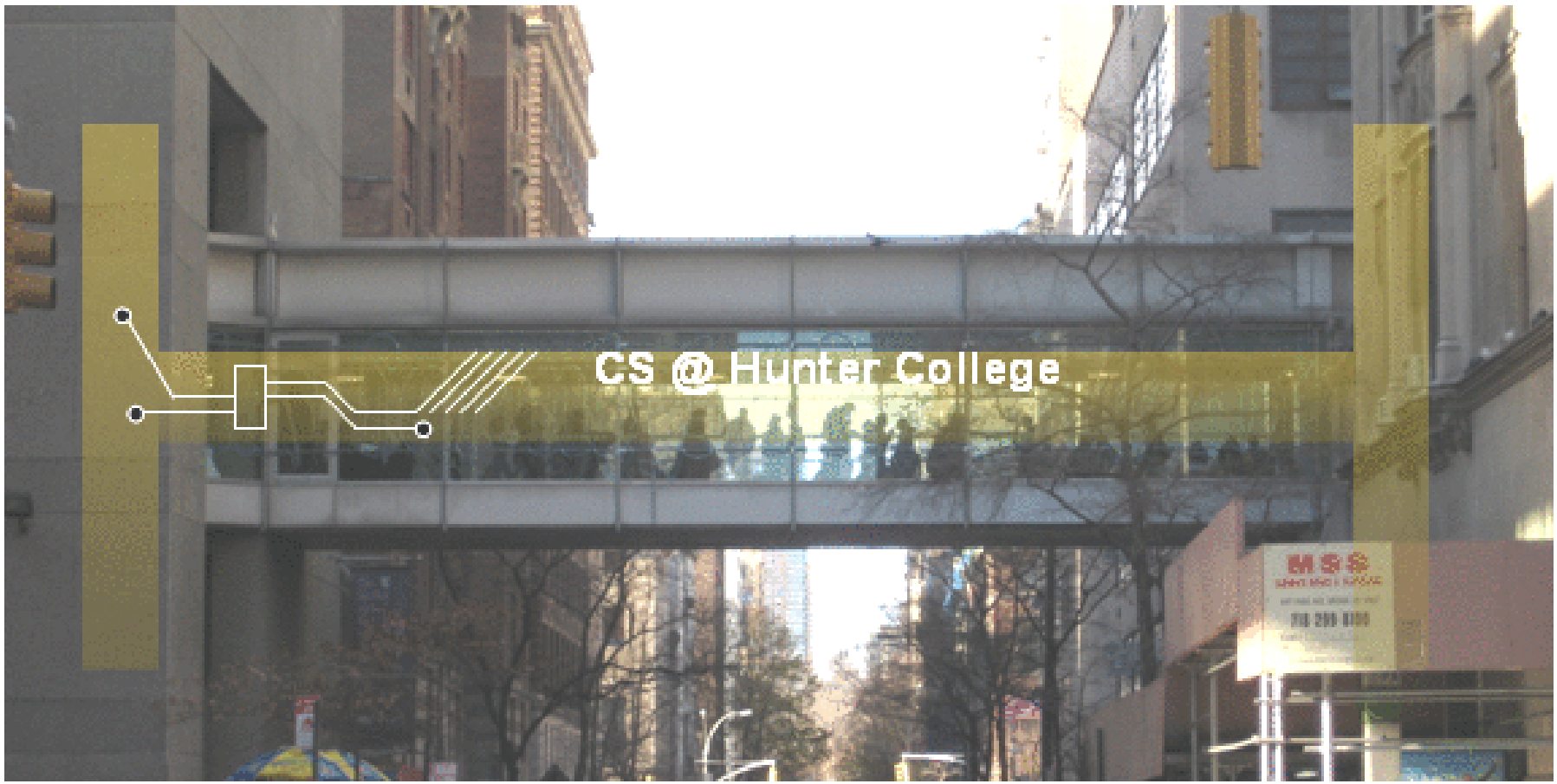
CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Frequently Asked Questions

From lecture slips & recitation sections.

- Who/why all the visitors?

Frequently Asked Questions

From lecture slips & recitation sections.

- Who/why all the visitors?
  It's outreach to NYC schools & prospective students.

Frequently Asked Questions

From lecture slips & recitation sections.

- Who/why all the visitors?
  It's outreach to NYC schools & prospective students.
- I have a conflict with the final– what should I do?

Frequently Asked Questions

From lecture slips & recitation sections.

- Who/why all the visitors?
  It's outreach to NYC schools & prospective students.
- I have a conflict with the final– what should I do?
  Fill in the pink slip with when you will take the final.

Frequently Asked Questions

From lecture slips & recitation sections.

- Who/why all the visitors?
  It's outreach to NYC schools & prospective students.
- I have a conflict with the final– what should I do?
  Fill in the pink slip with when you will take the final.
  Note: if not the assigned time, include reason for dean's approval for the non-standard time.

Frequently Asked Questions

From lecture slips & recitation sections.

- Who/why all the visitors?
  It's outreach to NYC schools & prospective students.
- I have a conflict with the final– what should I do?
  Fill in the pink slip with when you will take the final.
  Note: if not the assigned time, include reason for dean's approval for the non-standard time.
- I'm worried about my grade. Should I do Credit/NoCredit?

Frequently Asked Questions

From lecture slips & recitation sections.

- Who/why all the visitors?
  It's outreach to NYC schools & prospective students.
- I have a conflict with the final– what should I do?
  Fill in the pink slip with when you will take the final.
  Note: if not the assigned time, include reason for dean's approval for the non-standard time.
- I'm worried about my grade. Should I do Credit/NoCredit?
  It's fine with us, but check with your advisor to make sure it's accepted for your program of study.

Frequently Asked Questions

From lecture slips & recitation sections.

- Who/why all the visitors?
  It's outreach to NYC schools & prospective students.
- I have a conflict with the final– what should I do?
  Fill in the pink slip with when you will take the final.
  Note: if not the assigned time, include reason for dean's approval for the non-standard time.
- I'm worried about my grade. Should I do Credit/NoCredit?
  It's fine with us, but check with your advisor to make sure it's accepted for your program of study.
- I want to learn more– what should I take next?

Frequently Asked Questions

From lecture slips & recitation sections.

- Who/why all the visitors?
  It's outreach to NYC schools & prospective students.
- I have a conflict with the final– what should I do?
  Fill in the pink slip with when you will take the final.
  Note: if not the assigned time, include reason for dean's approval for the non-standard time.
- I'm worried about my grade. Should I do Credit/NoCredit?
  It's fine with us, but check with your advisor to make sure it's accepted for your program of study.
- I want to learn more– what should I take next?
  - *Majors: CSci 135/136 (C++, MWTh 12:10-1pm + section) & CSci 150 (Discrete Structures, MTh 1:10-2:25pm + section)*

Frequently Asked Questions

From lecture slips & recitation sections.

- Who/why all the visitors?
  It's outreach to NYC schools & prospective students.
- I have a conflict with the final– what should I do?
  Fill in the pink slip with when you will take the final.
  Note: if not the assigned time, include reason for dean's approval for the non-standard time.
- I'm worried about my grade. Should I do Credit/NoCredit?
  It's fine with us, but check with your advisor to make sure it's accepted for your program of study.
- I want to learn more– what should I take next?
  - *Majors: CSci 135/136 (C++, MWTh 12:10-1pm + section) & CSci 150 (Discrete Structures, MTh 1:10-2:25pm + section)*
  - *Minors: CSci 133 (More Python: multiple times) & CSci 232 (Databases, multiple times)*

Frequently Asked Questions

From lecture slips & recitation sections.

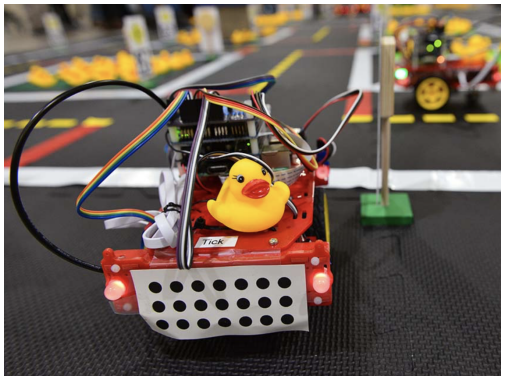
- Who/why all the visitors?
  It's outreach to NYC schools & prospective students.
- I have a conflict with the final– what should I do?
  Fill in the pink slip with when you will take the final.
  Note: if not the assigned time, include reason for dean's approval for the non-standard time.
- I'm worried about my grade. Should I do Credit/NoCredit?
  It's fine with us, but check with your advisor to make sure it's accepted for your program of study.
- I want to learn more– what should I take next?
  - *Majors: CSci 135/136 (C++, MWTh 12:10-1pm + section) & CSci 150 (Discrete Structures, MTh 1:10-2:25pm + section)*
  - *Minors: CSci 133 (More Python: multiple times) & CSci 232 (Databases, multiple times)*
- What's a mock exam? I see it on the webpage...

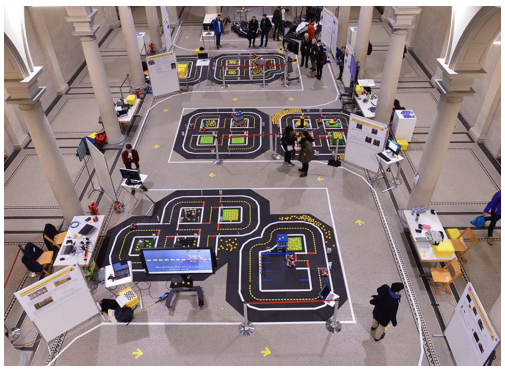
Frequently Asked Questions

From lecture slips & recitation sections.

