MET CS 688
Web Scraping & Web Crawling

Leila Ghaedi – Fall 2024

Creator: cemagraphics | Credit: Getty Images/iStockphoto

Web Scraping Applications

- Web scrapping can read pages, that is hard for human to read and summarize.
  - Example: 'cheapest flight from X to Y.'
- Web Scrapping is so useful that some web sites provide their own API (Application Programable Interface) for scrapping their data such as Twitter API, Instagram API, Youtube API, WikiPedia API, …
- Many applications benefit from web scrapping
  - Market forecasting and market studies, like scrapping online product review from Amazon, identifying public opinion about a corporation from twitter, what are trends on crypto currencies, etc.
  - Machine-language translation, like using web text as a template to reconstruct a sentence correctly .
  - Medical diagnostics (retrieve and analyze data from news sites, translated texts, and health forums), like reading health forums to identify a symptom of a drug in large scale.

Web scraping basic steps:

1. Retrieving HTML data from a domain name
2. Parsing that data for target information
3. Storing the target information
4. Optionally, moving to another page to repeat the process

Some Available Data Examples

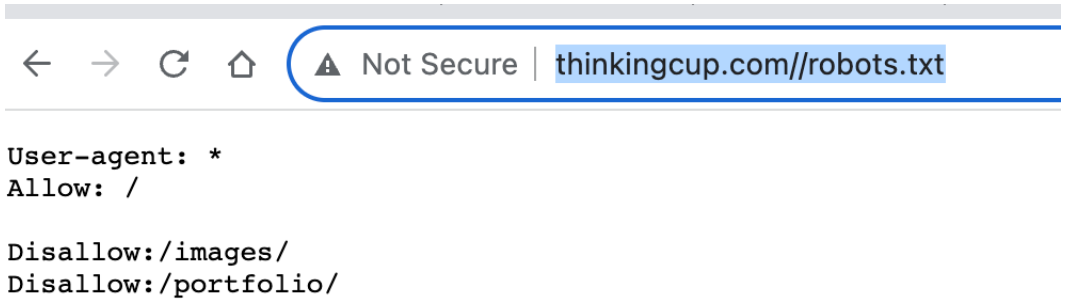
- Crowdfunding
  - Indiegogo https://webrobots.io/indiegogo-dataset/
  - Kickstarter https://webrobots.io/kickstarter-datasets/
- Academia
  - Pubmed: PubMed® comprises more than 36 million citations for biomedical literature from MEDLINE, life science journals, and online books. Citations may include links to full text content from PubMed Central and publisher web sites. https://www.nlm.nih.gov/databases/download/pubmed_medline.html
  - DBLP: (Digital Bibliography & Library Project) dataset is a comprehensive bibliographic database of computer science research papers and proceedings. https://dblp.uni-trier.de/xml/

Scraping Example

- How to find out the website allows Scraping:

  Take the root of the url and add '/robots.txt' and enter it to the browser

  http://thinkingcup.com/robots.txt

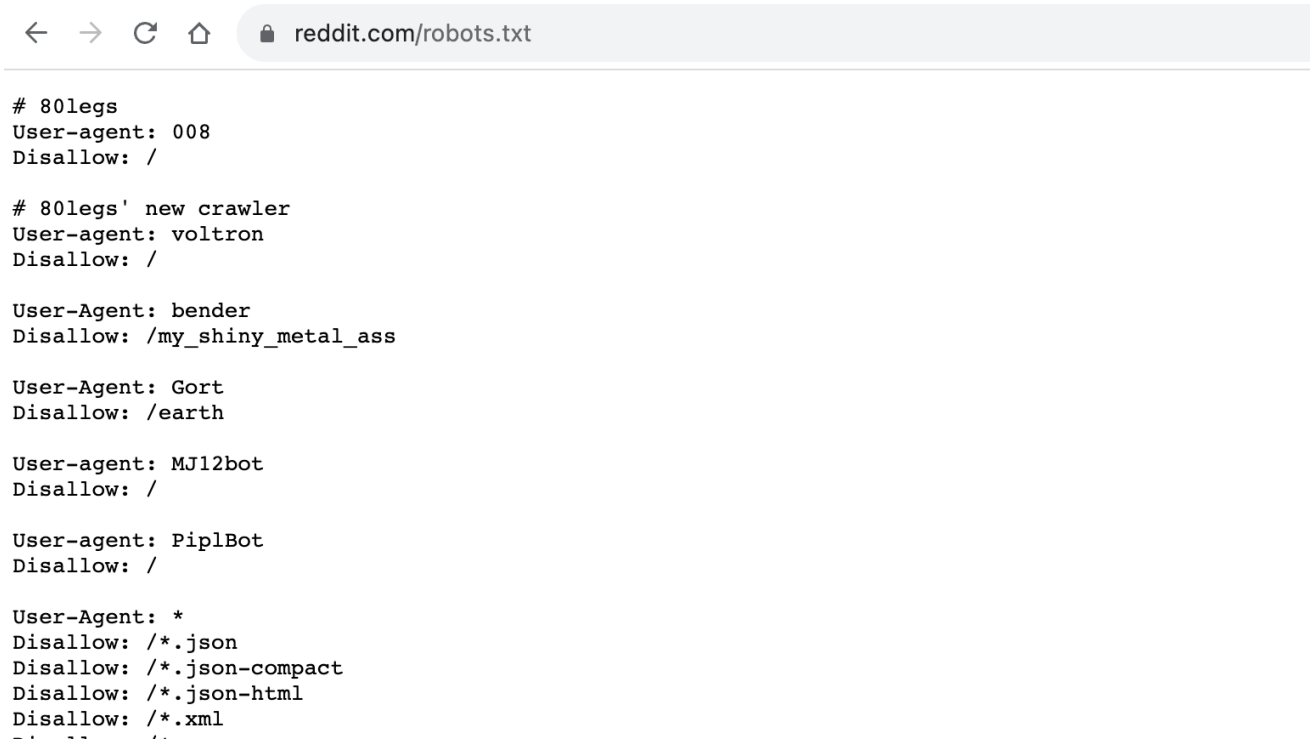


What is a user agent in crawling?

- It’s a string which helps the destination server identify which browser, device and operating system is being used.
- "userAgent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.0.0 Safari/537.36,gzip(gfe)”

Scraping Example (few months ago)

- https://www.reddit.com/robots.txt



robots.txt

- robots.txt is the filename used for implementing the **Robots Exclusion Protocol**, a standard used by websites to indicate to visiting web crawlers and other web robots which portions of the website they are allowed to visit.
- This relies on voluntary compliance. Not all robots comply with the standard; email harvesters, spambots, malware and robots that scan for security vulnerabilities may even start with the portions of the website where they have been told to stay out.

