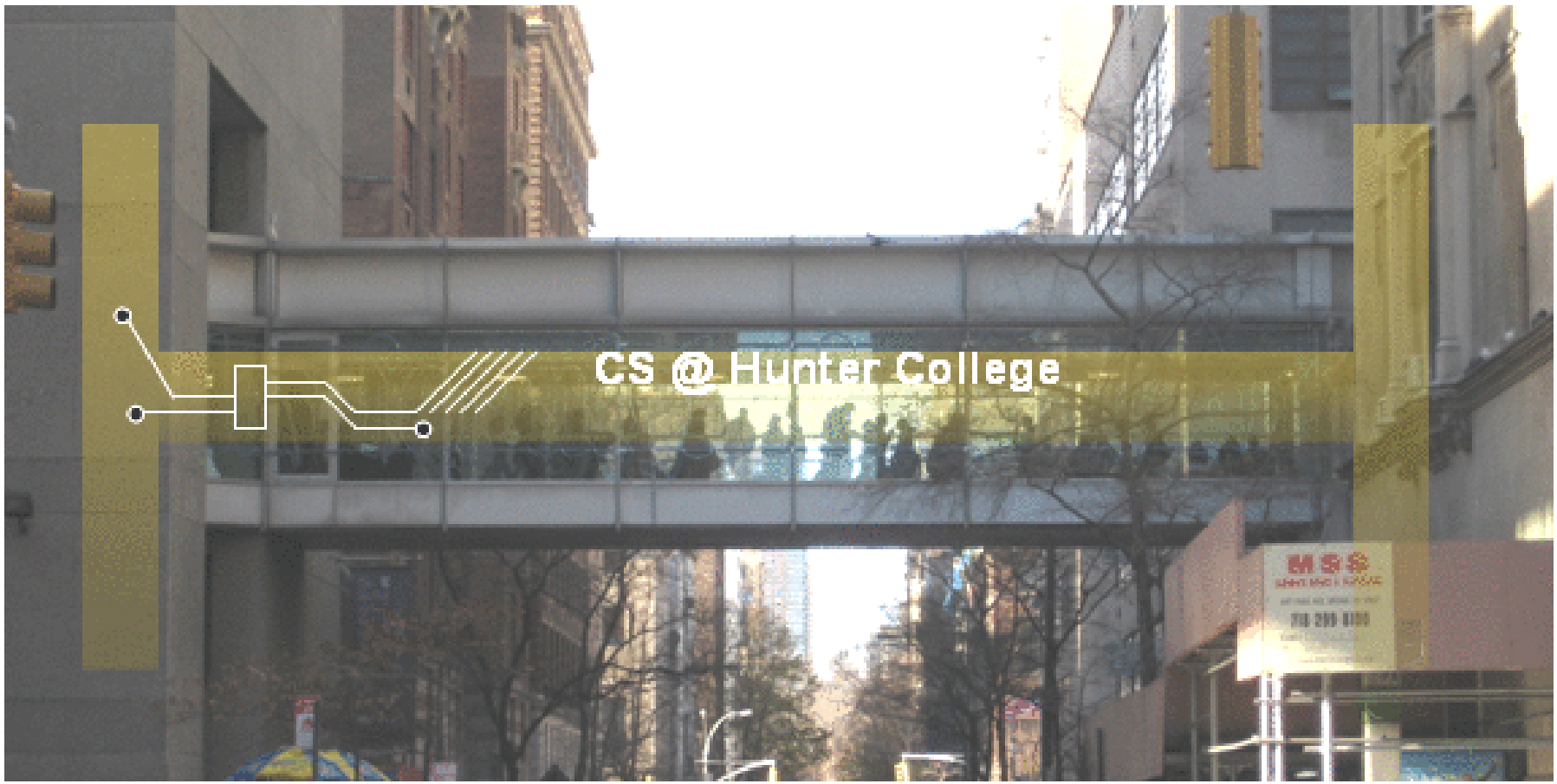
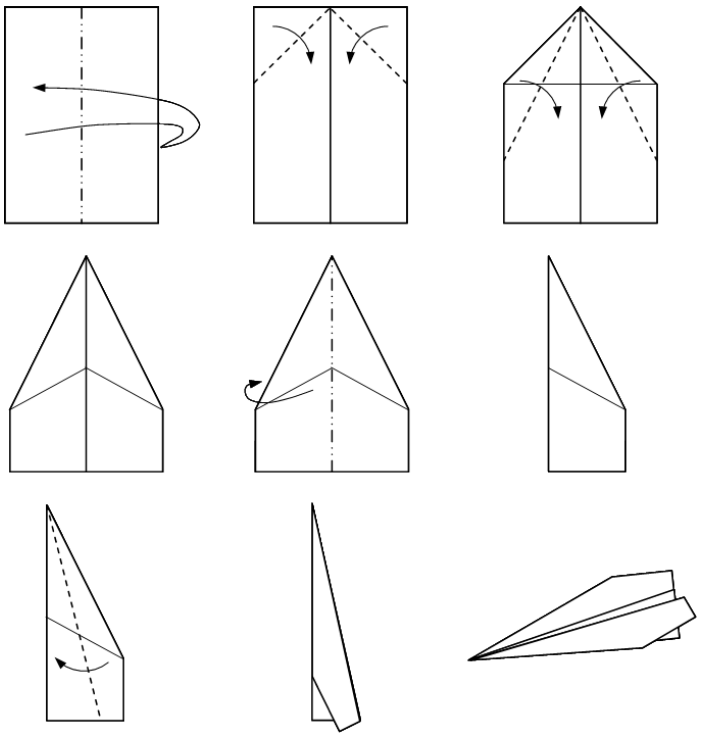
CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Announcements



- Guest Lecturer: Katherine Howitt

Frequently Asked Questions

From lecture slips & recitation sections.