- Who/why all the visitors?
  It's outreach to NYC schools & prospective students.
- I have a conflict with the final– what should I do?
  Fill in the pink slip with when you will take the final.
  Note: if not the assigned time, include reason for dean's approval for the non-standard time.
- I'm worried about my grade. Should I do Credit/NoCredit?
  It's fine with us, but check with your advisor to make sure it's accepted for your program of study.
- I want to learn more– what should I take next?
  - *Majors: CSci 135/136 (C++, MWTh 12:10-1pm + section) & CSci 150 (Discrete Structures, MTh 1:10-2:25pm + section)*
  - *Minors: CSci 133 (More Python: multiple times) & CSci 232 (Databases, multiple times)*
- What's a mock exam? I see it on the webpage...
  It's a practice exam that we're holding on 10 December.
  More details at the end of lecture.

Announcements



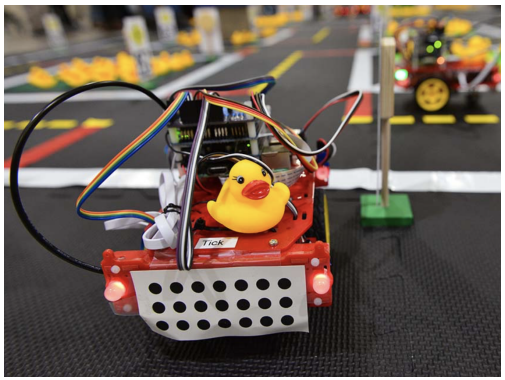


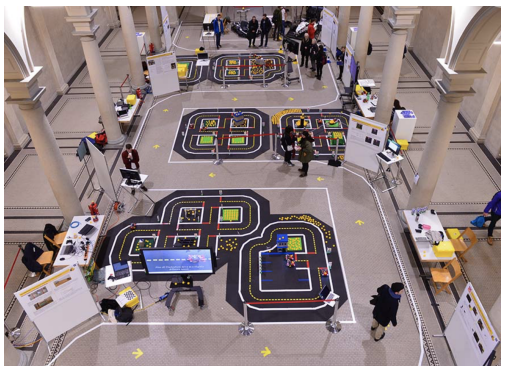
DuckieTown

(ETH Zurich, 2018)

- Two handouts today:
  - Lecture slip, and
  - Final exam plans (pink slip).

Announcements



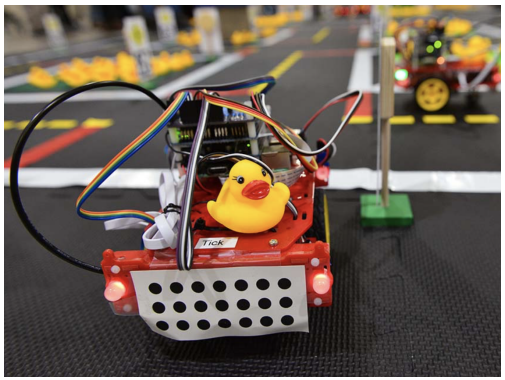


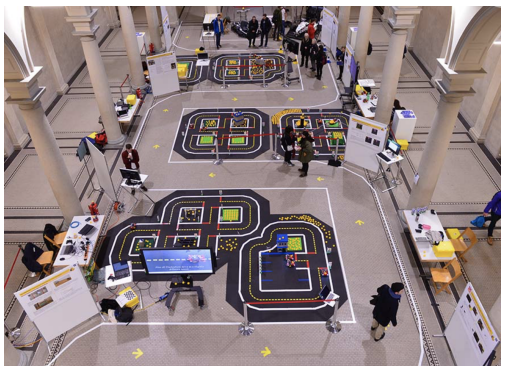
DuckieTown

(ETH Zurich, 2018)

- Two handouts today:
  - Lecture slip, and
  - Final exam plans (pink slip).
- 2 December: Deadline for January workshop on autonomous navigation: `https://bit.ly/2DhLLAN`

Announcements



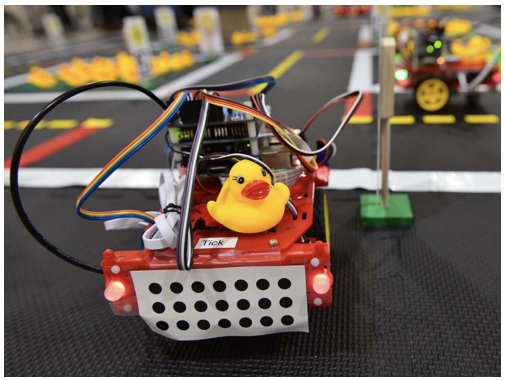


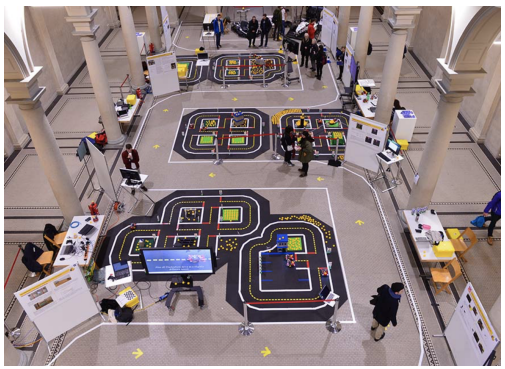
DuckieTown

(ETH Zurich, 2018)

- Two handouts today:
  - Lecture slip, and
  - Final exam plans (pink slip).
- 2 December: Deadline for January workshop on autonomous navigation: `https://bit.ly/2DhLLAN`
- Two weeks: Mock Exam (more at end of lecture).

Announcements





DuckieTown

(ETH Zurich, 2018)

- Two handouts today:
  - Lecture slip, and
  - Final exam plans (pink slip).
- 2 December: Deadline for January workshop on autonomous navigation: `https://bit.ly/2DhLLAN`
- Two weeks: Mock Exam (more at end of lecture).
- 20 days: Final Exam (more at end of lecture).

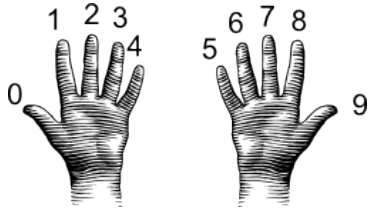
Today's Topics

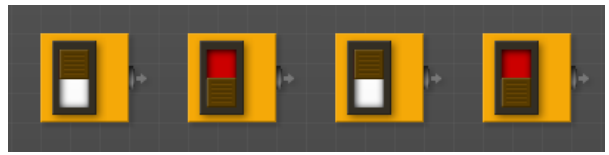
- Recap: Incrementer Design Challenge
- C++: Basic Format & Variables
- I/O and Definite Loops in C++
- More Info on the Final Exam

Today's Topics

- **Recap: Incrementer Design Challenge**
- C++: Basic Format & Variables
- I/O and Definite Loops in C++
- More Info on the Final Exam

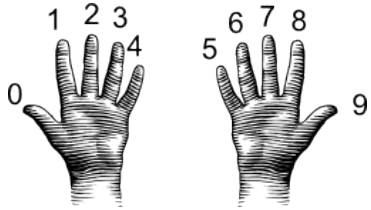
Recap: Design Challenge: Incrementers

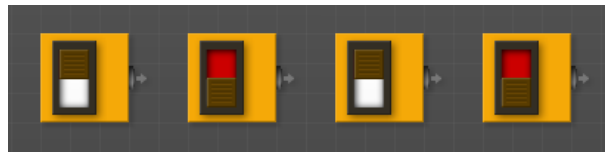




- Simplest arithmetic: add one ("increment") a variable.

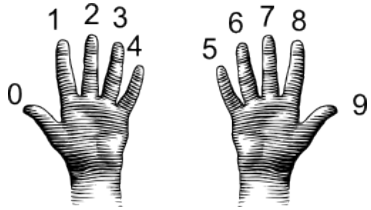
Recap: Design Challenge: Incrementers

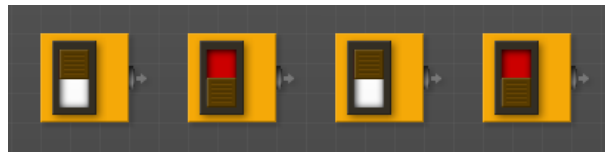




- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

Recap: Design Challenge: Incrementers

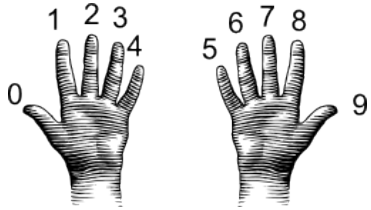


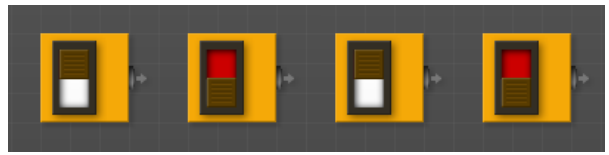


- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

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

Recap: Design Challenge: Incrementers



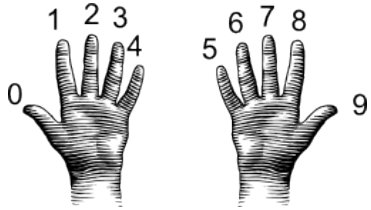


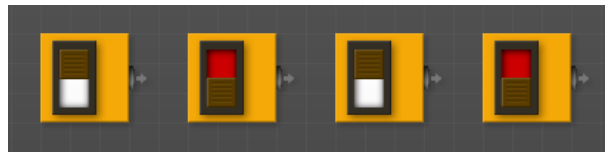
- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