https://en.wikipedia.org/wiki/Robots.txt

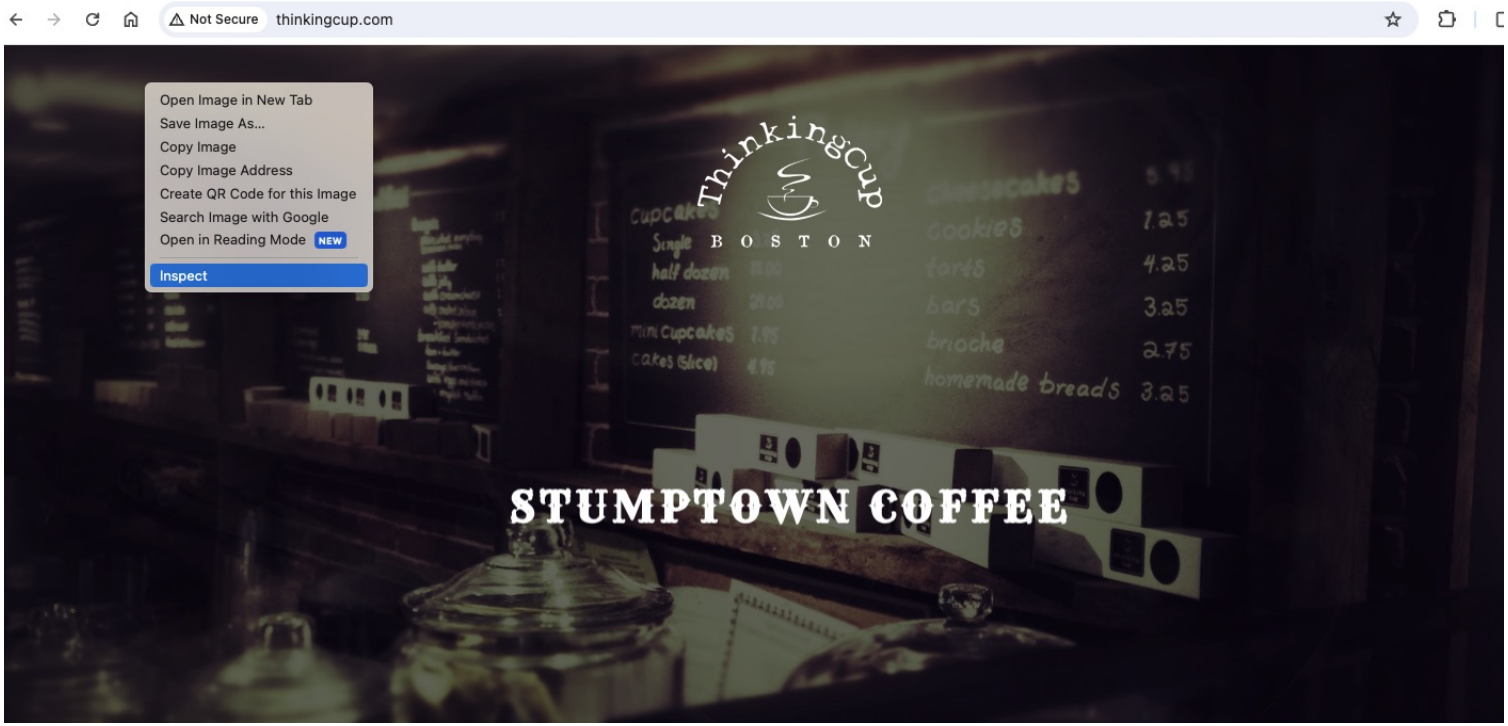
Scraping Example (now)

- https://www.reddit.com/robots.txt

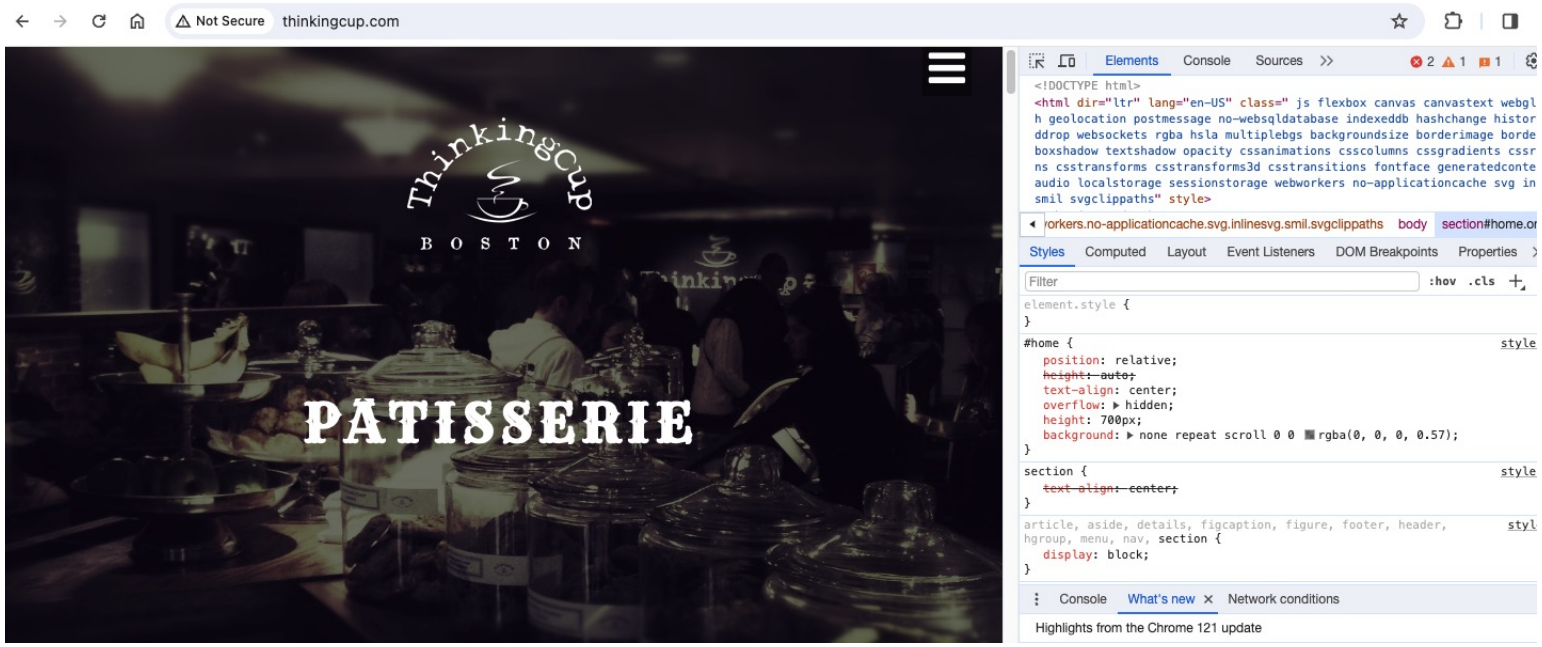
HTML Source in Browser

- You can view the HTML source of a website in your browser:
- Chrome:
  - Navigate to the web page you would like to examine. Right-click the page and look at the menu that appears. From that menu, click View page source. The source code for that page will now appear as a new tab in the browser

HTML Source in Browser



HTML Source in Browser



Scraping Example

- pip install requests
- Pip install beatifulsoup4
- Requests library intends to make parsing HTML (e.g. scraping the web) as simple and intuitive as possible.
  Beautiful Soup is a Python library for pulling data out of HTML and XML files.
- The prettify() method in Beautiful Soup is used to visualize and understand the structure of HTML and XML documents.

Scraping Example

In [1]:
```
import requests
from bs4 import BeautifulSoup as bs
```

In [2]:
```
x = requests.get('http://thinkingcup.com/')
soup = bs(x.content)
print(soup.prettify())
```

```
<!DOCTYPE html>
<html class="no-js" dir="ltr" lang="en-US">
 <head>
  <meta charset="utf-8"/>
  <title>
   Thinking Cup Coffee Shop Official Website - Boston
  </title>
  <meta content="Thinking Cup is pleased to serve stumptown coffee, organic teas and freshly baked pastries. We are
located in Downtown Boston." name="description"/>
  <meta content="thinking cup, coffee, specialty, gourmet coffee, coffee beans, coffee blends, coffee gifts, organi
c, tea, herbal, green tea, black tea,stumptown  coffee, espresso, lattes, downtown boston" name="keywords"/>
  <link href="http://thinkingcup.com" rel="canonical"/>
  <link href="http://thinkingcup.com/favicon.ico" rel="icon" type="image/x-icon"/>
  <meta content="IgSirmFO1Bji93b2mGsnyR3eTTBs42s_0U2xF7sAsMc" name="google-site-verification"/>
  <meta content="EwKGDZSL31rFBBHQ8A-WXYMDGQxE2hUlodEP5RvGydE" name="google-site-verification"/>
  <meta content="width=device-width, user-scalable=no, initial-scale=1, maximum-scale=1" name="viewport"/>
```

Extract <h1> tag from Beautiful Soup Object

- HTML <h1> to <h6> Tags
-
```
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>
```
- `.find()` method

Scraping Example

In [3]:
```
first_header = soup.find("h1")
first_header
```

Out[3]:
```
<h1>Stumptown Coffee</h1>
```

In [4]:
```
headers = soup.find_all("h1")
print(headers)
```

```
[<h1>Stumptown Coffee</h1>, <h1 style="font-size:33px;">Internationally Recognized Barista</h1>, <h1>Pâtisserie</h1>,
<h1 style="color:white"><a href="https://shop.thinkingcup.com/" style="color:white">Shop Online</a></h1>, <h1>Breakfa
st &amp;amp; Lunch</h1>, <h1 class="section-title" style="color:white">Order Ahead Onlinefor In-store Pickup <i class="ic
on-external-link"></i></h1>, <h1 class="section-title">About us</h1>, <h1 class="section-title">Menus</h1>, <h1 class
="section-title">Using only the finest coffee beans from various regions around the globe</h1>, <h1 class="section-ti
tle">Press &amp;amp; Mentions</h1>, <h1 class="section-title">Jobs</h1>, <h1 class="section-title">Helping to sustain a g
reen environment</h1>, <h1 class="section-title">Locations &amp;amp; Hours</h1>, <h1 class="section-title">Go to our Onli
ne Shop <i class="icon-external-link"></i></h1>]
```

