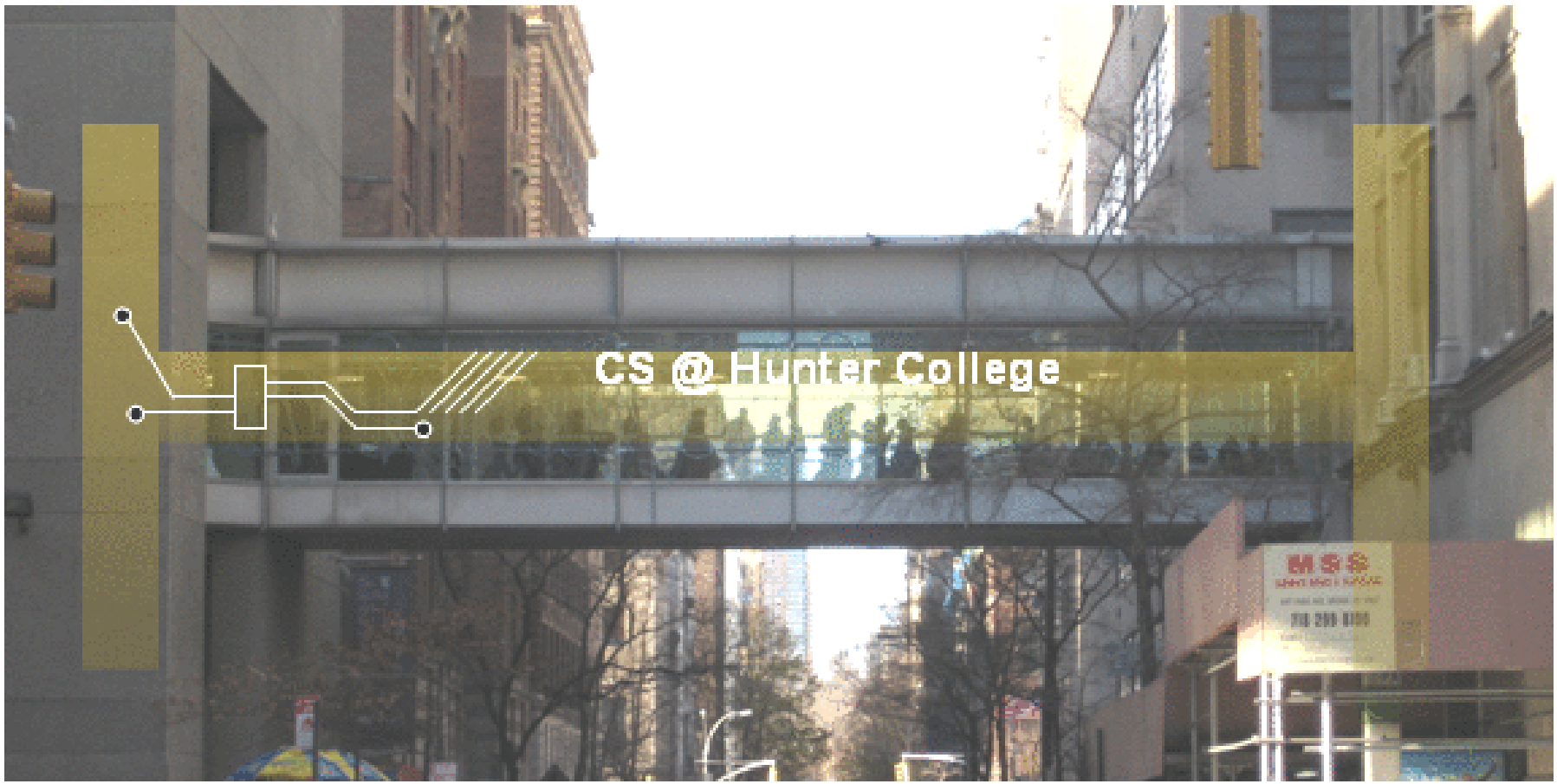
CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

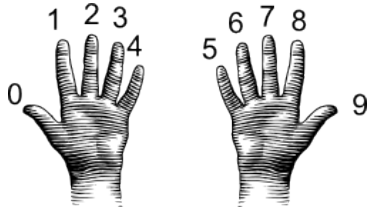
Today's Topics

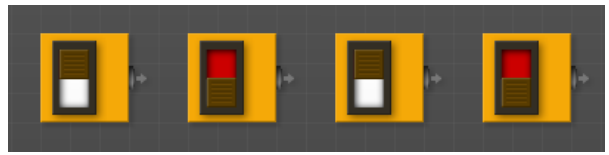
- Recap: Incrementer Design Challenge
- C++: Basic Format & Variables
- I/O and Definite Loops in C++

Today's Topics

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

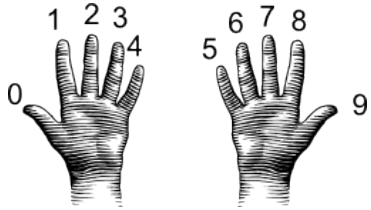
Recap: Design Challenge: Incrementers

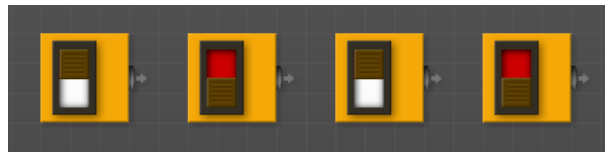




- Simplest arithmetic: add one ("increment") a variable.