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

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

Recap: Design Challenge: Incrementers



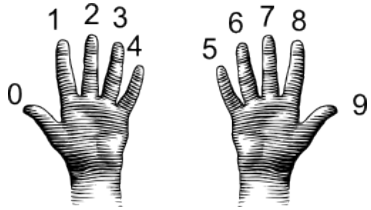


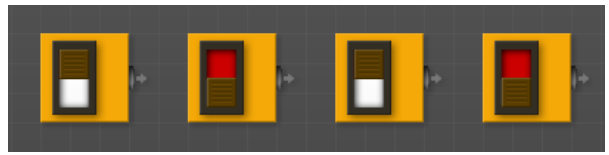
- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

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

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

Recap: Design Challenge: Incrementers





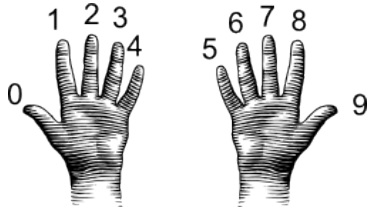
- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

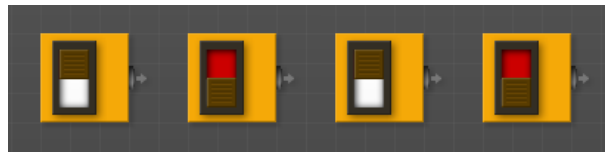
```
def addOne(n):
    m = n+1
    return(m)
```

- 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





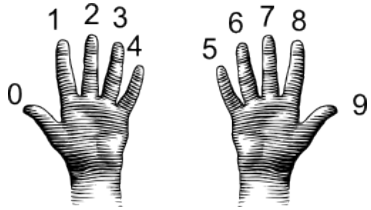
- Simplest arithmetic: add one ("increment") a variable.
- Example: Increment a decimal number:

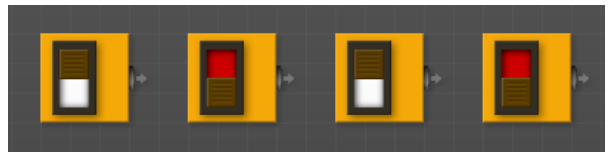
```
def addOne(n):
    m = n+1
    return(m)
```

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

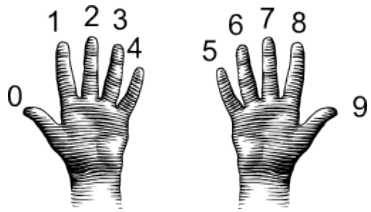
```
def addOne(n):
    m = n+1
    return(m)
```

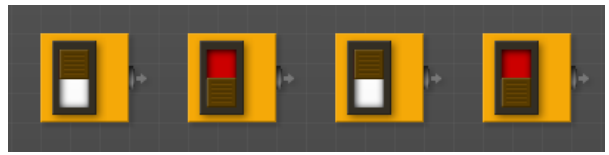
- Challenge: Write an algorithm for incrementing numbers expressed as words.
  Example: `"forty one"` → `"forty two"`

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

  Example: `"1001"` → `"1010"`

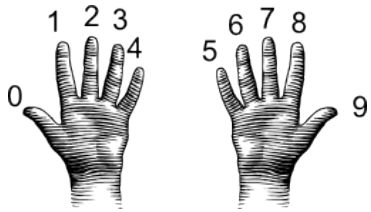
Recap: Incrementer Design Challenge

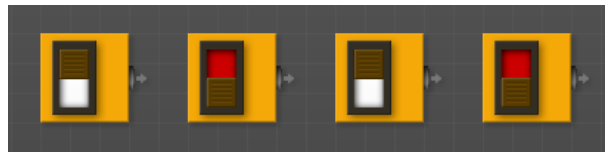




- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`

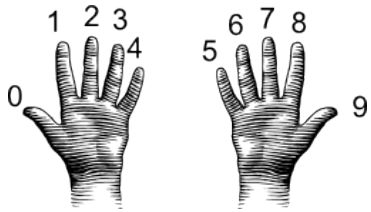
Recap: Incrementer Design Challenge

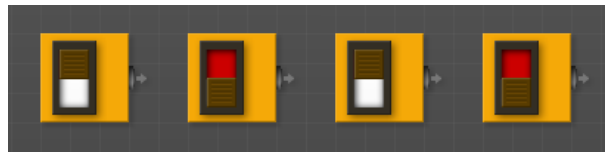




- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`
- *Hint: Convert to numbers, increment, and convert back to strings.*

Recap: Incrementer Design Challenge



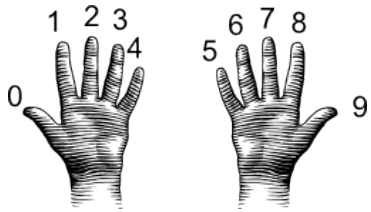


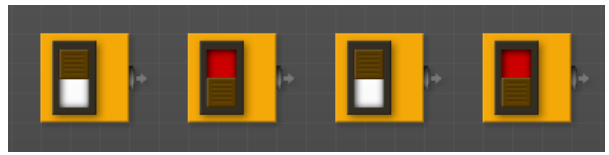
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`
- *Hint: Convert to numbers, increment, and convert back to strings.*

Pseudocode same for both questions:

1. Get user input.

Recap: Incrementer Design Challenge



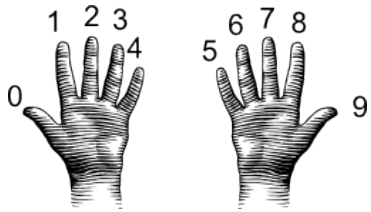


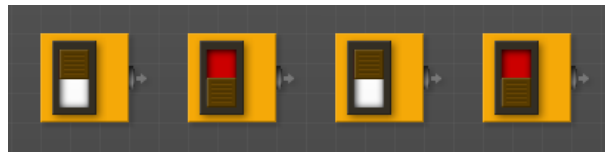
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`
- *Hint: Convert to numbers, increment, and convert back to strings.*

Pseudocode same for both questions:

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

Recap: Incrementer Design Challenge



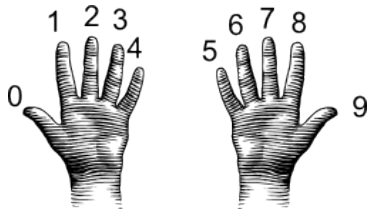


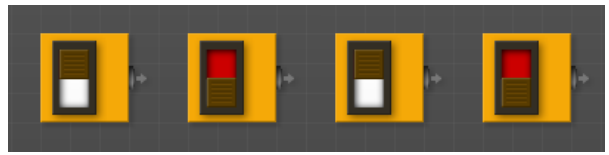
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` → `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` → `"1010"`
- *Hint: Convert to numbers, increment, and convert back to strings.*

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



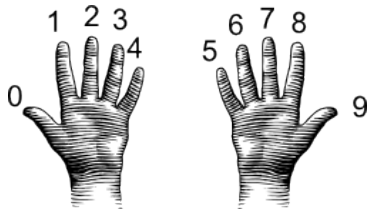


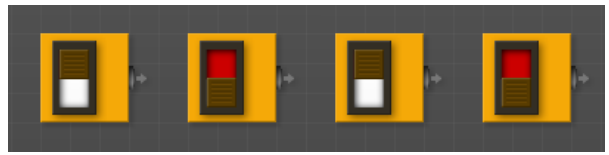
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`
- *Hint: Convert to numbers, increment, and convert back to strings.*

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



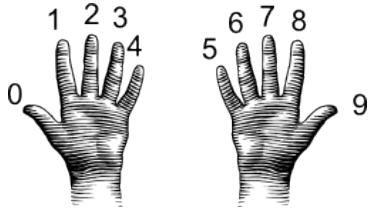


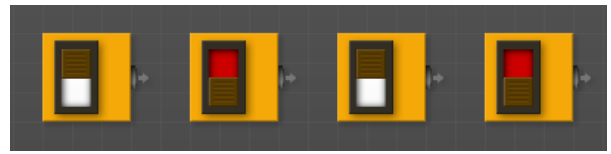
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` → `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` → `"1010"`
- *Hint: Convert to numbers, increment, and convert back to strings.*

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



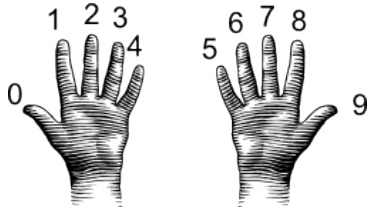


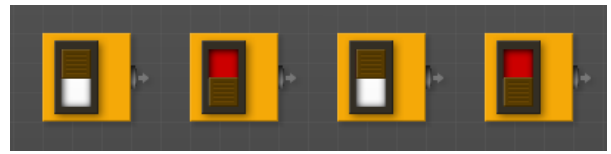
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`

Pseudocode same for both questions:

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

Recap: Incrementer Design Challenge



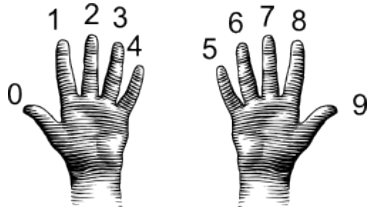


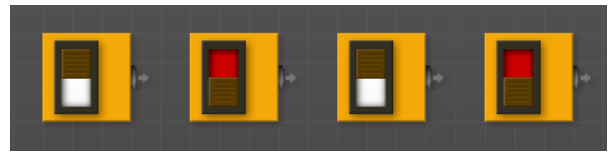
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`