Scraping Example

In [5]:
```
h1_h2_headers = soup.find_all(["h1" , "h2"])
h1_h2_headers
```

Out[5]:
```
[<h1>Stumptown Coffee</h1>,
 <h1 style="font-size:33px;">Internationally Recognized Barista</h1>,
 <h1>Pâtisserie</h1>,
 <h1 style="color:white"><a href="https://shop.thinkingcup.com/" style="color:white">Shop Online</a></h1>,
 <h1>Breakfast &amp;amp; Lunch</h1>,
 <h1 class="section-title" style="color:white">Order Ahead Onlinefor In-store Pickup <i class="icon-external-link"></
i></h1>,
 <h2 class="section-pre-title">Our Story.....</h2>,
 <h1 class="section-title">About us</h1>,
 <h2>Awards</h2>,
 <h2 class="section-pre-title">drink and eat</h2>,
 <h1 class="section-title">Menus</h1>,
 <h2><strong>Coffee</strong></h2>,
 <h2><strong>Tea &amp;amp; More</strong></h2>,
 <h2><strong>Breakfast</strong></h2>,
 <h2><strong>Sandwiches</strong></h2>,
 <h2 class="section-pre-title">Gluten Free Menu</h2>,
 <h2 class="section-pre-title">Dairy Free Menu</h2>,
 <h2 class="section-pre-title">THINKING CUP PHILOSOPHY</h2>,
 <h1 class="section-title">Using only the finest coffee beans from various regions around the globe</h1>,
 <h2 class="section-pre-title">they talk about us</h2>,
 <h1 class="section-title">Press &amp;amp; Mentions</h1>,
```

Scraping Example

- select() method is similar to find_all() method.
- select_one() method is like find() method.
- The .select() method in Beautiful Soup allows us to find elements in an HTML document using CSS selectors. It returns a list of matching elements, which we can then use to extract information or navigate further within the document.
- The HTML <p> element defines a paragraph.

Scraping Example

```
In [6]: content = soup.select("p")
        content
```

```
Out[6]: [<p>Thinking Cup is pleased to be the first coffee shop in Downtown Boston to serve "Stumptown Coffee" and "Third W
        ave Coffee" products exclusively.  Stumptown, originating in Portland Oregon had been named "Best Coffee in the Wor
        ld" by NPR, The New York Times, Food &amp;amp; Wine Magazine, and USA Today.  To ensure exceptional quality and freshne
        ss we obtain our beans directly from Stumptown's roasting facility in Brooklyn, New York.</p>,
         <p>Opened its doors in December 2010, Thinking Cup is ideally located just steps from the Boston Common.  We are c
        ommitted to providing the premier coffee experience in Boston by serving only the finest coffees, teas, and freshly
        baked pastries.
         </p>,
         <p>
         <strong>THINKING CUP PHILOSOPHY</strong>
         <br/>
         <i class="fa fa-check"></i> Using only the finest coffee beans from various regions around the globe <br/>
         <i class="fa fa-check"></i> Offering outstanding service from knowledgeable baristas willing to educate our guests
        about coffee <br/>
```

Scraping Example

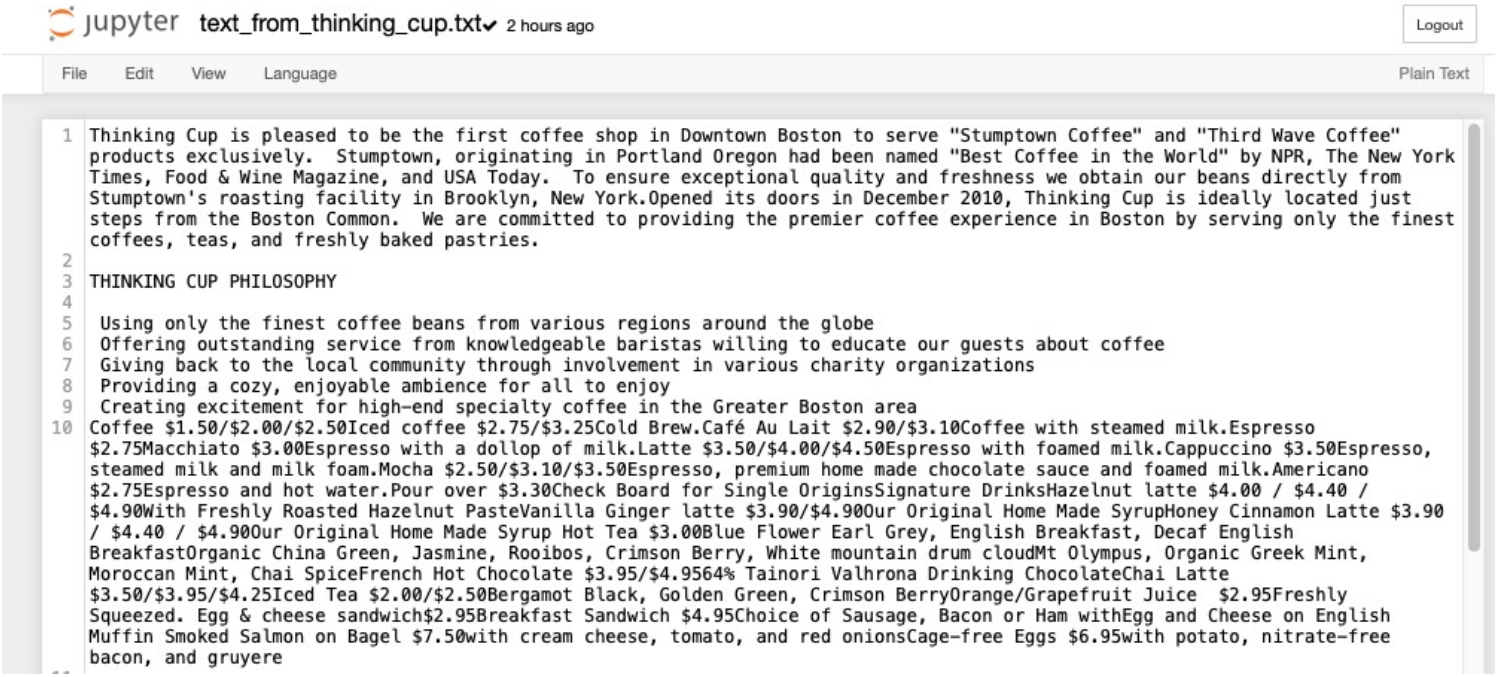
- How to extract paragraphs from a website and save it as a text file.

```
In [7]: f = open("text_from_thinking_cup.txt", "w")

        # paragraphs from soup
        for data in soup.find_all("p"):
            sum = data.get_text()
            f.writelines(sum)

        f.close()
```

Scraping Example



ebay Scraping Example

- In HTML, the span tag is a generic inline container element.
- The tag is an HTML element that groups related parts of a web page. It's used to wrap sections of text for styling purposes or to add attributes to a section of text without creating a new line of content. For example, you could use to change the color of a word in a paragraph.
- Let's imagine we are interested in "xbox series s console" and want to explore the ebay listings.

ebay Scraping Example

1. Import the libraries
2. Send a get request to the URL of the target website, which in this case is the results page of a search for "xbox series s console"
3. Print the title element of the webpage