Frequently Asked Questions

From lecture slips & recitation sections.

- Can I get a copy of the lecture slides and programs from lecture?

Frequently Asked Questions

From lecture slips & recitation sections.

- Can I get a copy of the lecture slides and programs from lecture?
  Yes, the slides are posted on the class website.

Frequently Asked Questions

From lecture slips & recitation sections.

- Can I get a copy of the lecture slides and programs from lecture?
  Yes, the slides are posted on the class website.
- Can I work ahead on programs?

Frequently Asked Questions

From lecture slips & recitation sections.

- Can I get a copy of the lecture slides and programs from lecture?
  Yes, the slides are posted on the class website.
- Can I work ahead on programs?
  Yes, you'll get the most out of lecture if you're 5 or so programs ahead. We give an extra 7-10 days on deadlines from when material is presented.

Frequently Asked Questions

From lecture slips & recitation sections.

- Can I get a copy of the lecture slides and programs from lecture?
  Yes, the slides are posted on the class website.
- Can I work ahead on programs?
  Yes, you'll get the most out of lecture if you're 5 or so programs ahead. We give an extra 7-10 days on deadlines from when material is presented.
- I'm sure I did Problem 9 correctly, but Gradescope disagrees. Why?

Frequently Asked Questions

From lecture slips & recitation sections.

- Can I get a copy of the lecture slides and programs from lecture?
  Yes, the slides are posted on the class website.
- Can I work ahead on programs?
  Yes, you'll get the most out of lecture if you're 5 or so programs ahead. We give an extra 7-10 days on deadlines from when material is presented.
- I'm sure I did Problem 9 correctly, but Gradescope disagrees. Why?
  Some of the grading scripts are really nicky about spacing and new lines. Let us know{ some we can x, some have to match exactly.

Frequently Asked Questions

From lecture slips & recitation sections.

- Can I get a copy of the lecture slides and programs from lecture?
  Yes, the slides are posted on the class website.
- Can I work ahead on programs?
  Yes, you'll get the most out of lecture if you're 5 or so programs ahead. We give an extra 7-10 days on deadlines from when material is presented.
- I'm sure I did Problem 9 correctly, but Gradescope disagrees. Why?
  Some of the grading scripts are really nicky about spacing and new lines. Let us know{ some we can x, some have to match exactly.
- I don't get the brackets. Why are using them? Why not parenthesis?

Frequently Asked Questions

From lecture slips & recitation sections.

- Can I get a copy of the lecture slides and programs from lecture?
  Yes, the slides are posted on the class website.
- Can I work ahead on programs?
  Yes, you'll get the most out of lecture if you're 5 or so programs ahead. We give an extra 7-10 days on deadlines from when material is presented.
- I'm sure I did Problem 9 correctly, but Gradescope disagrees. Why?
  Some of the grading scripts are really nicky about spacing and new lines. Let us know{ some we can x, some have to match exactly.
- I don't get the brackets. Why are using them? Why not parenthesis?
  Parenthesis are for functions:

Frequently Asked Questions

From lecture slips & recitation sections.

- Can I get a copy of the lecture slides and programs from lecture?
  Yes, the slides are posted on the class website.
- Can I work ahead on programs?
  Yes, you'll get the most out of lecture if you're 5 or so programs ahead. We give an extra 7-10 days on deadlines from when material is presented.
- I'm sure I did Problem 9 correctly, but Gradescope disagrees. Why?
  Some of the grading scripts are really nicky about spacing and new lines. Let us know{ some we can x, some have to match exactly.
- I don't get the brackets. Why are using them? Why not parenthesis?
  Parenthesis are for functions: `print("Hi!")`

Frequently Asked Questions

From lecture slips & recitation sections.

- Can I get a copy of the lecture slides and programs from lecture?
  Yes, the slides are posted on the class website.
- Can I work ahead on programs?
  Yes, you'll get the most out of lecture if you're 5 or so programs ahead. We give an extra 7-10 days on deadlines from when material is presented.
- I'm sure I did Problem 9 correctly, but Gradescope disagrees. Why?
  Some of the grading scripts are really nicky about spacing and new lines. Let us know{ some we can x, some have to match exactly.
- I don't get the brackets. Why are using them? Why not parenthesis?
  Parenthesis are for functions: `print("Hi!")` *or* `tori.left(90)`

Frequently Asked Questions

From lecture slips & recitation sections.

- Can I get a copy of the lecture slides and programs from lecture?
  Yes, the slides are posted on the class website.
- Can I work ahead on programs?
  Yes, you'll get the most out of lecture if you're 5 or so programs ahead. We give an extra 7-10 days on deadlines from when material is presented.
- I'm sure I did Problem 9 correctly, but Gradescope disagrees. Why?
  Some of the grading scripts are really nicky about spacing and new lines. Let us know{ some we can x, some have to match exactly.
- I don't get the brackets. Why are using them? Why not parenthesis?
  Parenthesis are for functions: `print("Hi!")` *or* `tori.left(90)`
  Brackets are for accessing part of a list or string:

Frequently Asked Questions

From lecture slips & recitation sections.

