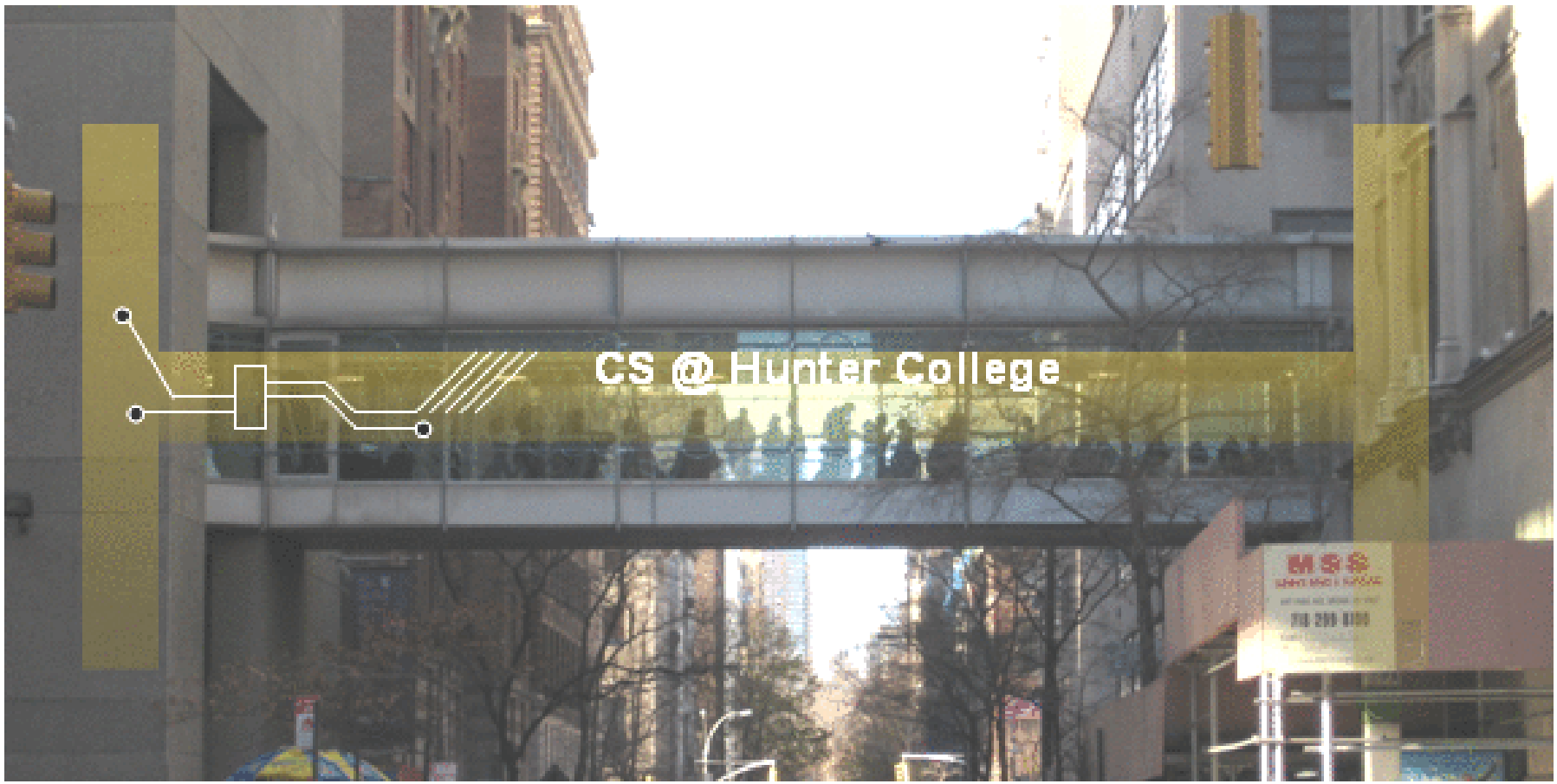
CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Frequently Asked Questions

From email

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**
  You don't submit the lab, you **read the lab.**

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**
  You don't submit the lab, you **read the lab.**
  When you are done, got to Gradescope where you take the Lab Quiz, then submit this class's 5 programming assignments.

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**
  You don't submit the lab, you **read the lab.**
  When you are done, got to Gradescope where you take the Lab Quiz, then submit this class's 5 programming assignments.
- **I accidentally submitted the Lab Quiz before completing it. Can I retake it?**

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**
  You don't submit the lab, you **read the lab.**
  When you are done, got to Gradescope where you take the Lab Quiz, then submit this class's 5 programming assignments.
- **I accidentally submitted the Lab Quiz before completing it. Can I retake it?**
  Lab Quiz (Gradescope) can be submitted only once.

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**
  You don't submit the lab, you **read the lab.**
  When you are done, got to Gradescope where you take the Lab Quiz, then submit this class's 5 programming assignments.
- **I accidentally submitted the Lab Quiz before completing it. Can I retake it?**
  Lab Quiz (Gradescope) can be submitted only once.
  Unfortunately we cannot reopen quizzes, but don't worry! Your grade on the final exam will replace any missing or lower quiz grades.