In [1]:
```
import requests
from bs4 import BeautifulSoup
response = requests.get("https://www.ebay.com/sch/i.html?_from=R40&_trksid=\
p2334524.m570.l1311&_nkw=xbox+series+x&_sacat=0&LH_TitleDesc=0&_odkw=xbox&_osacat=0")
soup = BeautifulSoup(response.text, 'html.parser')
print(soup.title)
```

```
<title>xbox series x for sale | eBay</title>
```

ebay Scraping Example

Review HTML file for the data you are looking for
Find span tag for price

In [2]:
```
# print html
print(soup.html)
```

```
<html lang="en"><!--<![endif]--><!--M#s0-2--><noscript class="x-page-config"></noscript><!--M/--><head><meta conten
t="IE=edge" http-equiv="X-UA-Compatible"/><script>"use strict";if(window.PerformanceObserver&&performance&&performa
nce.mark&&performance.getEntriesByName){window.SRP=window.SRP||{};var paintObserver=new window.PerformanceObserver
((function(e){var r=e.getEntries();if(r.sort((function(e,r){return e.startTime-r.startTime})),r&&!(r.length<2)){var
n=r[1].startTime;window.SRP.TTI_TIMER={lastInteractiveWindow:n};var t=new window.PerformanceObserver((function(e){f
or(var r=e.getEntries(),i=0,a=r.length;i<a;i++)r[i].startTime-n>=5e3&&(window.SRP.TTI_TIMER.timeToInteract=n,t.disc
onnect()),n=r[i].startTime+r[i].duration,window.SRP.TTI_TIMER.lastInteractiveWindow=n}));t.observe({entryTypes:["lo
ngtask"]}),paintObserver.disconnect()})));paintObserver.observe({entryTypes:["paint"]})};</script><title>xbox serie
s x for sale | eBay</title><meta content="Get the best deals for xbox series x at eBay.com. We have a great online
selection at the lowest prices with Fast &amp;amp; Free shipping on many items!" name="description"><meta content="unsa
fe-url" name="referrer"/><meta content="https://ir.ebaystatic.com/cr/v/c1/ebay-logo-1-1200x630-margin.png" property
="og:image"/><meta content="34E98E6F27109BE1A9DCF19658EEEE33" name="msvalidate.01"/><link href="https://ir.ebaystat
```

ebay Scraping Example

Review HTML file for the data you are looking for
Find span tag for price

In [2]:
```
# print html
print(soup.html)
```

```
<html lang="en"><!--<![endif]--><!--M#s0-2--><noscript class="x-page-config"></noscript><!--M/--><head><meta conten
t="IE=edge" http-equiv="X-UA-Compatible"/><script>"use strict";if(window.PerformanceObserver&&performance&&performa
nce.mark&&performance.getEntriesByName){window.SRP=window.SRP||{};var paintObserver=new window.PerformanceObserver
((function(e){var r=e.getEntries();if(r.sort((function(e,r){return e.startTime-r.startTime})),r&&!(r.length<2)){var
n=r[1].startTime;window.SRP.TTI_TIMER={lastInteractiveWindow:n};var t=new window.PerformanceObserver((function(e){f
or(var r=e.getEntries(),i=0,a=r.length;i<a;i++)r[i].startTime-n>=5e3&&(window.SRP.TTI_TIMER.timeToInteract=n,t.disc
onnect()),n=r[i].startTime+r[i].duration,window.SRP.TTI_TIMER.lastInteractiveWindow=n}));t.observe({entryTypes:["lo
ngtask"]}),paintObserver.disconnect()})));paintObserver.observe({entryTypes:["paint"]})};</script><title>xbox serie
s x for sale | eBay</title><meta content="Get the best deals for xbox series x at eBay.com. We have a great online
selection at the lowest prices with Fast &amp;amp; Free shipping on many items!" name="description"><meta content="unsa
fe-url" name="referrer"/><meta content="https://ir.ebaystatic.com/cr/v/c1/ebay-logo-1-1200x630-margin.png" property
="og:image"/><meta content="34E98E6F27109BE1A9DCF19658EEEE33" name="msvalidate.01"/><link href="https://ir.ebaystat
```

ebay Scraping Example

The <a> tag defines a hyperlink, which is used to link from one page to another.

```
In [3]: # extract all the URLs found within a page's <a> tags:
        for link in soup.find_all('a'):
            print(link.get('href'))
```

```
#mainContent
https://www.ebay.com/
https://www.ebay.com/sch/ebayadvsearch
https://signin.ebay.com/ws/eBayISAPI.dll?SignIn&_trksid=m570.l3348
https://www.ebay.com/deals
https://www.ebay.com/b/Brand-Outlet/bn_7115532402
https://ocsnext.ebay.com/ocs/home
https://www.ebay.com/sl/sell
https://www.ebay.com/mye/myebay/watchlist
#gh-wl-click
https://signin.ebay.com/ws/eBayISAPI.dll?SignIn&ru=ru
https://www.ebay.com/mys/home?source=GBH
#gh-eb-My
https://www.ebay.com/mye/myebay/summary
https://www.ebay.com/mye/myebay/rvi
https://www.ebay.com/mye/myebay/v2/bidsoffers
```

ebay Scraping Example

```
In [4]: # extract all the text from a page
        print(soup.get_text())
```

```
xbox series x for sale | eBay
 Skip to main content Shop by categoryShop by categoryEnter your search keyword All CategoriesAntiques Art Baby Boo
ks & Magazines Business & Industrial Cameras & Photo Cell Phones & Accessories Clothing, Shoes & Accessories Coins
& Paper Money Collectibles Computers/Tablets & Networking Consumer Electronics Crafts Dolls & Bears Entertainment M
emorabilia Everything Else Gift Cards & Coupons Health & Beauty Home & Garden Jewelry & Watches Movies & TV Music M
usical Instruments & Gear Pet Supplies Pottery & Glass Real Estate Specialty Services Sporting Goods Sports Mem, Ca
rds & Fan Shop Stamps Tickets & Experiences Toys & Hobbies Travel Video Games & ConsolesAdvancedHi (Sign in to bid
or buy) Daily Deals Brand Outlet Help & Contact SellWatchlistExpand Watch ListLoading...Sign in to see your user in
formation My eBayExpand My eBay Summary Recently Viewed Bids/Offers Watchlist Purchase History Buy Again Selling Sa
ved Searches Saved Sellers My Garage Messages Collection beta The eBay vaultNotification  Expand CartLoading...Some
thing went wrong. View cart for details.  {"delay":300}  SponsoredSponsoredSponsoredSponsoredSponsoredSponsoredInc
lude descriptionFilterCategoryAllVideo Games & ConsolesVideo GamesVideo Game AccessoriesSelected categoryVideo Game
ConsolesReplacement Parts & ToolsOriginal Game Cases & BoxesVideo Game MerchandisePrepaid Gaming CardsManuals, Inse
rts & Box ArtStrategy Guides & CheatsMixed LotsMoreEverything ElseConsumer ElectronicsSports Mem, Cards & Fan ShopC
```

ebay Scraping Example

In [5]:
```
prices = soup.find_all("span", {"class", "s-item__price"})

for price in prices:
    print(price.text)
```

```
$20.00
$276.69
$469.99
$278.57
$349.99
$385.00
$325.00
$345.00
$380.00
$330.00
$267.00
$280.00
```

ebay Scraping Example

```
In [6]: print(type(prices))
        print(type(price))
```

```
<class 'bs4.element.ResultSet'>
<class 'bs4.element.Tag'>
```

ebay Scraping Example

In [7]:
```
logisticsCosts = soup.find_all("span", {"class", "s-item__shipping s-item__logisticsCost"})

for logisticsCost in logisticsCosts:
    print(logisticsCost.text)
```

```
Free shipping
Free shipping
Free shipping
Free shipping
Free shipping
+$29.99 shipping
Free shipping
Free shipping
Free shipping
+$39.10 shipping
+$33.25 shipping
+$27.05 shipping
+$83.00 shipping
+$17.95 shipping
+$26.55 shipping
Free shipping
```

ebay Scraping Example

In [8]:
```
divisions = soup.find_all("div", class_= "s-item__detail s-item__detail--primary")

for division in divisions:
    print(division.text)
```

```
$20.00
or Best Offer
 Sponsored
$276.69
or Best Offer
Free shipping
16 watchers
 Sponsored
$469.99
Buy It Now
Free shipping
Free returns
eBay Refurbished
1,961 sold
 Sponsored
$278.57
or Best Offer
```

ebay Scraping Example

In [8]:
```
divisions = soup.find_all("div", class_= "s-item__detail s-item__detail--primary")

for division in divisions:
    print(division.text)
```

```
$20.00
or Best Offer
 Sponsored
$276.69
or Best Offer
Free shipping
16 watchers
 Sponsored
$469.99
Buy It Now
Free shipping
Free returns
eBay Refurbished
1,961 sold
 Sponsored
$278.57
or Best Offer
```

Worldometers Scraping Example

- Worldometers has a lot of data in table format, the challenge is to store the data from HTML into a Pandas data frame.
- An HTML table has two kinds of cells:
  - Header cells - contains header information (created with the <th> element)
  - Data cells - contains data (created with the <td> element)
- The <tr> tag defines a row in a HTML table.