Pseudocode same for both questions:

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

Recap: Incrementer Design Challenge



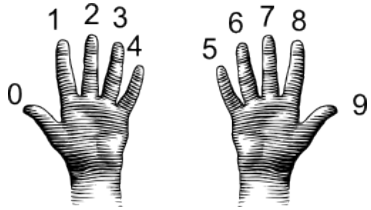


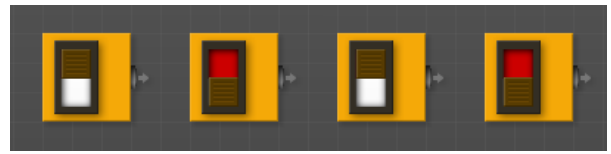
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`

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



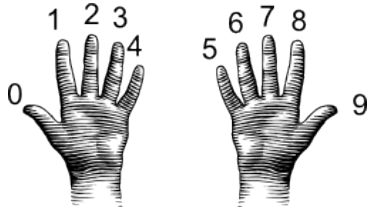


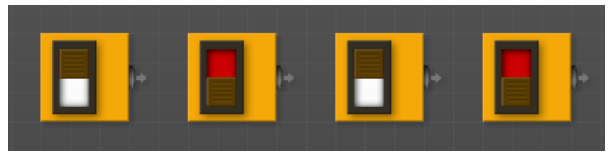
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`

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



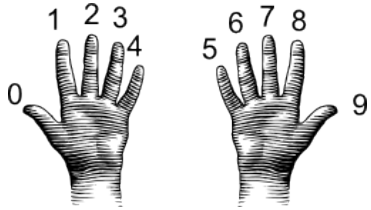


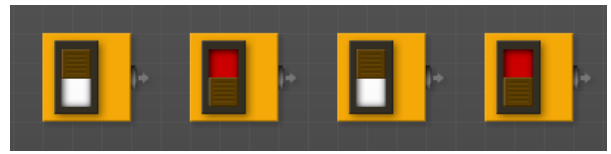
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` → `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` → `"1010"`

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



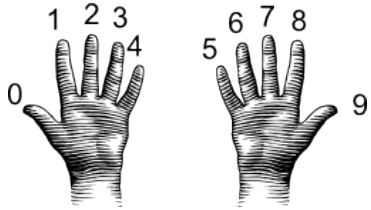


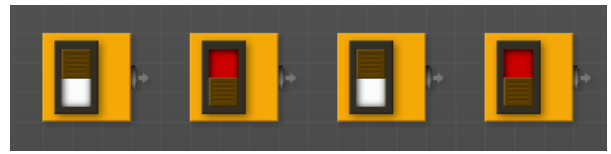
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`

Pseudocode same for both questions:

1. Get user input: `"1001"`

Recap: Incrementer Design Challenge



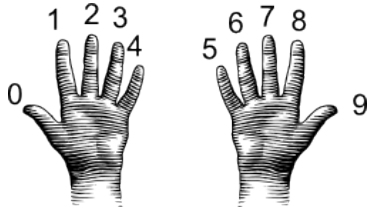


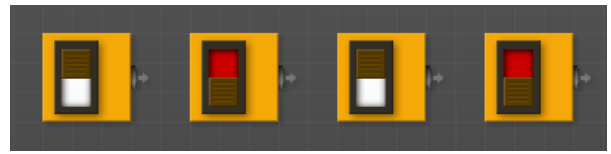
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`

Pseudocode same for both questions:

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

Recap: Incrementer Design Challenge



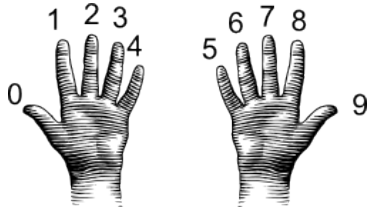


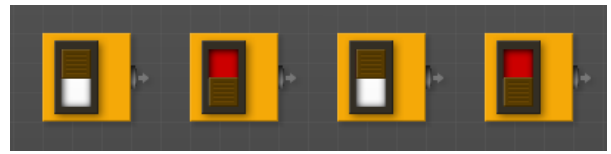
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`

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



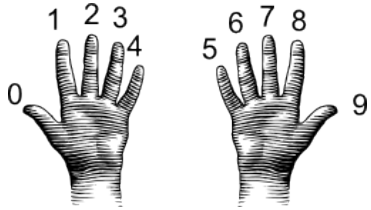


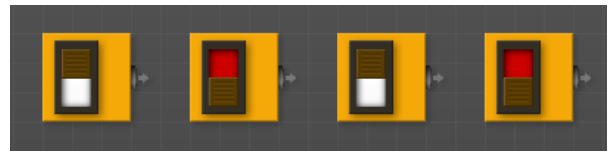
- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`

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



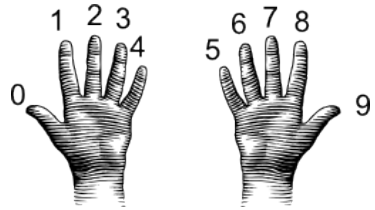


- Challenge: Write an algorithm for incrementing numbers expressed as words. Example: `"forty one"` $\rightarrow$ `"forty two"`
- Challenge: Write an algorithm for incrementing binary numbers. Example: `"1001"` $\rightarrow$ `"1010"`

Pseudocode same for both questions:

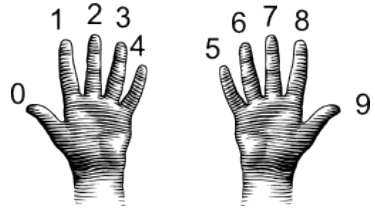
1. Get user input: `"1001"`
2. Convert to standard decimal number: 9
3. Add one (increment) the standard decimal number: 10
4. Convert back to your format: `"1010"`
5. Print the result.

Recap: Incrementer Design Challenge



Focus on: Convert to standard decimal number:

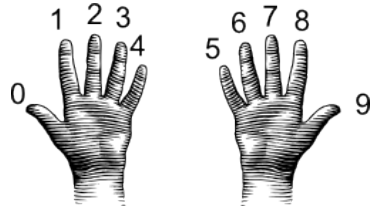
Recap: Incrementer Design Challenge



Focus on: Convert to standard decimal number:

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

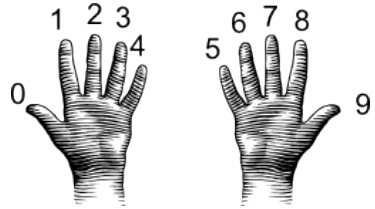
Recap: Incrementer Design Challenge



Focus on: Convert to standard decimal number:

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

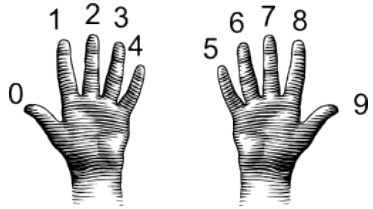
Recap: Incrementer Design Challenge



Focus on: Convert to standard decimal number:

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

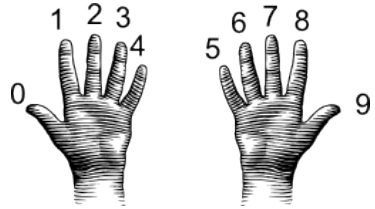
Recap: Incrementer Design Challenge



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)
```

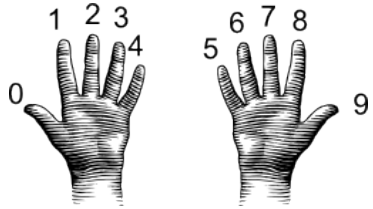
Recap: Incrementer Design Challenge



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)
```

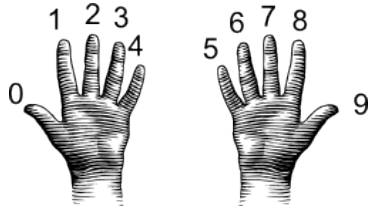
Recap: Incrementer Design Challenge



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

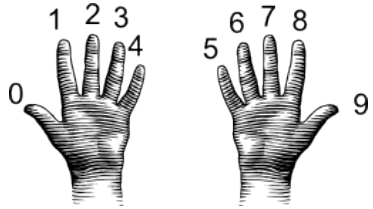


Focus on: Convert to standard decimal number:

```
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

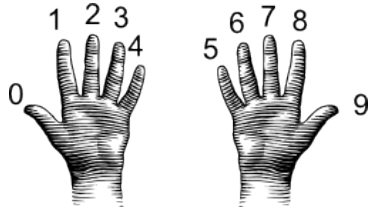


Focus on: Convert to standard decimal number:

```
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

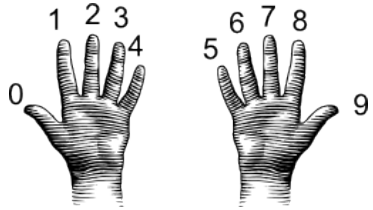


Focus on: Convert to standard decimal number:

```
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



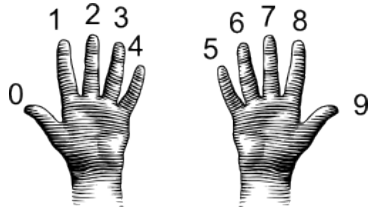
Focus on: Convert to standard decimal number:

```
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



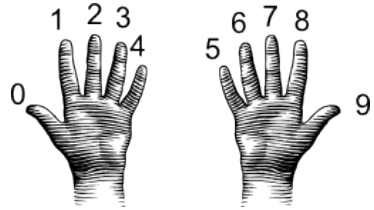
Focus on: Convert to standard decimal number:

```
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)?

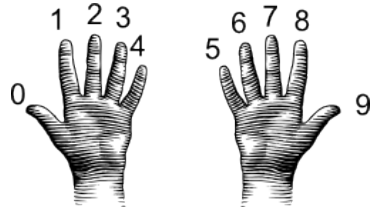
Unit Testing: testing individual units/functions/blocks of code to verify correctness. Often automated (e.g. gradescope).

Unit Testing: Incrementer Design Challenge



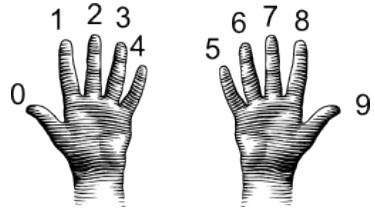
- Unit Testing: testing individual units/functions/blocks of code to verify correctness. Often automated (e.g. gradescope).
- To test all branches of code, would need to test all inputs: `"zero"`, `"one"`,..., `"nine"`, & some bad inputs.

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:

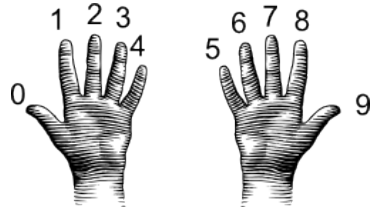
Unit Testing: Incrementer Design Challenge



- 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"]
```

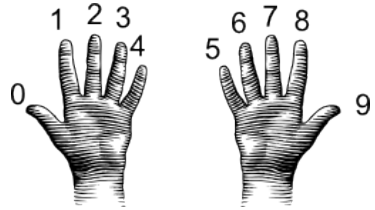
Unit Testing: Incrementer Design Challenge



- 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)
```

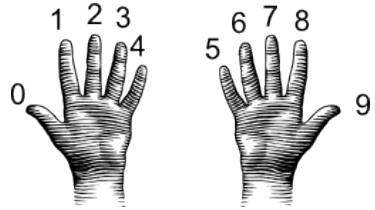
Unit Testing: Incrementer Design Challenge



- 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]):
```

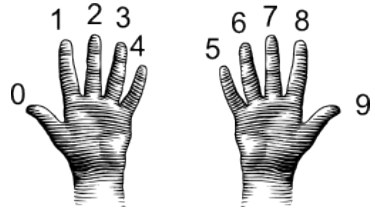
Unit Testing: Incrementer Design Challenge



- 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
```

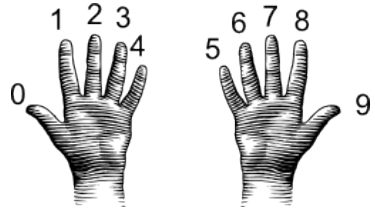
Unit Testing: Incrementer Design Challenge



- 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++
- More Info on the Final Exam

In Pairs or Triples:

- *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++

```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.

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++

```
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!!";`

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 >>`:

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;`

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++

```
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 }
```

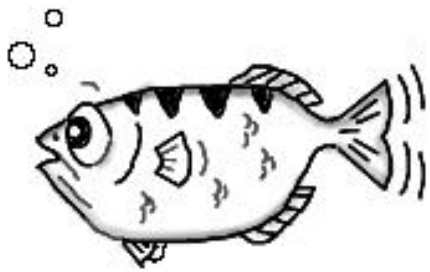
- Programs are organized in functions.
- Variables must be **declared**:
  `int num;`
- Many types available:
  `int, float, char, ...`
- Semicolons separate commands:
  `num = 5; more = 2*num;`
- To print, we'll use `cout <<`:
  `cout << "Hello!!";`
- To get input, we'll use `cin >>`:
  `cin >> num;`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

In Pairs or Triples:

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;
}
```

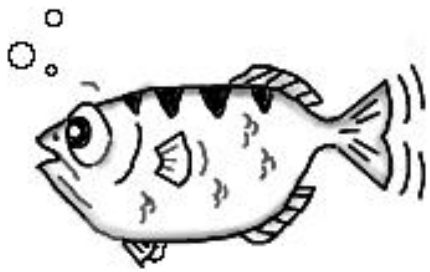
Side Note: gdb



gdb.org

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

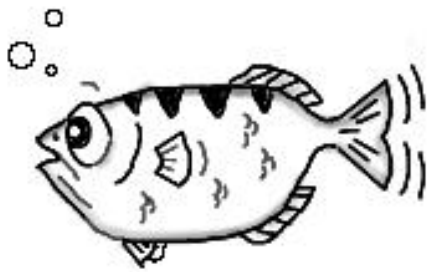
Side Note: gdb



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.

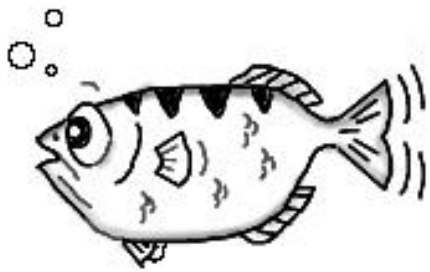
Side Note: gdb



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`)

In Pairs or Triples...

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++**
- More Info on the Final Exam

In Pairs or Triples:

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;
...

}

In Pairs or Triples:

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`)

In Pairs or Triples:

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;
}
```

C++ Demo

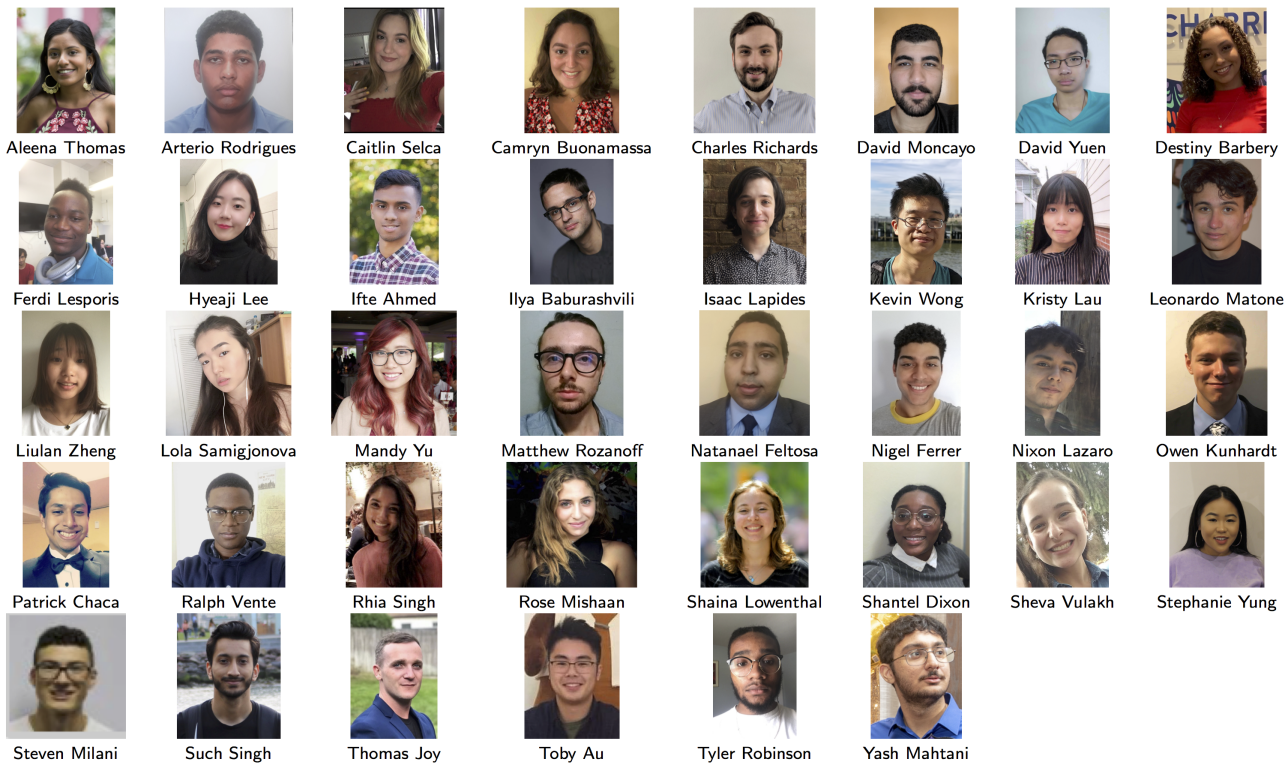
```cpp
//Growth example
#include <iostream>
using namespace std;

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;
}
```

(Demo with `onlinegdb`)

Lecture Slips

Which UTA have you spoken with most? Why?

Introductions: Undergraduate Teaching Assistants



Lecture Slips

In pairs or triples: **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;
}
```

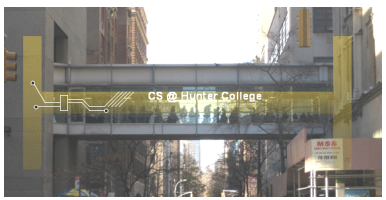
Lecture Slips

Translate line-by-line:

```
//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;
}
```

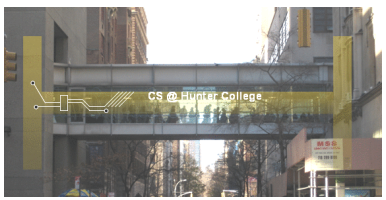
Recap: C++

- On lecture slip, write down a topic you wish we had spent more time (and why).

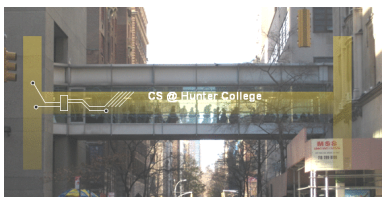


Recap: C++

- On lecture slip, write down a topic you wish we had spent more time (and why).
- C++ is a popular programming language that extends C.