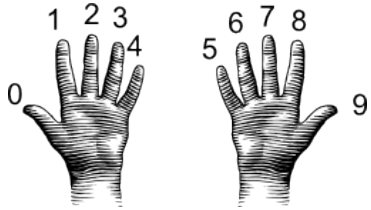
Recap: Design Challenge: Incrementers

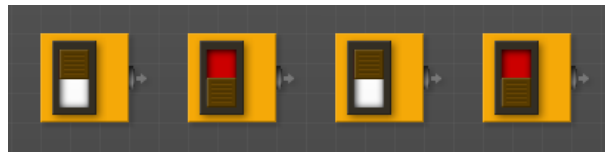




- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

Recap: Design Challenge: Incrementers

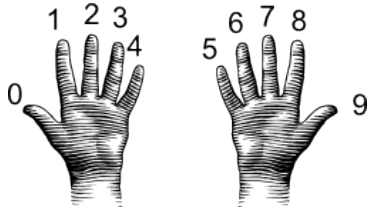


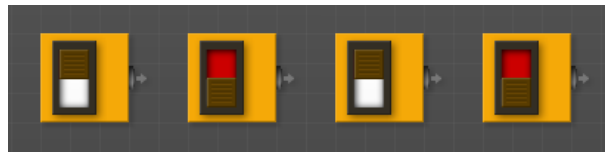


- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

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

Recap: Design Challenge: Incrementers



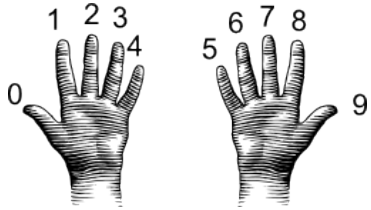


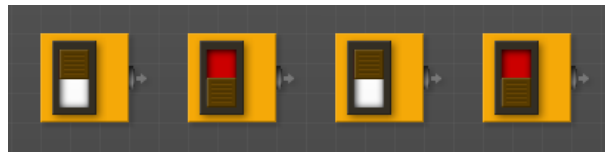
- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

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

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

Recap: Design Challenge: Incrementers



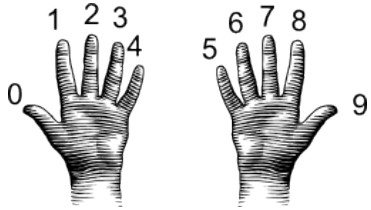


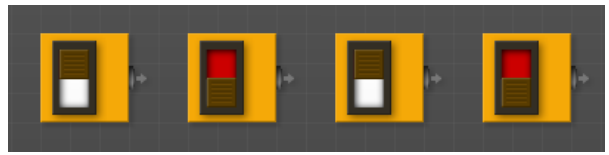
- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

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

- Challenge: Write an algorithm for incrementing numbers expressed as words.
  Example: `"forty one"` $\rightarrow$ `"forty two"`

Recap: Design Challenge: Incrementers





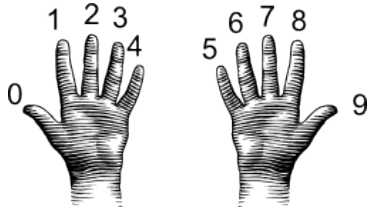
- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

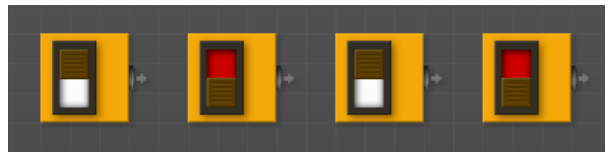
```
def addOne(n):
    m = n+1
    return(m)
```

- 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.

Recap: Design Challenge: Incrementers





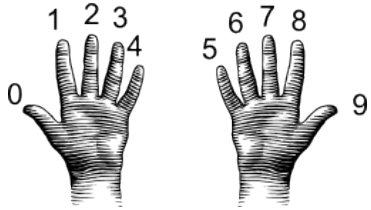
- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

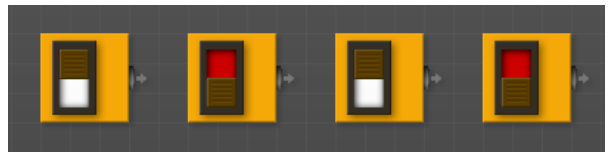
```
def addOne(n):
    m = n+1
    return(m)
```

- 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.

