python™

FROM NOTHING TO ANYTHING

Murugan, Naveenkumar
HEWLETT PACKARD

Guide to Using Jupyter Notebooks

For a complete User Manual check out the Bryn Mawr College Computer Science Guide.

Numbers and more in Python!

In this lecture, we will learn about numbers in Python and how to use them.

We'll learn about the following topics:

```
1.) Types of Numbers in Python
2.) Basic Arithmetic
3.) Differences between Python 2 vs 3 in division
4.) Object Assignment in Python
```

Types of numbers

Python has various "types" of numbers (numeric literals). We'll mainly focus on integers and floating point numbers.

Integers are just whole numbers, positive or negative. For example: 2 and -2 are examples of integers.

Floating point numbers in Python are notable because they have a decimal point in them, or use an exponential (e) to define the number. For example 2.0 and -2.1 are examples of floating point numbers. 4E2 (4 times 10 to the power of 2) is also an example of a floating point number in Python.

Throughout this course we will be mainly working with integers or simple float number types.

Here is a table of the two main types we will spend most of our time working with some examples:

Examples	Number "Type"
1,2,-5,1000	Integers
1.2,-0.5,2e2,3E2	Floating-point numbers

Now let's start with some basic arithmetic.

Basic Arithmetic

In [1]:
```
# Addition
2+1
```

Out[1]: 3

In [2]:
```
# Subtraction
2-1
```

Out[2]: 1

In [3]:
```
# Multiplication
2*2
```

Out[3]: 4

In [1]:
```
# Division
3/2
```

Out[1]: 1.5

Python 2 Alert!

In Python 2, the / symbol performs what is known as "*classic*" division, this means that the decimal points are truncated (cut off). In Python 3 however, a single / performs "*true*" division. So you would get 1.5 if you had inputed 3/2 in Python 3.

So what do we do if we are using Python 2 to avoid this?

There are two options:

Specify one of the numbers to be a float:

In [11]:
```
# Specifying one of the numbers as a float
3.0/2
```

Out[11]: 1.5

In [12]:
```
# Works for either number
3/2.0
```

Out[12]: 1.5

We could also "cast" the type using a function that basically turns integers into floats. This function, unsurprisingly, is called float().

In [14]:
```
# We can use this float() function to cast integers as floats:
float(3)/2
```

Out[14]: 1.5

Arithmetic continued

In [16]:
```
# Powers
2**3
```

Out[16]: 8

In [17]:
```
# Can also do roots this way
4**0.5
```

Out[17]: 2.0

In [18]:
```
# Order of Operations followed in Python
2 + 10 * 10 + 3
```

Out[18]: 105

In [19]:
```
# Can use parenthesis to specify orders
(2+10) * (10+3)
```

Out[19]: 156

Python executes expression based on PEMDAS order of operations

P - Parenthesis

E - Exponential

M - Multiplication

D - Division

A - Addition

S - Subtraction

Variable Assignments

Now that we've seen how to use numbers in Python as a calculator let's see how we can assign names and create variables.

We use a single equals sign to assign labels to variables. Let's see a few examples of how we can do this.

In [37]:
```
# Let's create an object called "a" and assign it the number 5
a = 5
```

Now if I call *a* in my Python script, Python will treat it as the number 5.

In [38]:
```
# Adding the objects
a+a
```

Out[38]: 10

What happens on reassignment? Will Python let us write it over?

In [39]:
```
# Reassignment
a = 10
```

In [40]:
```
# Check
a
```

Out[40]: 10

Yes! Python allows you to write over assigned variable names. We can also use the variables themselves when doing the reassignment. Here is an example of what I mean:

In [41]:
```
# Check
a
```

Out[41]: 10

In [42]:
```
# Use A to redefine A
a = a + a
```

In [43]:
```
# Check
a
```

Out[43]: 20

The names you use when creating these labels need to follow a few rules:

1. Names can not start with a number.
2. There can be no spaces in the name, use _ instead.
3. Can't use any of these symbols :'",<>/?|\()!@#$%^&*~-+
3. It's considered best practice (PEP8) that the names are lowercase.

Using variable names can be a very useful way to keep track of different variables in Python. For example:

In [2]:
```
# Use object names to keep better track of what's going on in your code!
my_income = 100

tax_rate = 0.1

my_taxes = my_income*tax_rate
```

In [3]:
```
# Show my taxes!
my_taxes
```

Out[3]: 10.0

Up next we'll learn about Strings!

Strings

Strings are used in Python to record text information, such as name. Strings in Python are actually a *sequence*, which basically means Python keeps track of every element in the string as a sequence. For example, Python understands the string "hello' to be a sequence of letters in a specific order. This means we will be able to use indexing to grab particular letters (like the first letter, or the last letter).

This idea of a sequence is an important one in Python and we will touch upon it later on in the future.

In this lecture we'll learn about the following:

```
1.) Creating Strings
2.) Printing Strings
3.) Differences in Printing in Python 2 vs 3
4.) String Indexing and Slicing
5.) String Properties
6.) String Methods
7.) Print Formatting
```

Creating a String

To create a string in Python you need to use either single quotes or double quotes. For example:

In [1]:
```
# Single word
'hello'
```

Out[1]:
```
'hello'
```

In [2]:
```
# Entire phrase
'This is also a string'
```

Out[2]:
```
'This is also a string'
```

In [3]:
```
# We can also use double quote
"String built with double quotes"
```

Out[3]:
```
'String built with double quotes'
```

In [4]:
```
# Be careful with quotes!
' I'm using single quotes, but will create an error'
```

```
  File "<ipython-input-4-6565b0b7b5e3>", line 2
    ' I'm using single quotes, but will create an error'
       ^
SyntaxError: invalid syntax
```

The reason for the error above is because the single quote in I'm stopped the string. You can use combinations of double and single quotes to get the complete statement.

In [10]:
```
"Now I'm ready to use the single quotes inside a string!"
```

Out[10]:
```
"Now I'm ready to use the single quotes inside a string!"
```

Now let's learn about printing strings!

Printing a String

Using Jupyter notebook with just a string in a cell will automatically output strings, but the correct way to display strings in your output is by using a print function.

We can use a print statement to print a string.

In [26]:
```
print('This is the first string am printing')
```

```
This is the first string am printing
```

Python 2 Alert!

Something to note. In Python 3, print is a function, not a statement. So you would print statements like this: print('Hello World')

If you want to use this functionality in Python2, you can import form the **future** module.

A word of caution, after importing this you won't be able to choose the print statement method anymore. So pick whichever one you prefer depending on your Python installation and continue on with it.

In [32]:
```
# To use print function from Python 3 in Python 2
from __future__ import print_function

print('Hello World')
```

```
Hello World
```

String Basics

We can also use a function called len() to check the length of a string!

In [33]:
```
len('Hello World')
```

Out[33]:
```
11
```

String Indexing

We know strings are a sequence, which means Python can use indexes to call parts of the sequence. Let's learn how this works.

In Python, we use brackets [] after an object to call its index. We should also note that indexing starts at 0 for Python. Let's create a new object called s and the walk through a few examples of indexing.

