Google Earth

November 12 2015

Regex exercises

- Let's try to create the foll
  expressions:

- Capitalized words
- Hyphenated names
- Dates like 10/13/2014 or 4/4/03
- Words in quotes

Web Scraping Introduction

- Why isn't there a nice "importXML" in Python?

- *xml.etree.ElementTree* module

Using *xml.etree.ElementTree*

```
import xml.etree.ElementTree as et

filename = 'example.xml'

file = open(filename, 'r')
contents = file.read()
file.close()

tree = et.fromstring(contents)
tree.tag

for node in tree.findall('car'):
  for subnode in node.findall('year'):
    print(subnode.tag, ": ", subnode.text)
  for subnode in node.findall('color'):
    print(subnode.tag, ": ", subnode.text)
  print('---')
```

contents:

```
<inventory>
  <car>
    <year> 2010 </year>
    <color> black </color>
  </car>
  <car>
    <year> 2012 </year>
    <color> red </color>
  </car>
  <car>
    <year> 2014 </year>
    <color> yellow </color>
  </car>
</inventory>
```

Using *xml.etree.ElementTree*

```
import xml.etree.ElementTree as et

filename = 'example.xml'

file = open(filename, 'r')
contents = file.read()
file.close()

tree = et.fromstring(contents)
tree.tag

for node in tree.findall('car'):
  for subnode in node.findall('year'):
    print(subnode.tag, ": ", subnode.text)
  for subnode in node.findall('color'):
    print(subnode.tag, ": ", subnode.text)
  print('---')
```

Internal rep. of the XML

contents:

```
<inventory>
  <car>
    <year> 2010 </year>
    <color> black </color>
  </car>
  <car>
    <year> 2012 </year>
    <color> red </color>
  </car>
  <car>
    <year> 2014 </year>
    <color> yellow </color>
  </car>
</inventory>
```

Using *xml.etree.ElementTree*

```
import xml.etree.ElementTree as et

filename = 'example.xml'

file = open(filename, 'r')
contents = file.read()
file.close()

tree = et.fromstring(contents)
tree.tag

for node in tree.findall('car'):
  for subnode in node.findall('year'):
    print(subnode.tag, ": ", subnode.text)
  for subnode in node.findall('color'):
    print(subnode.tag, ": ", subnode.text)
  print('---')
```

Gives back **list** of "car" nodes

contents:

```
<inventory>
  <car>
    <year> 2010 </year>
    <color> black </color>
  </car>
  <car>
    <year> 2012 </year>
    <color> red </color>
  </car>
  <car>
    <year> 2014 </year>
    <color> yellow </color>
  </car>
</inventory>
```

Using *xml.etree.ElementTree*

```
import xml.etree.ElementTree as et

filename = 'example.xml'

file = open(filename, 'r')
contents = file.read()
file.close()

tree = et.fromstring(contents)
tree.tag

for node in tree.findall('car'):
  for subnode in node.findall('year'):
    print(subnode.tag, ": ", subnode.text)
  for subnode in node.findall('color'):
    print(subnode.tag, ": ", subnode.text)
  print('---')
```

Searches for “year” inside “ car

contents:

```
<inventory>
  <car>
    <year> 2010 </year>
    <color> black </color>
  </car>
  <car>
    <year> 2012 </year>
    <color> red </color>
  </car>
  <car>
    <year> 2014 </year>
    <color> yellow </color>
  </car>
</inventory>
```

Using *xml.etree.ElementTree*

```
import xml.etree.ElementTree as et

filename = 'example.xml'

file = open(filename, 'r')
contents = file.read()
file.close()

tree = et.fromstring(contents)
tree.tag

for node in tree.findall('car'):
  for subnode in node.findall('year'):
    print(subnode.tag, ": ", subnode.text)
  for subnode in node.findall('color'):
    print(subnode.tag, ": ", subnode.text)
  print('---')
```

