CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Today's Topics



- Recap: Incrementer Design Challenge
- C++: Basic Format & Variables
- $\bullet\,$ I/O and Definite Loops in C++

Today's Topics



- Recap: Incrementer Design Challenge
- C++: Basic Format & Variables
- $\bullet\,$ I/O and Definite Loops in C++





• Simplest arithmetic: add one ("increment") a variable.





- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:



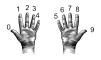


- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
def addOne(n):
    m = n+1
    return(m)
```

Summer 2020

4 / 30





- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
def addOne(n):
    m = n+1
    return(m)
```

Challenge: Write an algorithm for incrementing numbers expressed as words.





- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
def addOne(n):
    m = n+1
    return(m)
```

ullet Challenge: Write an algorithm for incrementing numbers expressed as words. Example: "forty one" \to "forty two"



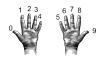


- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
def addOne(n):
    m = n+1
    return(m)
```

Challenge: Write an algorithm for incrementing numbers expressed as words.
 Example: "forty one" → "forty two"

Hint: Convert to numbers, increment, and convert back to strings.





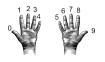
- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
def addOne(n):
    m = n+1
    return(m)
```

Challenge: Write an algorithm for incrementing numbers expressed as words.
 Example: "forty one" → "forty two"

Hint: Convert to numbers, increment, and convert back to strings.

• Challenge: Write an algorithm for incrementing binary numbers.





- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