Worldometers Scraping Example

In [1]:
```
import bs4
import requests
import pandas as pd
```

In [2]:
```
response = requests.get("https://www.worldometers.info/coronavirus/")
soup = bs4.BeautifulSoup(response.text, 'html')
```

Worldometers Scraping Example

In [3]:
```
table_soup = soup.find_all('table')[0]
table_soup
```

Out[3]:
```
<table class="table table-bordered table-hover main_table_countries" id="main_table_countries_today"
00%;margin-top: 0px !important;display:none;">
<thead>
<tr>
<th width="1%">#</th>
<th width="100">Country,<br/>Other</th>
<th width="20">Total<br/>Cases</th>
<th width="30">New<br/>Cases</th>
<th width="30">Total<br/>Deaths</th>
<th width="30">New<br/>Deaths</th>
<th width="30">Total<br/>Recovered</th>
<th width="30">New<br/>Recovered</th>
<th width="30">Active<br/>Cases</th>
<th width="30">Serious,<br/>Critical</th>
<th width="30">Tot Cases/<br/>1M pop</th>
<th width="30">Deaths/<br/>1M pop</th>
<th width="30">Total<br/>Tests</th>
<th width="30">Tests/<br/>
<nobr>1M pop</nobr>
```

In [4]:
```
type(table_soup)
```

Out[4]:
```
bs4.element.Tag
```

Worldometers Scraping Example

In [5]:
```
# Get all the tags that define a header cell in a table
title_data = table_soup.find_all('th')

# Get only the text and store it in the list in Python
title_text = [title.text.strip() for title in title_data]
print(title_text)
```

```
['#', 'Country,Other', 'TotalCases', 'NewCases', 'TotalDeaths', 'NewDeaths', 'TotalRecovered', 'NewRecovered', 'Activ
eCases', 'Serious,Critical', 'Tot\xa0Cases/1M pop', 'Deaths/1M pop', 'TotalTests', 'Tests/\n1M pop', 'Population', 'C
ontinent', '1 Caseevery X ppl', '1 Deathevery X ppl', '1 Testevery X ppl', 'New Cases/1M pop', 'New Deaths/1M pop',
'Active Cases/1M pop']
```

In [6]:
```
rows = table_soup.find_all('tr')
```

Worldometers Scraping Example

In [7]:
```
# Create a list to store the data in each row
# For each row in the list of rows
# Scrape the cell data on each row and store it in each_row_data
# Strip surrounding characters and get only text and store in row_data

row_data = []

for row in rows:
    each_row_data = row.find_all('td')
    row_data_1 = [data.text.strip() for data in each_row_data]
    row_data.append(row_data_1)
```

Worldometers Scraping Example

```
In [8]: # The empty lists have to be removed
        row_data
```

```
Out[8]: [[],
         ['',
          'North America',
          '128,020,172',
          '',
          '1,644,502',
          '',
          '123,658,790',
          '+241',
          '2,716,880',
          '6,536',
          '',
          '',
          '',
          '',
          '',
          'North America',
          '',
```

Worldometers Scraping Example

In [9]:
```
# Only get the list is not empty
row_data = [alpha for alpha in row_data if alpha != []]
```

In [10]:
```
row_data
```

Out[10]:
```
[['',
  'North America',
  '128,020,172',
  '',
  '1,644,502',
  '',
  '123,658,790',
  '+241',
  '2,716,880',
  '6,536',
  '',
  '',
  '',
  '',
  '',
  'North America',
  '',
  '',
```

Worldometers Scraping Example

```
In [11]: df = pd.DataFrame(data=row_data,columns=title_text)
         display(df.iloc[:10] )
```

	#	Country,Other	TotalCases	NewCases	TotalDeaths	NewDeaths	TotalRecovered	NewRecovered	ActiveCases	Serious,Critical	...	TotalTests	Tests/\n1 pc
0		North America	128,020,172		1,644,502		123,658,790	+241	2,716,880	6,536	...		
1		Asia	220,719,754	+33	1,549,619		204,304,984	+40,819	14,865,151	15,146	...		
2		Europe	249,919,346	+73	2,070,549		246,008,706	+1,477	1,840,091	5,482	...		
3		South America	68,975,740		1,360,574		66,517,342	+1,480	1,097,824	10,069	...		
4		Oceania	14,635,319		30,289		14,500,581		104,449	49	...		
5		Africa	12,837,218		258,829		12,088,233		490,156	547	...		
6			721		15		706		0	0	...		
7		World	695,108,270	+106	6,914,377	0	667,079,342	+44,017	21,114,551	37,829	...		
8	1	USA	108,296,305		1,174,580		106,111,280		1,010,445	1,370	...	1,186,093,362	3,542,6
9	2	India	44,997,710		531,930		44,465,194		586	N/A	...	930,797,975	661,7

10 rows × 22 columns

Scrapy Example

1. Create a new Scrapy project
2. Write a spider to crawl a site and extract data
3. Export the scraped data using the command line
4. Change the spider to recursively follow links
5. Use spider arguments

https://docs.scrapy.org/en/latest/intro/tutorial.html

Scrapy Example

- Create the project in the terminal
  - scrapy startproject tutorial
- This will create tutorial directory with these contents:

```
tutorial/
    scrapy.cfg            # deploy configuration file

    tutorial/             # project's Python module, you'll import your code from here
        __init__.py

        items.py          # project items definition file

        middlewares.py    # project middlewares file

        pipelines.py      # project pipelines file

        settings.py       # project settings file

        spiders/          # a directory where you'll later put your spiders
            __init__.py
```

Scrapy Example (Build a spider)

- Spiders are classes that you define and Scrapy uses to scrape information from a website (or a group of websites).
- They must subclass Spider and define the initial requests to make, optionally how to follow links in the pages, and how to parse the downloaded page content to extract data.
- Save it in a file named quotes_spider.py under the tutorial/spiders directory in your project:

Scrapy Example (Build a spider)

```python
from pathlib import Path

import scrapy


class QuotesSpider(scrapy.Spider):
    name = "quotes"

    def start_requests(self):
        urls = [
            "https://quotes.toscrape.com/page/1/",
            "https://quotes.toscrape.com/page/2/",
        ]
        for url in urls:
            yield scrapy.Request(url=url, callback=self.parse)

    def parse(self, response):
        page = response.url.split("/")[-2]
        filename = f"quotes-{page}.html"
        Path(filename).write_bytes(response.body)
        self.log(f"Saved file {filename}")
```

Scrapy Example (Build a spider)

- Our Spider subclasses scrapy.Spider and defines some attributes and methods:
- name: identifies the Spider. It must be unique within a project, that is, you can’t set the same name for different Spiders.
- start_requests(): must return an iterable of Requests (you can return a list of requests or write a generator function) which the Spider will begin to crawl from. Subsequent requests will be generated successively from these initial requests.
- parse(): method usually parses the response, extracting the scraped data as dicts and also finding new URLs to follow and creating new requests (Request) from them.

How to run our spider?