In [1]:
```
# Assign s as a string
s = 'Hello World'
```

In [35]:
```
#Check
s
```

Out[35]:
```
'Hello World'
```

In [36]:
```
# Print the object
print(s)
```

```
Hello World
```

Let's start indexing!

In [21]:
```
# Show first element (in this case a letter)
s[0]
```

Out[21]:
```
'H'
```

In [22]:
```
s[1]
```

Out[22]:
```
'e'
```

In [23]:
```
s[2]
```

Out[23]:
```
'l'
```

We can use a : to perform *slicing* which grabs everything up to a designated point. For example:

In [24]:
```
# Grab everything past the first term all the way to the length of s which
 is len(s)
s[1:]
```

Out[24]:
```
'ello World'
```

In [25]:
```
# Note that there is no change to the original s
s
```

Out[25]:
```
'Hello World'
```

In [26]:
```
# Grab everything UP TO the 3rd index
s[:3]
```

Out[26]:
```
'Hel'
```

Note the above slicing. Here we're telling Python to grab everything from 0 up to 3. It doesn't include

the 3rd index. You'll notice this a lot in Python, where statements and are usually in the context of "up to, but not including".

In [27]:
```
#Everything
s[:]
```

Out[27]:
```
'Hello World'
```

We can also use negative indexing to go backwards.

In [28]:
```
# Last letter (one index behind 0 so it loops back around)
s[-1]
```

Out[28]:
```
'd'
```

In [29]:
```
# Grab everything but the last letter
s[:-1]
```

Out[29]:
```
'Hello Worl'
```

We can also use index and slice notation to grab elements of a sequence by a specified step size (the default is 1). For instance we can use two colons in a row and then a number specifying the frequency to grab elements. For example:

In [42]:
```
# Grab everything, but go in steps size of 1
s[::1]
```

Out[42]:
```
'Hello World'
```

In [46]:
```
# Grab everything, but go in step sizes of 2
s[::2]
```

Out[46]:
```
'HloWrd'
```

In [47]:
```
# We can use this to print a string backwards
s[::-1]
```

Out[47]:
```
'dlroW olleH'
```

String Properties

Its important to note that strings have an important property known as immutability. This means that once a string is created, the elements within it can not be changed or replaced. For example:

In [4]:
```
s='Hello World'
```

In [5]:
```
# Let's try to change the first letter to 'x'
s[0] = 'x'
```

```
---------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call last
)
```

```
<ipython-input-5-3a9c668aa5ab> in <module>()
      1 # Let's try to change the first letter to 'x'
----> 2 s[0] = 'x'

TypeError: 'str' object does not support item assignment
```

Notice how the error tells us directly what we can't do, change the item assignment!

Something we can do is concatenate strings!

In [6]:
```
s
```

Out[6]:
```
'Hello World'
```

In [7]:
```
# Concatenate strings!
s + ' concatenate me!'
```

Out[7]:
```
'Hello World concatenate me!'
```

In [8]:
```
# We can reassign s completely though!
s = s + ' concatenate me!'
```

In [9]:
```
print(s)
```

```
Hello World concatenate me!
```

In [10]:
```
s
```

Out[10]:
```
'Hello World concatenate me!'
```

We can use the multiplication symbol to create repetition!

In [ ]:
```
letter = 'z'
```

In [ ]:
```
letter*10
```

Basic Built-in String methods

Objects in Python usually have built-in methods. These methods are functions inside the object (we will learn about these in much more depth later) that can perform actions or commands on the object itself.

We call methods with a period and then the method name. Methods are in the form:

object.method(parameters)

Where parameters are extra arguments we can pass into the method. Don't worry if the details don't make 100% sense right now. Later on we will be creating our own objects and functions!

Here are some examples of built-in methods in strings:

In [1]:
```
s= 'Hello world!'
```

In [ ]:
```
# Upper Case a string
s.upper()
```

In [ ]:
```
# Lower case
s.lower()
```

In [ ]:
```
# Split a string by blank space (this is the default) and it will return y
ou a list Object
s.split()
```

In [ ]:
```
# Split by a specific element (doesn't include the element that was split
on)
s.split('W')
```

In [12]:
```
# endswith() function or method used to check the last letter of the strin
g
# It returns boolean i.e. True or False
s.endswith('!')
```

Out[12]: True

In [2]:
```
# startswith() function or method used to check the last letter of the str
ing
s.startswith('H')
```

Out[2]: True

In [3]:
```
#capitalize() - Capitalizes the first letter of every statement
s.capitalize()
```

Out[3]: 'Hello world!'

In [4]:
```
# counts the number of letters or strings in the given string
s.count('o')
```

Out[4]: 2

In [5]:
```
# Returns the index of the letter passed to the method
s.index('o')
```

Out[5]: 4

In [7]:
```
# Finds the index or position of the letter passed to the method(First Occ
urance)
s.find('o')
```

Out[7]: 4

In [9]:
```
# Replaces the letter or string passed
s.replace('o','O')
```

Out[9]: 'HellO wOrld!'

In [13]:
```
# Check the variable contains only space or empty
```

```
s=' '
s.isspace()
```

Out[13]: True

In [15]:
```
# Check the variable contains only Numeric
s='3'
s.isnumeric()
```

Out[15]: True

In [25]:
```
# Check the variable contains only Alphabets
s='PYTHON'
s.isalpha()
```

Out[25]: True

In [24]:
```
# Check the variable contains only Alpha Numeric ( Alphabets + Numbers)
s='PYTHON123'
s.isalnum()
```

Out[24]: True

In [21]:
```
#Check the variable contains only
s='412'
s.isdigit()
```

Out[21]: True

There are many more methods than the ones covered here. Visit the advanced String section to find out more!

Print Formatting

We can use the .format() method to add formatted objects to printed string statements.

The easiest way to show this is through an example:

In [27]:
```
print("I'm just string without any variable or formatting")
```

```
I'm just string without any variable or formatting
```

In [37]:
```
# Formatting with numbered index {0} {1}
# It will follow the order of parm passed in format()
First_name = 'Naveenkumar'
Last_name = 'Murugan'
print("My first name is:{0} and My Last Name is:{1}".format(First_name,Las
t_name))
```

```
My first name is:Naveenkumar and My Last Name is:Murugan
```

In [39]:
```
# Formatting with numbered index {0} {1}
# It will defaultly assigns index 0 to First_name and 1 to Last_name
First_name = 'Naveenkumar'
```

```
Last_name = 'Murugan'
print("My first name is:{} and My Last Name is:{}".format(First_name,Last_
name))
```

```
My first name is:Naveenkumar and My Last Name is:Murugan
```

In [40]:
```
# Formatting with Name
# It will defaultly assigns index 0 to First_name and 1 to Last_name
First_name = 'Naveenkumar'
Last_name = 'Murugan'
print("My first name is:{fn} and My Last Name is:{ln}".format(fn=First_nam
e,ln=Last_name))
```

```
My first name is:Naveenkumar and My Last Name is:Murugan
```

In [42]:
```
# Formatting with casting and concatenation
First_name = 'Naveenkumar'
Last_name = 'Murugan'
digit = 123 # Casted to String using str()
print('My first name is: '+ First_name + ' My Last Name is: ' + Last_name
+' Roll number ' + str(digit))
```

```
My first name is: Naveenkumar My Last Name is: Murugan Roll number 123
```

We will revisit this string formatting topic in later sections when we are building our projects!