Gives back **list** of " year" nodes

contents:

```
<inventory>
  <car>
    <year> 2010 </year>
    <color> black </color>
  </car>
  <car>
    <year> 2012 </year>
    <color> red </color>
  </car>
  <car>
    <year> 2014 </year>
    <color> yellow </color>
  </car>
</inventory>
```

Using *xml.etree.ElementTree*

```
import xml.etree.ElementTree as et

filename = 'example.xml'

file = open(filename, 'r')
contents = file.read()
file.close()

tree = et.fromstring(contents)
tree.tag

for node in tree.findall('car'):
  for subnode in node.findall('year'):
    print(subnode.tag, ": ", subnode.text)
  for subnode in node.findall('color'):
    print(subnode.tag, ": ", subnode.text)
  print('---')
```

contents:

```
<inventory>
  <car>
    <year> 2010 </year>
    <color> black </color>
  </car>
  <car>
    <year> 2012 </year>
    <color> red </color>
  </car>
  <car>
    <year> 2014 </year>
    <color> yellow </color>
  </car>
</inventory>
```

subnode.text gives back text
subnode.tag gives back tag

Using *xml.etree.ElementTree*

```
import xml.etree.ElementTree as et

filename = 'example.xml'

file = open(filename, 'r')
contents = file.read()
file.close()

tree = et.fromstring(contents)
tree.tag

for node in tree.findall('car'):
  for subnode in node.findall('year'):
    print(subnode.tag, ": ", subnode.text)
  for subnode in node.findall('color'):
    print(subnode.tag, ": ", subnode.text)
  print('---')
```

contents:

```
<inventory>
  <car c='good' h='10'>
    <year> 2010 </year>
    <color> black </color>
  </car>
  <car>
    <year> 2012 </year>
    <color> red </color>
  </car>
  <car>
    <year> 2014 </year>
    <color> yellow </color>
  </car>
</inventory>
```

node.attrib gives back attribute dictionary:

```
{'c': 'good', 'h':'10}
```

Getting data from the web

1) Use urllib.request

```
url = 'http://www.example.com/cars.xml'

remoteFile = urllib.request.urlopen(url)
contents = remoteFile.read()
remoteFile.close()
```

2) Use *xml.etree.ElementTree*

```
for node in tree.findall('car'):
    print node.text
    print node.tag
    attributes = node.attrib
    for key in attributes.keys:
        print(key, attributes[key])
```

Show me how that's useful!

- Go to:
  - https://maps.googleapis.com/maps/api/geocode/xml?address=115+Waterman+Street,+Providence,+RI

- This is Google letting you:
  - Give an address
  - Get back an XML containing latitude/longitude

```
<location>
  <lat>41.8271609</lat>
  <lng>-71.3995390</lng>
</location>
```

Show me how that's useful!

- Go to:
  - https://maps.googleapis.com/maps/api/geocode/xml?latlng=41.826273,-71.401628

- This is Google letting you:
  - Give latitude/longitude
  - Get back an XML with address information

```
<address_component>
  <long_name>Brown University</long_name>
  <short_name>Brown University</short_name>
  <type>establishment</type>
</address_component>
<address_component>
  <long_name>George Street</long_name>
  <short_name>George St</short_name>
  <type>route</type>
</address_component>
<address_component>
  <long_name>College Hill</long_name>
  <short_name>College Hill</short_name>
  <type>neighborhood</type>
  <type>political</type>
```

This happens a lot!

- Web services *expose functionality* via *XML*
- Or via something called *JSON*
  - Even *more convenient*!