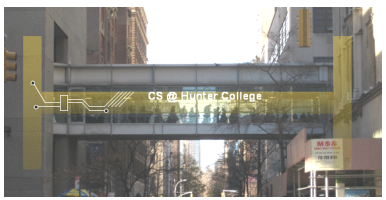


Recap: C++



- On lecture slip, write down a topic you wish we had spent more time (and why).
- C++ is a popular programming language that extends C.
- Input/Output (I/O):
  - `cin >>`
  - `cout <<`

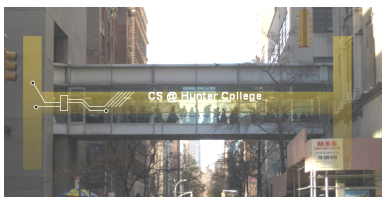
Recap: C++



- On lecture slip, write down a topic you wish we had spent more time (and why).
- C++ is a popular programming language that extends C.
- Input/Output (I/O):
  - `cin >>`
  - `cout <<`
- Definite loops:

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

Recap: C++



- On lecture slip, write down a topic you wish we had spent more time (and why).
- C++ is a popular programming language that extends C.
- Input/Output (I/O):
  - `cin >>`
  - `cout <<`
- Definite loops:

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

- Pass your lecture slip to the aisles for UTA's to collect.

Today's Topics

- Recap: Incrementer Design Challenge
- C++: Basic Format & Variables
- I/O and Definite Loops in C++
- **More Info on the Final Exam**

Final Exam: When

- The final exam is **Tuesday, 21 May, 9am-11am**, Assembly Hall (118 HN).

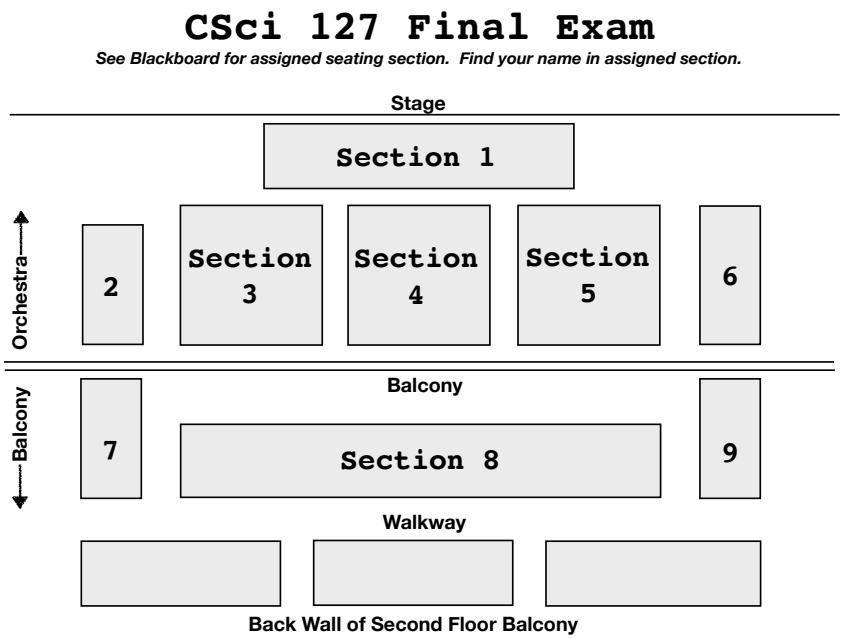
Final Exam: When

- The final exam is **Tuesday, 21 May, 9am-11am**, Assembly Hall (118 HN).
- If you have a conflict, the alternative time is: Wednesday, 15 May, 8:30-10:30am, 1001E HN.

Final Exam: When

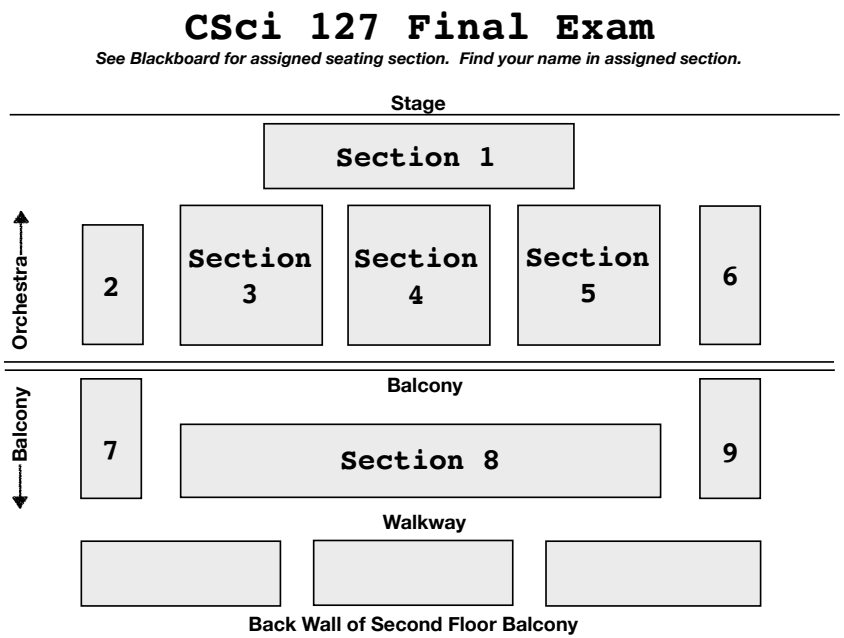
- The final exam is **Tuesday, 21 May, 9am-11am**, Assembly Hall (118 HN).
- If you have a conflict, the alternative time is: Wednesday, 15 May, 8:30-10:30am, 1001E HN.
- If you have accommodations via the Accessibility Office, we will send the exam to their testing center. (Must complete by noon, Tuesday, 21 May.)

Final Exam: Logistics



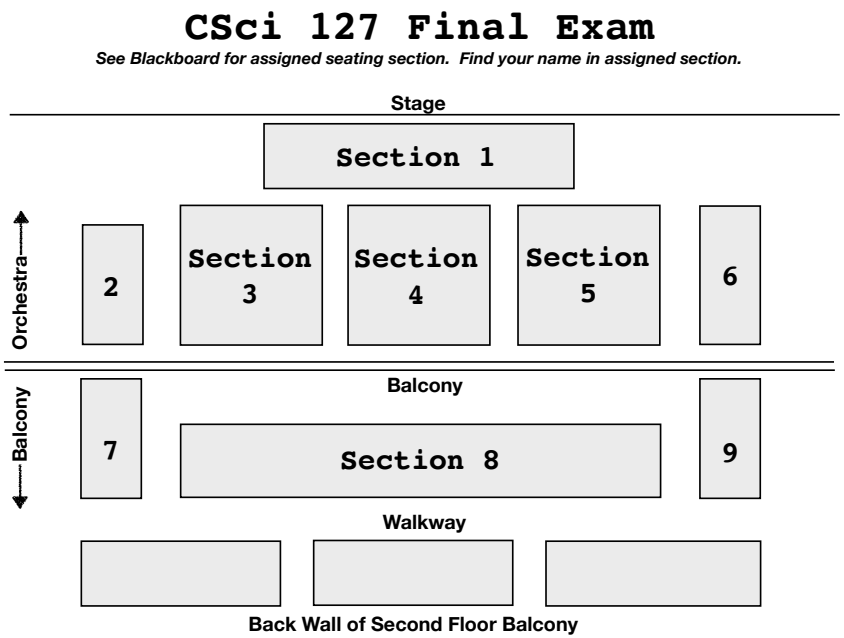
- Bring ID, note sheet, pencils or pens.

Final Exam: Logistics



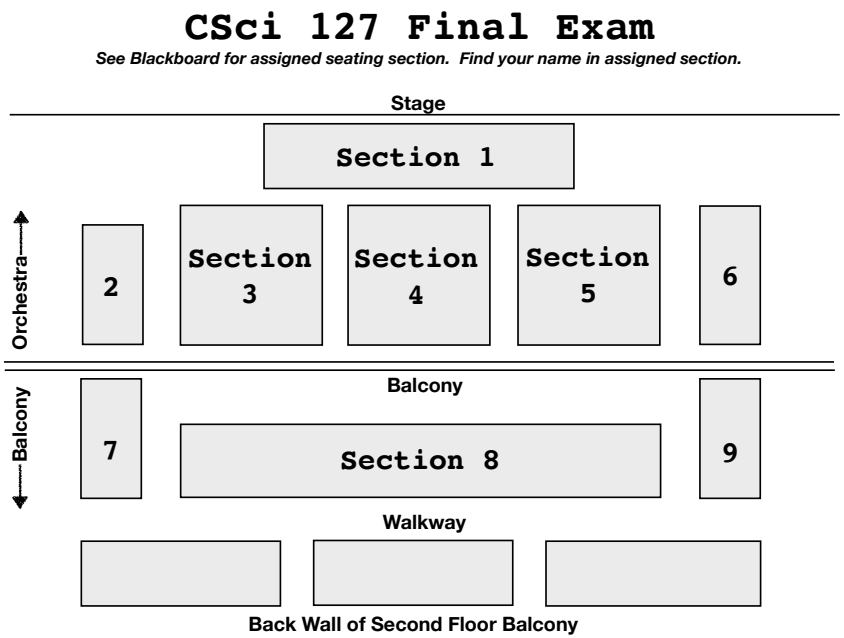
- Bring ID, note sheet, pencils or pens.
- Seating is assigned. See Blackboard for assignments.