Next up: Lists!

Lists

Earlier when discussing strings we introduced the concept of a *sequence* in Python. Lists can be thought of the most general version of a *sequence* in Python. Unlike strings, they are mutable, meaning the elements inside a list can be changed!

In this section we will learn about:

```
1.) Creating lists
2.) Indexing and Slicing Lists
3.) Basic List Methods
4.) Nesting Lists
5.) Introduction to List Comprehensions
```

Lists are constructed with brackets [] and commas separating every element in the list.

Let's go ahead and see how we can construct lists!

In [1]:
```
# Assign a list to an variable named my_list
my_list = [1,2,3]
```

We just created a list of integers, but lists can actually hold different object types. For example:

In [2]:
```
my_list = ['A string',23,100.232,'o']
```

Just like strings, the len() function will tell you how many items are in the sequence of the list.

In [3]:
```
len(my_list)
```

Out[3]:
```
4
```

Indexing and Slicing

Indexing and slicing works just like in strings. Let's make a new list to remind ourselves of how this works:

In [4]:
```
my_list = ['one','two','three',4,5]
```

In [5]:
```
# Grab element at index 0
my_list[0]
```

Out[5]:
```
'one'
```

In [6]:
```
# Grab index 1 and everything past it
my_list[1:]
```

Out[6]:
```
['two', 'three', 4, 5]
```

In [7]:
```
# Grab everything UP TO index 3
my_list[:3]
```

Out[7]:
```
['one', 'two', 'three']
```

We can also use + to concatenate lists, just like we did for strings.

In [8]:
```
my_list + ['new item']
```

Out[8]:
```
['one', 'two', 'three', 4, 5, 'new item']
```

Note: This doesn't actually change the original list!

In [9]:
```
my_list
```

Out[9]:
```
['one', 'two', 'three', 4, 5]
```

You would have to reassign the list to make the change permanent.

In [10]:
```
# Reassign
my_list = my_list + ['add new item permanently']
```

In [11]:
```
my_list
```

Out[11]:
```
['one', 'two', 'three', 4, 5, 'add new item permanently']
```

We can also use the * for a duplication method similar to strings:

In [12]:
```
# Make the list double
my_list * 2
```

Out[12]:
```
['one',
 'two',
 'three',
 4,
 5,
 'add new item permanently',
 'one',
 'two',
 'three',
 4,
 5,
 'add new item permanently']
```

In [13]:
```
# Again doubling not permanent
my_list
```

Out[13]:
```
['one', 'two', 'three', 4, 5, 'add new item permanently']
```

Basic List Methods

If you are familiar with another programming language, you might start to draw parallels between arrays in another language and lists in Python. Lists in Python however, tend to be more flexible than arrays in other languages for a two good reasons: they have no fixed size (meaning we don't have to specify how big a list will be), and they have no fixed type constraint (like we've seen above).

Let's go ahead and explore some more special methods for lists:

In [14]:
```
# Create a new list
l = [1,2,3]
```

Use the **append** method to permanently add an item to the end of a list:

In [15]:
```
# Append
l.append('append me!')
```

In [16]:
```
# Show
l
```

Out[16]:
```
[1, 2, 3, 'append me!']
```

Use **pop** to "pop off" an item from the list. By default pop takes off the last index, but you can also specify which index to pop off. Let's see an example:

In [17]:
```
# Pop off the 0 indexed item
l.pop(0)
```

Out[17]:
```
1
```

In [18]:
```
# Show
l
```

Out[18]:
```
[2, 3, 'append me!']
```

In [19]:
```
# Assign the popped element, remember default popped index is -1(last elem
ent)
popped_item = l.pop()
```

In [20]:
```
popped_item
```

Out[20]:
```
'append me!'
```

In [21]:
```
# Show remaining list
l
```

Out[21]:
```
[2, 3]
```

It should also be noted that lists indexing will return an error if there is no element at that index. For example:

In [22]:
```
l[100]
```

```
---------------------------------------------------------------------------
```

```
-
IndexError                                Traceback (most recent call last
)
<ipython-input-22-3e7ce3111e95> in <module>()
----> 1 l[100]

IndexError: list index out of range
```

We can use the **sort** method and the **reverse** methods to also effect your lists:

In [23]:
```
new_list = ['a','e','x','b','c']
```

In [24]:
```
#Show
new_list
```

Out[24]:
```
['a', 'e', 'x', 'b', 'c']
```

In [25]:
```
# Use reverse to reverse order (this is permanent!)
new_list.reverse()
```

In [26]:
```
new_list
```

Out[26]:
```
['c', 'b', 'x', 'e', 'a']
```

In [27]:
```
# Use sort to sort the list (in this case alphabetical order, but for numb
ers it will go ascending)
new_list.sort()
```

In [28]:
```
new_list
```

Out[28]:
```
['a', 'b', 'c', 'e', 'x']
```

In [29]:
```
# What will you do if you need sort the objects in Descending Order
new_list.sort(reverse=True)
print(new_list)
```

```
['x', 'e', 'c', 'b', 'a']
```

In [30]:
```
# count of objects in the sequence of list
li = ['Hello','Buddy','How','Are','You']
li.count('Buddy')
```

Out[30]:
```
1
```

In [31]:
```
# remove()- Removes the item from the list and returns the index of the it
em it removed
li.remove('Hello')
print(li)
```

```
['Buddy', 'How', 'Are', 'You']
```

In [32]:
```
# Insert() - Inserts the string or any object into a list
li.insert(0,'Hello')
print(li)
```

```
['Hello', 'Buddy', 'How', 'Are', 'You']
```

In [33]:
```
# Let's insert number object into the list
li.insert(0,143)
print(li)
```

```
[143, 'Hello', 'Buddy', 'How', 'Are', 'You']
```

In [34]:
```
# Let's clear everything in the list using clear()
li.clear()
print(li)
```

```
[]
```

Nesting Lists

A great feature of of Python data structures is that they support *nesting*. This means we can have data structures within data structures. For example: A list inside a list.

Let's see how this works!