- Can I get a copy of the lecture slides and programs from lecture?
  Yes, the slides are posted on the class website.
- Can I work ahead on programs?
  Yes, you'll get the most out of lecture if you're 5 or so programs ahead. We give an extra 7-10 days on deadlines from when material is presented.
- I'm sure I did Problem 9 correctly, but Gradescope disagrees. Why?
  Some of the grading scripts are really nicky about spacing and new lines. Let us know{ some we can x, some have to match exactly.
- I don't get the brackets. Why are using them? Why not parenthesis?
  Parenthesis are for functions: `print("Hi!")` *or* `tori.left(90)`
  Brackets are for accessing part of a list or string: `words[1]`

Frequently Asked Questions

From lecture slips & recitation sections.

- Can I get a copy of the lecture slides and programs from lecture?
  Yes, the slides are posted on the class website.
- Can I work ahead on programs?
  Yes, you'll get the most out of lecture if you're 5 or so programs ahead. We give an extra 7-10 days on deadlines from when material is presented.
- I'm sure I did Problem 9 correctly, but Gradescope disagrees. Why?
  Some of the grading scripts are really nicky about spacing and new lines. Let us know{ some we can x, some have to match exactly.
- I don't get the brackets. Why are using them? Why not parenthesis?
  Parenthesis are for functions: `print("Hi!")` *or* `tori.left(90)`
  Brackets are for accessing part of a list or string: `words[1]` *or* `mess[3:6]`.

Frequently Asked Questions

From lecture slips & recitation sections.

- Can I get a copy of the lecture slides and programs from lecture?
  Yes, the slides are posted on the class website.
- Can I work ahead on programs?
  Yes, you'll get the most out of lecture if you're 5 or so programs ahead. We give an extra 7-10 days on deadlines from when material is presented.
- I'm sure I did Problem 9 correctly, but Gradescope disagrees. Why?
  Some of the grading scripts are really nicky about spacing and new lines. Let us know{ some we can x, some have to match exactly.
- I don't get the brackets. Why are using them? Why not parenthesis?
  Parenthesis are for functions: `print("Hi!")` *or* `tori.left(90)`
  Brackets are for accessing part of a list or string: `words[1]` *or* `mess[3:6]`.
- Could you explain more about arithmetic (especially modulo!) in Python?

Frequently Asked Questions

From lecture slips & recitation sections.

- Can I get a copy of the lecture slides and programs from lecture?
  Yes, the slides are posted on the class website.
- Can I work ahead on programs?
  Yes, you'll get the most out of lecture if you're 5 or so programs ahead. We give an extra 7-10 days on deadlines from when material is presented.
- I'm sure I did Problem 9 correctly, but Gradescope disagrees. Why?
  Some of the grading scripts are really nicky about spacing and new lines. Let us know{ some we can x, some have to match exactly.
- I don't get the brackets. Why are using them? Why not parenthesis?
  Parenthesis are for functions: `print("Hi!")` *or* `tori.left(90)`
  Brackets are for accessing part of a list or string: `words[1]` *or* `mess[3:6]`.
- Could you explain more about arithmetic (especially modulo!) in Python?
  Yes, will do! We'll start out with arithmetic.

Frequently Asked Questions

From lecture slips & recitation sections.

- Can I get a copy of the lecture slides and programs from lecture?
  Yes, the slides are posted on the class website.
- Can I work ahead on programs?
  Yes, you'll get the most out of lecture if you're 5 or so programs ahead. We give an extra 7-10 days on deadlines from when material is presented.
- I'm sure I did Problem 9 correctly, but Gradescope disagrees. Why?
  Some of the grading scripts are really nicky about spacing and new lines. Let us know{ some we can x, some have to match exactly.
- I don't get the brackets. Why are using them? Why not parenthesis?
  Parenthesis are for functions: `print("Hi!")` *or* `tori.left(90)`
  Brackets are for accessing part of a list or string: `words[1]` *or* `mess[3:6]`.
- Could you explain more about arithmetic (especially modulo!) in Python?
  Yes, will do! We'll start out with arithmetic.
- One more time on all the `range()` options?

Frequently Asked Questions

From lecture slips & recitation sections.

- Can I get a copy of the lecture slides and programs from lecture?
  Yes, the slides are posted on the class website.
- Can I work ahead on programs?
  Yes, you'll get the most out of lecture if you're 5 or so programs ahead. We give an extra 7-10 days on deadlines from when material is presented.
- I'm sure I did Problem 9 correctly, but Gradescope disagrees. Why?
  Some of the grading scripts are really nicky about spacing and new lines. Let us know{ some we can x, some have to match exactly.
- I don't get the brackets. Why are using them? Why not parenthesis?
  Parenthesis are for functions: `print("Hi!")` *or* `tori.left(90)`
  Brackets are for accessing part of a list or string: `words[1]` *or* `mess[3:6]`.
- Could you explain more about arithmetic (especially modulo!) in Python?
  Yes, will do! We'll start out with arithmetic.
- One more time on all the `range()` options?
  We'll have some in group work and a quick review.