Recap: Design Challenge: Incrementers





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

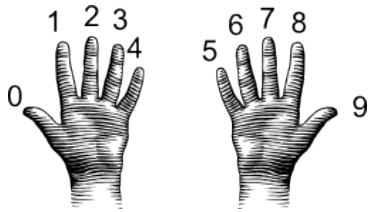
```
def addOne(n):
    m = n+1
    return(m)
```

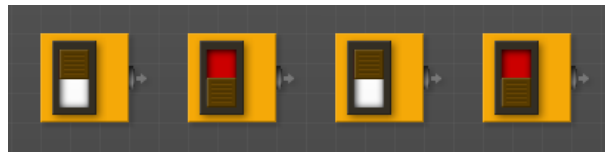
- Challenge: Write an algorithm for incrementing numbers expressed as words.
  Example: `"forty one"` → `"forty two"`

  Hint: Convert to numbers, increment, and convert back to strings.
- Challenge: Write an algorithm for incrementing binary numbers.

  Example: `"1001"` → `"1010"`

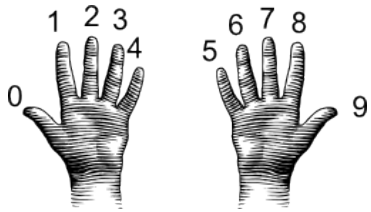
Recap: Incrementer Design Challenge

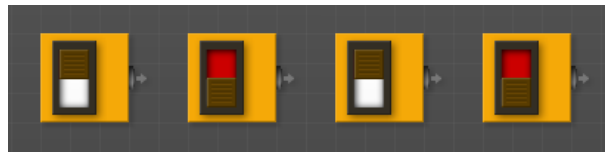




- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`

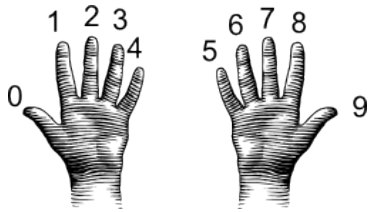
Recap: Incrementer Design Challenge

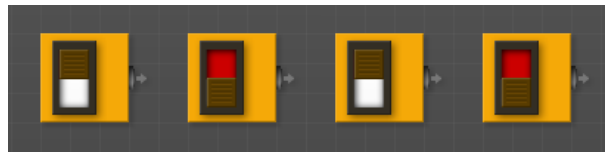




- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`
- *Hint: Convert to numbers, increment, and convert back to strings.*

Recap: Incrementer Design Challenge



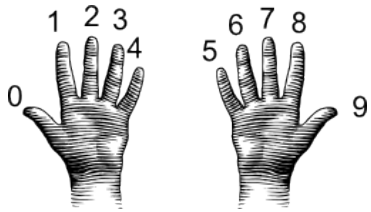


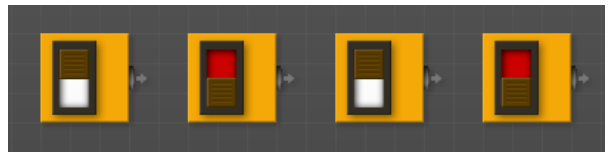
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`
- *Hint: Convert to numbers, increment, and convert back to strings.*

Pseudocode same for both questions:

1. Get user input.

Recap: Incrementer Design Challenge



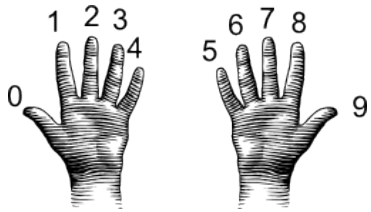


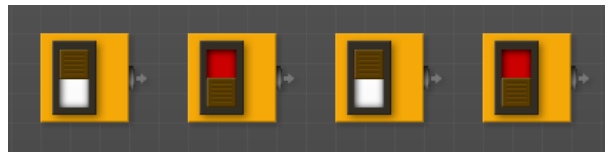
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`
- *Hint: Convert to numbers, increment, and convert back to strings.*

Pseudocode same for both questions:

1. Get user input.
2. Convert to standard decimal number.

Recap: Incrementer Design Challenge



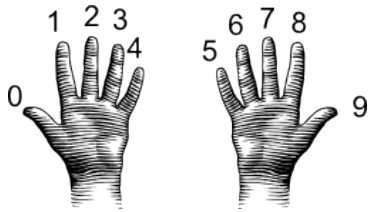


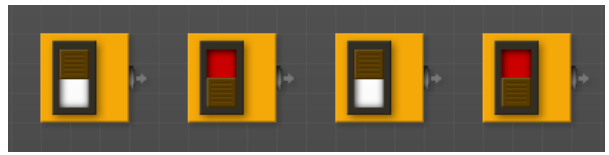
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`
- *Hint: Convert to numbers, increment, and convert back to strings.*

Pseudocode same for both questions:

1. Get user input.
2. Convert to standard decimal number.
3. Add one (increment) the standard decimal number.

Recap: Incrementer Design Challenge



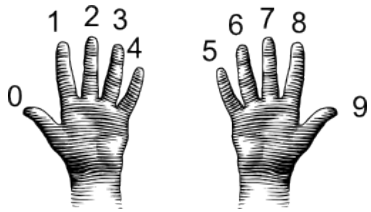


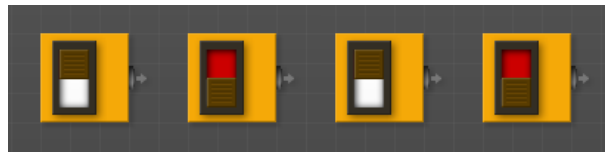
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`
- *Hint: Convert to numbers, increment, and convert back to strings.*