In [35]:
```
# Let's make three lists
lst_1=[1,2,3]
lst_2=[4,5,6]
lst_3=[7,8,9]

# Make a list of lists to form a matrix
matrix = [lst_1,lst_2,lst_3]
```

In [36]:
```
# Show
matrix
```

Out[36]:
```
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Now we can again use indexing to grab elements, but now there are two levels for the index. The items in the matrix object, and then the items inside that list!

In [37]:
```
# Grab first item in matrix object
matrix[0]
```

Out[37]:
```
[1, 2, 3]
```

In [38]:
```
# Grab first item of the first item in the matrix object
matrix[0][0]
```

Out[38]:
```
1
```

Copying list (Shallow Copy Vs Deep Copy)

Copying Object with '=' sign

Both the source and copied objects shares the same memory. so, when you change an element of source or copy that will be affected in both object

In [10]:
```
my_list = [1,2,3,4]
id(my_list)
```

Out[10]: 2485560403848

In [11]:
```
copied_list = my_list
id(copied_list)
```

Out[11]: 2485560403848

In [9]:
```
#When you change the element in either one of the source or copy object wi
ll affect other object too.
my_list[0] = 'Changed'
```

In [6]:
```
print(copied_list)
```

```
['Changed', 2, 3, 4]
```

In [7]:
```
print(my_list)
```

```
['Changed', 2, 3, 4]
```

Copying Object with copy() method available for list. Shallow copy.

In [19]:
```
my_list = [1,2,3,4]
id(my_list)
```

Out[19]: 2485561683592

In [20]:
```
shallow_copied_list = my_list.copy()
id(shallow_copied_list)
```

Out[20]: 2485561683912

Note:

Here Object change in either source or copied object will not affect other object if it's not containing other nested object.

In [21]:
```
#When you change the element in either one of the source or copy object wi
ll not affect other object too.
my_list[0] = 'Changed'
print(my_list)
print(shallow_copied_list)
```

```
['Changed', 2, 3, 4]
[1, 2, 3, 4]
```

In [30]:
```
## Lets Try to change the value of nested list ['A','B].
```

```
## The change will affect the source and object if its nested object
```

In [33]:
```
my_nested_list = [1,2,3,4,['A','B']]
id(my_nested_list)
```

Out[33]:
```
2485560448008
```

In [24]:
```
shallow_copied_list = my_nested_list.copy()
id(shallow_copied_list)
```

Out[24]:
```
2485560447112
```

In [26]:
```
my_nested_list[4][0] = 'XXXX'
```

In [28]:
```
print(my_nested_list)
print(shallow_copied_list)
```

```
[1, 2, 3, 4, ['XXXX', 'B']]
[1, 2, 3, 4, ['XXXX', 'B']]
```

How can we overcome the shallow copy's nested object problem?

In [34]:
```
## Deepcopy is the solution.
## it will use separate memory for each object so it will never interfere with other object's memory
import copy
deep_copied_list = copy.deepcopy(my_nested_list)
```

List Comprehensions

Python has an advanced feature called list comprehensions. They allow for quick construction of lists. To fully understand list comprehensions we need to understand for loops. So don't worry if you don't completely understand this section, and feel free to just skip it since we will return to this topic later.

But in case you want to know now, here are a few examples!

In [39]:
```
# Regular List contruction methods

# Declare the list
l=[]

# Adding Item to a list

for x in range(9):
    l.append(x)

# printing list
print(l)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8]
```

In [40]:
```
# Here is the quickest way to contruct list
```

```
numbers = [x for x in range(50)]
```

In [41]:
```
print(numbers)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 3
9, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49]
```

In [44]:
```
#list comprehension with for loop and if
# Print only Even Numbers
numbers = [x for x in range(50) if x % 2 == 0]
```

In [45]:
```
print(numbers)
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38
, 40, 42, 44, 46, 48]
```

In [46]:
```
# Build a list comprehension by deconstructing a for loop within a []
first_col = [row[0] for row in matrix]
```

In [47]:
```
first_col
```

Out[47]:
```
[1, 4, 7]
```

We used list comprehension here to grab the first element of every row in the matrix object. We will cover this in much more detail later on!

Dictionaries

We've been learning about *sequences* in Python but now we're going to switch gears and learn about *mappings* in Python. If you're familiar with other languages you can think of these Dictionaries as hash tables.

This section will serve as a brief introduction to dictionaries and consist of:

```
1.) Constructing a Dictionary
2.) Accessing objects from a dictionary
3.) Nesting Dictionaries
4.) Basic Dictionary Methods
```

So what are mappings? Mappings are a collection of objects that are stored by a *key*, unlike a sequence that stored objects by their relative position. This is an important distinction, since mappings won't retain order since they have objects defined by a key.

A Python dictionary consists of a key and then an associated value. That value can be almost any Python object.

Constructing a Dictionary

Let's see how we can construct dictionaries to get a better understanding of how they work!

In [1]:
```
# Make a dictionary with {} and : to signify a key and a value
my_dict = {'key1':'value1','key2':'value2'}
```

In [2]:
```
# Call values by their key
my_dict['key2']
```

Out[2]:
```
'value2'
```

Other Way to create a Dictionary

You can create a Dictionary using built-in dict() function

In [9]:
```
my_dict = dict(key1='value1',key2='value2',key3='value3')
```

In [10]:
```
my_dict['key1']
```

Out[10]:
```
'value1'
```

Its important to note that dictionaries are very flexible in the data types they can hold. For example:

In [13]:
```
my_dict = {'key1':123,'key2':[12,23,33],'key3':['item0','item1','item2']}
```

In [4]:
```
#Lets call items from the dictionary
my_dict['key3']
```

Out[4]:
```
['item0', 'item1', 'item2']
```

In [5]:
```
# Can call an index on that value
my_dict['key3'][0]
```

Out[5]:
```
'item0'
```

In [7]:
```
#Can then even call methods on that value
my_dict['key3'][0].upper()
```

Out[7]:
```
'ITEM0'
```

We can effect the values of a key as well. For instance:

In [14]:
```
my_dict['key1']
```

Out[14]:
```
123
```

In [15]:
```
# Subtract 123 from the value
my_dict['key1'] = my_dict['key1'] - 123
```

In [16]:
```
#Check
my_dict['key1']
```

Out[16]:
```
0
```

A quick note, Python has a built-in method of doing a self subtraction or addition (or multiplication or division). We could have also used += or -= for the above statement. For example:

In [17]:
```
# Set the object equal to itself minus 123
my_dict['key1'] -= 123
my_dict['key1']
```

Out[17]:
```
-123
```

We can also create keys by assignment. For instance if we started off with an empty dictionary, we could continually add to it:

In [21]:
```
# Create a new dictionary
d = {}
```

In [22]:
```
# Create a new key through assignment
d['animal'] = 'Dog'
```

In [24]:
```
# Can do this with any object
d['answer'] = 42
```

In [25]:
```
#Show
d
```

Out[25]:
```
{'animal': 'Dog', 'answer': 42}
```

Nesting with Dictionaries

Hopefully you're starting to see how powerful Python is with its flexibility of nesting objects and calling methods on them. Let's see a dictionary nested inside a dictionary:

In [26]:
```
# Dictionary nested inside a dictionary nested in side a dictionary
d = {'key1':{'nestkey':{'subnestkey':'value'}}}
```

Wow! That's a quite the inception of dictionaries! Let's see how we can grab that value:

In [29]:
```
# Keep calling the keys
d['key1']['nestkey']['subnestkey']
```

Out[29]:
```
'value'
```

A few Dictionary Methods

There are a few methods we can call on a dictionary. Let's get a quick introduction to a few of them:

In [2]:
```
# Create a typical dictionary
d = {'key1':1,'key2':2,'key3':3}
```

In [35]:
```
# Method to return a list of all keys
d.keys()
```

Out[35]:
```
['key3', 'key2', 'key1']
```

In [36]:
```
# Method to grab all values
d.values()
```

Out[36]:
```
[3, 2, 1]
```

In [33]:
```
# Method to return tuples of all items  (we'll learn about tuples soon)
d.items()
```

Out[33]:
```
[('key3', 3), ('key2', 2), ('key1', 1)]
```

In [4]:
```
# get method used to get the value of the key you're passing in.
# if the key is not found then ,
# it will return the message you want to display like - 'Key not found Bro
'


d.get('key3','Key not found Bro')
```

Out[4]:
```
3
```

In [5]:
```
# get method used to get the value of the key you're passing in.
# if the key is not found then ,
# it will return the message you want to display like - 'Key not found Bro
```

