

CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Announcements

- Final Exam Next Class!

Frequently Asked Questions

- What's the best way to study for the final exam?

Frequently Asked Questions

- What's the best way to study for the final exam?

The final exam problems are variations on the homework, quizzes, lecture examples, and lab examples.

Frequently Asked Questions

- What's the best way to study for the final exam?

The final exam problems are variations on the homework, quizzes, lecture examples, and lab examples.

Past exams (and answer keys) are on-line. Do 7-10 previous exams: allow 1 hour and work through, grade yourself, update note sheet, and repeat.

- Why do you care about cheating?

Frequently Asked Questions

- What's the best way to study for the final exam?

The final exam problems are variations on the homework, quizzes, lecture examples, and lab examples.

Past exams (and answer keys) are on-line. Do 7-10 previous exams: allow 1 hour and work through, grade yourself, update note sheet, and repeat.

- Why do you care about cheating?

First: it gives unfair advantage & is immoral.

Frequently Asked Questions

- What's the best way to study for the final exam?

The final exam problems are variations on the homework, quizzes, lecture examples, and lab examples.

Past exams (and answer keys) are on-line. Do 7-10 previous exams: allow 1 hour and work through, grade yourself, update note sheet, and repeat.

- Why do you care about cheating?

First: it gives unfair advantage & is immoral.

Second: it degrades the quality of our students.

Frequently Asked Questions

- What's the best way to study for the final exam?

The final exam problems are variations on the homework, quizzes, lecture examples, and lab examples.

Past exams (and answer keys) are on-line. Do 7-10 previous exams: allow 1 hour and work through, grade yourself, update note sheet, and repeat.

- Why do you care about cheating?

First: it gives unfair advantage & is immoral.

Second: it degrades the quality of our students.

Third: it's a standard question on faculty references.

Frequently Asked Questions

- What's the best way to study for the final exam?

The final exam problems are variations on the homework, quizzes, lecture examples, and lab examples.

Past exams (and answer keys) are on-line. Do 7-10 previous exams: allow 1 hour and work through, grade yourself, update note sheet, and repeat.

- Why do you care about cheating?

First: it gives unfair advantage & is immoral.

Second: it degrades the quality of our students.

Third: it's a standard question on faculty references.

Industry & graduate schools hate it: don't want someone who falsifies work.

Frequently Asked Questions

- What's the best way to study for the final exam?

The final exam problems are variations on the homework, quizzes, lecture examples, and lab examples.

Past exams (and answer keys) are on-line. Do 7-10 previous exams: allow 1 hour and work through, grade yourself, update note sheet, and repeat.

- Why do you care about cheating?

First: it gives unfair advantage & is immoral.

Second: it degrades the quality of our students.

Third: it's a standard question on faculty references.

Industry & graduate schools hate it: don't want someone who falsifies work.

Today's Topics

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
    float kg, lbs;
    cout << "Enter kg: ";
    cin >> kg;
    lbs = kg * 2.2;
    cout << endl << "Lbs: " << lbs << "\n\n";
    return 0;
}
```

- Recap: I/O & Definite Loops in C++
- Conditionals in C++
- Indefinite Loops in C++
- Recap: C++ & Python
- More info on the Final Exam

Today's Topics

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
    float kg, lbs;
    cout << "Enter kg: ";
    cin >> kg;
    lbs = kg * 2.2;
    cout << endl << "Lbs: " << lbs << "\n\n";
    return 0;
}
```

- Recap: I/O & Definite Loops in C++
- Conditionals in C++
- Indefinite Loops in C++
- Recap: C++ & Python
- More info on the Final Exam

Recap: Basic Form & I/O in C++

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
    float kg, lbs;
    cout << "Enter kg: ";
    cin >> kg;
    lbs = kg * 2.2;
    cout << endl << "Lbs: " << lbs << "\n\n";
    return 0;
}
```

Recap: Basic Form & I/O in C++

- Efficient for systems programming.

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
    float kg, lbs;
    cout << "Enter kg: ";
    cin >> kg;
    lbs = kg * 2.2;
    cout << endl << "Lbs: " << lbs << "\n\n";
    return 0;
}
```

Recap: Basic Form & I/O in C++

- Efficient for systems programming.
- Programs are organized in functions.

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
    float kg, lbs;
    cout << "Enter kg: ";
    cin >> kg;
    lbs = kg * 2.2;
    cout << endl << "Lbs: " << lbs << "\n\n";
    return 0;
}
```

Recap: Basic Form & I/O in C++

- Efficient for systems programming.
- Programs are organized in functions.
- Must declare variables:

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
    float kg, lbs;
    cout << "Enter kg: ";
    cin >> kg;
    lbs = kg * 2.2;
    cout << endl << "Lbs: " << lbs << "\n\n";
    return 0;
}
```

Recap: Basic Form & I/O in C++

- Efficient for systems programming.
- Programs are organized in functions.
- Must declare variables: int num;

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
    float kg, lbs;
    cout << "Enter kg: ";
    cin >> kg;
    lbs = kg * 2.2;
    cout << endl << "Lbs: " << lbs << "\n\n";
    return 0;
}
```

Recap: Basic Form & I/O in C++

- Efficient for systems programming.
- Programs are organized in functions.
- Must declare variables: `int num;`
- Many types available:

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
    float kg, lbs;
    cout << "Enter kg: ";
    cin >> kg;
    lbs = kg * 2.2;
    cout << endl << "Lbs: " << lbs << "\n\n";
    return 0;
}
```

Recap: Basic Form & I/O in C++

- Efficient for systems programming.
- Programs are organized in functions.
- Must declare variables: `int num;`
- Many types available:
`int, float, char, ...`

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
    float kg, lbs;
    cout << "Enter kg: ";
    cin >> kg;
    lbs = kg * 2.2;
    cout << endl << "Lbs: " << lbs << "\n\n";
    return 0;
}
```

Recap: Basic Form & I/O in C++

- Efficient for systems programming.
- Programs are organized in functions.
- Must declare variables: `int num;`
- Many types available:
`int, float, char, ...`
- To print:

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
    float kg, lbs;
    cout << "Enter kg: ";
    cin >> kg;
    lbs = kg * 2.2;
    cout << endl << "Lbs: " << lbs << "\n\n";
    return 0;
}
```