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**
  You don't submit the lab, you **read the lab.**
  When you are done, got to Gradescope where you take the Lab Quiz, then submit this class's 5 programming assignments.
- **I accidentally submitted the Lab Quiz before completing it. Can I retake it?**
  Lab Quiz (Gradescope) can be submitted only once.
  Unfortunately we cannot reopen quizzes, but don't worry! Your grade on the final exam will replace any missing or lower quiz grades.
- **Can I work ahead?**

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**
  You don't submit the lab, you **read the lab.**
  When you are done, got to Gradescope where you take the Lab Quiz, then submit this class's 5 programming assignments.
- **I accidentally submitted the Lab Quiz before completing it. Can I retake it?**
  Lab Quiz (Gradescope) can be submitted only once.
  Unfortunately we cannot reopen quizzes, but don't worry! Your grade on the final exam will replace any missing or lower quiz grades.
- **Can I work ahead?**
  Absolutely! Submission is open on Gradescope, 3 classes before the deadline.

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**
  You don't submit the lab, you **read the lab.**
  When you are done, got to Gradescope where you take the Lab Quiz, then submit this class's 5 programming assignments.
- **I accidentally submitted the Lab Quiz before completing it. Can I retake it?**
  Lab Quiz (Gradescope) can be submitted only once.
  Unfortunately we cannot reopen quizzes, but don't worry! Your grade on the final exam will replace any missing or lower quiz grades.
- **Can I work ahead?**
  Absolutely! Submission is open on Gradescope, 3 classes before the deadline.
- **When is the midterm?**

Frequently Asked Questions

From email

- **I am not sure how to submit the Lab.**
  You don't submit the lab, you **read the lab.**
  When you are done, got to Gradescope where you take the Lab Quiz, then submit this class's 5 programming assignments.
- **I accidentally submitted the Lab Quiz before completing it. Can I retake it?**
  Lab Quiz (Gradescope) can be submitted only once.
  Unfortunately we cannot reopen quizzes, but don't worry! Your grade on the final exam will replace any missing or lower quiz grades.
- **Can I work ahead?**
  Absolutely! Submission is open on Gradescope, 3 classes before the deadline.
- **When is the midterm?**
  There is no midterm. Instead there's required quizzes and programming assignments.

Today's Topics

- **For-loops**
- `range()`
- Variables
- Characters
- Strings

In Pairs or Triples...

Some review and some novel challenges:

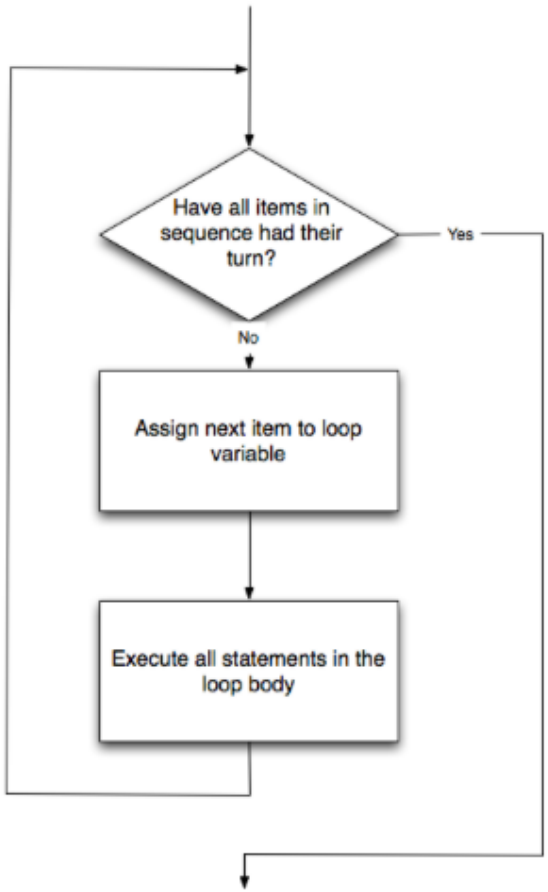
```
1  #Predict what will be printed:
2  for i in range(4):
3      print('The world turned upside down')
4  for j in [0,1,2,3,4,5]:
5      print(j)
6  for count in range(6):
7      print(count)
8  for color in ['red', 'green', 'blue']:
9      print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

Python Tutor

```
1  #Predict what will be printed:
2  for i in range(4):
3      print('The world turned upside down')
4  for j in [0,1,2,3,4,5]:
5      print(j)
6  for count in range(6):
7      print(count)
8  for color in ['red', 'green', 'blue']:
9      print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

(Demo with pythonTutor)

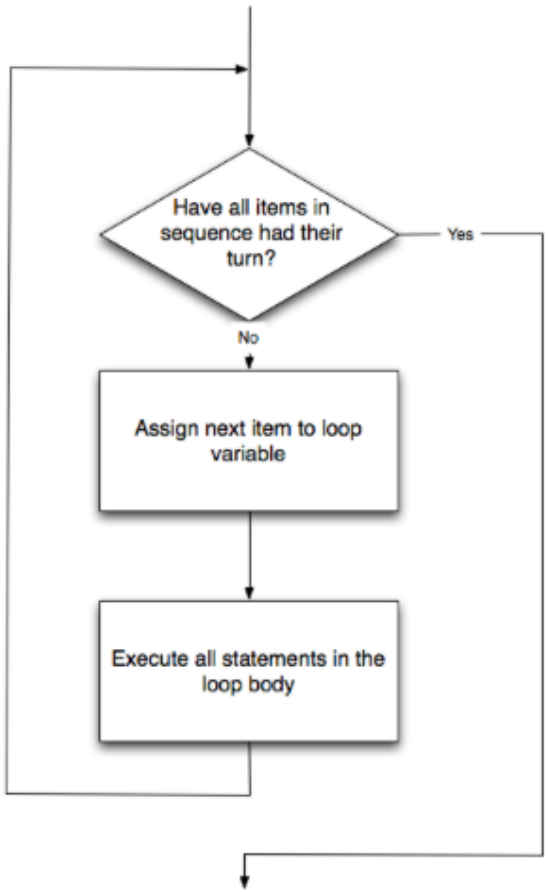
for-loop



How to Think Like CS, x4.5

```
for  i in list:
     statement1
     statement2
     statement3
```

for-loop



How to Think Like CS, x4.5

```
for  i in list:
     statement1
     statement2
     statement3
```

where `list` is a list of items:

- stated explicitly (e.g. [1,2,3]) or
- generated by a function, e.g. `range()`.