```
def addOne(n):
    m = n+1
    return(m)
```

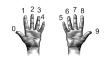
Challenge: Write an algorithm for incrementing numbers expressed as words.

Example: "forty one" \rightarrow "forty two"

Hint: Convert to numbers, increment, and convert back to strings.

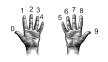
• Challenge: Write an algorithm for incrementing binary numbers.

Example: "1001" \to "1010"



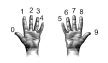


 \bullet Challenge: Write an algorithm for incrementing numbers expressed as words. Example: "forty one" \to "forty two"





- \bullet Challenge: Write an algorithm for incrementing numbers expressed as words. Example: "forty one" \to "forty two"
- Challenge: Write an algorithm for incrementing binary numbers. Example: "1001" \rightarrow "1010"
- Hint: Convert to numbers, increment, and convert back to strings.

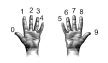




- ullet Challenge: Write an algorithm for incrementing numbers expressed as words. Example: "forty one" o "forty two"
- Challenge: Write an algorithm for incrementing binary numbers. Example: "1001" \rightarrow "1010"
- Hint: Convert to numbers, increment, and convert back to strings.

Pseudocode same for both questions:

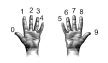
Get user input.





- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: "forty one" \rightarrow "forty two"
- Challenge: Write an algorithm for incrementing binary numbers. Example: "1001" \to "1010"
- Hint: Convert to numbers, increment, and convert back to strings.

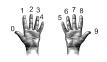
- Get user input.
- Convert to standard decimal number.





- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: "forty one" \rightarrow "forty two"
- Challenge: Write an algorithm for incrementing binary numbers. Example: "1001" \to "1010"
- Hint: Convert to numbers, increment, and convert back to strings.

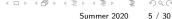
- Get user input.
- Convert to standard decimal number.
- Add one (increment) the standard decimal number.

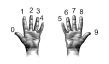




- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: "forty one" \rightarrow "forty two"
- Challenge: Write an algorithm for incrementing binary numbers. Example: "1001" \to "1010"
- Hint: Convert to numbers, increment, and convert back to strings.

- Get user input.
- Convert to standard decimal number.
- Add one (increment) the standard decimal number.
- 4 Convert back to your format.



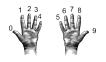




- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: "forty one" \rightarrow "forty two"
- Challenge: Write an algorithm for incrementing binary numbers. Example: "1001" \to "1010"
- Hint: Convert to numbers, increment, and convert back to strings.

- Get user input.
- Convert to standard decimal number.
- Add one (increment) the standard decimal number.
- Convert back to your format.
- Print the result.







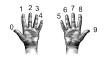
 Challenge: Write an algorithm for incrementing numbers expressed as words. Example: "forty one" → "forty two"

• Challenge: Write an algorithm for incrementing binary numbers.

Example: "1001" \rightarrow "1010"

Pseudocode same for both questions:

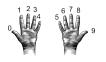
① Get user input: "forty one"





- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: "forty one" → "forty two"
- Challenge: Write an algorithm for incrementing binary numbers.
 Example: "1001" → "1010"

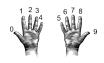
- Get user input: "forty one"
- 2 Convert to standard decimal number: 41





- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: "forty one" → "forty two"
- Challenge: Write an algorithm for incrementing binary numbers. Example: "1001" \rightarrow "1010"

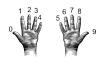
- Get user input: "forty one"
- 2 Convert to standard decimal number: 41
- 3 Add one (increment) the standard decimal number: 42





- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: "forty one" → "forty two"
- Challenge: Write an algorithm for incrementing binary numbers. Example: "1001" \to "1010"

- Get user input: "forty one"
- Convert to standard decimal number: 41
- Add one (increment) the standard decimal number: 42
- 4 Convert back to your format: "forty two"

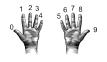




- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: "forty one" → "forty two"
- Challenge: Write an algorithm for incrementing binary numbers. Example: "1001" \to "1010"

- Get user input: "forty one"
- Convert to standard decimal number: 41
- Add one (increment) the standard decimal number: 42
- Convert back to your format: "forty two"
- Print the result.







 Challenge: Write an algorithm for incrementing numbers expressed as words. Example: "forty one" → "forty two"

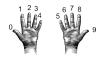
• Challenge: Write an algorithm for incrementing binary numbers.

Example: "1001" \rightarrow "1010"

Pseudocode same for both questions:

Get user input: "1001"

7 / 30

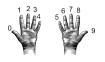




- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: "forty one" → "forty two"
- Challenge: Write an algorithm for incrementing binary numbers.
 Example: "1001" → "1010"

Pseudocode same for both questions:

- 1 Get user input: "1001"
- 2 Convert to standard decimal number: 9

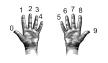




- ullet Challenge: Write an algorithm for incrementing numbers expressed as words. Example: "forty one" \to "forty two"
- Challenge: Write an algorithm for incrementing binary numbers. Example: "1001" \rightarrow "1010"

Pseudocode same for both questions:

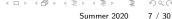
- ① Get user input: "1001"
- 2 Convert to standard decimal number: 9
- 3 Add one (increment) the standard decimal number: 10





- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: "forty one" → "forty two"
- Challenge: Write an algorithm for incrementing binary numbers. Example: "1001" \to "1010"

- Get user input: "1001"
- Convert to standard decimal number: 9
- Add one (increment) the standard decimal number: 10
- 4 Convert back to your format: "1010"







- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: "forty one" \rightarrow "forty two"
- Challenge: Write an algorithm for incrementing binary numbers. Example: "1001" \to "1010"

- Get user input: "1001"
- Convert to standard decimal number: 9
- Add one (increment) the standard decimal number: 10
- Convert back to your format: "1010"
- Print the result.





Focus on: Convert to standard decimal number:



Focus on: Convert to standard decimal number: def convert2Decimal(numString):



Focus on: Convert to standard decimal number:

def convert2Decimal(numString):

#Start with one-digit numbers: zero, one,..., nine



```
Focus on: Convert to standard decimal number:
def convert2Decimal(numString):
    #Start with one-digit numbers: zero,one,...,nine
    if numString == "zero":
        return(0)
```