Recap: Basic Form & I/O in C++

- Efficient for systems programming.
- Programs are organized in functions.
- Must declare variables: int num;
- Many types available:
int, float, char, ...
- To print: cout << "Hello!!";

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
    float kg, lbs;
    cout << "Enter kg: ";
    cin >> kg;
    lbs = kg * 2.2;
    cout << endl << "Lbs: " << lbs << "\n\n";
    return 0;
}
```

Recap: Basic Form & I/O in C++

- Efficient for systems programming.
- Programs are organized in functions.
- Must declare variables: int num;
- Many types available:
int, float, char, ...
- To print: cout << "Hello!!";
- To get input:

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
    float kg, lbs;
    cout << "Enter kg: ";
    cin >> kg;
    lbs = kg * 2.2;
    cout << endl << "Lbs: " << lbs << "\n\n";
    return 0;
}
```

Recap: Basic Form & I/O in C++

- Efficient for systems programming.
- Programs are organized in functions.
- Must declare variables: `int num;`
- Many types available:
`int, float, char, ...`
- To print: `cout << "Hello!!";`
- To get input: `cin >> num;`

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
    float kg, lbs;
    cout << "Enter kg: ";
    cin >> kg;
    lbs = kg * 2.2;
    cout << endl << "Lbs: " << lbs << "\n\n";
    return 0;
}
```

Recap: Basic Form & I/O in C++

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
    float kg, lbs;
    cout << "Enter kg: ";
    cin >> kg;
    lbs = kg * 2.2;
    cout << endl << "Lbs: " << lbs << "\n\n";
    return 0;
}
```

- Efficient for systems programming.
- Programs are organized in functions.
- Must declare variables: int num;
- Many types available:
int, float, char, ...
- To print: cout << "Hello!!";
- To get input: cin >> num;
- To use those I/O functions:

Recap: Basic Form & I/O in C++

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
    float kg, lbs;
    cout << "Enter kg: ";
    cin >> kg;
    lbs = kg * 2.2;
    cout << endl << "Lbs: " << lbs << "\n\n";
    return 0;
}
```

- Efficient for systems programming.
- Programs are organized in functions.
- Must declare variables: int num;
- Many types available:
int, float, char, ...
- To print: cout << "Hello!!";
- To get input: cin >> num;
- To use those I/O functions:
`#include <iostream>`
`using namespace std;`

Recap: Basic Form & I/O in C++

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
    float kg, lbs;
    cout << "Enter kg: ";
    cin >> kg;
    lbs = kg * 2.2;
    cout << endl << "Lbs: " << lbs << "\n\n";
    return 0;
}
```

- Efficient for systems programming.
- Programs are organized in functions.
- Must declare variables: int num;
- Many types available:
int, float, char, ...
- To print: cout << "Hello!!";
- To get input: cin >> num;
- To use those I/O functions:
`#include <iostream>`
`using namespace std;`
- Definite loops:

Recap: Basic Form & I/O in C++

- Efficient for systems programming.
- Programs are organized in functions.
- Must declare variables: int num;
- Many types available:
int, float, char, ...
- To print: cout << "Hello!!";
- To get input: cin >> num;
- To use those I/O functions:
`#include <iostream>
using namespace std;`
- Definite loops:
`for (i = 0; i < 10; i++) {...}`

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
    float kg, lbs;
    cout << "Enter kg: ";
    cin >> kg;
    lbs = kg * 2.2;
    cout << endl << "Lbs: " << lbs << "\n\n";
    return 0;
}
```

Recap: Basic Form & I/O in C++

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
    float kg, lbs;
    cout << "Enter kg: ";
    cin >> kg;
    lbs = kg * 2.2;
    cout << endl << "Lbs: " << lbs << "\n\n";
    return 0;
}
```

- Efficient for systems programming.
- Programs are organized in functions.
- Must declare variables: int num;
- Many types available:
int, float, char, ...
- To print: cout << "Hello!!";
- To get input: cin >> num;
- To use those I/O functions:
`#include <iostream>`
`using namespace std;`
- Definite loops:
`for (i = 0; i < 10; i++) {...}`
- Blocks of code uses '{' and '}'.

Recap: Basic Form & I/O in C++

- Efficient for systems programming.
- Programs are organized in functions.
- Must declare variables: `int num;`
- Many types available:
`int, float, char, ...`
- To print: `cout << "Hello!!";`
- To get input: `cin >> num;`
- To use those I/O functions:
`#include <iostream>
using namespace std;`
- Definite loops:
`for (i = 0; i < 10; i++) {...}`
- Blocks of code uses '{' and '}'.
- Commands generally end in ';'.

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
    float kg, lbs;
    cout << "Enter kg: ";
    cin >> kg;
    lbs = kg * 2.2;
    cout << endl << "Lbs: " << lbs << "\n\n";
    return 0;
}
```

Today's Topics

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
    float kg, lbs;
    cout << "Enter kg: ";
    cin >> kg;
    lbs = kg * 2.2;
    cout << endl << "Lbs: " << lbs << "\n\n";
    return 0;
}
```

- Recap: I/O & Definite Loops in C++
- **Conditionals in C++**
- Indefinite Loops in C++
- Recap: C++ & Python
- More info on the Final Exam

Challenge::

Predict what the following pieces of code will do:

```
//Demonstrates conditionals
#include <iostream>
using namespace std;

int main ()
{
    int yearBorn;
    cout << "Enter year born: ";
    cin >> yearBorn;
    if (yearBorn < 1946)
    {
        cout << "Greatest Generation";
    }
    else if (yearBorn <= 1964)
    {
        cout << "Baby Boomer";
    }
    else if (yearBorn <= 1984)
    {
        cout << "Generation X";
    }
    else if (yearBorn <= 2004)
    {
        cout << "Millennial";
    }
    else
    {
        cout << "TBD";
    }

    return 0;
}
```

```
using namespace std;

int main ()
{
    string conditions = "blowing snow";
    int winds = 100;
    float visibility = 0.2;

    if ( (winds > 35) && (visibility < 0.25) )
        (conditions == "blowing snow") ||
        (conditions == "heavy snow") )
    cout << "Blizzard!\n";

    string origin = "South Pacific";

    if (winds > 74)
        cout << "Major storm, called a ";
    if ((origin == "Indian Ocean")
        ||(origin == "South Pacific"))
        cout << "cyclone.\n";
    else if (origin == "North Pacific")
        cout << "typhoon.\n";
    else
        cout << "hurricane.\n";
```