Pseudocode same for both questions:

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

Recap: Incrementer Design Challenge



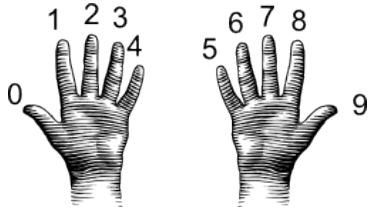


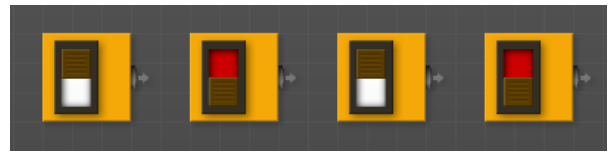
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`
- *Hint: Convert to numbers, increment, and convert back to strings.*

Pseudocode same for both questions:

1. Get user input.
2. Convert to standard decimal number.
3. Add one (increment) the standard decimal number.
4. Convert back to your format.
5. Print the result.

Recap: Incrementer Design Challenge



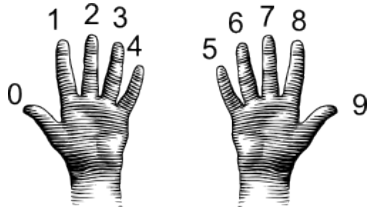


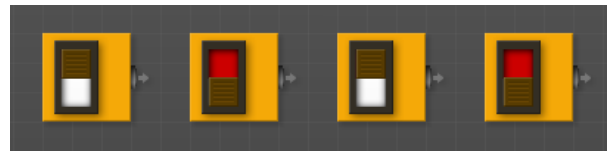
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`

Pseudocode same for both questions:

1. Get user input: `"forty one"`

Recap: Incrementer Design Challenge



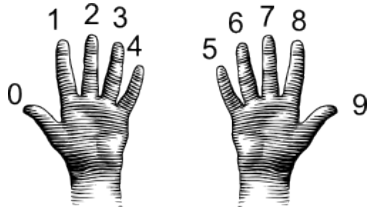


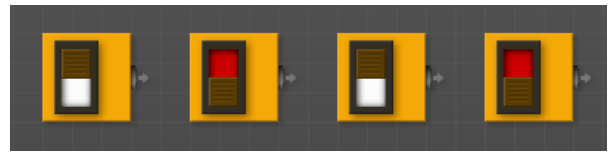
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`

Pseudocode same for both questions:

1. Get user input: `"forty one"`
2. Convert to standard decimal number: `41`

Recap: Incrementer Design Challenge



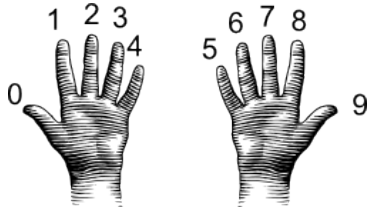


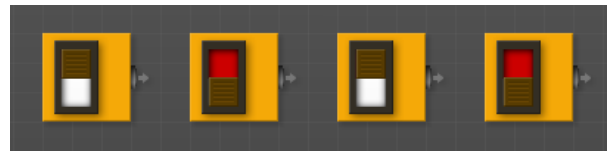
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`

Pseudocode same for both questions:

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

Recap: Incrementer Design Challenge



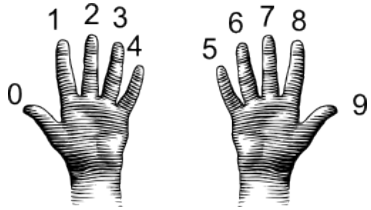


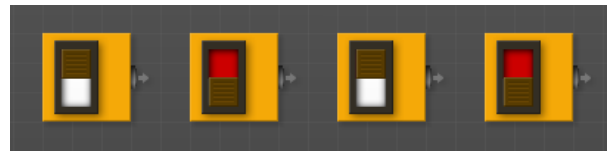
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`

Pseudocode same for both questions:

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

Recap: Incrementer Design Challenge



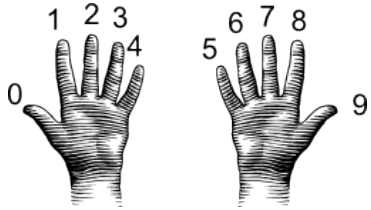


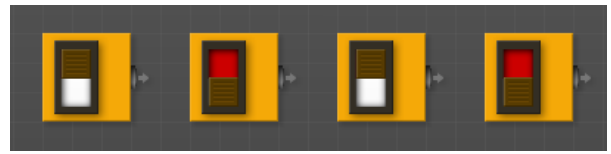
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` → `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` → `"1010"`

Pseudocode same for both questions:

1. Get user input: `"forty one"`
2. Convert to standard decimal number: `41`
3. Add one (increment) the standard decimal number: `42`
4. Convert back to your format: `"forty two"`
5. Print the result.