Today's Topics

- Arithmetic
- Indexing and Slicing Lists
- Design Challenge: Planes
- Colors & Hexadecimal Notation

Today's Topics

- **Arithmetic**
- Indexing and Slicing Lists
- Design Challenge: Planes
- Colors & Hexadecimal Notation

Arithmetic

Some arithmetic operators in Python:

- Addition:

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction:

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication:

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division:

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division: `ave = total / n`

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division: `ave = total / n`
- Floor or Integer Division:

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division: `ave = total / n`
- Floor or Integer Division:
  `weeks = totalDays // 7`

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division: `ave = total / n`
- Floor or Integer Division:
  `weeks = totalDays // 7`
- Remainder or Modulus:

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division: `ave = total / n`
- Floor or Integer Division:
  `weeks = totalDays // 7`
- Remainder or Modulus:
  `days = totalDays % 7`

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division: `ave = total / n`
- Floor or Integer Division:
  `weeks = totalDays // 7`
- Remainder or Modulus:
  `days = totalDays % 7`
- Exponentiaion:

Arithmetic

Some arithmetic operators in Python:

- Addition: `sum = sum + 3`
- Subtraction: `deb = deb - item`
- Multiplication: `area = h * w`
- Division: `ave = total / n`
- Floor or Integer Division:
  `weeks = totalDays // 7`
- Remainder or Modulus:
  `days = totalDays % 7`
- Exponentiaion:
  `pop = 2**time`

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 9 and 2.

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 9 and 2.
- If the user enters, 12 and 4.

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 9 and 2.
- If the user enters, 12 and 4.
- If the user enters, 8 and 20.

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 9 and 2.
- If the user enters, 12 and 4.
- If the user enters, 8 and 20.
- If the user enters, 11 and 1.

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 9 and 2.

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 9 and 2.

```
Enter starting time: 9
Enter how long: 2
Your event starts at 9 o'clock.
Your event ends at 11 o'clock.
```

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 12 and 4.

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 12 and 4.

```
Enter starting time: 12
Enter how long: 4
Your event starts at 12 o'clock.
Your event ends at 4 o'clock.
```

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 8 and 20.

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 8 and 20.

```
Enter starting time: 8
Enter how long: 20
Your event starts at 8 o'clock.
Your event ends at 4 o'clock.
```

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 11 and 1.

In Pairs or Triples...

What does this code do?

```
#Mystery code for lecture 3

startTime = int(input('Enter starting time: '))
duration = int(input('Enter how long: '))

print('Your event starts at', startTime, "o'clock.")

endTime = (startTime+duration)%12
print('Your event ends at', endTime, "o'clock.")
```

In particular, what is printed...

- If the user enters, 11 and 1.

```
Enter starting time: 11
Enter how long: 1
Your event starts at 11 o'clock.
Your event ends at 0 o'clock.
```

Today's Topics

- Arithmetic
- **Indexing and Slicing Lists**
- Design Challenge: Planes
- Colors & Hexadecimal Notation

In Pairs or Triples...

Mostly review:

```
 1  for d in range(10, 0, -1):
 2      print(d)
 3  print("Blast off!")
 4
 5  for num in range(5,8):
 6      print(num, 2*num)
 7
 8  s = "City University of New York"
 9  print(s[3], s[0:3], s[:3])
10  print(s[5:8], s[-1])
11
12  names = ["Eleanor", "Anna", "Alice", "Edith"]
13  for n in names:
14      print(n)
```

Python Tutor

```
1  for d in range(10, 0, -1):
2      print(d)
3  print("Blast off!")
4
5  for num in range(5,8):
6      print(num, 2*num)
7
8  s = "City University of New York"
9  print(s[3], s[0:3], s[:3])
10 print(s[5:8], s[-1])
11
12 names = ["Eleanor", "Anna", "Alice", "Edith"]
13 for n in names:
14     print(n)
```

(Demo with pythonTutor)

Review: range()

The three versions:

Review: range()

The three versions:

- `range(stop)`

Review: range()

The three versions:

- `range(stop)`
- `range(start, stop)`

Review: range()

The three versions:

- `range(stop)`
- `range(start, stop)`
- `range(start, stop, step)`

Slices

- Similar to `range()`, you can take portions or **slices** of lists and strings:

```
1  for d in range(10, 0, -1):
2      print(d)
3  print("Blast off!")
4
5  for num in range(5,8):
6      print(num, 2*num)
7
8  s = "City University of New York"
9  print(s[3], s[0:3], s[:3])
10 print(s[5:8], s[-1])
11
12 names = ["Eleanor", "Anna", "Alice", "Edith"]
13 for n in names:
14     print(n)
```

Slices

```
1  for d in range(10, 0, -1):
2      print(d)
3  print("Blast off!")
4
5  for num in range(5,8):
6      print(num, 2*num)
7
8  s = "City University of New York"
9  print(s[3], s[0:3], s[:3])
10 print(s[5:8], s[-1])
11
12 names = ["Eleanor", "Anna", "Alice", "Edith"]
13 for n in names:
14     print(n)
```

- Similar to `range()`, you can take portions or **slices** of lists and strings:

  `s[5:8]`

  gives: `"Uni"`

Slices

```
1  for d in range(10, 0, -1):
2      print(d)
3  print("Blast off!")
4
5  for num in range(5,8):
6      print(num, 2*num)
7
8  s = "City University of New York"
9  print(s[3], s[0:3], s[:3])
10 print(s[5:8], s[-1])
11
12 names = ["Eleanor", "Anna", "Alice", "Edith"]
13 for n in names:
14     print(n)
```

- Similar to `range()`, you can take portions or **slices** of lists and strings:

  `s[5:8]`

  gives: `"Uni"`
- Also works for lists:

Slices

```
1  for d in range(10, 0, -1):
2      print(d)
3  print("Blast off!")
4
5  for num in range(5,8):
6      print(num, 2*num)
7
8  s = "City University of New York"
9  print(s[3], s[0:3], s[:3])
10 print(s[5:8], s[-1])
11
12 names = ["Eleanor", "Anna", "Alice", "Edith"]
13 for n in names:
14     print(n)
```

- Similar to `range()`, you can take portions or **slices** of lists and strings:

  `s[5:8]`

  gives: `"Uni"`
- Also works for lists:

  `names[1:3]`

Slices

```
1  for d in range(10, 0, -1):
2      print(d)
3  print("Blast off!")
4
5  for num in range(5,8):
6      print(num, 2*num)
7
8  s = "City University of New York"
9  print(s[3], s[0:3], s[:3])
10 print(s[5:8], s[-1])
11
12 names = ["Eleanor", "Anna", "Alice", "Edith"]
13 for n in names:
14     print(n)
```

- Similar to `range()`, you can take portions or **slices** of lists and strings:

  `s[5:8]`

  gives: `"Uni"`
- Also works for lists:

  `names[1:3]`

  gives: `["Anna", "Alice"]`

Slices

```
1  for d in range(10, 0, -1):
2      print(d)
3  print("Blast off!")
4
5  for num in range(5,8):
6      print(num, 2*num)
7
8  s = "City University of New York"
9  print(s[3], s[0:3], s[:3])
10 print(s[5:8], s[-1])
11
12 names = ["Eleanor", "Anna", "Alice", "Edith"]
13 for n in names:
14     print(n)
```

- Similar to `range()`, you can take portions or **slices** of lists and strings:

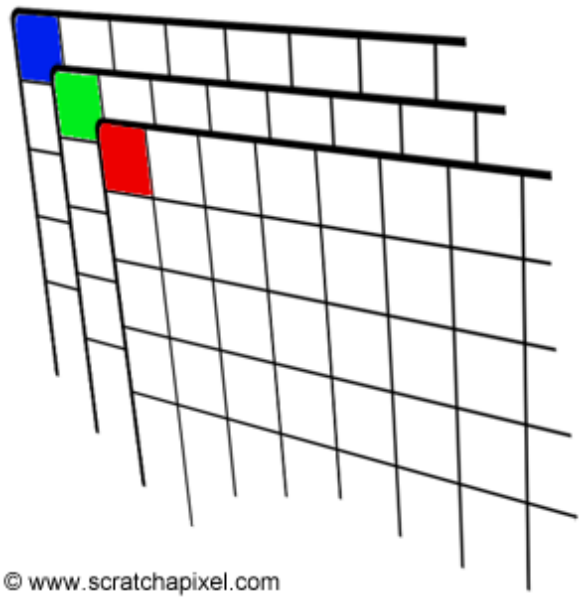
  `s[5:8]`

  gives: `"Uni"`
- Also works for lists:

  `names[1:3]`

  gives: `["Anna", "Alice"]`
- Python also lets you \count backwards": last element has index: `-1`.

Preview: Images



Preview: Images

img[i,j,0]

img[i,j,1]

img[i,j,2]

This image has 287 rows, 573 columns, and 4 color channels (for red, green, blue, and a 4th for how transparent).

Today's Topics

Arithmetic

Indexing and Slicing Lists

Design Challenge: Planes

Colors & Hexadecimal Notation

Design Challenge: Planes

Design Challenge: Planes

A classic write-an-algorithm challenge for introductory programming.

Design Challenge: Planes

A classic write-an-algorithm challenge for introductory programming.

With a slight twist:

Design Challenge: Planes

A classic write-an-algorithm challenge for introductory programming.

With a slight twist: re ning designs

Design Challenge: Planes

A classic write-an-algorithm challenge for introductory programming.

With a slight twist: re ning designs

- As a team, write down your design.

Design Challenge: Planes

A classic write-an-algorithm challenge for introductory programming.

With a slight twist: re ning designs

- As a team, write down your design.
- Exchange with another team.

Design Challenge: Planes

A classic write-an-algorithm challenge for introductory programming.

With a slight twist: re ning designs

- As a team, write down your design.
- Exchange with another team.
- They build an airplane to your design (test plane) without consulting you .

Design Challenge: Planes

A classic write-an-algorithm challenge for introductory programming.

With a slight twist: re ning designs

- As a team, write down your design.
- Exchange with another team.
- They build an airplane to your design (test plane) without consulting you .
- You exchange test planes, and revise your algorithm .

Design Challenge: Planes

A classic write-an-algorithm challenge for introductory programming.