- Stay tuned!
- Next class we *interact with Google Maps!*

Google Earth

- Download and install Goo
  - If you haven't done so, a TA

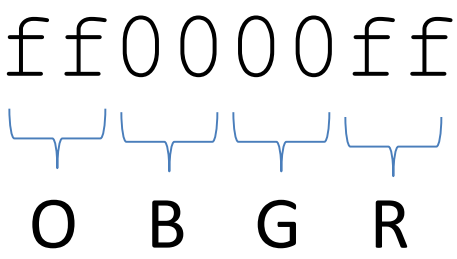
In groups, do Task 1

- Download CIT.kml from the course webpage
  - Open the file on Google Earth
  - Open the file using
    - TextEdit (Mac)
    - Notepad (Win)

Hmm, let's talk about the color in the KML file

KML Color Codes

O = Opacity
B = Blue
G = Green
R = Red

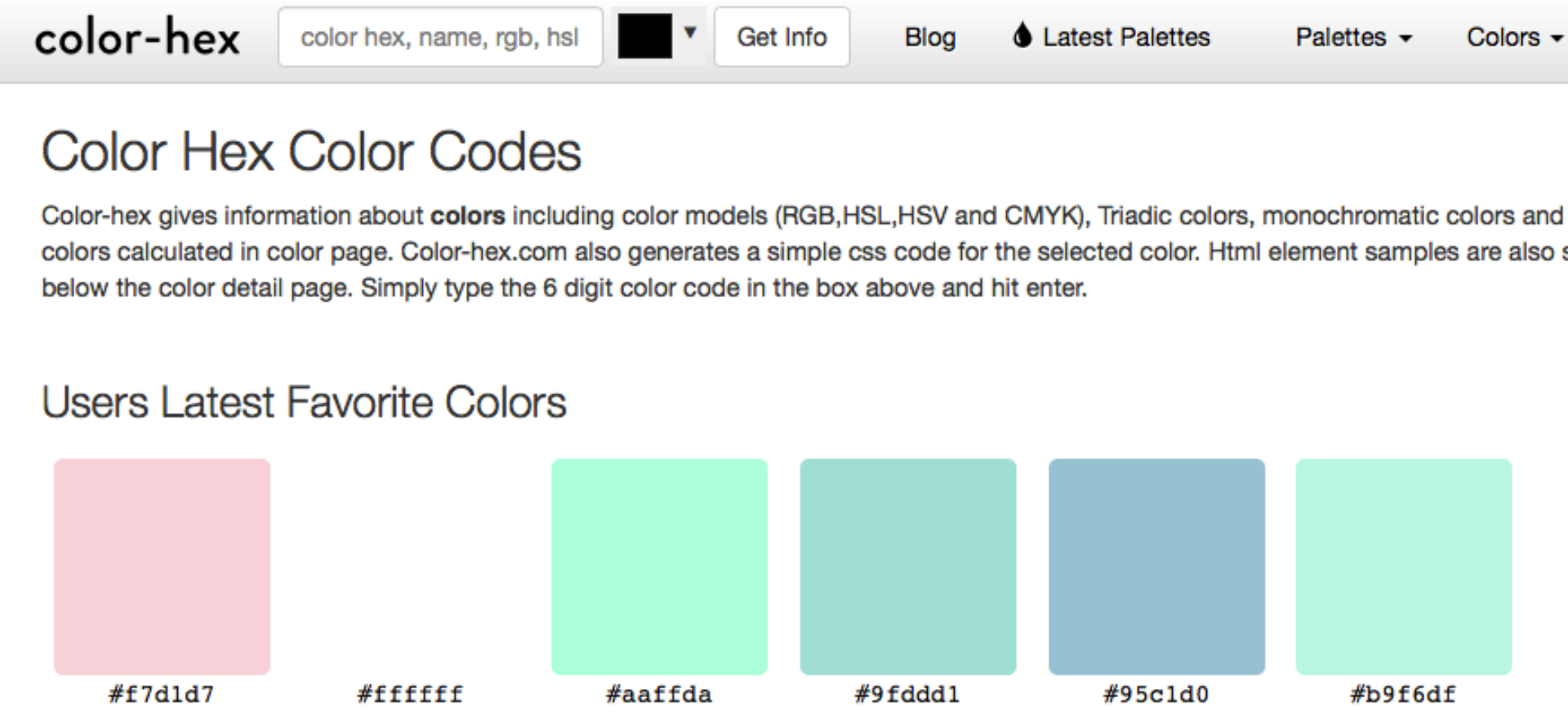


Hexadecimal (16 digits): 0-9,a-f

00 01 ... 08 09 0a 0b 0c 0d 0e 0f 10 11 ... fe ff

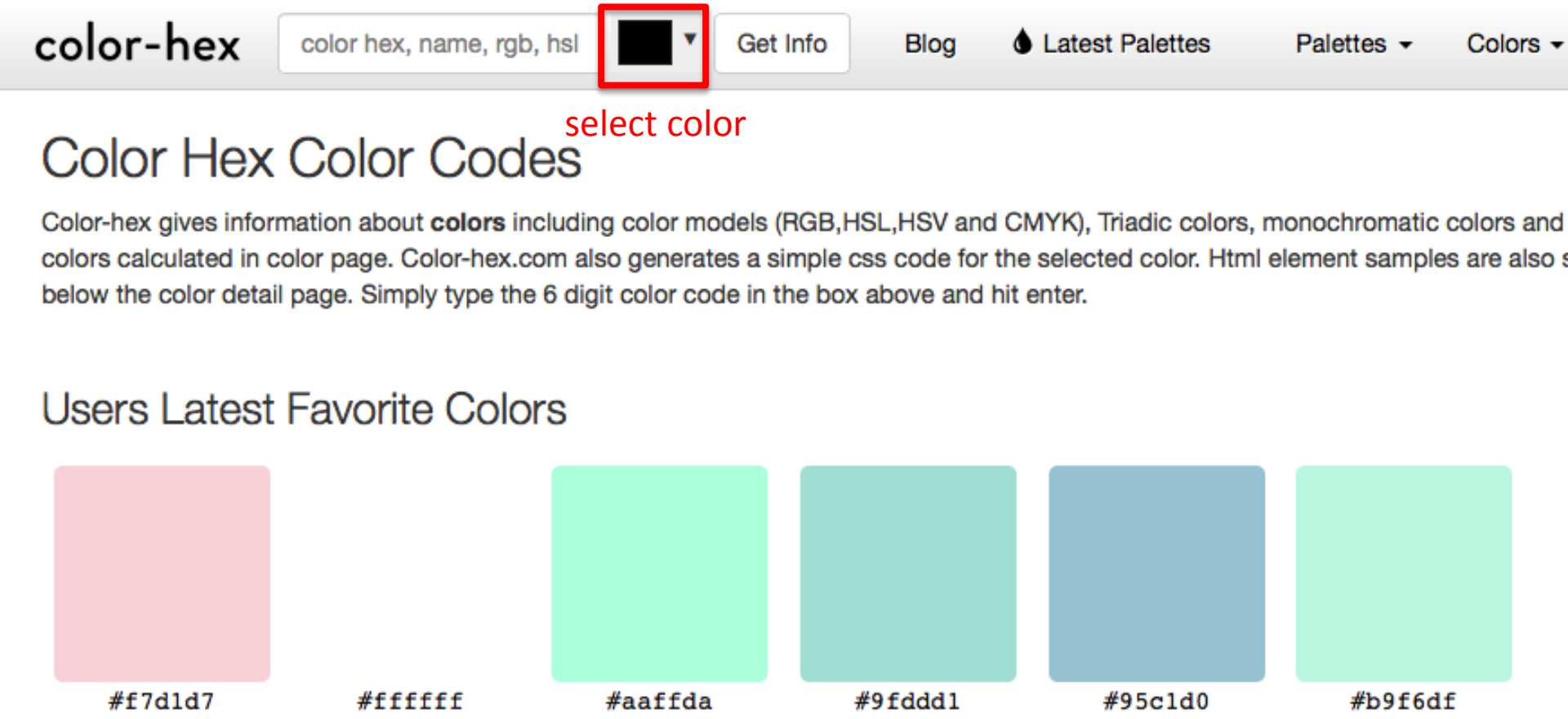
KML Color Codes