Recap: Incrementer Design Challenge



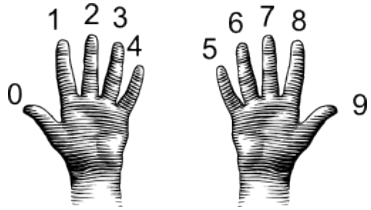


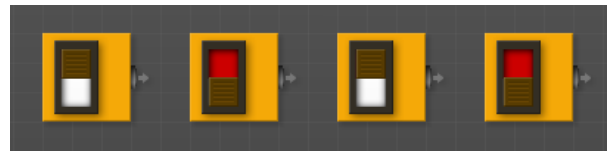
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`

Pseudocode same for both questions:

1. Get user input: `"1001"`

Recap: Incrementer Design Challenge



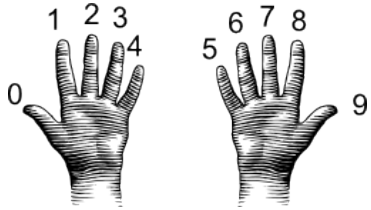


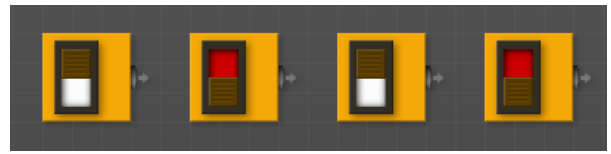
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`

Pseudocode same for both questions:

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

Recap: Incrementer Design Challenge



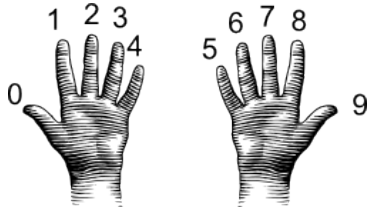


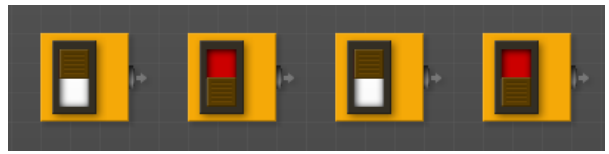
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`

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

Recap: Incrementer Design Challenge



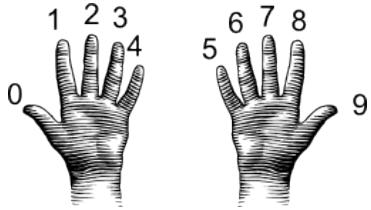


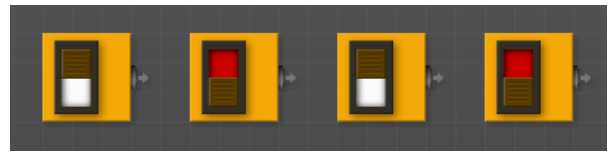
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`

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
4. Convert back to your format: `"1010"`

Recap: Incrementer Design Challenge



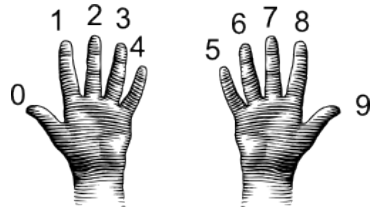


- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`

Pseudocode same for both questions:

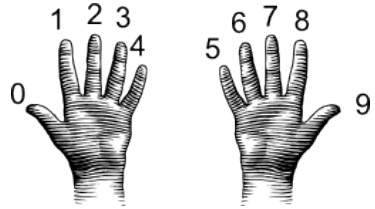
1. Get user input: `"1001"`
2. Convert to standard decimal number: 9
3. Add one (increment) the standard decimal number: 10
4. Convert back to your format: `"1010"`
5. Print the result.

Recap: Incrementer Design Challenge



Focus on: Convert to standard decimal number:

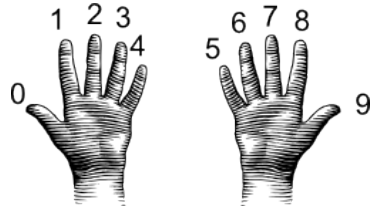
Recap: Incrementer Design Challenge



Focus on: Convert to standard decimal number:

```
def convert2Decimal(numString):
```

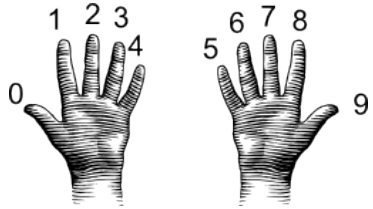
Recap: Incrementer Design Challenge



Focus on: Convert to standard decimal number:

```
def convert2Decimal(numString):
    #Start with one-digit numbers:  zero,one,...,nine
```

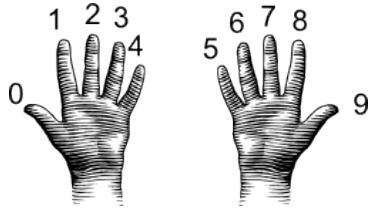
Recap: Incrementer Design Challenge



Focus on: Convert to standard decimal number:

```
def convert2Decimal(numString):
    #Start with one-digit numbers:  zero,one,...,nine
    if numString == "zero":
        return(0)
```