With a slight twist: re ning designs

- As a team, write down your design.
- Exchange with another team.
- They build an airplane to your design (test plane) without consulting you .
- You exchange test planes, and revise your algorithm .
- The build team makes 3 copies of your paper airplane,

Design Challenge: Planes

A classic write-an-algorithm challenge for introductory programming.

With a slight twist: re ning designs

- As a team, write down your design.
- Exchange with another team.
- They build an airplane to your design (test plane) without consulting you .
- You exchange test planes, and revise your algorithm .
- The build team makes 3 copies of your paper airplane, and ies it from the balcony (must be behind rst row of seats).

Design Challenge: Planes

A classic write-an-algorithm challenge for introductory programming.

With a slight twist: re ning designs

- As a team, write down your design.
- Exchange with another team.
- They build an airplane to your design (test plane) without consulting you .
- You exchange test planes, and revise your algorithm .
- The build team makes 3 copies of your paper airplane, and ies it from the balcony (must be behind rst row of seats).
- Will be judged on closeness to the stage.

Design Challenge: Planes

A classic write-an-algorithm challenge for introductory programming.

With a slight twist: re ning designs

- As a team, write down your design.
- Exchange with another team.
- They build an airplane to your design (test plane) without consulting you .
- You exchange test planes, and revise your algorithm .
- The build team makes 3 copies of your paper airplane, and ies it from the balcony (must be behind rst row of seats).
- Will be judged on closeness to the stage.
- Winning design/build team gets chocolate.

Design Challenge: Planes

A classic write-an-algorithm challenge for introductory programming.

With a slight twist: re ning designs

- As a team, write down your design.
- Exchange with another team.
- They build an airplane to your design (test plane) without consulting you .
- You exchange test planes, and revise your algorithm .
- The build team makes 3 copies of your paper airplane, and ies it from the balcony (must be behind rst row of seats).
- Will be judged on closeness to the stage.
- Winning design/build team gets chocolate.

Remember to pick up all your airplanes!

Design Challenge: Initial Design (20 Minutes)

A classic write-an-algorithm challenge for introductory programming.

With a slight twist: re ning designs

- As a team, write down your design.
- Exchange with another team.
- They build an airplane to your design (test plane) without consulting you.
- You exchange test planes, and revise your algorithm.
- The build team makes 3 copies of your paper airplane, and ies it from the balcony (must be behind rst row of seats).
- Will be judged on closeness to the stage.
- Winning design/build team gets chocolate.

Remember to pick up all your airplanes!

Design Challenge: Test Build (2 Minutes)

A classic write-an-algorithm challenge for introductory programming.

With a slight twist: re ning designs

- As a team, write down your design.
- Exchange with another team.
- They build an airplane to your design (test plane) without consulting you.
- You exchange test planes, and revise your algorithm.
- The build team makes 3 copies of your paper airplane, and ies it from the balcony (must be behind rst row of seats).
- Will be judged on closeness to the stage.
- Winning design/build team gets chocolate.

Remember to pick up all your airplanes!

Design Challenge: Revise Design (3 Minutes)

A classic write-an-algorithm challenge for introductory programming.

With a slight twist: re ning designs

- As a team, write down your design.
- Exchange with another team.
- They build an airplane to your design (test plane) without consulting you.
- You exchange test planes, and revise your algorithm .
- The build team makes 3 copies of your paper airplane, and ies it from the balcony (must be behind rst row of seats).
- Will be judged on closeness to the stage.
- Winning design/build team gets chocolate.

Remember to pick up all your airplanes!

Design Challenge: Build Final Planes (25 Minutes)

A classic write-an-algorithm challenge for introductory programming.

With a slight twist: re ning designs

- As a team, write down your design.
- Exchange with another team.
- They build an airplane to your design (test plane) without consulting you.
- You exchange test planes, and revise your algorithm.
- The build team makes 3 copies of your paper airplane, and ies it from the balcony (must be behind rst row of seats).
- Will be judged on closeness to the stage.
- Winning design/build team gets chocolate.

Remember to pick up all your airplanes!

Design Challenge: Test Planes (3 Minutes)

A classic write-an-algorithm challenge for introductory programming.

With a slight twist: re ning designs

- As a team, write down your design.
- Exchange with another team.
- They build an airplane to your design (test plane) without consulting you.
- You exchange test planes, and revise your algorithm.
- The build team makes 3 copies of your paper airplane,and ies it from the balcony (must be behind rst row of seats) .
- Will be judged on closeness to the stage.
- Winning design/build team gets chocolate.

Remember to pick up all your airplanes!

Design Challenge: Retrieve Planes (25 Minutes)

A classic write-an-algorithm challenge for introductory programming.

With a slight twist: re ning designs

- As a team, write down your design.
- Exchange with another team.
- They build an airplane to your design (test plane) without consulting you.
- You exchange test planes, and revise your algorithm.
- The build team makes 3 copies of your paper airplane, and ies it from the balcony (must be behind rst row of seats).
- Will be judged on closeness to the stage.
- Winning design/build team gets chocolate.

Remember to pick up all your airplanes!

Today's Topics

Arithmetic

Indexing and Slicing Lists

Design Challenge: Planes

Colors & Hexadecimal Notation

Colors

Can specify by name.

Colors

Can specify by name.

Can specify by numbers:

Colors

Can specify by name.

Can specify by numbers:

- Amount of Red, Green, and Blue (RGB).

Colors

Can specify by name.