Final Exam: Logistics



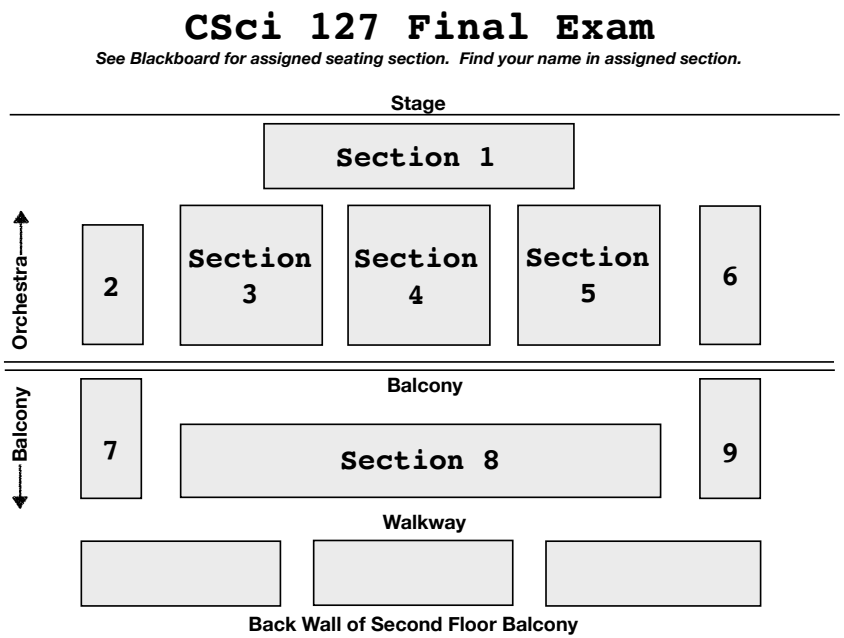
- Bring ID, note sheet, pencils or pens.
- Seating is assigned. See Blackboard for assignments.
- Sign out when you turn in your exam.

Final Exam: Logistics



- Bring ID, note sheet, pencils or pens.
- Seating is assigned. See Blackboard for assignments.
- Sign out when you turn in your exam.
- Cannot leave during the first 45 minutes of the exam.

Final Exam: Logistics



- Bring ID, note sheet, pencils or pens.
- Seating is assigned. See Blackboard for assignments.
- Sign out when you turn in your exam.
- Cannot leave during the first 45 minutes of the exam.
- Cannot start the exam after students start leaving.

Final Exam: Format

- The exam is 2 hours long.

Final Exam: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.

Final Exam: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.

Final Exam: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.

Final Exam: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
  - With notes, examples, programs: what will help you on the exam.

Final Exam: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
  - With notes, examples, programs: what will help you on the exam.
  - No origami– it's distracting to others taking the exam.

Final Exam: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
  - With notes, examples, programs: what will help you on the exam.
  - No origami– it's distracting to others taking the exam.
  - Best if you design/write yours since excellent way to study.

Final Exam: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
  - With notes, examples, programs: what will help you on the exam.
  - No origami– it's distracting to others taking the exam.
  - Best if you design/write yours since excellent way to study.
- The exam format:

Final Exam: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
  - With notes, examples, programs: what will help you on the exam.
  - No origami– it's distracting to others taking the exam.
  - Best if you design/write yours since excellent way to study.
- The exam format:
  - Printed on both sides of the paper.

Final Exam: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
  - With notes, examples, programs: what will help you on the exam.
  - No origami– it's distracting to others taking the exam.
  - Best if you design/write yours since excellent way to study.
- The exam format:
  - Printed on both sides of the paper.
  - 10 questions, each worth 10 points.

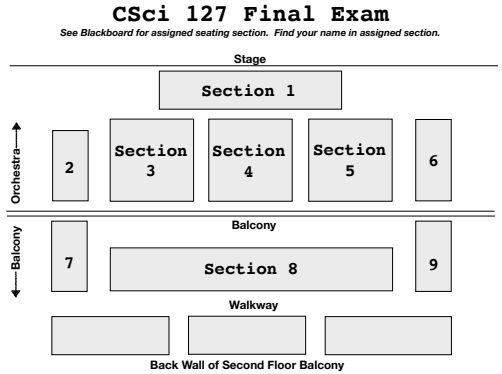
Final Exam: Format

- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
  - With notes, examples, programs: what will help you on the exam.
  - No origami– it's distracting to others taking the exam.
  - Best if you design/write yours since excellent way to study.
- The exam format:
  - Printed on both sides of the paper.
  - 10 questions, each worth 10 points.
  - Style of questions: what does the code do? short answer, write functions, top down design, & write complete programs.

Final Exam: Format

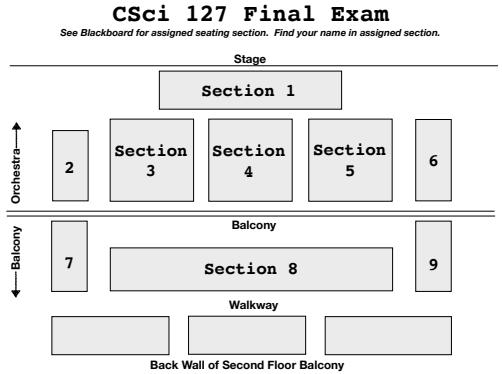
- The exam is 2 hours long.
- There are 4 different versions to discourage copying.
- It is on paper. No use of computers, phones, etc. allowed.
- You may have 1 piece of **8.5" x 11"** piece of paper.
  - With notes, examples, programs: what will help you on the exam.
  - No origami– it's distracting to others taking the exam.
  - Best if you design/write yours since excellent way to study.
- The exam format:
  - Printed on both sides of the paper.
  - 10 questions, each worth 10 points.
  - 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).

Mock Final



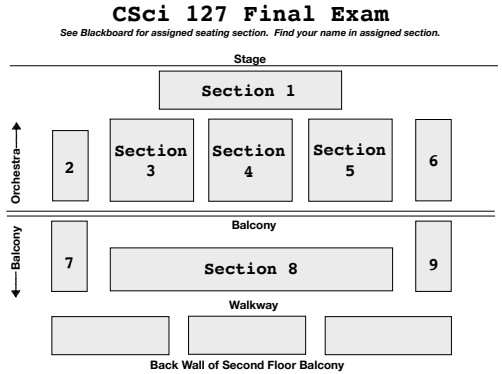
- Given in lecture on 10 December.

Mock Final



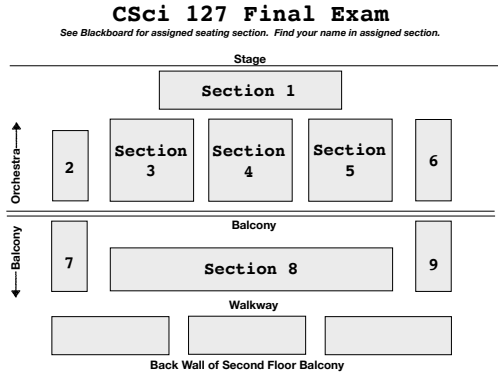
- Given in lecture on 10 December.
- Practice exam: the same format as the final (except 1, not full 2 hours).

Mock Final



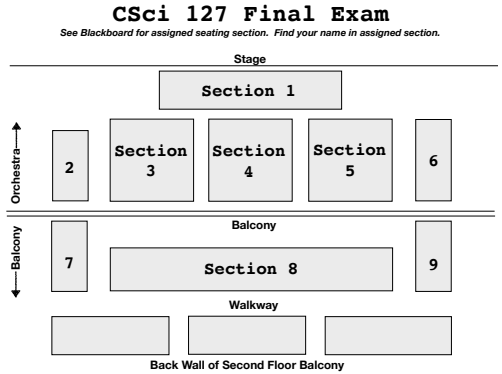
- Given in lecture on 10 December.
- Practice exam: the same format as the final (except 1, not full 2 hours).
- Bring ID & 1 page of notes (will check IDs during exam).

Mock Final



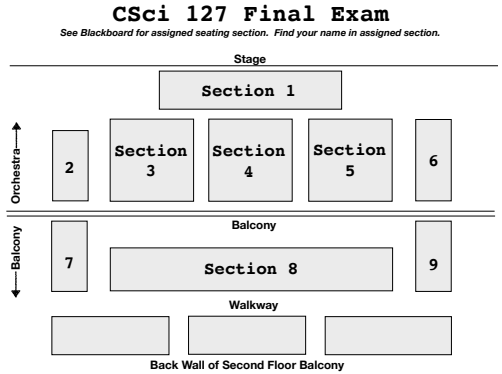
- Given in lecture on 10 December.
- Practice exam: the same format as the final (except 1, not full 2 hours).
- Bring ID & 1 page of notes (will check IDs during exam).
- No electronics (i.e. computers, tablets, smart watches, ear buds, etc.).
- Seating is assigned.

Mock Final



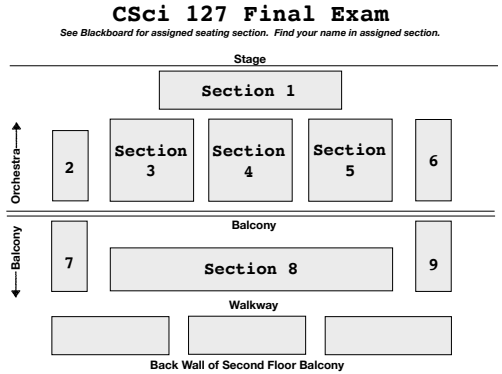
- Given in lecture on 10 December.
- Practice exam: the same format as the final (except 1, not full 2 hours).
- Bring ID & 1 page of notes (will check IDs during exam).
- No electronics (i.e. computers, tablets, smart watches, ear buds, etc.).
- Seating is assigned.
  - Assigned seating (see Blackboard): every other seat, every other row.