C++ Demo

```
//Demonstrates conditionals
#include <iostream>
using namespace std;

int main ()
{
    int yearBorn;
    cout << "Enter year born: ";
    cin >> yearBorn;
    if (yearBorn < 1946)
    {
        cout << "Greatest Generation";
    }
    else if (yearBorn <= 1964)
    {
        cout << "Baby Boomer";
    }
    else if (yearBorn <= 1984)
    {
        cout << "Generation X";
    }
    else if (yearBorn <= 2004)
    {
        cout << "Millennial";
    }
    else
    {
        cout << "TBD";
    }

    return 0;
}
```

(Demo with [onlinedgdb](#))

Conditionals

General format:

```
//Demonstrates conditionals
#include <iostream>
using namespace std;

int main ()
{
    int yearBorn;
    cout << "Enter year born: ";
    cin >> yearBorn;
    if (yearBorn < 1946)
    {
        cout << "Greatest Generation";
    }
    else if (yearBorn <= 1964)
    {
        cout << "Baby Boomer";
    }
    else if (yearBorn <= 1984)
    {
        cout << "Generation X";
    }
    else if (yearBorn <= 2004)
    {
        cout << "Millennial";
    }
    else
    {
        cout << "TBD";
    }
    return 0;
}
```

```
if ( logical expression )
{
    command1;
    ...
}
else if ( logical expression )
{
    command1;
    ...
}
else
{
    command1;
    ...
}
```

Logical Operators in C++

Very similar, just different names: `&&`, `||`, and `!`:

Logical Operators in C++

Very similar, just different names: `&&`, `||`, and `!`:

and (`&&`)

in1		in2	<i>returns:</i>
False	<code>&&</code>	False	False
False	<code>&&</code>	True	False
True	<code>&&</code>	False	False
True	<code>&&</code>	True	True

Logical Operators in C++

Very similar, just different names: `&&`, `||`, and `!`:

and (`&&`)

in1		in2	<i>returns:</i>
False	<code>&&</code>	False	False
False	<code>&&</code>	True	False
True	<code>&&</code>	False	False
True	<code>&&</code>	True	True

or (`||`)

in1		in2	<i>returns:</i>
False	<code> </code>	False	False
False	<code> </code>	True	True
True	<code> </code>	False	True
True	<code> </code>	True	True

Logical Operators in C++

Very similar, just different names: `&&`, `||`, and `!`:

and (`&&`)

in1		in2	<i>returns:</i>
False	<code>&&</code>	False	False
False	<code>&&</code>	True	False
True	<code>&&</code>	False	False
True	<code>&&</code>	True	True

or (`||`)

in1		in2	<i>returns:</i>
False	<code> </code>	False	False
False	<code> </code>	True	True
True	<code> </code>	False	True
True	<code> </code>	True	True

not (`!`)

	in1	<i>returns:</i>
!	False	True
!	True	False

Today's Topics

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
    float kg, lbs;
    cout << "Enter kg: ";
    cin >> kg;
    lbs = kg * 2.2;
    cout << endl << "Lbs: " << lbs << "\n\n";
    return 0;
}
```

- Recap: I/O & Definite Loops in C++
- Conditionals in C++
- **Indefinite Loops in C++**
- Recap: C++ & Python
- More info on the Final Exam

Challenge:

Predict what the following pieces of code will do:

```
//While Growth example
#include <iostream>
using namespace std;

int main ()
{
    int population = 100;
    int year = 0;
    cout << "Year\tPopulation\n";
    while (population < 1000)
    {
        cout << year << "\t" << population << "\n";
        population = population * 2;
    }
    return 0;
}
```



C++ Demo

```
//While Growth example
#include <iostream>
using namespace std;

int main ()
{
    int population = 100;
    int year = 0;
    cout << "Year\tPopulation\n";
    while (population < 1000)
    {
        cout << year << "\t" << population << "\n";
        population = population * 2;
    }
    return 0;
}
```

(Demo with onlinegdb)

Indefinite Loops: while

```
//While Growth example
#include <iostream>
using namespace std;

int main ()
{
    int population = 100;
    int year = 0;
    cout << "Year\tPopulation\n";
    while (population < 1000)
    {
        cout << year << "\t" << population << "\n";
        population = population * 2;
    }
    return 0;
}
```

General format:

```
while ( logical expression )
{
    command1;
    command2;
    command3;
    ...
}
```

Challenge:

Predict what the following piece of code will do:

```
//Demonstrates loops
#include <iostream>
using namespace std;

int main ()
{
    int num;
    cout << "Enter an even number: ";
    cin >> num;
    while (num % 2 != 0)
    {
        cout << "\nThat's odd!\n";
        cout << "Enter an even number: ";
        cin >> num;
    }
    cout << "You entered: "
        << num << ".\n";
    return 0;
}
```

C++ Demo

```
//Demonstrates loops
#include <iostream>
using namespace std;

int main ()
{
    int num;
    cout << "Enter an even number: ";
    cin >> num;
    while (num % 2 != 0)
    {
        cout << "\nThat's odd!\n";
        cout << "Enter an even number: ";
        cin >> num;
    }
    cout << "You entered: "
        << num << ".\n";
    return 0;
}
```

(Demo with onlinegdb)

Indefinite Loops: while

```
//Demonstrates loops
#include <iostream>
using namespace std;

int main ()
{
    int num;
    cout << "Enter an even number: ";
    cin >> num;
    while (num % 2 != 0)
    {
        cout << "\nThat's odd!\n";
        cout << "Enter an even number: ";
        cin >> num;
    }
    cout << "You entered: "
        << num << ".\n";
    return 0;
}
```

General format:

```
while ( logical expression )
{
    command1;
    command2;
    command3;
    ...
}
```

Challenge:

Predict what the following pieces of code will do:

```
//Demonstrates do-while loops
#include <iostream>
using namespace std;

int main ()
{
    int num;
    do
    {
        cout << "Enter an even number: ";
        cin >> num;
    } while (num % 2 != 0);

    cout << "You entered: "
        << num << ".\n";
    return 0;
}
```

C++ Demo

```
//Demonstrates do-while loops
#include <iostream>
using namespace std;

int main ()
{
    int num;
    do
    {
        cout << "Enter an even number: ";
        cin >> num;
    } while (num % 2 != 0);

    cout << "You entered: "
        << num << ".\n";
    return 0;
}
```

(Demo with onlinegdb)

Indefinite Loops: do-while

