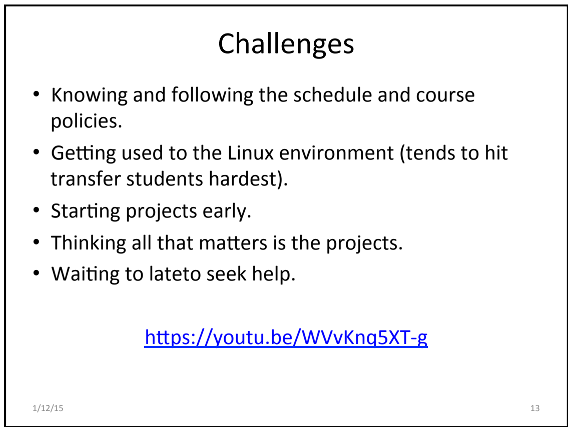


Development Environment

- You will use the GL Linux systems and GCC (GNU Compiler Collection) suite for development.
- You will learn to be semi-literate in Linux and shell usage.
- You will learn to use a text editor — Emacs is recommended.
- You may use IDEs such as Eclipse or XCode, but support will not be provided, and…

Your programs must compile and function correctly on the GL Linux systems.

1/12/15 13



Challenges

- Knowing and following the schedule and course policies.
- Getting used to the Linux environment (tends to hit transfer students hardest).
- Starting projects early.
- Thinking all that matters is the projects.
- Waiting to lateto seek help.

https://youtu.be/WVvKnq5XT-g

1/12/15 13