- From project's top level directory and run:

  scrapy crawl quotes

- This command runs the spider. You will get an output similar to this:

```
... (omitted for brevity)
2016-12-16 21:24:05 [scrapy.core.engine] INFO: Spider opened
2016-12-16 21:24:05 [scrapy.extensions.logstats] INFO: Crawled 0 pages (at 0 pages/min), scrap
2016-12-16 21:24:05 [scrapy.extensions.telnet] DEBUG: Telnet console listening on 127.0.0.1:60
2016-12-16 21:24:05 [scrapy.core.engine] DEBUG: Crawled (404) <GET https://quotes.toscrape.com/
2016-12-16 21:24:05 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://quotes.toscrape.com/
2016-12-16 21:24:05 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://quotes.toscrape.com/
2016-12-16 21:24:05 [quotes] DEBUG: Saved file quotes-1.html
2016-12-16 21:24:05 [quotes] DEBUG: Saved file quotes-2.html
2016-12-16 21:24:05 [scrapy.core.engine] INFO: Closing spider (finished)
...
```

Scrapy Example

- Two new files have been created in the directory : *quotes-1.html* and *quotes-2.html*, with the content for the respective URLs, as our parse method instructs.
- Extract Data:
  - scrapy shell 'https://quotes.toscrape.com/page/1/'

```
[ ... Scrapy log here ... ]
2016-09-19 12:09:27 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://quotes.toscrape.c
[s] Available Scrapy objects:
[s]   scrapy     scrapy module (contains scrapy.Request, scrapy.Selector, etc)
[s]   crawler    <scrapy.crawler.Crawler object at 0x7fa91d888c90>
[s]   item       {}
[s]   request    <GET https://quotes.toscrape.com/page/1/>
[s]   response   <200 https://quotes.toscrape.com/page/1/>
[s]   settings   <scrapy.settings.Settings object at 0x7fa91d888c10>
[s]   spider     <DefaultSpider 'default' at 0x7fa91c8af990>
[s] Useful shortcuts:
[s]   shelp()           Shell help (print this help)
[s]   fetch(req_or_url) Fetch request (or URL) and update local objects
[s]   view(response)    View response in a browser
```

Each quote in https://quotes.toscrape.com is represented by HTML elements that look like this:

```
<div class="quote">
    <span class="text">“The world as we have created it is a process of our
    thinking. It cannot be changed without changing our thinking.”</span>
    <span>
        by <small class="author">Albert Einstein</small>
        <a href="/author/Albert-Einstein">(about)</a>
    </span>
    <div class="tags">
        Tags:
        <a class="tag" href="/tag/change/page/1/">change</a>
        <a class="tag" href="/tag/deep-thoughts/page/1/">deep-thoughts</a>
        <a class="tag" href="/tag/thinking/page/1/">thinking</a>
        <a class="tag" href="/tag/world/page/1/">world</a>
    </div>
</div>
```

Having figured out how to extract each bit, we can now iterate over all the quotes elements and put them together into a Python dictionary:

```
>>> for quote in response.css("div.quote"):
...     text = quote.css("span.text::text").get()
...     author = quote.css("small.author::text").get()
...     tags = quote.css("div.tags a.tag::text").getall()
...     print(dict(text=text, author=author, tags=tags))
...
{'text': '“The world as we have created it is a process of our thinking. It cannot be chang
{'text': '“It is our choices, Harry, that show what we truly are, far more than our abiliti
...
```

Legalities of Web Scraping

- There are three types of intellectual properties:
  - Copyrights: Protect works of authorship, including novels, poetry, art, and music. Circle C ©
  - Patents: Protect inventions that are new and nonobvious over existing technology.
  - Trademarks: Anything that represents a brand. These symbols are signifying trademark rights: Circle R (®), TM, and SM.

Trademark

- According to the US Patent and Trademark Office:
  - A trademark is a word, phrase, symbol, and/or design that identifies and distinguishes the source of the goods of one party from those of others. A service mark is a word, phrase, symbol, and/or design that identifies and distinguishes the source of a service rather than goods. The term “trademark” is often used to refer to both trademarks and service marks.
- Unlike with patents, the ownership of a trademark depends heavily on the context in which it is used.
- For example, if we wish to publish a blog post with an accompanying picture of the Tesla logo, we could do that (as long as our post wasn’t implying that our blog post was sponsored by, or published by, Tesla). If we wanted to manufacture a T-shirt with the same Tesla logo displayed on it, that is a trademark infringement.

Copyright

- We should never directly publish copyrighted material without permission from the original author or copyright holder.
- *Copyright protection extends to **creative works only**. It does not cover statistics or facts. Fortunately, much of what web scrapers are after are statistics and facts. [Mitchell '18]*
- Example: The poetry published online, by an individual who is alive. The poetry itself in a raw format is a copy righted material. However, its word counts, sentiment, and other quantitative information are not copyrighted material.

Trespass to Chattels

- It applies when we access to property that we are not allowed to access it.
- According to https://trespass.uslegal.com/trespass-to-chattel:

> Trespass to chattels is a particular type of trespass whereby a person has intentionally interfered with another person's lawful possession of a chattel. A chattel refers to the movable or immovable personal property of an individual except real estate. Generally, the basic elements of a claim of trespass to chattels are lack of an owner's consent to trespass, interference with possessory interest, and intention of the trespasser

Trespass to Chattels - Violation

- Three criteria need to be met (all together) for web scrapers to violate Trespass to chattels:
  - Lack of consent: Web servers are open to everyone, they are generally "giving consent" to web scrapers as well. However, many websites' Terms of Service agreements specifically prohibit the use of scrapers in a file called robot.txt. In addition, any explicit notices delivered to you from the web server revoke this consent.
  - Actual harm: Servers are expensive properties. Besides, if the scrapers take a website down, or limit its ability to serve other users, this can add to the "harm" you cause.
  - Intentionality: If you're writing the code that performs some harm, such as DDoS attack.

MET CS 688 Statistics & Probability I

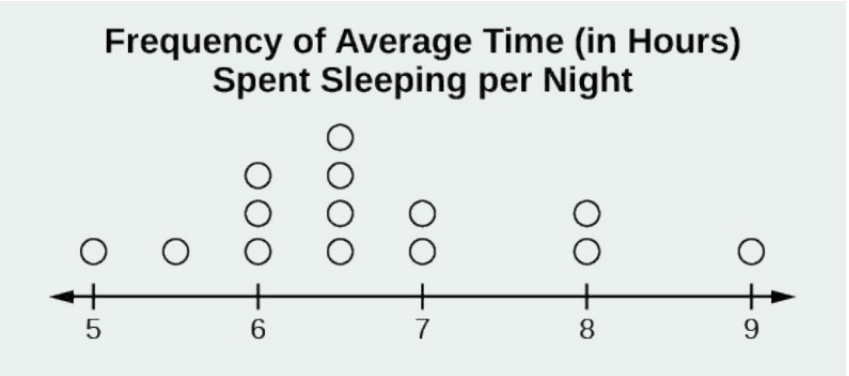
Leila Ghaedi – Fall 2024

Creator: cemagraphics | Credit: Getty Images/iStockphoto

Descriptive Statistics

Descriptive Statistics Example

- Imagine we ask 14 people to write down the average time (in hours, to the nearest half hour) they sleep per night.
- 5, 5.5, 6, 6, 6, 6.5, 6.5, 6.5, 6.5, 7, 7, 8, 8, 9



Descriptive vs Inferential Statistics

- Descriptive Statistics
  - Organizing and summarizing data is called descriptive statistics. Two ways to summarize data are by graphing and by using numbers (for example, finding the average or probability).
- Inferential Statistics
  - Uses probability to determine how confident we can be that our conclusions are correct. Effective interpretation of data (inference) is based on good procedures for producing data and thoughtful examination of the data.

Descriptive vs Inferential Statistics

- **Descriptive statistics** state facts and proven outcomes from a population.
- **Inferential statistics** analyze samplings to make predictions about larger populations.

Probability

- Probability is a mathematical tool used to study randomness. It deals with the chance (the likelihood) of an event occurring.
- For example, if you toss a fair coin four times, the outcomes may not be two heads and two tails. However, if you toss the same coin 4,000 times, the outcomes will be close to half heads and half tails. The expected theoretical probability of heads in any one toss is 1 2 or 0.5. Even though the outcomes of a few repetitions are uncertain, there is a regular pattern of outcomes when there are many repetitions.

Population vs Sample

- In statistics, we generally want to study a population. You can think of a population as a collection of persons, things, or objects under study.
- To study the population, we select a sample. The idea of sampling is to select a portion (or subset) of the larger population and study that portion (the sample) to gain information about the population.
- Data are the result of sampling from a population.
- Because it takes a lot of time and money to examine an entire population, sampling is a very practical technique.
- In presidential elections, opinion poll samples of 1,000–2,000 people are taken. The opinion poll is supposed to represent the views of the people in the entire country.

Representative Sample

- A statistic is a number that represents a property of the sample.
- For example, if we consider one math class to be a sample of the population of all math classes, then the average number of points earned by students in that one math class is an example of a statistic.
- One of the main concerns in the field of statistics is how accurately a statistic estimates a parameter. The accuracy really depends on how well the sample represents the population. The sample must contain the characteristics of the population in order to be a representative sample.

Variable, Value

- A variable, is a characteristic or measurement that can be determined for each member of a population.
- Variables may be numerical or categorical.
  - Numerical Variables take on values with equal units such as weight in pounds and time in hours.
  - Categorical Variables place the person or thing into a category.