Today's Topics

- For-loops
- **range()**
- Variables
- Characters
- Strings

More on range():

```
1  #Predict what will be printed:
2
3  for num in [2,4,6,8,10]:
4      print(num)
5
6  sum = 0
7  for x in range(0,12,2):
8      print(x)
9      sum = sum + x
10
11 print(sum)
12
13 for c in "ABCD":
14     print(c)
```

Python Tutor

```
1  #Predict what will be printed:
2
3  for num in [2,4,6,8,10]:
4      print(num)
5
6  sum = 0
7  for x in range(0,12,2):
8      print(x)
9      sum = sum + x
10
11 print(sum)
12
13 for c in "ABCD":
14     print(c)
```

(Demo with pythonTutor)

range()

Simplest version:

- `range(stop)`

`range()`

Simplest version:

- `range(stop)`
- Produces a list: [0,1,2,3,...,stop-1]

range()

Simplest version:

- `range(stop)`
- Produces a list: [0,1,2,3,...,stop-1]
- For example, if you want the the list [0,1,2,3,...,100], you would write:

range()

Simplest version:

- `range(stop)`
- Produces a list: [0,1,2,3,...,stop-1]
- For example, if you want the the list [0,1,2,3,...,100], you would write:

```
range(101)
```

range()

What if you wanted to start somewhere else:

range()

What if you wanted to start somewhere else:

- `range(start, stop)`

range()

What if you wanted to start somewhere else:

- `range(start, stop)`
- Produces a list:
  [start,start+1,...,stop-1]

range()

What if you wanted to start somewhere else:

- `range(start, stop)`
- Produces a list:
  [start,start+1,...,stop-1]
- For example, if you want the the list
  [10,11,...,20]
  you would write:

range()

What if you wanted to start somewhere else:

- `range(start, stop)`
- Produces a list:
  [start,start+1,...,stop-1]
- For example, if you want the the list
  [10,11,...,20]
  you would write:

```
range(10,21)
```

range()

What if you wanted to count by twos, or some other number:

range()

What if you wanted to count by twos, or some other number:

- `range(start, stop, step)`

range()

What if you wanted to count by twos, or some other number:

- `range(start, stop, step)`
- Produces a list: [start,start+step,start+2*step...,last] (where last is the largest start+k*step less than stop)

range()

What if you wanted to count by twos, or some other number:

- `range(start, stop, step)`
- Produces a list:
  [start,start+step,start+2*step...,last]
  (where last is the largest start+k*step less than stop)
- For example, if you want the the list [5,10,...,50]
  you would write:

range()

What if you wanted to count by twos, or some other number:

- `range(start, stop, step)`
- Produces a list:
  [start,start+step,start+2*step...,last]
  (where last is the largest start+k*step less than stop)
- For example, if you want the the list [5,10,...,50]
  you would write:

  ```
  range(5,51,5)
  ```

In summary: range()

The three versions:

In summary: range()

The three versions:

- `range(stop)`

In summary: range()

The three versions:

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

In summary: range()

The three versions:

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

Today's Topics

- For-loops
- `range()`
- **Variables**
- Characters
- Strings

Variables

- A **variable** is a reserved memory location for storing a value.

Variables

- A **variable** is a reserved memory location for storing a value.
- Di erent kinds, or **types**, of values need di erent amounts of space:
  - **int**: integer or whole numbers

Variables

- A **variable** is a reserved memory location for storing a value.
- Di erent kinds, or **types**, of values need di erent amounts of space:
  - **int**: integer or whole numbers
  - **oat**: oating point or real numbers

Variables

- A **variable** is a reserved memory location for storing a value.
- Di erent kinds, or **types**, of values need di erent amounts of space:
  - **int**: integer or whole numbers
  - **oat**: oating point or real numbers
  - **string**: sequence of characters

Variables

- A **variable** is a reserved memory location for storing a value.
- Di erent kinds, or **types**, of values need di erent amounts of space:
  - **int**: integer or whole numbers
  - **oat**: oating point or real numbers
  - **string**: sequence of characters
  - **list**: a sequence of items

Variables

- A **variable** is a reserved memory location for storing a value.
- Di erent kinds, or **types**, of values need di erent amounts of space:
  - **int**: integer or whole numbers
  - **oat**: oating point or real numbers
  - **string**: sequence of characters
  - **list**: a sequence of items
    e.g. `[3, 1, 4, 5, 9]` or `['violet','purple','indigo']`

Variables

- A **variable** is a reserved memory location for storing a value.
- Di erent kinds, or **types**, of values need di erent amounts of space:
  - **int**: integer or whole numbers
  - **oat**: oating point or real numbers
  - **string**: sequence of characters
  - **list**: a sequence of items
    e.g. `[3, 1, 4, 5, 9]` or `['violet','purple','indigo']`
  - **class variables**: for complex objects, like turtles.
- In Python (unlike other languages) you don't need to specify the type; it is deduced by its value.