```
'

d.get('XXXX','Key not found Bro')
```

Out[5]: 'Key not found Bro'

In [7]:
```
# How can you update the dictionary with other dictionary's key value pair
.
# update method is the solution
d1 = {'newkey1':'value1','newkey2':'value2'}
d.update(d1)
print(d)
```

```
{'key1': 1, 'key2': 2, 'key3': 3, 'newkey1': 'value1', 'newkey2': 'value2'
}
```

In [8]:
```
#Shallow Copying dictionary
copied_dict = d.copy()
print(copied_dict)
```

```
{'key1': 1, 'key2': 2, 'key3': 3, 'newkey1': 'value1', 'newkey2': 'value2'
}
```

In [12]:
```
# Clearing all the elements of Dictionary.
# You can make use of clear() method
d.clear()
# it will print empty dictionary as we cleared it.
print(d)
```

```
{}
```

In [15]:
```
# To remove the specified key from dictionary.
# You can make use of pop() method
d= {'key1': 1, 'key2': 2, 'key3': 3, 'newkey1': 'value1', 'newkey2': 'valu
e2'}
d.pop('newkey1')
print(d)
```

```
{'key1': 1, 'key2': 2, 'key3': 3, 'newkey2': 'value2'}
```

In [16]:
```
# To remove the last item inserted
d.popitem()
```

Out[16]: ('newkey2', 'value2')

Hopefully you now have a good basic understanding how to construct dictionaries. There's a lot more to go into here, but we will revisit dictionaries at later time. After this section all you need to know is how to create a dictionary and how to retrieve values from it.

Tuples

In Python tuples are very similar to lists, however, unlike lists they are *immutable* meaning they can not be changed. You would use tuples to present things that shouldn't be changed, such as days of the week, or dates on a calendar.

In this section, we will get a brief overview of the following:

```
1.) Constructing Tuples
2.) Basic Tuple Methods
3.) Immutability
4.) When to Use Tuples.
```

You'll have an intuition of how to use tuples based on what you've learned about lists. We can treat them very similarly with the major distinction being that tuples are immutable.

Constructing Tuples

The construction of a tuples use () with elements separated by commas. For example:

In [5]:
```
# Can create a tuple with mixed types
t = (1,2,3)
```

In [6]:
```
# Check len just like a list
len(t)
```

Out[6]:
```
3
```

In [8]:
```
# Can also mix object types
t = ('one',2)

# Show
t
```

Out[8]:
```
('one', 2)
```

In [4]:
```
# Use indexing just like we did in lists
t[0]
```

Out[4]:
```
'one'
```

In [11]:
```
# Slicing just like a list
t[-1]
```

Out[11]:
```
2
```

Basic Tuple Methods

Tuples have built-in methods, but not as many as lists do. Lets look at two of them:

In [12]:
```
# Use .index to enter a value and return the index
t.index('one')
```

Out[12]:
```
0
```

In [13]:
```
# Use .count to count the number of times a value appears
t.count('one')
```

Out[13]:
```
1
```

Immutability

It can't be stressed enough that tuples are immutable. To drive that point home:

In [14]:
```
t[0]= 'change'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-14-93def5f9b4bd> in <module>()
----> 1 t[0]= 'change'

TypeError: 'tuple' object does not support item assignment
```

Because of this immutability, tuples can't grow. Once a tuple is made we can not add to it.

In [15]:
```
t.append('nope')
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-15-799b3447c4d9> in <module>()
----> 1 t.append('nope')

AttributeError: 'tuple' object has no attribute 'append'
```

When to use Tuples

You may be wondering, "Why bother using tuples when they have fewer available methods?" To be honest, tuples are not used as often as lists in programming, but are used when immutability is necessary. If in your program you are passing around an object and need to make sure it does not get changed, then tuple become your solution. It provides a convenient source of data integrity.

You should now be able to create and use tuples in your programming as well as have an understanding of their immutability.

Set and Booleans

There are two other object types in Python that we should quickly cover. Sets and Booleans.

Sets

Sets are an unordered collection of *unique* and *immutable*(frozen set) elements.Common uses include membership testing, removing duplicates from a sequence, and computing standard math operations on sets such as intersection, union, difference, and symmetric difference.

Being an unordered collection, sets do not record element position or order of insertion. Accordingly, sets do not support indexing, slicing, or other sequence-like behavior.

Creating Set() from Iterable Python Objects(String, List, Tuple,Dictionaries)

In [1]:
```
# String
s = 'Im string , but am happy to be converted as sets'

# List Created
my_list = [x for x in range(20)]

# Tuple
tup = tuple([x for x in range(10)])

# Dictionary

my_dict = {'key1':'value1','key2':'value2','key3':'value3'}
```

In [2]:
```
# Creating sets from string

setfromstring = set(s)

print(setfromstring)
```

```
{'e', 'v', 't', 'I', 'p', 'g', 'u', 'i', 'd', 'm', 'y', 'c', 'a', 'o', 'h'
, ',', 'b', 'n', 'r', ' ', 's'}
```

In [4]:
```
#creating sets from list

setfromlist = set(my_list)

print(setfromlist)
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19}
```

In [5]:
```
#creating sets from list
```

```
setfromtuple = set(tup)

print(setfromtuple)
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [28]:
```
#creating sets from Dictionary

setfromdict = set(my_dict)

print(setfromdict)
```

```
{'key3', 'key1', 'key2'}
```

In [13]:
```
#Creating sets without set() built-in function
my_sets = {'Hello','Buddy','How','Are','You'}

type(my_sets)
```

Out[13]:
```
set
```

In [12]:
```
#This is not the way to create empty set
D = {}

type(D)
```

Out[12]:
```
dict
```

In [12]:
```
# Declaring empty set
x = set()
```

Sets can hold only Immutable or Hashable Objects (Numbers, strings, tuples)

In [13]:
```
# Adding integer to sets
x.add(1)
print(x)
```

```
{1}
```

In [19]:
```
# Adding string to sets
x.add('string')
print(x)
```

```
{1, 'string', (1, 2, 3)}
```

In [15]:
```
# Adding string to tuple
x.add((1,2,3))
print(x)
```

```
{1, 'string', (1, 2, 3)}
```