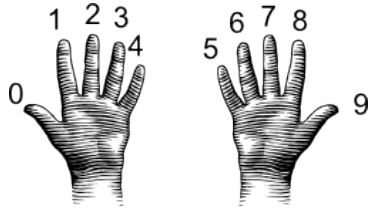
Recap: Incrementer Design Challenge



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)
```

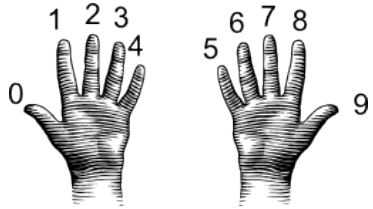
Recap: Incrementer Design Challenge



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)
```

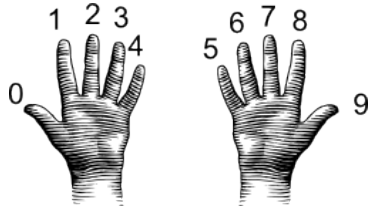
Recap: Incrementer Design Challenge



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)
```

Recap: Incrementer Design Challenge

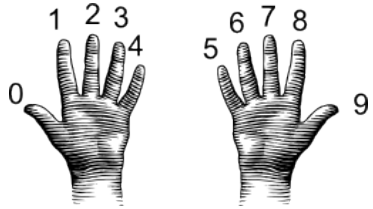


Focus on: Convert to standard decimal number:

```
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?

Recap: Incrementer Design Challenge

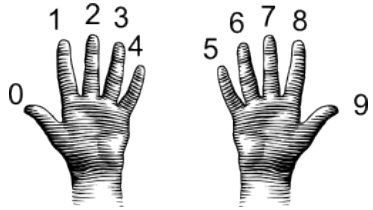


Focus on: Convert to standard decimal number:

```
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?

Unit Testing: Incrementer Design Challenge

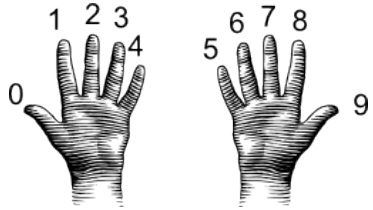


Focus on: Convert to standard decimal number:

```
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? What inputs would find the error(s)?

Unit Testing: Incrementer Design Challenge



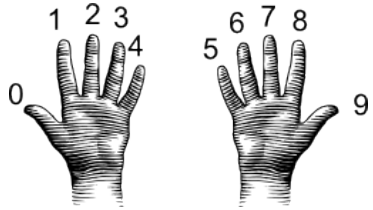
Focus on: Convert to standard decimal number:

```
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? What inputs would find the error(s)?

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

Unit Testing: Incrementer Design Challenge



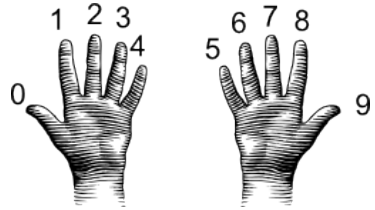
Focus on: Convert to standard decimal number:

```
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? What inputs would find the error(s)?

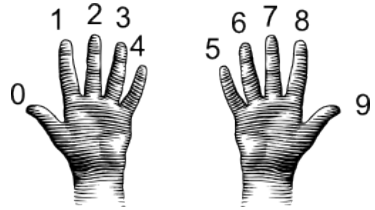
Unit Testing: testing individual units/functions/blocks of code to verify correctness. Often automated (e.g. gradescope).

Unit Testing: Incrementer Design Challenge



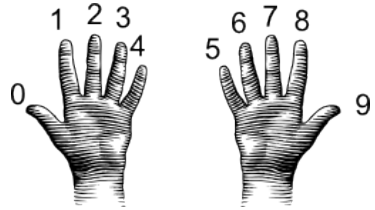
- Unit Testing: testing individual units/functions/blocks of code to verify correctness. Often automated (e.g. gradescope).
- To test all branches of code, would need to test all inputs: `"zero"`, `"one"`,..., `"nine"`, & some bad inputs.

Unit Testing: Incrementer Design Challenge



- 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:

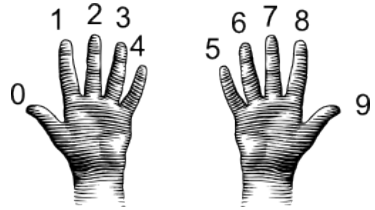
Unit Testing: Incrementer Design Challenge



- 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"]
```

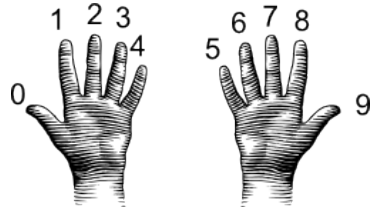
Unit Testing: Incrementer Design Challenge



- 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)
```

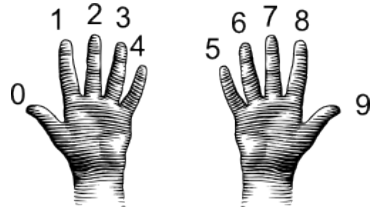
Unit Testing: Incrementer Design Challenge