Variable Names

- There's some rules about valid names for variables.

Variable Names

- There's some rules about valid names for variables.
- Can use the underscore ('_'), upper and lower case letters.

Variable Names

- There's some rules about valid names for variables.
- Can use the underscore ('_'), upper and lower case letters.
- Can also use numbers, just can't start a name with a number.

Variable Names

- There's some rules about valid names for variables.
- Can use the underscore ('_'), upper and lower case letters.
- Can also use numbers, just can't start a name with a number.
- Can't use symbols (like '+' or '*') since used for arithmetic.

Variable Names

- There's some rules about valid names for variables.
- Can use the underscore ('_'), upper and lower case letters.
- Can also use numbers, just can't start a name with a number.
- Can't use symbols (like '+' or '*') since used for arithmetic.
- Can't use some words that Python has reserved for itself (e.g. `for`).
  (List of reserved words in *Think CS*, *x*2.5.)

Today's Topics

- For-loops
- `range()`
- Variables
- **Characters**
- Strings

Standardized Code for Characters

American Standard Code for Information Interchange (ASCII), 1960.

Standardized Code for Characters

American Standard Code for Information Interchange (ASCII), 1960.

(New version called: Unicode).

Standardized Code for Characters

American Standard Code for Information Interchange (ASCII), 1960.

(New version called: Unicode).

(wiki)

Converting from Character to Code:

(There is a link to the ASCII table on the course webpage, under 'Useful Links'.)

Converting from Character to Code:

(There is a link to the ASCII table on the course webpage, under 'Useful Links'.)

ord(c) : returns Unicode (ASCII) of the character.

Converting from Character to Code:

(There is a link to the ASCII table on the course webpage, under 'Useful Links'.)

ord(c) : returns Unicode (ASCII) of the character.

Example: ord('a') returns 97.

Converting from Character to Code:

(There is a link to the ASCII table on the course webpage, under 'Useful Links'.)

ord(c) : returns Unicode (ASCII) of the character.

Example: ord('a') returns 97.

chr(x) : returns the character whose Unicode is x.

Converting from Character to Code:

(There is a link to the ASCII table on the course webpage, under 'Useful Links'.)

ord(c) : returns Unicode (ASCII) of the character.

Example: ord('a') returns 97.

chr(x) : returns the character whose Unicode is x.

Example: chr(97) returns 'a'.

Converting from Character to Code:

(There is a link to the ASCII table on the course webpage, under 'Useful Links'.)

ord(c) : returns Unicode (ASCII) of the character.

Example: ord('a') returns 97.

chr(x) : returns the character whose Unicode is x.

Example: chr(97) returns 'a'.

What is chr(33) ?

In Pairs or Triples...

Some review and some novel challenges:

Python Tutor

(Demo with pythonTutor)

Wrap

User Input

Covered in detail in Lab 2:

(Demo with pythonTutor)

Side Note: '+' for numbers and strings

`x = 3 + 5` stores the number 8 in memory location `x`.

Side Note: '+' for numbers and strings

`x = 3 + 5` stores the number 8 in memory location `x`.

`x = x + 1` increases `x` by 1.

Side Note: '+' for numbers and strings

`x = 3 + 5` stores the number 8 in memory location `x`.

`x = x + 1` increases `x` by 1.

`s = "hi" + "Mom"` stores `"hiMom"` in memory location `s`.

Side Note: '+' for numbers and strings

`x = 3 + 5` stores the number 8 in memory location `x`.

`x = x + 1` increases `x` by 1.

`s = "hi" + "Mom"` stores `"hiMom"` in memory location `s`.

`s = s + "A"` adds the letter `"A"` to the end of the string `s`.

Today's Topics

For-loops

range()

Variables

Characters

Strings

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
```

The first line creates a variable, called s, that stores the string: "FridaysSaturdaysSundays"

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
```

The first line creates a variable, called s, that stores the string: "FridaysSaturdaysSundays"

There are many useful functions for strings (more in Lab 2).

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
```

The first line creates a variable, called s, that stores the string: "FridaysSaturdaysSundays"

There are many useful functions for strings (more in Lab 2).

s.count(x) will count the number of times the pattern x, appears in s.

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
```

The first line creates a variable, called s, that stores the string: "FridaysSaturdaysSundays"

There are many useful functions for strings (more in Lab 2).

s.count(x) will count the number of times the pattern x, appears in s.

- s.count("s") counts the number of lower case s that occurs.

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
```

The first line creates a variable, called s, that stores the string: "FridaysSaturdaysSundays"

There are many useful functions for strings (more in Lab 2).

s.count(x) will count the number of times the pattern x, appears in s.

- s.count("s") counts the number of lower case s that occurs.
- num = s.count("s") stores the result in the variable num, for later.

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
```

The first line creates a variable, called s, that stores the string: "FridaysSaturdaysSundays"

There are many useful functions for strings (more in Lab 2).

s.count(x) will count the number of times the pattern x, appears in s.

- s.count("s") counts the number of lower case s that occurs.
- num = s.count("s") stores the result in the variable num, for later.
- What would print(s.count("sS")) output?