```
//Demonstrates do-while loops
#include <iostream>
using namespace std;

int main ()
{
    int num;
    do
    {
        cout << "Enter an even number: ";
        cin >> num;
    } while (num % 2 != 0);

    cout << "You entered: "
        << num << ".\n";
    return 0;
}
```

General format:

```
do
{
    command1;
    command2;
    command3;
    ...
} while ( logical expression );
```

Today's Topics

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
    float kg, lbs;
    cout << "Enter kg: ";
    cin >> kg;
    lbs = kg * 2.2;
    cout << endl << "Lbs: " << lbs << "\n\n";
    return 0;
}
```

- Recap: I/O & Definite Loops in C++
- Conditionals in C++
- Indefinite Loops in C++
- **Recap: C++ & Python**
- More info on the Final Exam

Recap: C++ Control Structures

- I/O:

```
//Another C++ program; Demonstrates loops
#include <iostream>
using namespace std;

int main ()
{
    int i,j;
    for (i = 0; i < 4; i++)
    {
        cout << "The world turned upside down...\n";
    }

    for (j = 10; j > 0; j--)
    {
        cout << j << " ";
    }
    cout << "Blast off!!" << endl;
    return 0;
}
```

Recap: C++ Control Structures

- I/O: `cin >> ...;`

```
//Another C++ program; Demonstrates loops
#include <iostream>
using namespace std;

int main ()
{
    int i,j;
    for (i = 0; i < 4; i++)
    {
        cout << "The world turned upside down...\n";
    }

    for (j = 10; j > 0; j--)
    {
        cout << j << " ";
    }
    cout << "Blast off!!" << endl;
    return 0;
}
```

Recap: C++ Control Structures

- I/O: `cin >> ...;` & `cout << ...;`

```
//Another C++ program; Demonstrates loops
#include <iostream>
using namespace std;

int main ()
{
    int i,j;
    for (i = 0; i < 4; i++)
    {
        cout << "The world turned upside down...\n";
    }

    for (j = 10; j > 0; j--)
    {
        cout << j << " ";
    }
    cout << "Blast off!!" << endl;
    return 0;
}
```

Recap: C++ Control Structures

- I/O: `cin >> ...;` & `cout << ...;`
- Definite loops:

```
//Another C++ program; Demonstrates loops
#include <iostream>
using namespace std;

int main ()
{
    int i,j;
    for (i = 0; i < 4; i++)
    {
        cout << "The world turned upside down...\n";
    }

    for (j = 10; j > 0; j--)
    {
        cout << j << " ";
    }
    cout << "Blast off!!" << endl;
    return 0;
}
```

Recap: C++ Control Structures

- I/O: `cin >> ...;` & `cout << ...;`
- Definite loops:
`for (i = 0; i < 10; i++)`
`{`
 ...
`}`

```
//Another C++ program; Demonstrates loops
#include <iostream>
using namespace std;

int main ()
{
    int i,j;
    for (i = 0; i < 4; i++)
    {
        cout << "The world turned upside down...\n";
    }

    for (j = 10; j > 0; j--)
    {
        cout << j << " ";
    }
    cout << "Blast off!!" << endl;
    return 0;
}
```

Recap: C++ Control Structures

- I/O: `cin >> ...;` & `cout << ...;`

- Definite loops:

```
for (i = 0; i < 10; i++)  
{  
    ...  
}
```

- Conditionals:

```
//Another C++ program; Demonstrates loops  
#include <iostream>  
using namespace std;  
  
int main ()  
{  
    int i,j;  
    for (i = 0; i < 4; i++)  
    {  
        cout << "The world turned upside down...\n";  
    }  
  
    for (j = 10; j > 0; j--)  
    {  
        cout << j << " ";  
    }  
    cout << "Blast off!!" << endl;  
  
    return 0;  
}
```

Recap: C++ Control Structures

- I/O: `cin >> ...;` & `cout << ...;`

- Definite loops:

```
for (i = 0; i < 10; i++)  
{  
    ...  
}  
}
```

- Conditionals:

```
if (logical expression)  
{  
    ...  
}  
else  
{  
    ...  
}
```

```
//Another C++ program; Demonstrates loops  
#include <iostream>  
using namespace std;  
  
int main ()  
{  
    int i,j;  
    for (i = 0; i < 4; i++)  
    {  
        cout << "The world turned upside down...\n";  
    }  
  
    for (j = 10; j > 0; j--)  
    {  
        cout << j << " ";  
    }  
    cout << "Blast off!!" << endl;  
    return 0;  
}
```

Recap: C++ Control Structures

- I/O: `cin >> ...;` & `cout << ...;`

- Definite loops:

```
for (i = 0; i < 10; i++)  
{  
    ...  
}  
}
```

- Conditionals:

```
if (logical expression)  
{  
    ...  
}  
else  
{  
    ...  
}
```

```
//Another C++ program; Demonstrates loops  
#include <iostream>  
using namespace std;  
  
int main ()  
{  
    int i,j;  
    for (i = 0; i < 4; i++)  
    {  
        cout << "The world turned upside down...\n";  
    }  
  
    for (j = 10; j > 0; j--)  
    {  
        cout << j << " ";  
    }  
    cout << "Blast off!!" << endl;  
  
    return 0;  
}
```

- Indefinite loops:

Recap: C++ Control Structures

- I/O: `cin >> ...;` & `cout << ...;`

- Definite loops:

```
for (i = 0; i < 10; i++)  
{  
    ...  
}  
}
```

- Conditionals:

```
if (logical expression)  
{  
    ...  
}  
else  
{  
    ...  
}
```

- Indefinite loops:

```
while (logical expression)  
{  
    ...  
}
```

Challenge: Definite Loops in Python & C++

- Rewrite this program in C++:

```
for i in range(2017, 2000, -2):
    print("Year is", i)
```

- Rewrite this program in Python:

```
#include <iostream>
using namespace std;
int main()
{
    for (int i = 1; i < 50; i++)
    {
        cout << i << endl;
    }
    return 0;
}
```

Challenge:: Definite Loops in Python & C++

- Rewrite this program in C++:

```
for i in range(2017, 2000, -2):  
    print("Year is", i)
```