- 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]):
```

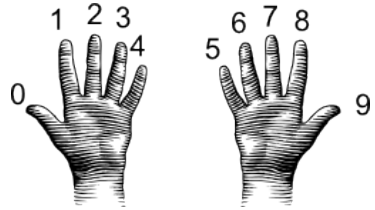
Unit Testing: Incrementer Design Challenge



- 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
```

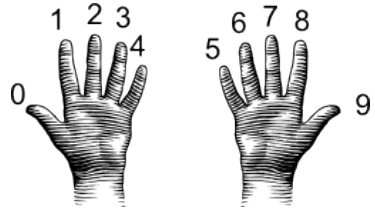
Unit Testing: Incrementer Design Challenge



- 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: Incrementer Design Challenge



- 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:
    #FAIL
```

Today's Topics

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

Challenge:

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

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

onlinegdb demo

```cpp
//Another C++ program, demonstrating variables
#include <iostream>
using namespace std;

int main ()
{
  int year;
  cout << "Enter a number: ";
  cin >> year;
  cout << "Hello " << year << "!!\n\n";
  return 0;
}
```

(Demo with `onlinegdb`)

Introduction to C++

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

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

Introduction to C++

```cpp
//Another C++ program, demonstrating variables
#include <iostream>
using namespace std;

int main ()
{
  int year;
  cout << "Enter a number: ";
  cin >> year;
  cout << "Hello " << year << "!!\n\n";
  return 0;
}
```

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

Introduction to C++

```cpp
//Another C++ program, demonstrating variables
#include <iostream>
using namespace std;

int main ()
{
  int year;
  cout << "Enter a number: ";
  cin >> year;
  cout << "Hello " << year << "!!\n\n";
  return 0;
}
```

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

Introduction to C++

```cpp
//Another C++ program, demonstrating variables
#include <iostream>
using namespace std;

int main ()
{
  int year;
  cout << "Enter a number: ";
  cin >> year;
  cout << "Hello " << year << "!!\n\n";
  return 0;
}
```

- 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++.

Introduction to C++

- Programs are organized in functions.

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

Introduction to C++

- Programs are organized in functions.

Example:

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

Introduction to C++

```
1  //Another C++ program, demonstrating variables
2  #include <iostream>
3  using namespace std;
4
5  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()`

Introduction to C++

```
1  //Another C++ program, demonstrating variables
2  #include <iostream>
3  using namespace std;
4
5  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()
{
```

Introduction to C++

```
1  //Another C++ program, demonstrating variables
2  #include <iostream>
3  using namespace std;
4
5  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()
{
    cout << "Hello world!";
    return(0);
}
```

Introduction to C++

- Programs are organized in functions.

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

Introduction to C++

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

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

Introduction to C++

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

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

Introduction to C++

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

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

Introduction to C++

```
1  //Another C++ program, demonstrating variables
2  #include <iostream>
3  using namespace std;
4
5  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:

Introduction to C++

```
1  //Another C++ program, demonstrating variables
2  #include <iostream>
3  using namespace std;
4
5  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:
  `num = 5; more = 2*num;`

Introduction to C++

```
1  //Another C++ program, demonstrating variables
2  #include <iostream>
3  using namespace std;
4
5  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:
  `num = 5; more = 2*num;`
- To print, we'll use `cout <<`:

Introduction to C++

```cpp
//Another C++ program, demonstrating variables
#include <iostream>
using namespace std;

int main ()
{
  int year;
  cout << "Enter a number: ";
  cin >> year;
  cout << "Hello " << year << "!!\n\n";
  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!!";`

Introduction to C++

```cpp
//Another C++ program, demonstrating variables
#include <iostream>
using namespace std;

int main ()
{
  int year;
  cout << "Enter a number: ";
  cin >> year;
  cout << "Hello " << year << "!!\n\n";
  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!!";`
- To get input, we'll use `cin >>`:

Introduction to C++

```cpp
//Another C++ program, demonstrating variables
#include <iostream>
using namespace std;

int main ()
{
  int year;
  cout << "Enter a number: ";
  cin >> year;
  cout << "Hello " << year << "!!\n\n";
  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!!";`
- To get input, we'll use `cin >>`:
  `cin >> num;`

Introduction to C++

```
1  //Another C++ program, demonstrating variables
2  #include <iostream>
3  using namespace std;
4
5  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:
  `num = 5; more = 2*num;`
- To print, we'll use `cout <<`:
  `cout << "Hello!!";`
- To get input, we'll use `cin >>`:
  `cin >> num;`
- To use those I/O functions, we put at the top of the program:

Introduction to C++