```
Focus on: Convert to standard decimal number:
def convert2Decimal(numString):
    #Start with one-digit numbers: zero,one,...,nine
    if numString == "zero":
        return(0)
    elif numString == "one":
        return(1)
```



```
Focus on: Convert to standard decimal number:
def convert2Decimal(numString):
    #Start with one-digit numbers: zero,one,...,nine
    if numString == "zero":
        return(0)
    elif numString == "one":
        return(1)
    elif numString == "two":
        return(1)
```



```
Focus on: Convert to standard decimal number:
def convert2Decimal(numString):
    #Start with one-digit numbers: zero,one,...,nine
    if numString == "zero":
        return(0)
    elif numString == "one":
        return(1)
    elif numString == "two":
        return(1)
    else:
        return(9)
```



```
Focus on: Convert to standard decimal number:
def convert2Decimal(numString):
    #Start with one-digit numbers: zero,one,...,nine
    if numString == "zero":
        return(0)
    elif numString == "one":
        return(1)
    elif numString == "two":
        return(1)
    else:
        return(9)
```

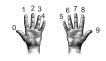
Will this work?

Recap: Incrementer Design Challenge



```
Focus on: Convert to standard decimal number:
def convert2Decimal(numString):
    #Start with one-digit numbers: zero,one,...,nine
    if numString == "zero":
        return(0)
    elif numString == "one":
        return(1)
    elif numString == "two":
        return(1)
    else:
        return(9)
```

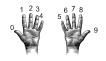
Will this work?



Focus on: Convert to standard decimal number: def convert2Decimal(numString):

```
#Start with one-digit numbers: zero,one,...,nine
if numString == "zero":
    return(0)
elif numString == "one":
    return(1)
elif numString == "two":
    return(1)
else:
    return(9)
```

Will this work? What inputs would find the error(s)?



Focus on: Convert to standard decimal number: def convert2Decimal(numString):

```
#Start with one-digit numbers: zero,one,...,nine
if numString == "zero":
    return(0)
elif numString == "one":
    return(1)
elif numString == "two":
    return(1)
else:
    return(9)
```

Will this work? What inputs would find the error(s)?

Unit Testing: testing individual units/functions/blocks of code to verify correctness.

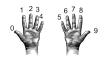


Focus on: Convert to standard decimal number: def convert2Decimal(numString):

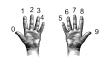
```
#Start with one-digit numbers: zero,one,...,nine
if numString == "zero":
    return(0)
elif numString == "one":
    return(1)
elif numString == "two":
    return(1)
else:
    return(9)
```

Will this work? What inputs would find the error(s)?

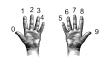
Unit Testing: testing individual units/functions/blocks of code to verify correctness. Often automated (e.g. gradescope).



- Unit Testing: testing individual units/functions/blocks of code to verify correctness. Often automated (e.g. gradescope).
- To test all branches of code, would need to test all inputs: "zero", "one",..., "nine", & some bad inputs.

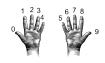


- Unit Testing: testing individual units/functions/blocks of code to verify correctness. Often automated (e.g. gradescope).
- To test all branches of code, would need to test all inputs: "zero",
 "one",..., "nine", & some bad inputs. Often do, if important or small.
- If large, design automated tests that will "cover" as many branches as possible and use randomly generated inputs:



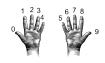
- Unit Testing: testing individual units/functions/blocks of code to verify correctness. Often automated (e.g. gradescope).
- To test all branches of code, would need to test all inputs: "zero",
 "one",..., "nine", & some bad inputs. Often do, if important or small.
- If large, design automated tests that will "cover" as many branches as possible and use randomly generated inputs:

```
names = ["zero","one",...,"nine"]
```



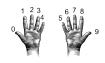
- Unit Testing: testing individual units/functions/blocks of code to verify correctness. Often automated (e.g. gradescope).
- To test all branches of code, would need to test all inputs: "zero",
 "one",..., "nine", & some bad inputs. Often do, if important or small.
- If large, design automated tests that will "cover" as many branches as possible and use randomly generated inputs:

```
names = ["zero","one",...,"nine"]
x = random.randrange(10)
```



- Unit Testing: testing individual units/functions/blocks of code to verify correctness. Often automated (e.g. gradescope).
- To test all branches of code, would need to test all inputs: "zero",
 "one",..., "nine", & some bad inputs. Often do, if important or small.
- If large, design automated tests that will "cover" as many branches as possible and use randomly generated inputs:

```
names = ["zero","one",...,"nine"]
x = random.randrange(10)
if x == convert2Decimal(names[x]):
```



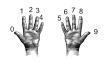
- Unit Testing: testing individual units/functions/blocks of code to verify correctness. Often automated (e.g. gradescope).
- To test all branches of code, would need to test all inputs: "zero",
 "one",..., "nine", & some bad inputs. Often do, if important or small.
- If large, design automated tests that will "cover" as many branches as possible and use randomly generated inputs:

```
names = ["zero","one",...,"nine"]
x = random.randrange(10)
if x == convert2Decimal(names[x]):
    #PASS
```



- Unit Testing: testing individual units/functions/blocks of code to verify correctness. Often automated (e.g. gradescope).
- To test all branches of code, would need to test all inputs: "zero",
 "one",..., "nine", & some bad inputs. Often do, if important or small.
- If large, design automated tests that will "cover" as many branches as possible and use randomly generated inputs:

```
names = ["zero","one",...,"nine"]
x = random.randrange(10)
if x == convert2Decimal(names[x]):
    #PASS
else:
```



- Unit Testing: testing individual units/functions/blocks of code to verify correctness. Often automated (e.g. gradescope).
- To test all branches of code, would need to test all inputs: "zero",
 "one",..., "nine", & some bad inputs. Often do, if important or small.
- If large, design automated tests that will "cover" as many branches as possible and use randomly generated inputs:

10 / 30

CSci 127 (Hunter) Lecture 12 Summer 2020

Today's Topics



- Recap: Incrementer Design Challenge
- C++: Basic Format & Variables
- $\bullet\,$ I/O and Definite Loops in C++

11 / 30

Challenge:

ullet Using what you know from Python, predict what the C++ code will do:

```
//Another C++ program, demonstrating variables
#include <iostream>
using namespace std;
int main ()
 int year;
cout << "Enter a number: ";</pre>
  cin >> year;
  cout << "Hello | << year << "!!\n\n";
  return 0;
```

12 / 30

onlinegdb demo

```
1 //Another (~e program, demonstrating variables 2 #include ciostreams 3 using numespace std; 4 5 int main () 6-{ 1 court (*e program of the program of the
```

(Demo with onlinegdb)

```
1 //Another G-e program, demonstrating variables
2 Sinclude -iostreams
3 using numespace std;
4 5 int main ()
6-{
7 int year;
8 cout < "Enter a number: ";
9 cin >> year;
10 cout < ""Ello | " << year << "!!\n\n";
11 return 0;
12 }
```

 C++ is a popular programming language that extends C.

```
1 //Mosther Gr. program, demonstrating variables
2 sinclude clostream
3 using namespace std;
4 int main ()
6 · {
7 int year;
8 cout < "Enter a number: ";
9 cin >> year;
10 cout < "Hell | " << year < "!!\n\n";
11 return 0;
```

- C++ is a popular programming language that extends C.
- Fast, efficient, and powerful.

```
1 //Mother Gr. program, demonstrating variables
2 shocked coistreom
3 using namespace std;
4 5 int main ()
6: {
7 int year;
8 cout < "Enter a number: ";
9 cin >> year;
10 cout < "Ello" << year << "!!\n\n";
11 return 0;
12 1
```

- C++ is a popular programming language that extends C.
- Fast, efficient, and powerful.
- Used for systems programming (and future courses!).