Challenge:: Definite Loops in Python & C++

- Rewrite this program in C++:

```
for i in range(2017, 2000, -2):  
    print("Year is", i)  
  
#include <iostream>  
using namespace std;
```

Challenge:: Definite Loops in Python & C++

- Rewrite this program in C++:

```
for i in range(2017, 2000, -2):  
    print("Year is", i)
```

```
#include <iostream>  
using namespace std;  
int main()
```

Challenge:: Definite Loops in Python & C++

- Rewrite this program in C++:

```
for i in range(2017, 2000, -2):  
    print("Year is", i)
```

```
#include <iostream>  
using namespace std;  
int main()  
{
```

Challenge:: Definite Loops in Python & C++

- Rewrite this program in C++:

```
for i in range(2017, 2000, -2):  
    print("Year is", i)
```

```
#include <iostream>  
using namespace std;  
int main()  
{  
    for (int i = 2017; i >= 2000; i=i-2)
```

Challenge:: Definite Loops in Python & C++

- Rewrite this program in C++:

```
for i in range(2017, 2000, -2):
    print("Year is", i)

#include <iostream>
using namespace std;
int main()
{
    for (int i = 2017; i >= 2000; i=i-2)
    {
        cout << "Year is" << i << endl;
```

Challenge:: Definite Loops in Python & C++

- Rewrite this program in C++:

```
for i in range(2017, 2000, -2):
    print("Year is", i)

#include <iostream>
using namespace std;
int main()
{
    for (int i = 2017; i >= 2000; i=i-2)
    {
        cout << "Year is" << i << endl;
    }
    return 0;
}
```

Challenge:: Definite Loops in Python & C++

- Rewrite this program in Python:

```
#include <iostream>
using namespace std;
int main()
{
    for (int i = 1; i < 50; i++)
    {
        cout << i << endl;
    }
    return 0;
}
```

Challenge:: Definite Loops in Python & C++

- Rewrite this program in Python:

```
#include <iostream>
using namespace std;
int main()
{
    for (int i = 1; i < 50; i++)
    {
        cout << i << endl;
    }
    return 0;
}
```

```
for i in range(1, 50):
```

Challenge:: Definite Loops in Python & C++

- Rewrite this program in Python:

```
#include <iostream>
using namespace std;
int main()
{
    for (int i = 1; i < 50; i++)
    {
        cout << i << endl;
    }
    return 0;
}

for i in range(1, 50):
    print(i)
```

Challenge:: Conditionals in Python & C++

- *Python: what is the output?*

```
year = 2016
if year % 4 == 0 and \
    (not (year % 100 == 0) or (year % 400 == 0)):
    print("Leap!!")
print("Year")
```

- *Write a C++ program that asks the user the number of times they plan to ride transit this week. Your program should then print if it is cheaper to buy single ride metro cards or 7-day unlimited card.*
(The 7-day card is \$33.00, and the cost of single ride, with bonus, is \$2.75).

Challenge:: Conditionals in Python & C++

- *Python: what is the output?*

```
year = 2016
if year % 4 == 0 and \
    (not (year % 100 == 0) or (year % 400 == 0)):
    print("Leap!!")
print("Year")
```

Challenge:: Conditionals in Python & C++

- *Python: what is the output?*

```
year = 2016
if year % 4 == 0 and \
    (not (year % 100 == 0) or (year % 400 == 0)):
    print("Leap!!")
print("Year")  year = 2016
```

```
if TRUE and \
    (not (year % 100 == 0) or (year % 400 == 0)):
    print("Leap!!")
print("Year")
```

Challenge:: Conditionals in Python & C++

- *Python: what is the output?*

```
year = 2016
if year % 4 == 0 and \
    (not (year % 100 == 0) or (year % 400 == 0)):
    print("Leap!!")
print("Year")
```

Challenge:: Conditionals in Python & C++

- *Python: what is the output?*

```
year = 2016
if year % 4 == 0 and \
    (not (year % 100 == 0) or (year % 400 == 0)):
    print("Leap!!")
print("Year")
```

```
year = 2016
if TRUE and \
    (not FALSE or (year % 400 == 0)):
    print("Leap!!")
print("Year")
```

Challenge:: Conditionals in Python & C++

- *Python: what is the output?*

```
year = 2016
if year % 4 == 0 and \
    (not (year % 100 == 0) or (year % 400 == 0)):
    print("Leap!!")
print("Year")
```

Challenge:: Conditionals in Python & C++

- *Python: what is the output?*

```
year = 2016
if year % 4 == 0 and \
    (not (year % 100 == 0) or (year % 400 == 0)):
    print("Leap!!")
print("Year")
```

```
year = 2016
if TRUE and \
    (TRUE or (year % 400 == 0)):
    print("Leap!!")
print("Year")
```

Challenge:: Conditionals in Python & C++

- *Python: what is the output?*

```
year = 2016
if year % 4 == 0 and \
    (not (year % 100 == 0) or (year % 400 == 0)):
    print("Leap!!")
print("Year")
```

Challenge:: Conditionals in Python & C++

- *Python: what is the output?*

```
year = 2016
if year % 4 == 0 and \
    (not (year % 100 == 0) or (year % 400 == 0)):
    print("Leap!!")
print("Year")
```

```
year = 2016
if TRUE and \
    (TRUE or FALSE):
    print("Leap!!")
print("Year")
```

Challenge:: Conditionals in Python & C++

- *Python: what is the output?*

```
year = 2016
if year % 4 == 0 and \
    (not (year % 100 == 0) or (year % 400 == 0)):
    print("Leap!!")
print("Year")
```

```
year = 2016
if TRUE and \
    (TRUE or FALSE):
    print("Leap!!")
print("Year")
```

Challenge:: Conditionals in Python & C++

- *Python: what is the output?*

```
year = 2016
if year % 4 == 0 and \
    (not (year % 100 == 0) or (year % 400 == 0)):
    print("Leap!!")
print("Year")
```

```
year = 2016
if TRUE and \
    (TRUE):
    print("Leap!!")
print("Year")
```

Challenge:: Conditionals in Python & C++

- *Python: what is the output?*

```
year = 2016
if year % 4 == 0 and \
    (not (year % 100 == 0) or (year % 400 == 0)):
    print("Leap!!")
print("Year")
```

```
year = 2016
if TRUE:
    print("Leap!!")
print("Year")
```

Challenge:: Conditionals in Python & C++

- *Python: what is the output?*

```
year = 2016
if year % 4 == 0 and \
    (not (year % 100 == 0) or (year % 400 == 0)):
    print("Leap!!")
print("Year")
```

```
year = 2016
if TRUE:
    print("Leap!!")
print("Year")
```

Prints: Leap!
Year

Challenge:: Conditionals in Python & C++

- Your program should then print if it is cheaper to buy single ride metro cards (\$2.75 per ride) or 7-day unlimited card (\$33.00).

