CSci 127: Introduction to Computer Science

Ceasar Ciper: hints for P9 of programming assignments

```python
word = input("Enter a string: ")
codedWord = ""
shift = 2 #shift two letters
for ch in word:
    offset = ord(ch) - ord('A') #distance to 'A'
    wrap = (offset + shift) % ? # %: remainder operator

    #TODO: compute new letter, call it newChar

    #TODO: add newChar to the end (right) of coded word


print("After shifting", shift, "letters,", \
    word, "becomes", codedWord)
```

Old Business: Reverse a String

Purpose: enter a string. Get its reversed version and print.

Input: a string

Output: reversed version of the input string

Process:

- Take interactive input from users.
- Initialize reversed string to be empty.
- Find each letter in the string, from left (beginning) to right (end),
  - concatenate the current letter to the left (aka front) of reversed string
- Print reversed string.

Code to reverse a string

```
#purpose: reverse a string

original = input("Enter a string: ")
reverse = ""
for ch in original:
    reverse = ch + reverse

print("reversed string is", reverse)
```

For more details or other implementations, watch video 1 and video 2.

Unicode (generalization of ASCII)

```
#google "subscript 2 unicode" and get 2082

print("x\u2082") #print x_subscript_2, \u means unicode
#2082 is unicode for subscript 2

#google "superscript 2 unicode" and get 00B2

print("x\u00B2") #print x_superscript_2
#00B2 is unicode for superscript 2
```

program to illustrate unicodes for subscript 2 and superscript 2

Today's Topics

- More on Strings

- Arithmetic

- Indexing and Slicing Lists or Strings

- Colors & Hexadecimal Notation

Indexing and Slicing Lists Example

```
daysList = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]

size = len(daysList)  #find out number of elements in daysList
for i in range(size):
    print(daysList[i], end=" ")  #ends = " " means items are separated by a
        space

print()  #print a new line
print("daysList[0] =", daysList[0])
print("daysList[-1] =", daysList[-1])
print("daysList[" + str(size-1) + "] = ", daysList[size-1])
print("daysList[" + str(-size) + "] =", daysList[-size])
print("daysList[" + str(-size+1) + "] =", daysList[-size+1])

print("daysList[0:2] =", daysList[0:2])
print("daysList[1:5:2] =", daysList[1:5:2])
print("daysList[1:6:2] =", daysList[1:6:2])
print("daysList[-5:-1:3] =", daysList[-5:-1:3])
```

Today's Topics

- **More on Strings**

- Arithmetic

- Indexing and Slicing Lists or Strings

- Colors & Hexadecimal Notation

More on Strings...

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

1. (a) What will the following Python code print:

```python
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```
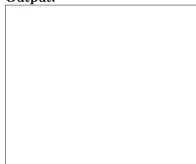
Output:

More on Strings...

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

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

More on Strings...

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- 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...

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- 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...

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- 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 3 `print()`.

More on Strings...
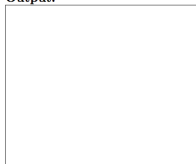
1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

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

More on Strings...

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

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

More on Strings...

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

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

More on Strings...

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

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

More on Strings...

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- 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 3 `print()`.
  - ▸ 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...

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

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

More on Strings...

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- 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✗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⊠Saturday⊠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✗Saturday✗Sunday"
  days = ['Friday', 'Saturday', 'Sunday']

- Different delimiters give different 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.

```
"FridayXSaturdayXSunday"
days = ['Friday', 'Saturday', 'Sunday']
```

- Different delimiters give different 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.

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

- Different delimiters give different lists:

  ```
  days = s[:-1].split("day")
  ```
  "Fri~~day~~sSatur~~day~~sSun~~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.

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

- Different delimiters give different lists:

  days = s[:-1].split("day")
  "Fri🗙🗙🗙sSatur🗙🗙🗙sSun🗙🗙🗙"
  days = ['Fri', 'sSatur', 'sSun']

More on Strings...

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

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

More on Strings...

1. (a) What will the following Python code print:

```python
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- 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 or Strings

- 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`

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 locations 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 locations s.

- s = s + "A" adds the letter "A" to the end of the strings s.

Challenge (Group Work): *What does this code do?*

```
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.")
```

- When dividend is divided by divisor, integer division (aka floor division) operation // return quotient without decimal numbers, and remainder operator % returns the remainder.
- For example, divide 11 pens among 5 students, each student get 2 pens (11 // 5 returns 2), and there is one pen left (11 % 5 returns 1).

```
      __2_
   5 / 11
      -10
      ---
        1
```

Challenge (Group Work): *What does this code do?*

```python
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.")
```

link to program in python tutor

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.