Mock Final



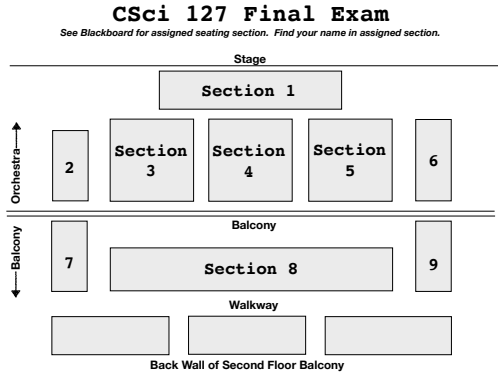
- Given in lecture on 10 December.
- Practice exam: the same format as the final (except 1, not full 2 hours).
- Bring ID & 1 page of notes (will check IDs during exam).
- No electronics (i.e. computers, tablets, smart watches, ear buds, etc.).
- Seating is assigned.
  - Assigned seating (see Blackboard): every other seat, every other row.
  - Fill out exam slip (pre-printed at your designated seat).

Mock Final



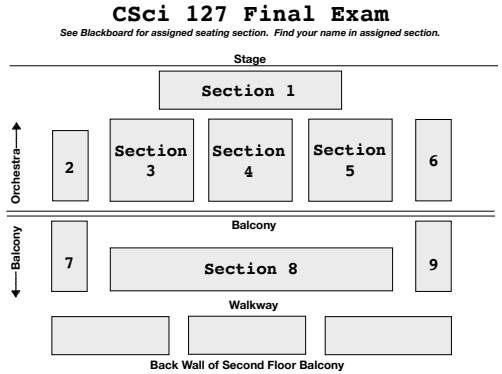
- Given in lecture on 10 December.
- Practice exam: the same format as the final (except 1, not full 2 hours).
- Bring ID & 1 page of notes (will check IDs during exam).
- No electronics (i.e. computers, tablets, smart watches, ear buds, etc.).
- Seating is assigned.
  - Assigned seating (see Blackboard): every other seat, every other row.
  - Fill out exam slip (pre-printed at your designated seat).
  - Sign out as you leave (clipboards for each section).

Mock Final



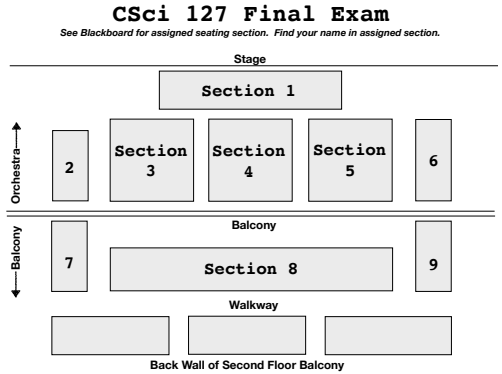
- Given in lecture on 10 December.
- Practice exam: the same format as the final (except 1, not full 2 hours).
- Bring ID & 1 page of notes (will check IDs during exam).
- No electronics (i.e. computers, tablets, smart watches, ear buds, etc.).
- Seating is assigned.
  - Assigned seating (see Blackboard): every other seat, every other row.
  - Fill out exam slip (pre-printed at your designated seat).
  - Sign out as you leave (clipboards for each section).
  - Cannot leave in first 30 minutes.

Mock Final



- Given in lecture on 10 December.
- Practice exam: the same format as the final (except 1, not full 2 hours).
- Bring ID & 1 page of notes (will check IDs during exam).
- No electronics (i.e. computers, tablets, smart watches, ear buds, etc.).
- Seating is assigned.
  - Assigned seating (see Blackboard): every other seat, every other row.
  - Fill out exam slip (pre-printed at your designated seat).
  - Sign out as you leave (clipboards for each section).
  - Cannot leave in first 30 minutes.
- Lecture slip for that week: exam slips & signing out at end of exam.

Mock Final



- Given in lecture on 10 December.
- Practice exam: the same format as the final (except 1, not full 2 hours).
- Bring ID & 1 page of notes (will check IDs during exam).
- No electronics (i.e. computers, tablets, smart watches, ear buds, etc.).
- Seating is assigned.
  - Assigned seating (see Blackboard): every other seat, every other row.
  - Fill out exam slip (pre-printed at your designated seat).
  - Sign out as you leave (clipboards for each section).
  - Cannot leave in first 30 minutes.
- Lecture slip for that week: exam slips & signing out at end of exam.
- Answer key will be available on webpage after lecture.

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: Top-Down Design & APIs

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

Final Overview: Top-Down Design & APIs

For each question, write **only 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.
- Write a function that takes a string and returns its length.
- Write a function that, given a DataFrame, returns the minimal value in the first column.
- Write a function that takes a whole number and returns the corresponding binary number as a string.
- 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 Overview: Top-Down Design & APIs

For each question, write **only 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.
- Write a function that takes a string and returns its length.
- Write a function that, given a DataFrame, returns the minimal value in the first column.
- Write a function that takes a whole number and returns the corresponding binary number as a string.
- Write a function that computes the total monthly payment when given the initial loan amount, annual interest rate, number of years of the loan.

(Hint: highlight key words, make list of inputs, list of outputs, then put together.)

Final Overview

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 Overview

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 Overview

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 Overview

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 Overview

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 Overview

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 Overview

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 Overview

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 Overview

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 Overview

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 Overview

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 Overview

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 Overview

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 Overview

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)
```

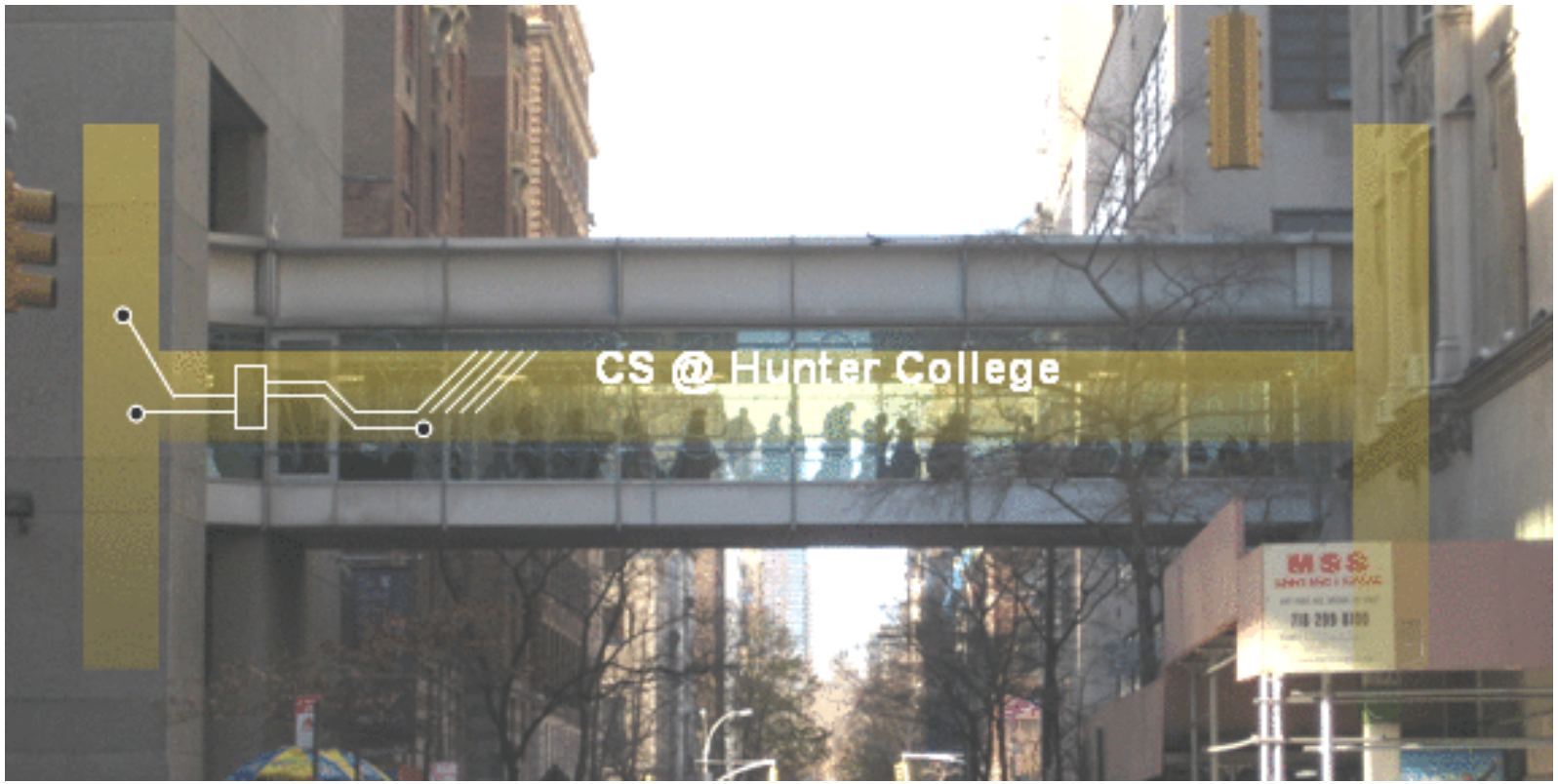
Final Overview

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)
```

Writing Boards



- Return writing boards as you leave...