```
1 //Another C= program, demonstrating variables 2 #Include (iostream 3 using namespace std; 4 int main () 6 - { int year; cout < "Enter a number: "; cout < "Enter a number: "; coit >> year; lo cout < "Hello" << year << "!!\n\n"; large = "!" large = "!"
```

- C++ is a popular programming language that extends C.
- Fast, efficient, and powerful.
- Used for systems programming (and future courses!).
- Today, we'll introduce the basic structure and simple input/output (I/O) in C/C++.

• Programs are organized in functions.

```
1 //Another C++ program, demonstrating variables
2 finclude instream
3 using numespace std;
4
5 int year;
6 cout < "Enter a number: ";
9 cin >> year;
10 cout < "Mello " << year << "!!\n\n";
11 return 0;
12 }
```

```
1 //mosher c+ program, demonstrating variables
2 Enclude clostream
3 using namespace std;
4 5 int main ()
6-{
7 int year;
9 cin >> year;
10 cout <= "Enter a number: ";
11 return 0;
12 }
```

Programs are organized in functions.

Example:

15 / 30

```
int main ()
```

Programs are organized in functions.

Example:

int main()

15 / 30

```
1 //Mnother C+ program, demonstrating variables
2 #Include viostreams
3 using nomespace std;
4 int main ()
6-{
7 int year;
8 cout <= "Enter a number: ";
9 cin >> year;
10 cout <= "Hello" << year << "!!\n\n";
11 return 0;
12 }
```

Programs are organized in functions.

```
Example:
int main()
{
```

```
1 //Mnother C++ program, demonstrating variables
2 #snclude viostreams
3 using namespace std;
4 int main ()
6 \cdot {
7 int year;
8 cout < "Enter a number: ";
9 cin >> year;
10 cout < "Hello" << year << "!!\n\n";
11 return 0;
12 }
```

Programs are organized in functions.

```
Example:
int main()
{
    cout << "Hello world!";
    return(0);
}</pre>
```

Programs are organized in functions.

```
1 //Mnother C+ program, demonstrating variables
2 sinclude viostream
3 using namespace std;
4 int main ()
6 { try ear;
8 cout <= "Enter a number: ";
9 cin >> year;
10 cout <= "Hello" << year <= "!!\n\n";
11 return 0;
12 }
```

- Programs are organized in functions.
- Variables must be declared:

```
1 //Mnother (-- program, demonstrating variables
2 finclude -iostream
3 using numespace std;
4 int main ()
6 {int year;
8 cout <= "Enter a number: ";
9 cin >> year;
10 cout <= "Mello" <= year <= "!!\n\n";
11 return 0;
12 }
```

- Programs are organized in functions.
- Variables must be **declared**: int num;

```
1 //hostner (-- program, demonstrating variables similar (solutions) a single membapace std;
5 int main () 6 {
    int year;
8 cout < "Enter a number: ";
9 cout < "Bello " < year < "!!\n\n";
11 |
```

- Programs are organized in functions.
- Variables must be **declared**: int num;
- Many types available: int, float, char, ...

```
1 //Another C+ program, demostrating variables #include clostream 
3 using namespace std; 
4 int main () 
6 { 
7 int year; 
8 cout < "Enter a number; "; 
9 cin >> year; 
10 cout < "Hello" << year << "!!\n\n"; 
11 return 0; 
12 }
```

- Programs are organized in functions.
- Variables must be **declared**: int num;
- Many types available: int, float, char, ...
- Semicolons separate commands:

16 / 30

CSci 127 (Hunter) Lecture 12 Summer 2020

- Programs are organized in functions.
- Variables must be **declared**: int num;
- Many types available: int, float, char, ...
- Semicolons separate commands: num = 5; more = 2*num;

- Programs are organized in functions.
- Variables must be declared: int num;
- Many types available: int, float, char, ...
- Semicolons separate commands: num = 5; more = 2*num;
- To print, we'll use cout <<:

CSci 127 (Hunter) Lecture 12

- Programs are organized in functions.
- Variables must be declared: int num;
- Many types available: int, float, char, ...
- Semicolons separate commands:
 num = 5; more = 2*num;
 To print we'll use cout.
- To print, we'll use cout <<:
 cout << "Hello!!";</pre>