Today's Topics

- More on Strings

- Arithmetic

- **Indexing and Slicing Lists or Strings**

- Colors & Hexadecimal Notation

Challenge (Group Work):

```python
for d in range(10, 0, -1):
    print(d)
print("Blast off!")

for num in range(5, 8):
    print(num, 2*num)

s = "City University of New York"
print(s[3], s[0:3], s[:3])
print(s[5:8], s[-1])

names = ["Eleanor", "Anna", "Alice", "Edith"]
for n in names:
    print(n)
```

link to program

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

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

   `s[5:8]`

   gives: `"Uni"`

```
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"`
- 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

- 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"]`

```
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"`

- 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 or Strings

- **Colors & Hexadecimal Notation**

Colors

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- Can specify by name.

Colors

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- Can specify by name.
- Can specify by numbers:

Colors

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	
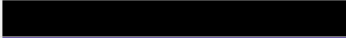
- Can specify by name.
- Can specify by numbers:
  - Amount of Red, Green, and Blue (RGB).

Colors

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- Can specify by name.
- Can specify by numbers:
  - Amount of Red, Green, and Blue (RGB).
  - Adding light, not paint:

Colors

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	
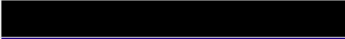
- Can specify by name.
- Can specify by numbers:
  - Amount of Red, Green, and Blue (RGB).
  - Adding light, not paint:
    - Black: 0% red, 0% green, 0% blue

Colors

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- 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

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- Can specify by numbers (RGB):

Colors

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- Can specify by numbers (RGB):
  - ▶ Fractions of each:

Colors

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- Can specify by numbers (RGB):
  - ▶ Fractions of each:
    e.g. (1.0, 0, 0) is 100% red, no green, and no blue.

Colors

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- 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

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- 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

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- 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 and Hexadecimal

	decimal	hexadecimal
base	10	16
digits	0-9	0-9, A (10) - F (15)
eg	$205 = 2 * 10^2 + 0 * 10^1 + 5 * 10^0$	$CD_{16} = 12 * 16^1 + 13 = 205_{10}$
	$255 = 2 * 10^2 + 5 * 10^1 + 5 * 10^0$	$FF_{16} = 15 * 16^1 + 15 = 255_{10}$

Colors

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- 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

Color Name	HEX	Color
Black	#000000	
Navy	#000080	
DarkBlue	#00008B	
MediumBlue	#0000CD	
Blue	#0000FF	

- 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. #0000FF is no red, no green, and 100% blue.

Challenge (Group Work): link to trinket

```python
import turtle
teddy = turtle.Turtle()

names = ["violet", "purple", "indigo", "lavender"]
for c in names:
    teddy.color(c)
    teddy.left(60)
    teddy.forward(40)
    teddy.dot(10)

teddy.penup()
teddy.forward(100)
teddy.pendown()

hexNames = ["#FF00FF", "#990099", "#550055", "#111111"]
for c in hexNames:
    teddy.color(c)
    teddy.left(60)
    teddy.forward(40)
    teddy.dot(10)
```
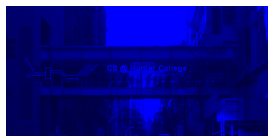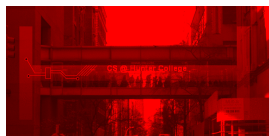
Recap

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

Practice Quiz & Final Questions

- Since you must pass the final exam to pass the course, we end every lecture with final 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.

Weekly Reminders!
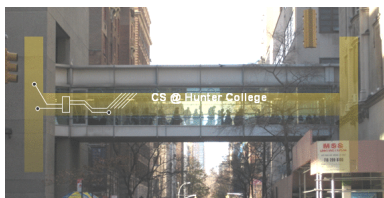
Before next lecture, don't forget to:

- Work on this week's Online Lab
- Schedule an appointment to take the Quiz in lab 1001G Hunter North
- If you haven't already, schedule an appointment to take the Code Review (**every week**) in lab 1001G Hunter North
- Submit this week's 5 programming assignments **(programs 11-15)**
- If you need help, schedule an appointment for Tutoring in lab 1001G 11:30am-5:30PM (the last appointment starts at 5:15PM)
- Take the Lecture Preview on Blackboard on Monday (or no later than 10:15am on Tuesday)

Lecture Slips & Writing Boards

- Return writing boards as you leave.