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

Challenge:: Conditionals in Python & C++

- Your program should then print if it is cheaper to buy single ride metro cards (\$2.75 per ride) or 7-day unlimited card (\$33.00).

```
#include <iostream>
using namespace std;
int main()
```

Challenge:: Conditionals in Python & C++

- Your program should then print if it is cheaper to buy single ride metro cards (\$2.75 per ride) or 7-day unlimited card (\$33.00).

```
#include <iostream>
using namespace std;
int main()
{
    int rides;
```

Challenge:: Conditionals in Python & C++

- Your program should then print if it is cheaper to buy single ride metro cards (\$2.75 per ride) or 7-day unlimited card (\$33.00).

```
#include <iostream>
using namespace std;
int main()
{
    int rides;
    cout << "Enter number of rides:";
```

Challenge:: Conditionals in Python & C++

- Your program should then print if it is cheaper to buy single ride metro cards (\$2.75 per ride) or 7-day unlimited card (\$33.00).

```
#include <iostream>
using namespace std;
int main()
{
    int rides;
    cout << "Enter number of rides:";
    cin >> rides;
```

Challenge:: Conditionals in Python & C++

- Your program should then print if it is cheaper to buy single ride metro cards (\$2.75 per ride) or 7-day unlimited card (\$33.00).

```
#include <iostream>
using namespace std;
int main()
{
    int rides;
    cout << "Enter number of rides:";
    cin >> rides;
    if (2.75 * rides < 33.00)
```

Challenge:: Conditionals in Python & C++

- Your program should then print if it is cheaper to buy single ride metro cards (\$2.75 per ride) or 7-day unlimited card (\$33.00).

```
#include <iostream>
using namespace std;
int main()
{
    int rides;
    cout << "Enter number of rides:";
    cin >> rides;
    if (2.75 * rides < 33.00)
    {
        cout << "Cheaper to buy single ride metro cards.\n";
    }
}
```

Challenge:: Conditionals in Python & C++

- Your program should then print if it is cheaper to buy single ride metro cards (\$2.75 per ride) or 7-day unlimited card (\$33.00).

```
#include <iostream>
using namespace std;
int main()
{
    int rides;
    cout << "Enter number of rides:";
    cin >> rides;
    if (2.75 * rides < 33.00)
    {
        cout << "Cheaper to buy single ride metro cards.\n";
    }
    else
```

Challenge:: Conditionals in Python & C++

- Your program should then print if it is cheaper to buy single ride metro cards (\$2.75 per ride) or 7-day unlimited card (\$33.00).

```
#include <iostream>
using namespace std;
int main()
{
    int rides;
    cout << "Enter number of rides:";
    cin >> rides;
    if (2.75 * rides < 33.00)
    {
        cout << "Cheaper to buy single ride metro cards.\n";
    }
    else
    {
        cout << "Cheaper to buy 7-day unlimited card.\n";
    }
}
```

Challenge:: Conditionals in Python & C++

- Your program should then print if it is cheaper to buy single ride metro cards (\$2.75 per ride) or 7-day unlimited card (\$33.00).

```
#include <iostream>
using namespace std;
int main()
{
    int rides;
    cout << "Enter number of rides:";
    cin >> rides;
    if (2.75 * rides < 33.00)
    {
        cout << "Cheaper to buy single ride metro cards.\n";
    }
    else
    {
        cout << "Cheaper to buy 7-day unlimited card.\n";
    }
    return 0;
}
```

Challenge:: Indefinite Loops in Python & C++

- Write Python code that repeatedly prompts for a non-empty string.
- Write C++ code that repeatedly prompts until an odd number is entered.

Challenge:: Indefinite Loops in Python & C++

- Write Python code that repeatedly prompts for a non-empty string.

Challenge:: Indefinite Loops in Python & C++

- Write Python code that repeatedly prompts for a non-empty string.

```
s = ""
```

Challenge:: Indefinite Loops in Python & C++

- Write Python code that repeatedly prompts for a non-empty string.

```
s = ""  
while s == "":
```

Challenge:: Indefinite Loops in Python & C++

- Write Python code that repeatedly prompts for a non-empty string.

```
s = ""  
while s == "":  
    s = input("Enter a non-empty string: ")
```

Challenge:: Indefinite Loops in Python & C++

- Write Python code that repeatedly prompts for a non-empty string.

```
s = ""  
while s == "":  
    s = input("Enter a non-empty string: ")  
print("You entered: ", s)
```

Challenge:: Indefinite Loops in Python & C++

- Write Python code that repeatedly prompts for a non-empty string.

```
s = ""  
while s == "":  
    s = input("Enter a non-empty string: ")  
print("You entered: ", s)
```

- Write C++ code that repeatedly prompts until an odd number is entered.

Challenge:: Indefinite Loops in Python & C++

- Write Python code that repeatedly prompts for a non-empty string.

```
s = ""  
while s == "":  
    s = input("Enter a non-empty string: ")  
print("You entered: ", s)
```

- Write C++ code that repeatedly prompts until an odd number is entered.