```
1 //Amother C++ program, demonstrating variables #include clostreams
3 using nomespace std;
5 int main ()
6 if year;
7 int year;
8 cout < "Enter a number: ";
9 cin >> year;
10 cout < "Hello |" << year << "!!\n\n";
11 return 0;
12 }
```

- Programs are organized in functions.
- Variables must be declared: int num;
- Many types available: int, float, char, ...
- Semicolons separate commands: num = 5; more = 2*num;
- To print, we'll use cout <<:
 cout << "Hello!!";</pre>
- To get input, we'll use cin >>:

```
1 //Moother (~* program, demonstrating variables 2 #include (lostream 3 using nomespace std; 4 thr main () 6-{ 1 th year; 8 cout < "Enter a number: "; 9 cin >> year; 10 cout < "Whello " << year << "!!\n\n"; 11 return 0; 12 }
```

- Programs are organized in functions.
- Variables must be declared: int num;
- Many types available: int, float, char, ...
- Semicolons separate commands: num = 5; more = 2*num;
- To print, we'll use cout <<:
 cout << "Hello!!";</pre>
- To get input, we'll use cin >>:cin >> num;

```
//Monther C-- program, demonstrating variables
2 Enclude (instreman)
3 using nomespace std;
4 int main ()
6: {
7 int year;
9 cin >> year;
10 cout <= "Enter a number: ";
9 cin >> year;
11 return 0;
12 return 0;
```

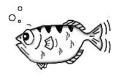
- Programs are organized in functions.
- Variables must be declared: int num;
- Many types available: int, float, char, ...
- Semicolons separate commands: num = 5; more = 2*num;
- To print, we'll use cout <<:
 cout << "Hello!!";</pre>
- To get input, we'll use cin >>:cin >> num;
- To use those I/O functions, we put at the top of the program:

```
//Monther C-- program, demonstrating variables
2 Enclude (instreman)
3 using nomespace std;
4 int main ()
6: {
7 int year;
9 cin >> year;
10 cout <= "Enter a number: ";
9 cin >> year;
11 return 0;
12 return 0;
```

- Programs are organized in functions.
- Variables must be **declared**: int num;
- Many types available: int, float, char, ...
- Semicolons separate commands: num = 5; more = 2*num;
- o To print, we'll use cout <<:
 cout << "Hello!!";</pre>
- To get input, we'll use cin >>: cin >> num;
- To use those I/O functions, we put at the top of the program: #include <iostream> using namespace std;

```
//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:
```

 Part of Richard Stallman's "GNU is Not Unix" (GNU) project.

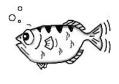


gdb.org



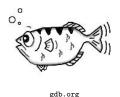
gdb.org

- Part of Richard Stallman's "GNU is Not Unix" (GNU) project.
- Written in 1986, gdb is the GNU debugger and based on dbx from the Berkeley Distribution of Unix.



gdb.org

- Part of Richard Stallman's "GNU is Not Unix" (GNU) project.
- Written in 1986, gdb is the GNU debugger and based on dbx from the Berkeley Distribution of Unix.
- Lightweight, widely-available program that allows you to "step through" your code line-by-line.



- Part of Richard Stallman's "GNU is Not Unix" (GNU) project.
- Written in 1986, gdb is the GNU debugger and based on dbx from the Berkeley Distribution of Unix.
- Lightweight, widely-available program that allows you to "step through" your code line-by-line.
- Available on the lab machines (via command-line and the IDE spyder) and on-line (onlinegdb.com).

C++ Demo