Categorical (Nominal vs Ordinal)

- There are two types of categorical variable:
  - Nominal (There is no intrinsic ordering to its categories, such as eye color, marital status)
  - Ordinal (ordinal variable has a clear ordering such as satisfaction rating [extremely dislike, dislike, neutral, like, extremely like] or income brackets [low income, middle income, high income]

Control, dependent and independent variable

- In an experiment:
  - Independent variable: is changed to test the dependent variable.
  - Dependent variable: is affected by the changes made to the independent variable.
  - Control Variable: is unchanged and constant.

Descriptive Statistics

1. Display data graphically and interpret graphs: histograms, and box plots.
2. Recognize, describe, and calculate the measures of location of data: quartiles and percentiles.
3. Recognize, describe, and calculate the measures of the center of data: mean, median, and mode.
4. Recognize, describe, and calculate the measures of the spread of data: variance, standard deviation, and range.

Histogram

- A histogram consists of adjoining boxes. It has both a horizontal axis and a vertical axis. The horizontal axis is labeled with what the data represents (for instance, distance from your home to school). The vertical axis is labeled either frequency or relative frequency (or percent frequency or probability).
- The histogram can give you the shape of the data, the center, and the spread of the data.
- The relative frequency is equal to the frequency for an observed value of the data divided by the total number of data values in the sample.

  f = frequency

  n = total number of data values

  RF = relative frequency, RF = f / n

Histogram

- pip install matplotlib

In [1]:
```
from matplotlib import pyplot as plt
import numpy as np
```

In [2]:
```
# creating dataset
# this is the number of times each person in our sample
# called customer service of their mobile provider in the past year

data = np.array([5, 6, 2, 4, 2, 3,
                 3, 4, 1, 6, 7, 5,
                 8, 4, 3, 5, 2, 4,
                 1, 2, 1, 2, 2, 3,
                 1, 3, 2, 4, 2, 1])
```

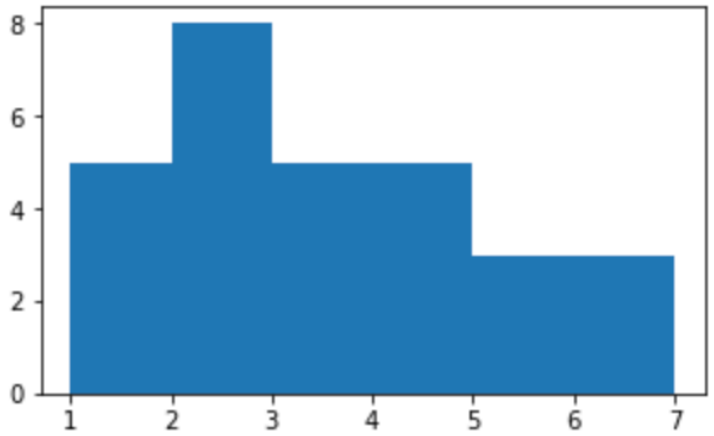
Histogram

- The plt.subplots() function in Matplotlib is used to create a figure and a set of subplots. It takes the following arguments:
- nrows: The number of rows of subplots.
- ncols: The number of columns of subplots.
- sharex: Whether to share the x-axis between all subplots.
- sharey: Whether to share the y-axis between all subplots.
- grid: Whether to add a grid to each subplot.
- **The subplots() function returns a tuple of two objects:**
- fig: A figure object.
- axs: An array of axes objects, one for each subplot.

Histogram

In [3]:
```
# Creating histogram
fig, axs = plt.subplots(figsize =(5, 3))
axs.hist(data, bins = [1, 2, 3, 4, 5, 6, 7])

# show plot
plt.show()
```

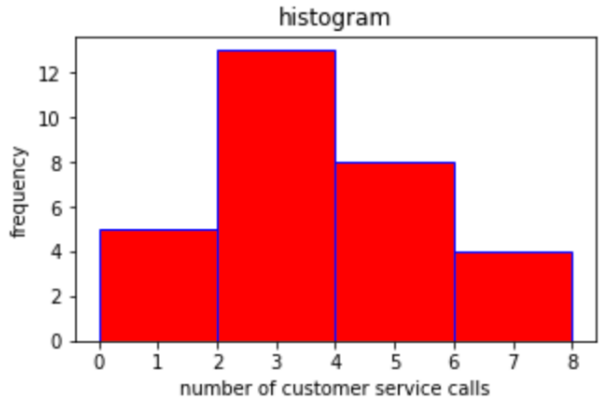


Histogram

In [4]:
```
# Creating histogram
fig, axs = plt.subplots(figsize =(5, 3))
axs .hist(data, bins = [0, 2, 4, 6, 8], color = "red", ec="blue" )

# add labels and titles
plt.xlabel("number of customer service calls")
plt.ylabel("frequency")
plt.title('histogram')

# show plot
plt.show()
```



Histogram

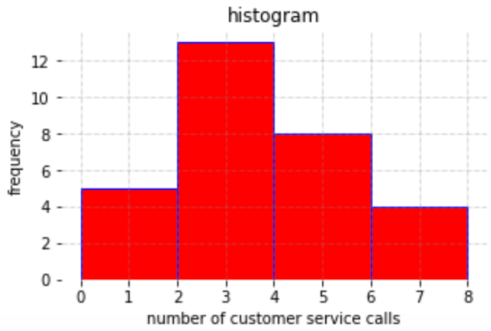
In [5]:
```
# Creating histogram
fig,  axs = plt.subplots(figsize =(5, 3))
axs.hist(data, bins = [0, 2, 4, 6, 8], color = "red", ec="blue" )

# add gridlines
axs.grid(visible = True, color ='grey', linestyle ='-.', linewidth = 0.5, alpha = 0.6)

#add labels and titles
plt.xlabel("number of customer service calls")
plt.ylabel("frequency")
plt.title('histogram')

# remove axes splines
for p in ['top', 'bottom', 'left', 'right']:
    axs.spines[p].set_visible(False)

# show plot
plt.show()
```



Measures of center of the data

- The "center" of a data set is a way of describing location.
  1. Mean (Average)
  2. Median
  3. Mode

Measures of center of the data

- To calculate the mean weight of 50 people, add the 50 weights together and divide by 50.
- To find the median weight of the 50 people, order the data and find the number that splits the data into two equal parts.
- The median is generally a better measure of the center when there are extreme values or outliers because it is not affected by the precise numerical values of the outliers. The mean is the most common measure of the center.

Measures of center of the data

- Another measure of the center is the mode. The mode is the most frequent value.

Means

- Arithmetic Mean $$\bar{X} = \frac{\sum_{i=1}^{n} x_i}{n}$$
- Harmonic Mean $$\frac{n}{\frac{1}{x_1} + \frac{1}{x_2} + \ldots + \frac{1}{x_n}} = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$$
- Geometrical Mean $$GM = \sqrt[n]{x_1 . x_2 . x_3 \ldots x_n}$$

Means

- Arithmetic Mean $$\bar{X} = \frac{\sum_{i=1}^{n} x_i}{n}$$
- Harmonic Mean

  Good for dealing with outlier.

  e.g. calculating the speed of moving object, profit of a company in consecutive years, …
- Geometrical Mean

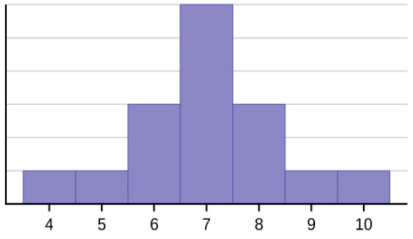
  Very sensitive to outlier and zero

  e.g. compare rooms with their capacity (width, height and length)

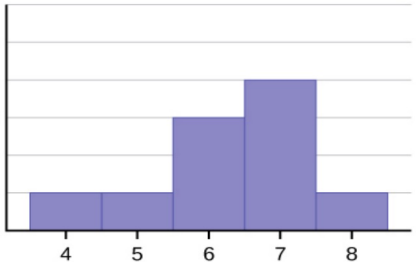
Skewness

- Symmetrical distribution: In a perfectly symmetrical distribution, the mean and the median are the same.
- Skewed to the left:
  - Mean < Median < Mode
- Skewed to the right:
  - Mode < Median < Mean

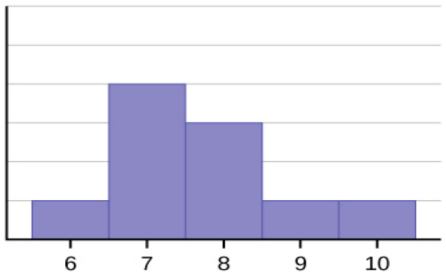
Symmetrical Distribution



Skewed to the left



Skewed to the right



Measures of center of the data for the example data set

```
In [6]: np.mean(data)
Out[6]: 3.2666666666666666

In [7]: np.median(data)
Out[7]: 3.0

In [8]: st.mode(data)
Out[8]: ModeResult(mode=array([2]), count=array([8]))
```

Quartiles and Percentiles

- The common measures of location are quartiles and percentiles Quartiles are special percentiles.
- The first quartile, Q1, is the same as the 25th percentile, and the third quartile, Q3, is the same as the 75th percentile.
- The median, M, is called both the second quartile (Q2) and the 50th percentile.
- To calculate quartiles and percentiles, the data must be ordered from smallest to largest. Quartiles divide ordered data into quarters. Percentiles divide ordered data into hundredths.