Go to http://www.color-hex.com



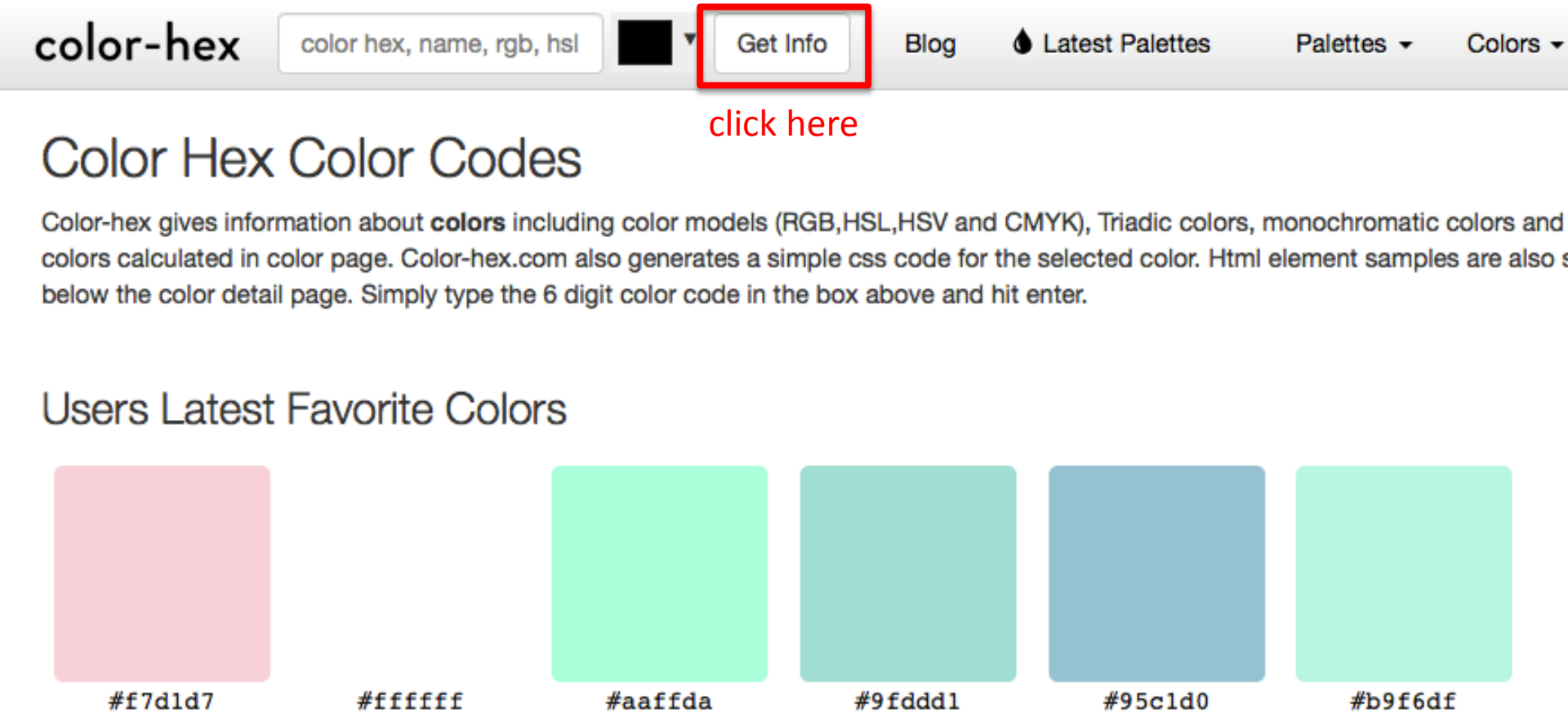
KML Color Codes

Go to http://www.color-hex.com



KML Color Codes

Go to http://www.color-hex.com



KML Color Codes

#00beef Color Hex

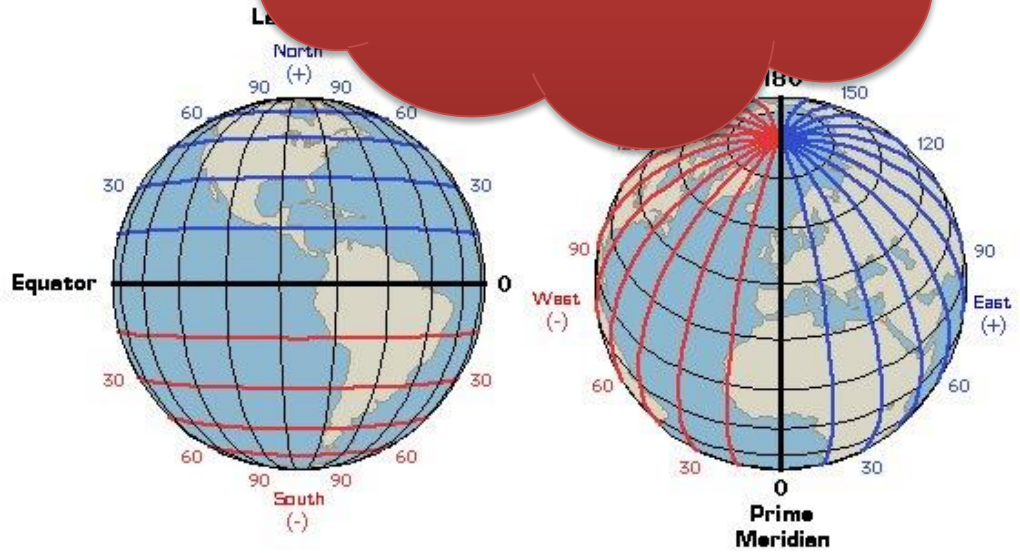
Use this in your KML

ffefbe00

Google Earth

- Now open CIT.kml in a tex
  changing the following:
  - Name
  - Description
  - Size of Pin
  - Coordinates
  - Color **ffefbe00**

In groups, do Task 2



http://geographyworldonline.com/tutorial/instructions.html

Google Map API

- Go to:
  - https://maps.googleapis.com/maps/api/geocode/xml?address=115+Waterman+Street,+Providence,+RI

- This is Google letting you:
  - Give an address
  - Get back an XML containing latitude/longitude

```
<location>
  <lat>41.8271609</lat>
  <lng>-71.3995390</lng>
</location>
```

Google Map API

- Go to:
  - https://maps.googleapis.com/maps/api/geocode/xml?latlng=41.826273,-71.401628

- This is Google letting you:
  - Give latitude/longitude
  - Get back an XML with address information

```
<address_component>
  <long_name>Brown University</long_name>
  <short_name>Brown University</short_name>
  <type>establishment</type>
</address_component>
<address_component>
  <long_name>George Street</long_name>
  <short_name>George St</short_name>
  <type>route</type>
</address_component>
<address_component>
  <long_name>College Hill</long_name>
  <short_name>College Hill</short_name>
  <type>neighborhood</type>
  <type>political</type>
```

Application Programming Interfaces

- Web services *expose functionality* via *XML*
- Or via something called *JSON*
  - Even *more convenient*!

- The Google Earth example is an example of an API
  - Application Programming Interface

Let's try the same with JSON

- Go to:
  - https://maps.googleapis.com/maps/api/geocode/**json**?latlng=41.826273,-71.401628

- This is Google letting you:
  - Give latitude/longitude
  - Get back a *JSON response* with address information