In [20]:
```
#Removing Element
x.remove('string')
```

```
print(x)
```

```
{1, (1, 2, 3)}
```

In [22]:
```
#Pop (No order arbitary element (Internal order))
x.pop()
```

Out[22]:
```
1
```

In [23]:
```
#clear set
x.clear()
print(x)
```

```
set()
```

Methods available for sets and frozenset()

Membership testing

In [3]:
```
basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
```

In [6]:
```


'bikekey' in basket
```

Out[6]:
```
False
```

In [7]:
```
'bikekey' not in basket
```

Out[7]:
```
True
```

In [8]:
```
s = 'I am a string'
```

In [9]:
```
set(s)
```

Out[9]:
```
{' ', 'I', 'a', 'g', 'i', 'm', 'n', 'r', 's', 't'}
```

In [10]:
```
a = set('abracadabra')
b = set('alacazam')
print(a)
print(b)
```

```
{'d', 'b', 'a', 'r', 'c'}
{'m', 'a', 'z', 'l', 'c'}
```

In [11]:
```
# Length of sets
len(a)
```

Out[11]:
```
5
```

In [6]:
```
# unique letters in a
print(a)
print(b)
```

```
{'r', 'b', 'c', 'd', 'a'}
{'z', 'c', 'a', 'l', 'm'}
```

Difference

In [12]:
```
# letters in a but not in b
a-b
```

Out[12]:
```
{'b', 'd', 'r'}
```

In [55]:
```
a.difference(b)
```

Out[55]:
```
{'b', 'd', 'r'}
```

Union

In [13]:
```
# letters in a or b or both
a | b
```

Out[13]:
```
{'a', 'b', 'c', 'd', 'l', 'm', 'r', 'z'}
```

In [14]:
```
a.union(b)
```

Out[14]:
```
{'a', 'b', 'c', 'd', 'l', 'm', 'r', 'z'}
```

Intersection

In [15]:
```
# letters in both a and b
a & b
```

Out[15]:
```
{'a', 'c'}
```

In [16]:
```
a.intersection(b)
```

Out[16]:
```
{'a', 'c'}
```

Symmetric Difference

In [17]:
```
# letters in a or b but not both
```

```
a ^ b
```

Out[17]: {'b', 'd', 'l', 'm', 'r', 'z'}

In [58]:
```
a.symmetric_difference(b)
```

Out[58]: {'b', 'd', 'l', 'm', 'r', 'z'}

Subset and Superset

In [26]:
```
a = {'a','b','c','d'}
b = {'a','b','c','d','e','f'}
```

In [27]:
```
a.issubset(b)
```

Out[27]: True

In [23]:
```
a<=b
```

Out[23]: True

In [28]:
```
b.issuperset(a)
```

Out[28]: True

In [7]:
```
b>=a
```

Out[7]: True

Copy

In [29]:
```
set_backup = a.copy()
```

In [30]:
```
print(set_backup)
```

```
{'a', 'd', 'c', 'b'}
```

Duplicate Removal

Notice how it won't place another 1 there. That's because a set is only concerned with unique elements! We can cast a list with multiple repeat elements to a set to get the unique elements. For example:

In [31]:
```
# Create a list with repeats
l = [1,1,2,2,3,4,5,6,1,1]
```

In [32]:
```
# Cast as set to get unique values
set(l)
```

Out[32]: {1, 2, 3, 4, 5, 6}

Frozen Set (Immutable)

set() - mutable : it has methods to add,pop,remove and clear methods to manipulate set items.

frozenset() - Immutable(add,pop,remove and clear methods are not there)

In [33]:
```
fs = frozenset({1,2,3,4,5,5})
```

In [34]:
```
print(fs)
```

```
frozenset({1, 2, 3, 4, 5})
```

In [ ]:
```
fs.
```

Booleans

Python comes with Booleans (with predefined True and False displays that are basically just the integers 1 and 0). It also has a placeholder object called None. Let's walk through a few quick examples of Booleans (we will dive deeper into them later in this course).

In [13]:
```
# Set object to be a boolean
a = True
```

In [16]:
```
#Show
a
```

Out[16]:
```
True
```

We can also use comparison operators to create booleans. We will go over all the comparison operators later on in the course.

In [17]:
```
# Output is boolean
1 > 2
```

Out[17]:
```
False
```

We can use None as a placeholder for an object that we don't want to reassign yet:

In [18]:
```
# None placeholder
b = None
```

Thats it! You should now have a basic understanding of Python objects and data structure types.

In [38]:
```
b=None
```

In [39]:
```
type(b)
```

Out[39]:
```
NoneType
```

Conculusion : Mutable Objects: List -Slice & Indexing possible Dictionary - Cannot Index or Slice , but can grab elements with key set() - Cannot Index or Slice Immutable or Hashable Objects: string, - slicing and indexing possible Tuple, - slicing and indexing possible frozenset - Cannot Index or Slice Number - Cannot Index or Slice

Comparison Operators

In this lecture we will be learning about Comparison Operators in Python. These operators will allow us to compare variables and output a Boolean value (True or False).

If you have any sort of background in Math, these operators should be very straight forward.

First we'll present a table of the comparison operators and then work through some examples:

Table of Comparison Operators

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(a == b) is not true.
!=	If values of two operands are not equal, then condition becomes true.	(a != b) is true
<>	If values of two operands are not equal, then condition becomes true.	(a <> b) is true. This is similar to != operator.
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.

Let's now work through quick examples of each of these.

Equal

In [3]:
```
2 == 2
```

Out[3]:
```
True
```

```
In [4]: 1 == 0
```

```
Out[4]: False
```

Not Equal

```
In [5]: 2 != 1
```

```
Out[5]: True
```

```
In [6]: 2 != 2
```

```
Out[6]: False
```

```
In [7]: 2 <> 1
```

```
Out[7]: True
```

```
In [8]: 2 <> 2
```

```
Out[8]: False
```

Greater Than

```
In [9]: 2 > 1
```

```
Out[9]: True
```

```
In [10]: 2 > 4
```

```
Out[10]: False
```

Less Than

```
In [11]: 2 < 4
```

```
Out[11]: True
```

```
In [12]: 2 < 1
```

```
Out[12]: False
```

Greater Than or Equal to

```
In [13]: 2 >= 2
```

```
Out[13]: True
```

```
In [14]: 2 >= 1
```

```
Out[14]: True
```

Less than or Equal to

In [15]:
```
2 <= 2
```

Out[15]: True

In [16]:
```
2 <= 4
```

Out[16]: True

Chained Comparison Operators

An interesting feature of Python is the ability to *chain* multiple comparisons to perform a more complex test. You can use these chained comparisons as a shorthand for larger Boolean Expressions.

In this lecture we will learn how to chain comparison operators and we will also introduce two other important statements in python: **and** and **or**.