Quartiles and Percentiles for the example data set

```
In [9]: print("data : ", data)
        print("Minimum of Data:       ", np.min(data))
        print("Q1 quantile of data :  ", np.quantile(data, .25))
        print("Q2 quantile of data :  ", np.quantile(data, .50))
        print("Q3 quantile of data :  ", np.quantile(data, .75))
        print("Maximum of Data:       ", np.max(data))
```

```
data :  [5 6 2 4 2 3 3 4 1 6 7 5 8 4 3 5 2 4 1 2 1 2 2 3 1 3 2 4 2 1]
Minimum of Data:       1
Q1 quantile of data :  2.0
Q2 quantile of data :  3.0
Q3 quantile of data :  4.0
Maximum of Data:       8
```

Measures of the Spread of the Data

- An important characteristic of any set of data is the variation in the data. In some data sets, the data values are concentrated closely near the mean; in other data sets, the data values are more widely spread out from the mean.
- The most common measure of variation, or spread, is the standard deviation.
- The standard deviation is a number that measures how far data values are from their mean

Calculating Standard Deviation

To calculate the standard deviation, we need to calculate the variance first. The variance is the average of the squares of the deviations (the $x - \bar{x}$ values for a sample, or the $x - \mu$ values for a population). The symbol σ^2 represents the population variance; the population standard deviation σ is the square root of the population variance.

Formulas for the Sample Standard Deviation

- $s = \sqrt{\frac{\Sigma(x - \bar{x})^2}{n - 1}}$ or $s = \sqrt{\frac{\Sigma f(x - \bar{x})^2}{n - 1}}$
- For the sample standard deviation, the denominator is ***n* - 1**, that is the sample size MINUS 1.

Formulas for the Population Standard Deviation

- $\sigma = \sqrt{\frac{\Sigma(x - \mu)^2}{N}}$ or $\sigma = \sqrt{\frac{\Sigma f(x - \mu)^2}{N}}$
- For the population standard deviation, the denominator is N, the number of items in the population.

Standard Deviation
for the example data set

In [10]:
```
print("Standard Deviation of Data:", np.std(data))
print("Variance of Data:          ", np.var(data))

print ("Is Variance=(Standard Deviation)**2 :" ,(((np.std(data))**2)== np.var(data)))
```

```
Standard Deviation of Data: 1.8427033281447003
Variance of Data:           3.3955555555555548
Is Variance=(Standard Deviation)**2 : True
```

Box and Whisker Plots

- Box plots give a good graphical image of the concentration of the data.
- They also show how far the extreme values are from most of the data.
- Two versions of box plot
  - A box plot is constructed from five values: the minimum value, the first quartile, the median, the third quartile, and the maximum value.
  - Box plots that have dots marking outlier values. In those cases, the whiskers are not extending to the minimum and maximum values, instead using interquartile range.

Interquartile Range and Outliers

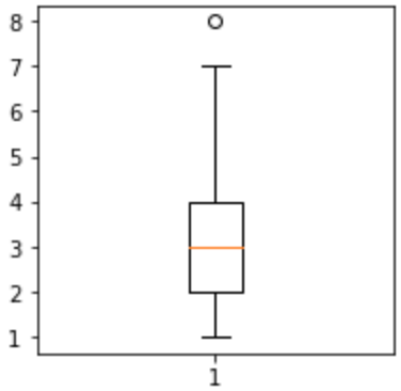
- The interquartile range is a number that indicates the spread of the middle half or the middle 50% of the data. It is the difference between the third quartile (Q3) and the first quartile (Q1).
- IQR = Q3 – Q1
- The IQR can help determine the outliers. A value is suspected to be a potential outlier if it is less than (1.5)(IQR) below the first quartile or more than (1.5)(IQR) above the third quartile. Potential outliers always require further investigation.
- In Python: The minimum and maximum values are defined as Q1–1.5 * IQR and Q3 + 1.5 * IQR respectively. Any points that fall outside of these limits are referred to as outliers.

Box and Whisker Plot for the example data set

In [11]:
```
fig = plt.figure(figsize =(3, 3))

# Creating plot
plt.boxplot(data)

# show plot
plt.show()
```



Violin Plots

- Like histograms and box plots, violin plots show an abstract representation of the probability distribution of the sample.
- Rather than showing counts of data points that fall into bins or order statistics, violin plots use kernel density estimation (KDE) to compute an empirical distribution of the sample.
- More advance violin plots with seaborn:
- https://seaborn.pydata.org/generated/seaborn.violinplot.html

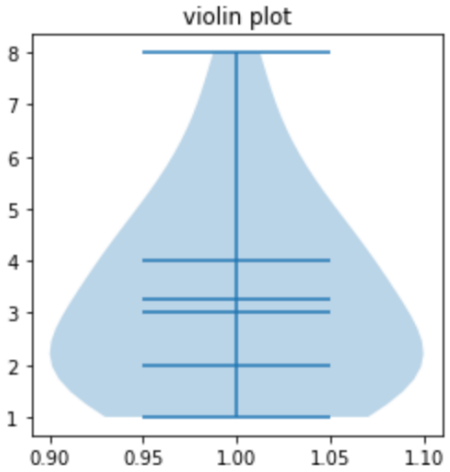
Violin Plot
for the example data set

In [12]:
```
fig, axs = plt.subplots(figsize=(4, 4))

axs.violinplot(data,  points=30, widths=0.2,
               showmeans=True, showextrema=True, showmedians=True,
               quantiles=[0.25, 0.75])

axs.set_title('violin plot')
```

Out[12]:
```
Text(0.5, 1.0, 'violin plot')
```



References

Müller, Andreas C.; Guido, Sarah. Introduction to Machine Learning with Python. O'Reilly Media. Kindle Edition.
Mitchell, Ryan. Web Scraping with Python. O'Reilly Media. Kindle Edition.

[Hey ’09] Hey, T., Tansley, S., & Tolle, K. M. (2009). The Fourth Paradigm: Data-Intensive Scientific Discovery (Vol. 1). Redmond, WA: Microsoft Research.

[Mitchell ’97] Mitchell, T.M. (1997) Machine Learning. McGraw Hill.

Illowsky, Barbara; Dean, Susan. Introductory Statistics . XanEdu Publishing Inc

[Géron ‘17] Géron, A. (2017). Hands-on Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O’Reilly.

[Provost ‘13] Provost, F., & Fawcett, T. (2013). Data Science for Business: What you need to know about data mining and data-analytic thinking. " O'Reilly Media, Inc.".

[Russell ’09] Russell, S., Norvig, P. (2019). Artificial intelligence: A Modern Approach (4th Edition). Prentice Hall.

[Chapmann ’17] Chapmann, J. (2017) Machine Learning: Fundamental Algorithms for Supervised and Unsupervised Learning With Real-World Applications.

[Knuth ‘97] Knuth, D. E. (1997). The Art of Computer Programming (Vol. 1). Pearson Education.

[Bhargava ‘16] Bhargava, A. (2016). Grokking Algorithms: An illustrated guide for programmers and other curious people. Manning Publications Co.

[Han ‘11] Han, J., Pei, J., & Kamber, M. (2011). *Data Mining: Concepts and Techniques*. 3rd Edition. Elsevier.