Can specify by numbers:

- Amount of Red, Green, and Blue (RGB).
- Adding light, not paint:

Colors

Can specify by name.

Can specify by numbers:

- Amount of Red, Green, and Blue (RGB).
- Adding light, not paint:
  - Black: 0% red, 0% green, 0% blue

Colors

Can specify by name.

Can specify by numbers:

- Amount of Red, Green, and Blue (RGB).
- Adding light, not paint:
  - Black: 0% red, 0% green, 0% blue
  - White: 100% red, 100% green, 100% blue

Colors

Can specify by numbers (RGB):

Colors

Can specify by numbers (RGB):

- Fractions of each:

Colors

Can specify by numbers (RGB):

- Fractions of each:
  e.g. (1.0, 0, 0) is 100% red, no green, and no blue.

Colors

Can specify by numbers (RGB):

- Fractions of each:
  e.g. (1.0, 0, 0) is 100% red, no green, and no blue.
- 8-bit colors: numbers from 0 to 255:

Colors

Can specify by numbers (RGB):

- Fractions of each:
  e.g. (1.0, 0, 0) is 100% red, no green, and no blue.
- 8-bit colors: numbers from 0 to 255:
  e.g. (0, 255, 0) is no red, 100% green, and no blue.

Colors

Can specify by numbers (RGB):

- Fractions of each:
  e.g. (1.0, 0, 0) is 100% red, no green, and no blue.
- 8-bit colors: numbers from 0 to 255:
  e.g. (0, 255, 0) is no red, 100% green, and no blue.
- Hexcodes (base-16 numbers)...

Decimal & Hexadecimal Numbers

Counting with 10 digits:

(from i-programmer.info)

Decimal

(from i-programmer.info)

Decimal

00 01 02 03 04 05 06 07 08 09

(from i-programmer.info)

Decimal

00 01 02 03 04 05 06 07 08 09
10 11 12 13 14 15 16 17 18 19

(from i-programmer.info)

Decimal

00 01 02 03 04 05 06 07 08 09
10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29

(from i-programmer.info)

Decimal

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39

(from i-programmer.info)

Decimal

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49

(from i-programmer.info)

Decimal

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59

(from i-programmer.info)

Decimal

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69

(from i-programmer.info)

Decimal

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79

(from i-programmer.info)

Decimal

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89

(from i-programmer.info)

Decimal

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99

(from i-programmer.info)

Decimal & Hexadecimal Numbers

Counting with 16 digits:

(from i-programmer.info)

Hexadecimal

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

(from i-programmer.info)

Hexadecimal

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F

(from i-programmer.info)

Hexadecimal

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F

(from i-programmer.info)

Hexadecimal

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F

(from i-programmer.info)

Hexadecimal

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

(from i-programmer.info)

Hexadecimal

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F

(from i-programmer.info)

Hexadecimal

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F

(from i-programmer.info)

Hexadecimal

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F

(from i-programmer.info)

Hexadecimal

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F

(from i-programmer.info)

Hexadecimal

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F

(from i-programmer.info)

Hexadecimal

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF

(from i-programmer.info)

Hexadecimal

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF

(from i-programmer.info)

Hexadecimal

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF

(from i-programmer.info)

Hexadecimal

(from i-programmer.info)

00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF

Hexadecimal

(from i-programmer.info)

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF

Hexadecimal

(from i-programmer.info)

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF
F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF

Colors

Can specify by numbers (RGB):

- Fractions of each:
  e.g. (1.0, 0, 0) is 100% red, no green, and no blue.
- 8-bit colors: numbers from 0 to 255:
  e.g. (0, 255, 0) is no red, 100% green, and no blue.
- Hexcodes (base-16 numbers):

Colors

Can specify by numbers (RGB):

- Fractions of each:
  e.g. (1.0, 0, 0) is 100% red, no green, and no blue.
- 8-bit colors: numbers from 0 to 255:
  e.g. (0, 255, 0) is no red, 100% green, and no blue.
- Hexcodes (base-16 numbers):
  e.g. #0000FFis no red, no green, and 100% blue.

In Pairs or Triples...

Some review and some novel challenges:

Trinkets

(Demo with trinkets)

Today's Topics

Arithmetic

Indexing and Slicing Lists

Design Challenge: Planes

Colors & Hexadecimal Notation

Today's Topics

Arithmetic

Indexing and Slicing Lists

Design Challenge: Planes

Colors & Hexadecimal Notation

Recap

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

Recap

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

In Python, we introduced:

Recap

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

In Python, we introduced:

- Indexing and Slicing Lists

Recap

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

In Python, we introduced:

- Indexing and Slicing Lists
- Colors

Recap

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

In Python, we introduced:

- Indexing and Slicing Lists
- Colors
- Hexadecimal Notation

Recap

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

In Python, we introduced:

- Indexing and Slicing Lists
- Colors
- Hexadecimal Notation

Pass your lecture slips to the end of the rows for the UTA's to collect.

Practice Quiz & Final Questions

Since you must pass the nal exam to pass the course, we end eve
lecture with nal exam review.

Practice Quiz & Final Questions

Since you must pass the nal exam to pass the course, we end eve
lecture with nal exam review.

Pull out something to write on (not to be turned in).

Practice Quiz & Final Questions

Since you must pass the nal exam to pass the course, we end eve
lecture with nal exam review.