Let's look at a few examples of using chains:

```
In [1]: 1 < 2 < 3
```

```
Out[1]: True
```

The above statement check if 1 was less than 2 **and** if 2 was less than 3. We could have written this using an **and** statement in Python:

```
In [2]: 1<2 and 2<3
```

```
Out[2]: True
```

The **and** is used to make sure two checks have to be true in order for the total check to be true. Let's see another example:

```
In [3]: 1 < 3 > 2
```

```
Out[3]: True
```

The above checks if 3 is larger than both the other numbers, so you could use **and** to rewrite it as:

```
In [4]: 1<3 and 3>2
```

```
Out[4]: True
```

Its important to note that Python is checking both instances of the comparisons. We can also use **or** to write comparisons in Python. For example:

```
In [5]: 1==2 or 2<3
```

```
Out[5]: True
```

Note how it was true, this is because with the **or** operator, we only need one *or* the other two be true. Let's see one more example to drive this home:

```
In [6]: 1==1 or 100==1
```

```
Out[6]: True
```

List Comprehension

List comprehension is an elegant way to define and create list in Python

simple list comprehension

In [4]:
```
# Here is the normal way to create a list

# You need to declare the before you append values to a list.
l = []

# iterate through range(0,10) elements and append to list.
for x in range(0,10):
    l.append(x)

# Print list
print(l)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [30]:
```
# Here is the pythonic way to create a list using list comprehension

l= [x for x in range(0,10)]

# print list
print(l)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [31]:
```
# you can also do some arithmatic operation on top of the number object th
at being stored.
l = [x+3 for x in range(0,10)]
```

list comprehension with for and if statements

In [10]:
```
# Here is the normal way to create a list

# You need to declare the before you append values to a list.
l = []

# iterate through range(0,10) elements and append only even numbers to lis
t.
for x in range(0,10):
    if x%2 == 0:

        l.append(x)

# Print list
print(l)
```

```
[0, 2, 4, 6, 8]
```

In [11]:
```
# Here is the pythonic way to create a list using list comprehension

l= [x for x in range(0,10) if x%2 == 0]

# print list
print(l)
```

```
[0, 2, 4, 6, 8]
```

list comprehension with for and two if statements

In [12]:
```
# Here is the normal way to create a list

# You need to declare the before you append values to a list.
l = []

# iterate through range(0,10) elements and append only even numbers to lis
t.
for x in range(0,10):
    if x%2 == 0:
        if x > 2:
            l.append(x)
# Print list
print(l)
```

```
[4, 6, 8]
```

In [14]:
```
# Here is the pythonic way to create a list using list comprehension

l= [x for x in range(0,10) if x%2 == 0 if x > 2]

# You can also use logical operator 'and' to replace if like

l= [x for x in range(0,10) if x%2 == 0 and x > 2]

# print list
print(l)
```

```
[4, 6, 8]
```

applying method on the objects being stored in list

In [19]:
```
fruits = ['Apple','Mango','Strawberry','Orange','Apricots']

# What you need to do if you want to create a list with only fruits that s
tart
# with letter 'A'

# list declaration
f = []
```

```
for fruit in fruits:
    if fruit.startswith('A'):
        f.append(fruit)

# printing list
print(f)
```

```
['Apple', 'Apricots']
```

In [21]:
```
# Here is the pythonic way to create a list using list comprehension

f= [fruit for fruit in fruits if fruit.startswith('A')]

# print list
print(f)
```

```
['Apple', 'Apricots']
```

Using Two For Loops in List comprehension

In [23]:
```
l = [[1,2,3],[4,5,6],[7,8,9]]
```

In [25]:
```
# Here is the normal way to create a list

# You need to declare the before you append values to a list.
flatten = []

# iterate through range(0,10) elements and append only even numbers to lis
t.
for nest_list in l:
    for element in nest_list:
        flatten.append(element)

# Print list
print(flatten)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [28]:
```
# Here is the pythonic way to create a list using list comprehension

flatten = [element for nest_list in l for element in nest_list ]

# Print list
print(flatten)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Applying Built-in function List comprehension

In [35]:
```
# what will you do if you need find the maximum number and append that to
a list.

l = [[1,2,3],[4,5,6],[7,8,9]]
```

```
# You need to declare the before you append values to a list.
maxl = []

# iterate through range(0,10) elements and append only even numbers to lis
t.
for nest_list in l:
    maxl.append(max(nest_list))

# Print list
print(maxl)
```

```
[3, 6, 9]
```

In [37]:
```
# you can achieve the same result in list comprehension
maxl = [max(nest_list) for nest_list in l ]

# printing
print(maxl)
```

```
[3, 6, 9]
```

Dictionary comprehension

Dict Comprehensions. On top of list comprehensions, Python now supports dict comprehensions, which allow you to express the creation of dictionaries at runtime using a similarly concise syntax.

A dictionary comprehension takes the form {key: value for (key, value) in iterable} .

You can use the dictionary comprehension for copying from different dictionary and for filtering keys or values from dictionary

In [39]:
```
d = {'k1':'v1','k2':'v2','k2':'v3'}
```

In [44]:
```
new_dic ={key: value for key, value in d.items()}

print(new_dic)
```

```
{'k1': 'v1', 'k2': 'v3'}
```

In [56]:
```
## reversing keys and values - normal way

d = {'k1':'v1','k2':'v2','k3':'v3'}

newd = {}

for k,v in d.items():
    newd[v]=k

print(newd)
```

```
{'v1': 'k1', 'v2': 'k2', 'v3': 'k3'}
```

In [57]:
```
# Pythonic way
```

```
newd = {v:k for k,v in d.items() }

print(newd)
```

```
{'v1': 'k1', 'v2': 'k2', 'v3': 'k3'}
```

Merging two dictionaries

In [63]:
```
d1= {'Rollno':1234,'Full Name':'Naveenkumar Murugan','Phone':9710410808}
d2 = {'salary':'I will not share :(','Company':'DXC'}

combined_dic={k: v for d in (d1, d2) for k, v in d.items()}

print(combined_dic)
```

```
{'Rollno': 1234, 'Full Name': 'Naveenkumar Murugan', 'Phone': 9710410808,
'salary': 'I will not share :(', 'Company': 'DXC'}
```

set comprehension

The syntax for set comprehensions is almost identical to that of list comprehensions, but it uses curly brackets instead of square brackets. The pattern is {EXPRESSION FOR ELEMENT IN SEQUENCE}.