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
```

The first line creates a variable, called s, that stores the string: "FridaysSaturdaysSundays"

There are many useful functions for strings (more in Lab 2).

s.count(x) will count the number of times the pattern x, appears in s.

- s.count("s") counts the number of lower case s that occurs.
- num = s.count("s") stores the result in the variable num, for later.
- What would print(s.count("sS")) output?
- What about:

```
mess = "10 20 21 9 101 35"
mults = mess.count("0 ")
print(mults)
```

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[0] is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[0] is 'F' .

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[1] is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[1] is 'r' .

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[-1] is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[-1] is 's' .

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[3:6] is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[3:6] is 'day' .

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[:3] is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[:3] is 'Fri' .

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[:-1] is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[:-1] is 'FridaysSaturdaysSunday'
(no trailing 's' at the end)

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

split() divides a string into a list.

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

split() divides a string into a list.

Cross out the delimiter, and the remaining items are the list.

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

split() divides a string into a list.

Cross out the delimiter, and the remaining items are the list.

"Friday~~s~~Saturday~~s~~Sunday"

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

split() divides a string into a list.

Cross out the delimiter, and the remaining items are the list.

"Friday~~s~~Saturday~~s~~Sunday"
days = ['Friday', 'Saturday', 'Sunday']

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

split() divides a string into a list.

Cross out the delimiter, and the remaining items are the list.

"Friday~~s~~Saturday~~s~~Sunday"
days = ['Friday', 'Saturday', 'Sunday']

Di erent delimiters give di erent lists:

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

split() divides a string into a list.

Cross out the delimiter, and the remaining items are the list.

"Friday~~s~~Saturday~~s~~Sunday"
days = ['Friday', 'Saturday', 'Sunday']

Di erent delimiters give di erent lists:

```
days = s[:-1].split("day")
```

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

split() divides a string into a list.

Cross out the delimiter, and the remaining items are the list.

"FridaysSaturdaysSunday"
days = ['Friday', 'Saturday', 'Sunday']

Di erent delimiters give di erent lists:

```
days = s[:-1].split("day")
```

"FridaysSaturdaysSundays"

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

split() divides a string into a list.

Cross out the delimiter, and the remaining items are the list.

"Friday~~s~~Saturday~~s~~Sunday"
days = ['Friday', 'Saturday', 'Sunday']

Di erent delimiters give di erent lists:

```
days = s[:-1].split("day")
```

"Fri~~day~~sSatur~~day~~sSun~~day~~"
days = ['Fri', 'sSatur', 'sSun']

Today's Topics

For-loops

range()

Variables

Characters

Strings

Recap

Recap

In Python, we introduced:

Recap

In Python, we introduced:

- For-loops

Recap

In Python, we introduced:

- For-loops
- range()

Recap

In Python, we introduced:

- For-loops
- range()
- Variables: ints and strings

Recap

In Python, we introduced:

- For-loops
- range()
- Variables: ints and strings
- Some arithmetic

Recap

In Python, we introduced:

- For-loops
- range()
- Variables: ints and strings
- Some arithmetic
- String concatenation

Recap

In Python, we introduced:

- For-loops
- range()
- Variables: ints and strings
- Some arithmetic
- String concatenation
- Functions: ord() and chr()

Recap

In Python, we introduced:

- For-loops
- range()
- Variables: ints and strings
- Some arithmetic
- String concatenation
- Functions: ord() and chr()
- String Manipulation

Recap

In Python, we introduced:

- For-loops
- range()
- Variables: ints and strings
- Some arithmetic
- String concatenation
- Functions: ord() and chr()
- String Manipulation

5 Minute Break!

Today we have a second lecture portion! Take a quick break.

Frequently Asked Questions

From emails.

> Gradescope does not give me credit but my program runs on my computer.

Frequently Asked Questions

From emails.

> Gradescope does not give me credit but my program runs on my computer.
> You must submit a le that contains python instructions and comments ONLY.

Frequently Asked Questions

From emails.

> Gradescope does not give me credit but my program runs on my computer.
> You must submit a le that contains python instructions and comments ONLY.
> Don't submit screenshots, those are images and the grading script cannot run and test your program that way.

Frequently Asked Questions

From emails.

Gradescope does not give me credit but my program runs on my computer.
You must submit a le that contains python instructions and comments ONLY.
Don't submit screenshots, those are images and the grading script cannot run and test your program that way.
Don't copy/paste from an interactive command prompt (repl.it or the python shell in IDLE) otherwise you will be copying extra characters and text that is not executable.

Frequently Asked Questions

From emails.

Gradescope does not give me credit but my program runs on my computer.
You must submit a le that contains python instructions and comments ONLY.
Don't submit screenshots, those are images and the grading script cannot run and test your program that way.
Don't copy/paste from an interactive command prompt (repl.it or the python shell in IDLE) otherwise you will be copying extra characters and text that is not executable.

I missed the deadline for a programming assignment. What should I do?

Frequently Asked Questions

From emails.

Gradescope does not give me credit but my program runs on my computer.
You must submit a le that contains python instructions and comments ONLY.
Don't submit screenshots, those are images and the grading script cannot run and test your program that way.
Don't copy/paste from an interactive command prompt (repl.it or the python shell in IDLE) otherwise you will be copying extra characters and text that is not executable.

I missed the deadline for a programming assignment. What should I do?
We do not accept late work but we drop the lowest 5 program grades.

Frequently Asked Questions

From emails.

Gradescope does not give me credit but my program runs on my computer.
You must submit a le that contains python instructions and comments ONLY.
Don't submit screenshots, those are images and the grading script cannot run and test your program that way.
Don't copy/paste from an interactive command prompt (repl.it or the python shell in IDLE) otherwise you will be copying extra characters and text that is not executable.

I missed the deadline for a programming assignment. What should I do?
We do not accept late work but we drop the lowest 5 program grades.
Due dates are one week late to allow exibility for emergencies.

Frequently Asked Questions

From emails.

Gradescope does not give me credit but my program runs on my computer.
You must submit a le that contains python instructions and comments ONLY.
Don't submit screenshots, those are images and the grading script cannot run and test your program that way.
Don't copy/paste from an interactive command prompt (repl.it or the python shell in IDLE) otherwise you will be copying extra characters and text that is not executable.

I missed the deadline for a programming assignment. What should I do?
We do not accept late work but we drop the lowest 5 program grades.
Due dates are one week late to allow exibility for emergencies.
You must work on THIS WEEK'S PROGRAMS , that way you will never miss a deadline!!!

Frequently Asked Questions

From emails.

Gradescope does not give me credit but my program runs on my computer.
You must submit a le that contains python instructions and comments ONLY.
Don't submit screenshots, those are images and the grading script cannot run and test your program that way.
Don't copy/paste from an interactive command prompt (repl.it or the python shell in IDLE) otherwise you will be copying extra characters and text that is not executable.

I missed the deadline for a programming assignment. What should I do?
We do not accept late work but we drop the lowest 5 program grades.
Due dates are one week late to allow exibility for emergencies.
You must work on THIS WEEK'S PROGRAMS , that way you will never miss a deadline!!!

There is a typo in the slides, should I report it?

Frequently Asked Questions

From emails.

Gradescope does not give me credit but my program runs on my computer.
You must submit a le that contains python instructions and comments ONLY.
Don't submit screenshots, those are images and the grading script cannot run and test your program that way.
Don't copy/paste from an interactive command prompt (repl.it or the python shell in IDLE) otherwise you will be copying extra characters and text that is not executable.

I missed the deadline for a programming assignment. What should I do?
We do not accept late work but we drop the lowest 5 program grades.
Due dates are one week late to allow exibility for emergencies.
You must work on THIS WEEK'S PROGRAMS , that way you will never miss a deadline!!!

There is a typo in the slides, should I report it?
Yes, great catch! We really appreciate it when you tell us about any typos or errors, please send us email.

Today's Topics

More on Strings

Arithmetic

Indexing and Slicing Lists

Colors & Hexadecimal Notation

Today's Topics

More on Strings

Arithmetic

Indexing and Slicing Lists

Colors & Hexadecimal Notation

More on Strings...

From Final Exam, Fall 2017, Version 1, #1:

More on Strings...

Some we have seen before, some we haven't.

More on Strings...

Some we have seen before, some we haven't.

Don't leave it blank{ write what you know & puzzle out as much as possible

More on Strings...

Some we have seen before, some we haven't.

Don't leave it blank{ write what you know & puzzle out as much as possible

First, go through and write down what we know:

More on Strings...

Some we have seen before, some we haven't.

Don't leave it blank{ write what you know & puzzle out as much as possible

First, go through and write down what we know:

- There are 3print() .

More on Strings...

Some we have seen before, some we haven't.

Don't leave it blank{ write what you know & puzzle out as much as possible

First, go through and write down what we know:

- There are 3print() .
- Output will have at least:

More on Strings...

Some we have seen before, some we haven't.

Don't leave it blank{ write what you know & puzzle out as much as possible

First, go through and write down what we know:

- There are 3print() .
- Output will have at least:
  There are ??? fun days in a week

More on Strings...

Some we have seen before, some we haven't.

Don't leave it blank{ write what you know & puzzle out as much as possible

First, go through and write down what we know:

- There are 3print() .
- Output will have at least:
  There are ??? fun days in a week
  Two of them are ???

More on Strings...

Some we have seen before, some we haven't.

Don't leave it blank{ write what you know & puzzle out as much as possible

First, go through and write down what we know:

- There are 3print() .
- Output will have at least:
  There are ??? fun days in a week
  Two of them are ???
  My favorite ??? is Saturday.

More on Strings...

Some we have seen before, some we haven't.

Don't leave it blank{ write what you know & puzzle out as much as possible

First, go through and write down what we know:

- There are 3print() .
- Output will have at least:
  There are ??? fun days in a week
  Two of them are ???
  My favorite ??? is Saturday.

Will get 1=3 to 1=2 points for writing down the basic structure.

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
```

The first line creates a variable, called s, that stores the string: "FridaysSaturdaysSundays"

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
```

The first line creates a variable, called s, that stores the string: "FridaysSaturdaysSundays"

There are many useful functions for strings (more in Lab 2).

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
```

The first line creates a variable, called s, that stores the string: "FridaysSaturdaysSundays"

There are many useful functions for strings (more in Lab 2).

s.count(x) will count the number of times the pattern x, appears in s.

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
```

The first line creates a variable, called s, that stores the string: "FridaysSaturdaysSundays"

There are many useful functions for strings (more in Lab 2).

s.count(x) will count the number of times the pattern x, appears in s.

- s.count("s") counts the number of lower case s that occurs.

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
```

The first line creates a variable, called s, that stores the string: "FridaysSaturdaysSundays"

There are many useful functions for strings (more in Lab 2).

s.count(x) will count the number of times the pattern x, appears in s.

- s.count("s") counts the number of lower case s that occurs.
- num = s.count("s") stores the result in the variable num, for later.