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

Challenge:: Indefinite Loops in Python & C++

- Write Python code that repeatedly prompts for a non-empty string.

```
s = ""  
while s == "":  
    s = input("Enter a non-empty string: ")  
print("You entered: ", s)
```

- Write C++ code that repeatedly prompts until an odd number is entered.

```
#include <iostream>  
using namespace std;  
int main()
```

Challenge:: Indefinite Loops in Python & C++

- Write Python code that repeatedly prompts for a non-empty string.

```
s = ""  
while s == "":  
    s = input("Enter a non-empty string: ")  
print("You entered: ", s)
```

- Write C++ code that repeatedly prompts until an odd number is entered.

```
#include <iostream>  
using namespace std;  
int main()  
{  
    int num = 0;
```

Challenge:: Indefinite Loops in Python & C++

- Write Python code that repeatedly prompts for a non-empty string.

```
s = ""  
while s == "":  
    s = input("Enter a non-empty string: ")  
print("You entered: ", s)
```

- Write C++ code that repeatedly prompts until an odd number is entered.

```
#include <iostream>  
using namespace std;  
int main()  
{  
    int num = 0;  
    while (num % 2 == 0)
```

Challenge:: Indefinite Loops in Python & C++

- Write Python code that repeatedly prompts for a non-empty string.

```
s = ""  
while s == "":  
    s = input("Enter a non-empty string: ")  
print("You entered: ", s)
```

- Write C++ code that repeatedly prompts until an odd number is entered.

```
#include <iostream>  
using namespace std;  
int main()  
{  
    int num = 0;  
    while (num % 2 == 0)  
    {  
        cout << "Enter an odd number:";
```

Challenge:: Indefinite Loops in Python & C++

- Write Python code that repeatedly prompts for a non-empty string.

```
s = ""  
while s == "":  
    s = input("Enter a non-empty string: ")  
print("You entered: ", s)
```

- Write C++ code that repeatedly prompts until an odd number is entered.

```
#include <iostream>  
using namespace std;  
int main()  
{  
    int num = 0;  
    while (num % 2 == 0)  
    {  
        cout << "Enter an odd number:";  
        cin >> num;
```

Challenge:: Indefinite Loops in Python & C++

- Write Python code that repeatedly prompts for a non-empty string.

```
s = ""  
while s == "":  
    s = input("Enter a non-empty string: ")  
print("You entered: ", s)
```

- Write C++ code that repeatedly prompts until an odd number is entered.

```
#include <iostream>  
using namespace std;  
int main()  
{  
    int num = 0;  
    while (num % 2 == 0)  
    {  
        cout << "Enter an odd number:";  
        cin >> num;  
    }
```

Challenge:: Indefinite Loops in Python & C++

- Write Python code that repeatedly prompts for a non-empty string.

```
s = ""  
while s == "":  
    s = input("Enter a non-empty string: ")  
print("You entered: ", s)
```

- Write C++ code that repeatedly prompts until an odd number is entered.

```
#include <iostream>  
using namespace std;  
int main()  
{  
    int num = 0;  
    while (num % 2 == 0)  
    {  
        cout << "Enter an odd number:";  
        cin >> num;  
    }  
    return 0;  
}
```

Today's Topics

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;