Pull out something to write on (not to be turned in).

Lightning rounds:

Practice Quiz & Final Questions

Since you must pass the nal exam to pass the course, we end eve
lecture with nal exam review.

Pull out something to write on (not to be turned in).

Lightning rounds:

- write as much you can for 60 seconds;

Practice Quiz & Final Questions

Since you must pass the nal exam to pass the course, we end eve
lecture with nal exam review.

Pull out something to write on (not to be turned in).

Lightning rounds:

- write as much you can for 60 seconds;
- followed by answer; and

Practice Quiz & Final Questions

- Since you must pass the nal exam to pass the course, we end every lecture with nal exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
  - write as much you can for 60 seconds;
  - followed by answer; and
  - repeat.

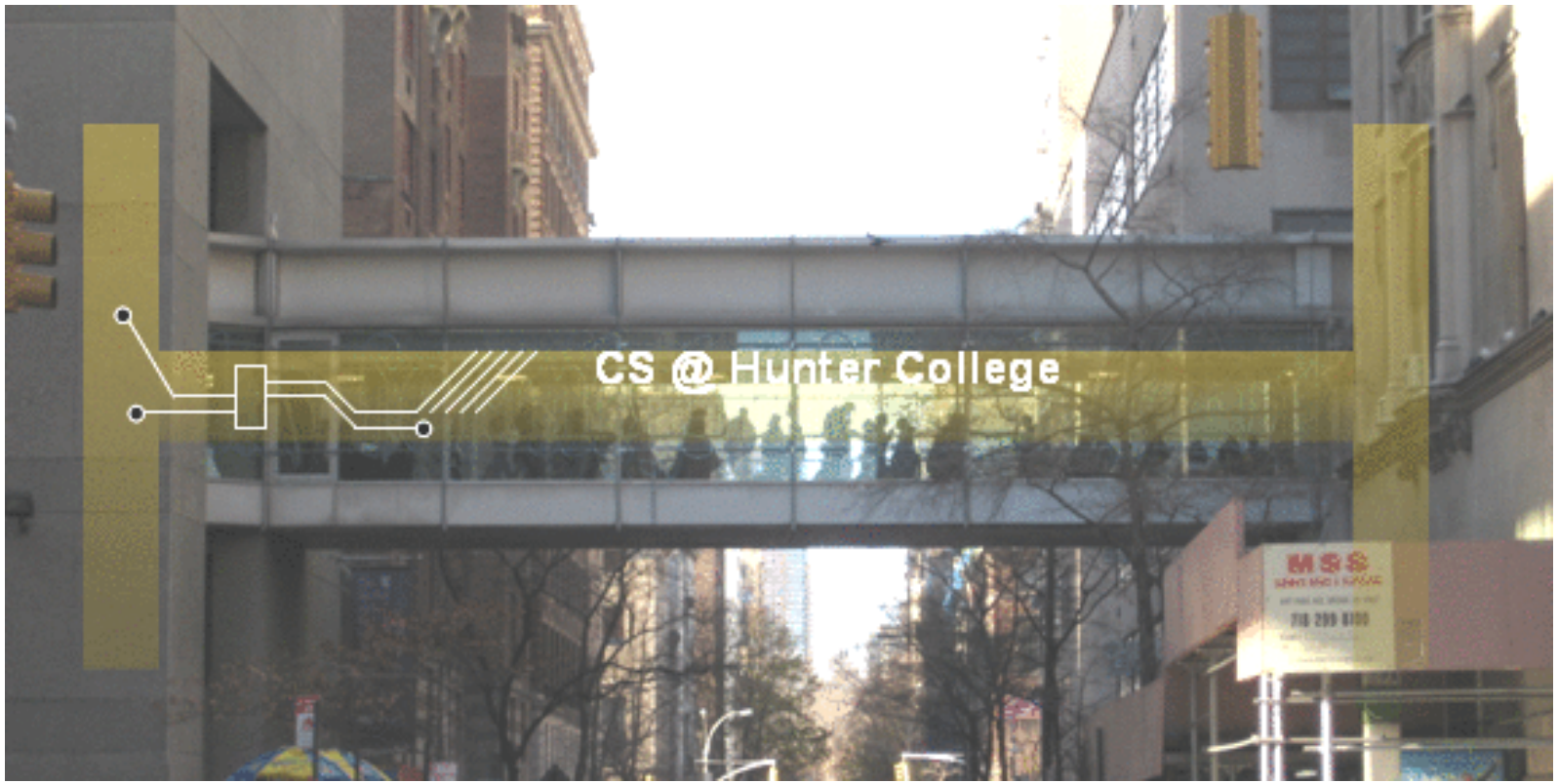
Practice Quiz & Final Questions

- Since you must pass the nal exam to pass the course, we end every lecture with nal exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
  - write as much you can for 60 seconds;
  - followed by answer; and
  - repeat.
- Past exams are on the webpage (under Final Exam Information).

Practice Quiz & Final Questions

- Since you must pass the nal exam to pass the course, we end every lecture with nal exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
  - write as much you can for 60 seconds;
  - followed by answer; and
  - repeat.
- Past exams are on the webpage (under Final Exam Information).
- We're starting with Fall 2017, Version 2.

Writing Boards



- Return writing boards as you leave...