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
```

The first line creates a variable, called s, that stores the string: "FridaysSaturdaysSundays"

There are many useful functions for strings (more in Lab 2).

s.count(x) will count the number of times the pattern x, appears in s.

- s.count("s") counts the number of lower case s that occurs.
- num = s.count("s") stores the result in the variable num, for later.
- What would print(s.count("sS")) output?

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
```

The first line creates a variable, called s, that stores the string: "FridaysSaturdaysSundays"

There are many useful functions for strings (more in Lab 2).

s.count(x) will count the number of times the pattern x, appears in s.

- s.count("s") counts the number of lower case s that occurs.
- num = s.count("s") stores the result in the variable num, for later.
- What would print(s.count("sS")) output?
- What about:

```
mess = "10 20 21 9 101 35"
mults = mess.count("0 ")
print(mults)
```

More on Strings...

Don't leave it blank{ write what you know & puzzle out as much as possible

More on Strings...

Don't leave it blank{ write what you know & puzzle out as much as possible

There are 3 fun days in a week
Two of them are ???
My favorite ??? is Saturday.

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[0] is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[0] is 'F' .

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[1] is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[1] is 'r' .

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[-1] is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[-1] is 's' .

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[3:6] is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[3:6] is 'day' .

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[:3] is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[:3] is 'Fri' .

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[:-1] is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

Strings are made up of individual characters (letters, numbers, etc.)

Useful to be able to refer to pieces of a string, either an individual location or a \substring" of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

s[:-1] is 'FridaysSaturdaysSunday'
(no trailing 's' at the end)

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

split() divides a string into a list.

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

split() divides a string into a list.

Cross out the delimiter, and the remaining items are the list.

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

split() divides a string into a list.

Cross out the delimiter, and the remaining items are the list.

"Friday~~s~~Saturday~~s~~Sunday"

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

split() divides a string into a list.

Cross out the delimiter, and the remaining items are the list.

"Friday~~s~~Saturday~~s~~Sunday"
days = ['Friday', 'Saturday', 'Sunday']

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

split() divides a string into a list.

Cross out the delimiter, and the remaining items are the list.

"Friday~~s~~Saturday~~s~~Sunday"
days = ['Friday', 'Saturday', 'Sunday']

Di erent delimiters give di erent lists:

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

split() divides a string into a list.

Cross out the delimiter, and the remaining items are the list.

"Friday~~s~~Saturday~~s~~Sunday"
days = ['Friday', 'Saturday', 'Sunday']

Di erent delimiters give di erent lists:

```
days = s[:-1].split("day")
```

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

split() divides a string into a list.

Cross out the delimiter, and the remaining items are the list.

"FridaysSaturdaysSunday"
days = ['Friday', 'Saturday', 'Sunday']

Di erent delimiters give di erent lists:

```
days = s[:-1].split("day")
```

"FridaysSaturdaysSundays"

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"
days = s[:-1].split("s")
```

split() divides a string into a list.

Cross out the delimiter, and the remaining items are the list.

"Friday~~s~~Saturday~~s~~Sunday"
days = ['Friday', 'Saturday', 'Sunday']

Di erent delimiters give di erent lists:

```
days = s[:-1].split("day")
```

"Fri~~day~~sSatur~~day~~sSun~~day~~"
days = ['Fri', 'sSatur', 'sSun']

More on Strings...

Don't leave it blank{ write what you know & puzzle out as much as possible

More on Strings...

Don't leave it blank{ write what you know & puzzle out as much as possible

There are 3 fun days in a week
Two of them are Friday Sunday
My favorite ??? is Saturday.

Today's Topics

More on Strings

Arithmetic

Indexing and Slicing Lists

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`

15 // 7 = 2

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`

15 // 7 = 2

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`

15 // 7 = 2

Remainder or Modulus:
`days = totalDays % 7`

15 % 7 = 1

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`

15 // 7 = 2

Remainder or Modulus:
`days = totalDays % 7`

15 % 7 = 1

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`

15 // 7 = 2

Remainder or Modulus:
`days = totalDays % 7`

15 % 7 = 1

Exponentiaion:
`pop = 2**time`

Challenge:

What does this code do?

Challenge:

What does this code do?

In particular, what is printed...

If the user enters, 9 and 2.

Challenge:

What does this code do?

In particular, what is printed...

If the user enters, 9 and 2.

If the user enters, 12 and 4.

Challenge:

What does this code do?

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.

Challenge:

What does this code do?

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.

Challenge:

What does this code do?

In particular, what is printed...

If the user enters, 9 and 2.

Challenge:

What does this code do?

In particular, what is printed...

If the user enters, 9 and 2.

Challenge:

What does this code do?

In particular, what is printed...

If the user enters, 12 and 4.

Challenge:

What does this code do?

In particular, what is printed...

If the user enters, 12 and 4.

Challenge:

What does this code do?

In particular, what is printed...

If the user enters, 8 and 20.

Challenge:

What does this code do?

In particular, what is printed...

If the user enters, 8 and 20.

Challenge:

What does this code do?

In particular, what is printed...

If the user enters, 11 and 1.

Challenge:

What does this code do?

In particular, what is printed...

If the user enters, 11 and 1.

Today's Topics

More on Strings

Arithmetic

Indexing and Slicing Lists

Colors & Hexadecimal Notation

Challenge:

Mostly review:

Python Tutor

(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:

Slices

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

```
s[5:8]
```

gives: "Uni"

Slices

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