  - Let's check its contents

```
{
   "results" : [
      {
         "address_components" : [
            {
               "long_name" : "Mountain View",
               "short_name" : "Mountain View",
               "types" : [ "locality", "political" ]
            },
            {
               "long_name" : "Santa Clara County",
               "short_name" : "Santa Clara County",
               "types" : [ "administrative_area_level_2", "political" ]
            },
            {
               "long_name" : "California",
               "short_name" : "CA",
               "types" : [ "administrative_area_level_1", "political" ]
            },
            {
               "long_name" : "United States",
               "short_name" : "US",
               "types" : [ "country", "political" ]
            },
            {
               "long_name" : "94043",
               "short_name" : "94043",
               "types" : [ "postal_code" ]
            }
         ],
```

```
         "formatted_address" : "1600 Amphitheatre Parkway, Mountain View, CA
94043, USA",
         "geometry" : {
            "location" : {
               "lat" : 37.4224764,
               "lng" : -122.0842499
            },
            "location_type" : "ROOFTOP",
            "viewport" : {
               "northeast" : {
                  "lat" : 37.4238253802915,
                  "lng" : -122.0829009197085
               },
               "southwest" : {
                  "lat" : 37.4211274197085,
                  "lng" : -122.0855988802915
               }
            }
         },
         "place_id" : "ChIJ2eUgeAK6j4ARbn5u_wAGqWA",
         "types" : [ "street_address" ]
      }
   ],
   "status" : "OK"
}
```

1) From place to latitude/longitude

- `response = # response to a JSON request`
- `If response["status"] is not "OK"`
  - `Tell user "Nothing here"`
- `Else`
  - `Latitude is in response["results"]`
    - which is a list…
  - `At position 0` (the most likely response)
    - which is a dictionary
  - `At key "geometry"`
    - which is another dictionary
  - `At key "location"`
    - which is yet another dictionary
  - `At keys "lat" or "lng"`
    - which are numbers!

From English to Python

```
def getCoordinates(placeName):
    '''INPUT: string query OUTPUT (latitude, logintude)
       string -> [float,float]'''

    googleGeocodeUrl ='http://maps.googleapis.com/maps/api/geocode/json?'
    query = placeName.encode('utf-8')
    params = {'address': query}

    url = googleGeocodeUrl + urllib.parse.urlencode(params)
    json_response = urllib.request.urlopen(url)
    response = json.loads(json_response.readall().decode('utf-8'))

    if response['status'] != 'OK':
        return [0.00, 0.00]
    else
        location = response['results'][0]['geometry']['location']
        latitude, longitude = location['lat'], location['lng']
        return [latitude, longitude]
```

From English to Python

```
def getCoordinates(placeName):
    '''INPUT: string query OUTPUT (latitude, logintude)
       string -> [float,float]'''
    googleGeocodeUrl ='http://maps.googleapis.com/maps/api/geocode/json?'
    query = placeName.encode('utf-8')
    params = {'address': query}

    url = googleGeocodeUrl + urllib.parse.urlencode(params)
    json_response = urllib.request.urlopen(url)
    response = json.loads(json_response.readall().decode('utf-8'))

    if response['status'] != 'OK':
        return [0.00, 0.00]
    else
        location = response['results'][0]['geometry']['location']
        latitude, longitude = location['lat'], location['lng']
        return [latitude, longitude]
```

Create URL, use urllib to get the JSON response

From English to Python

```
def getCoordinates(placeName):
    '''INPUT: string query OUTPUT (latitude, logintude)
       string -> [float,float]'''

    googleGeocodeUrl ='http://maps.googleapis.com/maps/api/geocode/json?'
    query = placeName.encode('utf-8')
    params = {'address': query}

    url = googleGeocodeUrl + urllib.parse.urlencode(params)
    json_response = urllib.request.urlopen(url)
    response = json.loads(json_response.readall().decode('utf-8'))

    if response['status'] != 'OK':
        return [0.00, 0.00]
    else
        location = response['results'][0]['geometry']['location']
        latitude, longitude = location['lat'], location['lng']
        return [latitude, longitude]
```

Transform string response into Python objects

From English to Python

```
def getCoordinates(placeName):
    '''INPUT: string query OUTPUT (latitude, logintude)
       string -> [float,float]'''

    googleGeocodeUrl ='http://maps.googleapis.com/maps/api/geocode/json?'
    query = placeName.encode('utf-8')
    params = {'address': query}

    url = googleGeocodeUrl + urllib.parse.urlencode(params)
    json_response = urllib.request.urlopen(url)
    response = json.loads(json_response.readall().decode('utf-8'))

    if response['status'] != 'OK':
        return [0.00, 0.00]
    else
        location = response['results'][0]['geometry']['location']
        latitude, longitude = location['lat'], location['lng']
        return [latitude, longitude]
```