```cpp
//Another C++ program, demonstrating variables
#include <iostream>
using namespace std;

int main ()
{
  int year;
  cout << "Enter a number: ";
  cin >> year;
  cout << "Hello " << year << "!!\n\n";
  return 0;
}
```

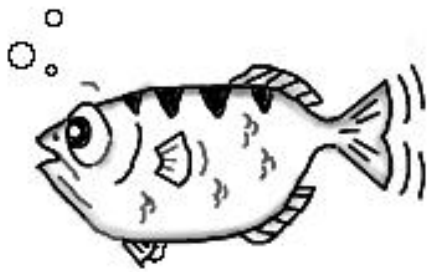
- Programs are organized in functions.
- Variables must be **declared**:
  `int num;`
- Many types available:
  `int, float, char, ...`
- Semicolons separate commands:
  `num = 5; more = 2*num;`
- To print, we'll use `cout <<`:
  `cout << "Hello!!";`
- To get input, we'll use `cin >>`:
  `cin >> num;`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

Challenge:

Predict what the following pieces of code will do:

```
//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;
}
```

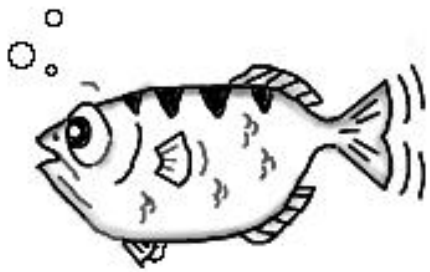
Side Note: gdb



gdb.org

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

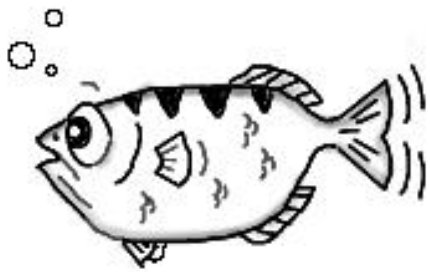
Side Note: gdb



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.

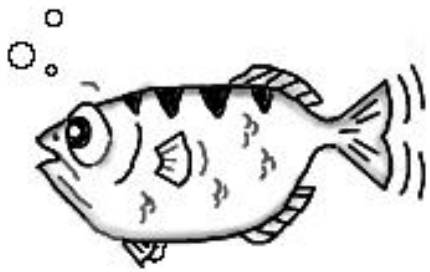
Side Note: gdb



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.

Side Note: gdb



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.
- Available on the lab machines (via command-line and the IDE spyder) and on-line (onlinegdb.com).

C++ Demo

```cpp
//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;
}
```

(Demo with `onlinegdb`)

Challenge:...

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

```cpp
//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;
}
```

Python Tutor

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

```cpp
//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;
}
```

(Write from scratch in `pythonTutor`.)

Today's Topics

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

Challenge:

Predict what the following pieces of code will do:

```
//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;
}
```

C++ Demo

```cpp
//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;
}
```

(Demo with `onlinegdb`)

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;
}
```

General format:

`for` (*initialization* ; *test* ; *updateAction*)
{

command1;
command2;
command3;
...

}

Challenge:

Predict what the following pieces of code will do:

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

int main ()
{
  int i,j,size;
  cout << "Enter size: ";
  cin >> size;
  for (i = 0; i < size; i++)
  {
    for (j = 0; j < size; j++)
      cout << "*";
    cout << endl;
  }
  cout << "\n\n";
  for (i = size; i > 0; i--)
  {
    for (j = 0; j < i; j++)
      cout << "*";
    cout << endl;
  }
  return 0;
}
```

C++ Demo

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

int main ()
{
  int i,j,size;
  cout << "Enter size: ";
  cin >> size;
  for (i = 0; i < size; i++)
  {
    for (j = 0; j < size; j++)
      cout << "*";
    cout << endl;
  }
  cout << "\n\n";
  for (i = size; i > 0; i--)
  {
    for (j = 0; j < i; j++)
      cout << "*";
    cout << endl;
  }
  return 0;
}
```

(Demo with `onlinegdb`)

Challenge:

Predict what the following pieces of code will do:

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

int main ()
{
  int population = 100;
  cout << "Year\tPopulation\n";
  for (int year = 0; year < 100; year= year+5)
  {
      cout << year << "\t" << population << "\n";
      population = population * 2;

  }
  return 0;
}
```

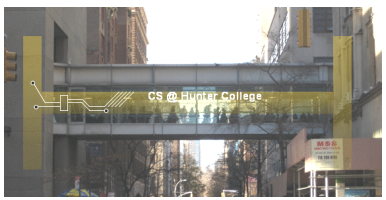
Challenge:

Translate the C++ program into Python:

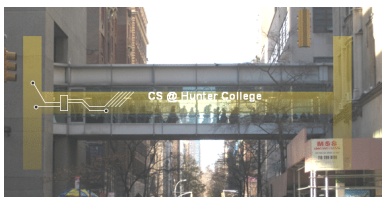
```cpp
//Growth example
#include <iostream>
using namespace std;

int main ()
{
  int population = 100;
  cout << "Year\tPopulation\n";
  for (int year = 0; year < 100; year= year+5)
  {
      cout << year << "\t" << population << "\n";
      population = population * 2;

  }
  return 0;
}
```

Recap: C++

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

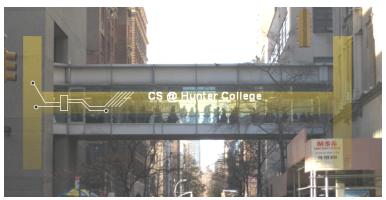


Recap: C++



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

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++) {
    ...
}
```