int main ()
{
    float kg, lbs;
    cout << "Enter kg: ";
    cin >> kg;
    lbs = kg * 2.2;
    cout << endl << "Lbs: " << lbs << "\n\n";
    return 0;
}
```

- Recap: I/O & Definite Loops in C++
- Conditionals in C++
- Indefinite Loops in C++
- Recap: C++ & Python
- **More info on the Final Exam**

Final Overview: Format

- Although the exam is remote, we still suggest you prepare 1 piece of **8.5" x 11"** paper.

Final Overview: Format

- Although the exam is remote, we still suggest you prepare 1 piece of **8.5" x 11"** paper.
 - ▶ With notes, examples, programs: what will help you on the exam.

Final Overview: Format

- Although the exam is remote, we still suggest you prepare 1 piece of **8.5" x 11"** paper.
 - ▶ With notes, examples, programs: what will help you on the exam.
 - ▶ Best if you design/write yours since excellent way to study.

Final Overview: Format

- Although the exam is remote, we still suggest you prepare 1 piece of **8.5" x 11"** paper.
 - ▶ With notes, examples, programs: what will help you on the exam.
 - ▶ Best if you design/write yours since excellent way to study.
- The exam format:

Final Overview: Format

- Although the exam is remote, we still suggest you prepare 1 piece of **8.5" x 11"** paper.
 - ▶ With notes, examples, programs: what will help you on the exam.
 - ▶ Best if you design/write yours since excellent way to study.
- The exam format:
 - ▶ 10 questions, each worth 10 points.

Final Overview: Format

- Although the exam is remote, we still suggest you prepare 1 piece of **8.5" x 11"** paper.
 - ▶ With notes, examples, programs: what will help you on the exam.
 - ▶ Best if you design/write yours since excellent way to study.
- The exam format:
 - ▶ 10 questions, each worth 10 points.
 - ▶ Questions correspond to the course topics, and are variations on the programming assignments, lab exercises, and lecture design challenges.

Final Overview: Format

- Although the exam is remote, we still suggest you prepare 1 piece of **8.5" x 11"** paper.
 - ▶ With notes, examples, programs: what will help you on the exam.
 - ▶ Best if you design/write yours since excellent way to study.
- The exam format:
 - ▶ 10 questions, each worth 10 points.
 - ▶ Questions correspond to the course topics, and are variations on the programming assignments, lab exercises, and lecture design challenges.
 - ▶ Style of questions: what does the code do? short answer, write functions, top down design, & write complete programs.

Final Overview: Format

- Although the exam is remote, we still suggest you prepare 1 piece of **8.5" x 11"** paper.
 - ▶ With notes, examples, programs: what will help you on the exam.
 - ▶ Best if you design/write yours since excellent way to study.
- The exam format:
 - ▶ 10 questions, each worth 10 points.
 - ▶ Questions correspond to the course topics, and are variations on the programming assignments, lab exercises, and lecture design challenges.
 - ▶ Style of questions: what does the code do? short answer, write functions, top down design, & write complete programs.
- Past exams available on webpage (includes answer keys).

How to Prepare

- Emphasis of this course is on analytic reasoning and problem solving.



How to Prepare

- Emphasis of this course is on analytic reasoning and problem solving.
- The best way to prepare to do problems (reading & watching videos can clarify but not replace problem solving).



How to Prepare

- Emphasis of this course is on analytic reasoning and problem solving.
- The best way to prepare to do problems (reading & watching videos can clarify but not replace problem solving).
- Repeat, while there are past exams:



How to Prepare



- Emphasis of this course is on analytic reasoning and problem solving.
- The best way to prepare to do problems (reading & watching videos can clarify but not replace problem solving).
- Repeat, while there are past exams:
 - ▶ Choose a past exam (see webpage).

How to Prepare



- Emphasis of this course is on analytic reasoning and problem solving.
- The best way to prepare to do problems (reading & watching videos can clarify but not replace problem solving).
- Repeat, while there are past exams:
 - ▶ Choose a past exam (see webpage).
 - ▶ With only a note sheet, work through in 1 hour (half the time).

How to Prepare



- Emphasis of this course is on analytic reasoning and problem solving.
- The best way to prepare to do problems (reading & watching videos can clarify but not replace problem solving).
- Repeat, while there are past exams:
 - ▶ Choose a past exam (see webpage).
 - ▶ With only a note sheet, work through in 1 hour (half the time).
 - ▶ Grade yourself (answers on webpage).

How to Prepare



- Emphasis of this course is on analytic reasoning and problem solving.
- The best way to prepare to do problems (reading & watching videos can clarify but not replace problem solving).
- Repeat, while there are past exams:
 - ▶ Choose a past exam (see webpage).
 - ▶ With only a note sheet, work through in 1 hour (half the time).
 - ▶ Grade yourself (answers on webpage).
 - ▶ Ask about those that don't make sense.

How to Prepare



- Emphasis of this course is on analytic reasoning and problem solving.
- The best way to prepare to do problems (reading & watching videos can clarify but not replace problem solving).
- Repeat, while there are past exams:
 - ▶ Choose a past exam (see webpage).
 - ▶ With only a note sheet, work through in 1 hour (half the time).
 - ▶ Grade yourself (answers on webpage).
 - ▶ Ask about those that don't make sense.
 - ▶ Rewrite answers & organize by type/question number.

How to Prepare



- Emphasis of this course is on analytic reasoning and problem solving.
- The best way to prepare to do problems (reading & watching videos can clarify but not replace problem solving).
- Repeat, while there are past exams:
 - ▶ Choose a past exam (see webpage).
 - ▶ With only a note sheet, work through in 1 hour (half the time).
 - ▶ Grade yourself (answers on webpage).
 - ▶ Ask about those that don't make sense.
 - ▶ Rewrite answers & organize by type/question number.
 - ▶ Adjust/rewrite note sheet to include what you wished you had.

How to Prepare



- Emphasis of this course is on analytic reasoning and problem solving.
- The best way to prepare to do problems (reading & watching videos can clarify but not replace problem solving).
- Repeat, while there are past exams:
 - ▶ Choose a past exam (see webpage).
 - ▶ With only a note sheet, work through in 1 hour (half the time).
 - ▶ Grade yourself (answers on webpage).
 - ▶ Ask about those that don't make sense.
 - ▶ Rewrite answers & organize by type/question number.
 - ▶ Adjust/rewrite note sheet to include what you wished you had.
- Aim to complete 7 to 10 past exams (one a day in the week leading up to the final).

Final Overview: Rules

You will get credit for your answers **only if:**

Final Overview: Rules

You will get credit for your answers **only if:**

- Your answer uses language constructs that were covered in the course.

Final Overview: Rules

You will get credit for your answers **only if:**

- Your answer uses language constructs that were covered in the course.
- Even if your answer is correct, it will get 0 points if the method was not covered in this course.

Final Overview: Rules

You will get credit for your answers **only if:**

- Your answer uses language constructs that were covered in the course.
- Even if your answer is correct, it will get 0 points if the method was not covered in this course.
- Your answer is not obviously copy/pasted from a website.

Final Overview: Rules

You will get credit for your answers **only if:**

- Your answer uses language constructs that were covered in the course.
- Even if your answer is correct, it will get 0 points if the method was not covered in this course.
- Your answer is not obviously copy/pasted from a website.
- Your answer is not oddly identically to that of another student.

Final Overview: Rules

You will get credit for your answers **only if:**

- Your answer uses language constructs that were covered in the course.
- Even if your answer is correct, it will get 0 points if the method was not covered in this course.
- Your answer is not obviously copy/pasted from a website.
- Your answer is not oddly identically to that of another student.

All acts of academic dishonesty will be reported to the Office of Academic and Student Affairs

Final Exam Practice Rounds:

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- **Write a function that takes a weight in kilograms and returns the weight in pounds.**

Final Exam Practice Rounds:

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a weight in kilograms and returns the weight in pounds.

```
def kg2lbs(kg):  
    ...  
    return(lbs)
```

Final Exam Practice Rounds:

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a weight in kilograms and returns the weight in pounds.

```
def kg2lbs(kg)
    lbs = kg * 2.2
    return(lbs)
```

Final Exam Practice Rounds:

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- **Write a function that takes a string and returns its length.**

Final Exam Practice Rounds:

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a string and returns its length.

```
def sLength(str):  
    ...  
    return(length)
```

Final Exam Practice Rounds:

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a string and returns its length.

```
def sLength(str):  
    length = len(str)  
    return(length)
```

Final Exam Practice Rounds:

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that, given a DataFrame, returns the minimal value in the “Manhattan” column.

Final Exam Practice Rounds:

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that, given a DataFrame, returns the minimal value in the “Manhattan” column.

```
def getMin(df):  
    ...  
    return(min)
```

Final Exam Practice Rounds:

For each question below, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that, given a DataFrame, returns the minimal value in the “Manhattan” column.

```
def getMin(df):  
    min = df['Manhattan'].min()  
    return(min)
```

Final Exam Practice Rounds:

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- **Write a function that takes a whole number and returns the corresponding binary number as a string.**

Final Exam Practice Rounds:

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a whole number and returns the corresponding binary number as a string.

```
def num2bin(num):  
    ...  
    return(bin)
```

Final Exam Practice Rounds:

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that takes a whole number and returns the corresponding binary number as a string.

```
def num2bin(num):  
    binStr = ""  
    while (num > 0):  
        #Divide by 2, and add the remainder to the string  
        r = num %2  
        binString = str(r) + binStr  
        num = num / 2  
    return(binStr)
```

Final Exam Practice Rounds:

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that computes the total monthly payment when given the initial loan amount, annual interest rate, number of years of the loan.

Final Exam Practice Rounds:

For each question, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that computes the total monthly payment when given the initial loan amount, annual interest rate, number of years of the loan.

```
def computePayment(loan,rate,year):  
    ....  
    return(payment)
```

Final Exam Practice Rounds:

For each question below, write the function header (name & inputs) and return values (often called the Application Programming Interface (API)):

- Write a function that computes the total monthly payment when given the initial loan amount, annual interest rate, number of years of the loan.

```
def computePayment(loan,rate,year):  
    (Some formula for payment)  
    return(payment)
```