```
s[5:8]
```

gives: "Uni"

Also works for lists:

Slices

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

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

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 .

Today's Topics

More on Strings

Arithmetic

Indexing and Slicing Lists

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

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)

$10^1 + 10^0$

Max Number = 99

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)

$10^1 + 10^0$

Max Number = 99

$90 = (9 \quad 10^1) + (0 \quad 10^0)$

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)

$10^1 + 10^0$

Max Number = 99

$90 = (9 \quad 10^1) + (0 \quad 10^0)$

$99 = (9 \quad 10^1) + (9 \quad 10^0)$

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

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

(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
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

(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
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

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

$16^1 + 16^0$

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
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

(from i-programmer.info)

$16^1 + 16^0$

Max Number = 255

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
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

(from i-programmer.info)

$16^1 + 16^0$

Max Number = 255

$F0 = (F \quad 16^1) + (0 \quad 16^0)$

$F0 = (240) + (0) = 240$

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
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

(from i-programmer.info)

$16^1 + 16^0$

Max Number = 255

$F0 = (F \quad 16^1) + (0 \quad 16^0)$

$F0 = (240) + (0) = 240$

$FF = (F \quad 16^1) + (F \quad 16^0)$

$FF = (240) + (15) = 255$

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.
- 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.

Challenge:

Some review and some novel challenges:

Trinkets

(Demo with trinkets)

Recap

In Python, we introduced:

Recap

In Python, we introduced:

- Indexing and Slicing Lists

Recap

In Python, we introduced:

- Indexing and Slicing Lists
- Arithmetic

Recap

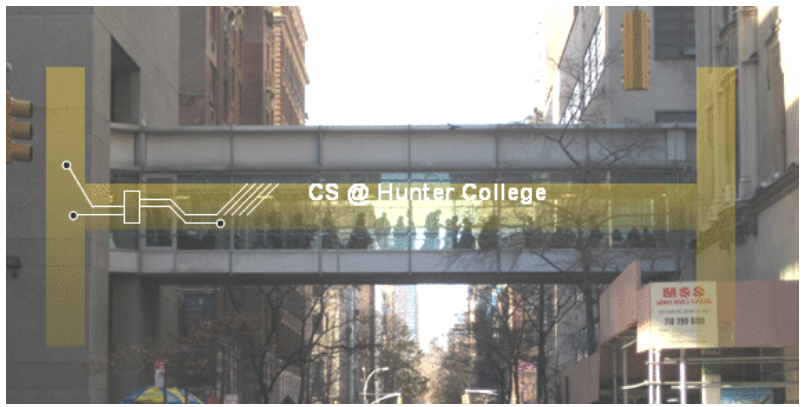
In Python, we introduced:

- Indexing and Slicing Lists
- Arithmetic
- Colors

Recap

- In Python, we introduced:
  - Indexing and Slicing Lists
  - Arithmetic
  - Colors
  - Hexadecimal Notation

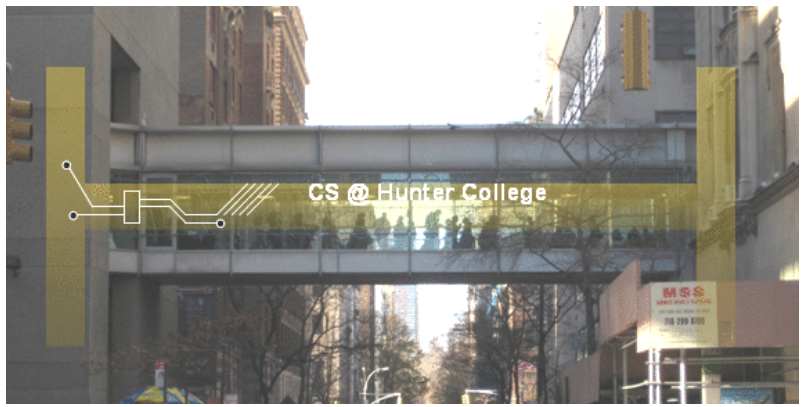
Class Reminders!



Before next class, don't forget to:

- Review this week's Lab

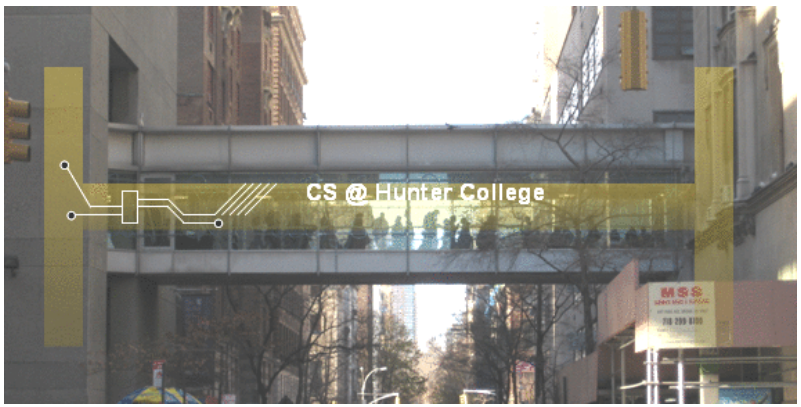
Class Reminders!



Before next class, don't forget to:

- Review this week's Lab
- Take the Lab Quiz on Gradescope by 6pm on today

Class Reminders!



Before next class, don't forget to:

- Review this week's Lab
- Take the Lab Quiz on Gradescope by 6pm on today
- Submit this class's 5 programming assignments (programs 6-15)