```
//Another C++ program, demonstrating I/O & arithmetic finclude clostream-
using namespace std;
int moin O {
   floot kg, lbs;
   cout << "Enter kg: ";
   lbs = kg * 2.2;
   cout << endl << "tbs: " << lbs << "\n\n";
   return 0;
}
```

(Demo with onlinegdb)

19 / 30

CSci 127 (Hunter) Lecture 12 Summer 2020

Challenge:...

Convert the C++ code to a **Python program**:

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std;
int main ()
  float kg, lbs;
  cout << "Enter kg: ";
  cin >> kg;
  lbs = kq * 2.2;
  cout << endl << "Lbs: " << lbs << "\n\n":
  return 0:
```

Python Tutor

Convert the C++ code to a **Python program**:

```
//Another C++ program, demonstrating I/O & arithmetic
#include <iostream>
using namespace std:
int main ()
  float kg, lbs;
  cout << "Enter kg: ";
  cin >> kg;
  lbs = kq * 2.2;
  cout << endl << "Lbs: " << lbs << "\n\n";
  return 0:
           (Write from scratch in pythonTutor.)
```

Today's Topics



- Recap: Incrementer Design Challenge
- C++: Basic Format & Variables
- I/O and Definite Loops in C++

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

C++ Demo

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

(Demo with onlinegdb)

CSci 127 (Hunter) Lecture 12 Summer 2020 24/30

Definite loops

```
//Another C++ program; Demonstrates loops
#include ciostream>
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;
}</pre>
```

```
General format:
for ( initialization ; test ; updateAction )
{
    command1;
    command2;
    command3;
    ...
}
```

```
//Another C++ program; Demonstrates loops
#include <iostream>
using namespace std;
int main ()
  int i,j,size;
  cout << "Enter size: ";</pre>
  cin >> size;
  for (i = 0; i < size; i++)
    for (j = 0; j < size; j++)
      cout << "*";
    cout << endl:
  cout << "\n\n";</pre>
  for (i = size: i > 0: i--)
    for (j = 0; j < i; j++)
      cout << "*":
    cout << endl;
  return 0;
```

C++ Demo

```
//Another C++ program; Demonstrates loops
#include <iostream>
using namespace std:
int main ()
  int i,j,size;
  cout << "Enter size: ";</pre>
  cin >> size:
  for (i = 0; i < size; i++)
    for (j = 0; j < size; j++)
                                               (Demo with onlinegdb)
    cout << "*";
   cout << endl:
  cout << "\n\n";
  for (i = size; i > 0; i--)
    for (j = 0; j < i; j++)
    cout << "*";
   cout << endl:
  return 0;
```

CSci 127 (Hunter) Lecture 12 Summer 2020 27 / 30

```
//Growth example
#include <iostream>
using namespace std;
int main ()
  int population = 100;
  cout << "Year\tPopulation\n";</pre>
  for (int year = 0; year < 100; year= year+5)
  {
      cout << year << "\t" << population << "\n";</pre>
      population = population * 2;
  return 0;
```

Translate the C++ program into Python:

```
//Growth example
#include <iostream>
using namespace std;
int main ()
  int population = 100;
  cout << "Year\tPopulation\n";</pre>
  for (int year = 0; year < 100; year= year+5)
      cout << year << "\t" << population << "\n";</pre>
      population = population * 2;
  return 0;
```

Recap: C++

 C++ is a popular programming language that extends C.



Recap: C++



- C++ is a popular programming language that extends C.
- Input/Output (I/O):
 - ▶ cin >>
 - **▶** cout <<

30 / 30

Recap: C++



- C++ is a popular programming language that extends C.
- Input/Output (I/O):
 - ▶ cin >>
 - ▶ cout <<
- Definite loops:
 for (i = 0; i < 10; i++) {</pre>

```
• •
```