The high-level dictionary must have status "OK", or we had no response

From English to Python

```
def getCoordinates(placeName):
    '''INPUT: string query OUTPUT (latitude, logintude)
       string -> [float,float]'''

    googleGeocodeUrl ='http://maps.googleapis.com/maps/api/geocode/json?'
    query = placeName.encode('utf-8')
    params = {'address': query}

    url = googleGeocodeUrl + urllib.parse.urlencode(params)
    json_response = urllib.request.urlopen(url)
    response = json.loads(json_response.readall().decode('utf-8'))

    if response['status'] != 'OK':
        return [0.00, 0.00]
    else
        location = response['results'][0]['geometry']['location']
        latitude, longitude = location['lat'], location['lng']
        return [latitude, longitude]
```

If we got an "OK" response, we query lat/lng

2) From latitude/longitude to place

```
def printCountry(latlonString) :
    googleGeocodeUrl ='http://maps.googleapis.com/maps/api/geocode/json?'
    query = latlonString.encode('utf-8')
    params = {'latlng': query}

    url = googleGeocodeUrl + urllib.parse.urlencode(params)
    json_response = urllib.request.urlopen(url)
    response = json.loads(json_response.readall().decode('utf-8'))

    if response['status'] != 'OK':
      print 'Nothing found'
    else
      descriptions = response['results'][0]['address_components']

      for description in descriptions:
         if 'country' in description['types'] :
            print description['long_name']
```

2) From latitude/longitude to place

```
def printCountry(latlonString) :
    googleGeocodeUrl ='http://maps.googleapis.com/maps/api/geocode/json?'
    query = latlonString.encode('utf-8')
    params = {'latlng': query}

    url = googleGeocodeUrl + urllib.parse.urlencode(params)
    json_response = urllib.request.urlopen(url)
    response = json.loads(json_response.readall().decode('utf-8'))

    if response['status'] != 'OK':
      print 'Nothing found'
    else
      descriptions = response['results'][0]['address_components']

      for description in descriptions:
         if 'country' in description['types'] :
            print description['long_name']
```

The high-level dictionary must have status "OK", or we had no response

2) From latitude/longitude to place

```
def printCountry(latlonString) :
    googleGeocodeUrl ='http://maps.googleapis.com/maps/api/geocode/json?'
    query = latlonString.encode('utf-8')
    params = {'latlng': query}

    url = googleGeocodeUrl + urllib.parse.urlencode(params)
    json_response = urllib.request.urlopen(url)
    response = json.loads(json_response.readall().decode('utf-8'))

    if response['status'] != 'OK':
      print 'Nothing found'
    else
      descriptions = response['results'][0]['address_components']

      for description in descriptions:
         if 'country' in description['types'] :
            print description['long_name']
```

Get "descriptions", which is a list of dictionaries (refer to the refer structure)

2) From latitude/longitude to place

```
def printCountry(latlonString) :
    googleGeocodeUrl ='http://maps.googleapis.com/maps/api/geocode/json?'
    query = latlonString.encode('utf-8')
    params = {'latlng': query}

    url = googleGeocodeUrl + urllib.parse.urlencode(params)
    json_response = urllib.request.urlopen(url)
    response = json.loads(json_response.readall().decode('utf-8'))

    if response['status'] != 'OK':
      print 'Nothing found'
    else
      descriptions = response['results'][0]['address_components']

      for description in descriptions:
        if 'country' in description['types']:
          print description['long_name']
```

For each dictionary, check if "types" has the kind of descriptor you're looking for

You can do much more than print countries!

- *Change* `printCountry`
  - Print country, state, and cit… d
  - Print "Nowhere that I know of otherw…

In groups, do Task 3

Hint: refer back to slide 14

API Details (with much more options)

https://developers.google.com/maps/documentation/geocoding/