In [65]:
```
my_set = {x for x in range(10)}

print(my_set)
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

Things to Remember about comprehension

1 You can use one or more for and if clause

2 You cannot create a list through list comprehension without for loop.

For loop is mandatory for all comprehension

3 It will follow same order of execution as like normal way .(for -> if -> object)

4 All the structure of dictionary,set comprehansion are similar to list comprehension.

Only difference is notation {},() - for generator comprehension

if,elif,else Statements

if Statements in Python allows us to tell the computer to perform alternative actions based on a certain set of results.

Verbally, we can imagine we are telling the computer:

"Hey if this case happens, perform some action"

We can then expand the idea further with elif and else statements, which allow us to tell the computer:

"Hey if this case happens, perform some action. Else if another case happens, perform some other action. Else-- none of the above cases happened, perform this action"

Let's go ahead and look at the syntax format for if statements to get a better idea of this:

```
if case1:
    perform action1
elif case2:
    perform action2
else:
    perform action 3
```

First Example

Let's see a quick example of this:

In [1]:
```
if True:
    print("It was true!")
```

```
It was true!
```

Let's add in some else logic:

In [3]:
```
x = False

if x:
    print('x was True!')
else:
    print('I will be printed in any case where x is not true')
```

```
I will be printed in any case where x is not true
```

Multiple Branches

Let's get a fuller picture of how far if, elif, and else can take us!

We write this out in a nested structure. Take note of how the if,elif,and else line up in the code. This can help you see what if is related to what elif or else statements.

We'll reintroduce a comparison syntax for Python.

In [2]:
```
company = 'newname'

if  company == 'HP':
    print('Welcome to HP')
elif company == 'HPE':
    print('Welcome to HPE')
elif company == 'DXC':
    print('Welcome to DXC')
else:
    print("Are you going to change the company name again?")
```

```
Are you going to change the company name again?
```

Note how the nested if statements are each checked until a True boolean causes the nested code below it to run. You should also note that you can put in as many elif statements as you want before you close off with an else.

Let's create two more simple examples for the if,elif, and else statements:

In [6]:
```
person = 'Naveeenkumar'
marks =100

if person == 'Naveeenkumar':
    if marks > 99:
        print('Congratualtions! ' + person)
```

```
Congratualtions! Naveeenkumar
```

Indentation

It is important to keep a good understanding of how indentation works in Python to maintain the structure and order of your code. We will touch on this topic again when we start building out functions!

for Loops

A **for** loop acts as an iterator in Python, it goes through items that are in a *sequence* or any other iterable item. Objects that we've learned about that we can iterate over include strings,lists,tuples, and even built in iterables for dictionaries, such as the keys or values.

We've already seen the **for** statement a little bit in past lectures but now lets formalize our understanding.

Here's the general format for a **for** loop in Python:

```
for item in object:
    statements to do stuff
```

The variable name used for the item is completely up to the coder, so use your best judgment for choosing a name that makes sense and you will be able to understand when revisiting your code. This item name can then be referenced inside you loop, for example if you wanted to use if statements to perform checks.

Let's go ahead and work through several example of **for** loops using a variety of data object types. we'll start simple and build more complexity later on.

Example 1

Iterating through a list.

In [5]:
```
# We'll learn how to automate this sort of list in the next lecture
l = [1,2,3,4,5,6,7,8,9,10]
```

In [6]:
```
for num in l:
    print(num)
```

```
1
2
3
4
5
6
7
8
9
10
```

Great! Hopefully this makes sense. Now lets add a if statement to check for even numbers. We'll first introduce a new concept here--the modulo.

Modulo

The modulo allows us to get the remainder in a division and uses the % symbol. For example:

In [5]:
```
17 % 5
```

Out[5]:
```
2
```

This makes sense since 17 divided by 5 is 3 remainder 2. Let's see a few more quick examples:

In [6]:
```
# 3 Remainder 1
10 % 3
```

Out[6]:
```
1
```

In [9]:
```
# 2 Remainder 4
18 % 7
```

Out[9]:
```
4
```

In [10]:
```
# 2 no remainder
4 % 2
```

Out[10]:
```
0
```

Notice that if a number is fully divisible with no remainder, the result of the modulo call is 0. We can use this to test for even numbers, since if a number modulo 2 is equal to 0, that means it is an even number!

Back to the **for** loops!

Example 2

Let's print only the even numbers from that list!

In [8]:
```
for num in l:
    if num % 2 == 0:
        print(num)
```

```
2
4
6
8
10
```

We could have also put in else statement in there:

In [9]:
```
for num in l:
    if num % 2 == 0:
        print(num)
    else:
        print('Odd number')
```

```
Odd number
2
Odd number
4
Odd number
6
Odd number
8
Odd number
10
```

Example 3

Another common idea during a **for** loop is keeping some sort of running tally during the multiple loops. For example, lets create a for loop that sums up the list:

In [10]:
```
# Start sum at zero
list_sum = 0

for num in l:
    list_sum = list_sum + num

print(list_sum)
```

```
55
```

Great! Read over the above cell and make sure you understand fully what is going on. Also we could have implemented a += to to the addition towards the sum. For example:

In [11]:
```
# Start sum at zero
list_sum = 0

for num in l:
    list_sum += num

print(list_sum)
```

```
55
```

Example 4

We've used for loops with lists, how about with strings? Remember strings are a sequence so when we iterate through them we will be accessing each item in that string.

In [13]:
```
for letter in 'This is a string.':
    print(letter)
```

```
T
h
i
s
```

```
i
s

a

s
t
r
i
n
g
.
```

Example 5

Let's now look at how a for loop can be used with a tuple:

In [14]:
```
tup = (1,2,3,4,5)

for t in tup:
    print(t)
```

```
1
2
3
4
5
```

Example 6

Tuples have a special quality when it comes to **for** loops. If you are iterating through a sequence that contains tuples, the item can actually be the tuple itself, this is an example of *tuple unpacking*. During the **for** loop we will be unpacking the tuple inside of a sequence and we can access the individual items inside that tuple!

In [16]:
```
l = [(2,4),(6,8),(10,12)]
```

In [17]:
```
for tup in l:
    print(tup)
```

```
(2, 4)
(6, 8)
(10, 12)
```

In [19]:
```
# Now with unpacking!
for (t1,t2) in l:
    print(t1)
```

```
2
6
10
```

Cool! With tuples in a sequence we can access the items inside of them through unpacking! The reason this is important is because many object will deliver their iterables through tuples. Let's start exploring iterating through Dictionaries to explore this further!

Example 7

In [1]:
```
d = {'k1':1,'k2':2,'k3':3}
```

In [3]:
```
for item in d:
    print(item)
```

```
k1
k2
k3
```

Notice how this produces only the keys. So how can we get the values? Or both the keys and the values?

Here is where we are going to have a Python 3 Alert!

Python 2 Alert!

Python 2: Use .iteritems() to iterate through

In Python 2 you should use .iteritems() to iterate through the keys and values of a dictionary. This basically creates a generator (we will get into generators later on in the course) that will generate the keys and values of your dictionary. Let's see it in action:

In [9]:
```
# Creates a generator
d.iteritems()
```

Out[9]:
```
<dictionary-itemiterator at 0x104365520>
```

Calling the items() method returns a list of tuples. Now we can iterate through them just as we did in the previous examples.

In [10]:
```
# Create a generator
for k,v in d.iteritems():
    print k
    print v
```

```
k3
3
k2
2
k1
1
```

Python 3: items()

In Python 3 you should use .items() to iterate through the keys and values of a dictionary. For example:

In [11]:
```
# For Python 3
for k,v in d.items():
    print(k)
    print(v)
```

```
k3
3
k2
2
k1
1
```

You might be wondering why this worked in Python 2. This is because of the introduction of generators to Python during its earlier years. (We will go over generators and what they are in a future section, but the basic notion is that generators don't store data in memory, but instead just yield it to you as it goes through an iterable item).

Originally, Python items() built a real list of tuples and returned that. That could potentially take a lot of extra memory.

Then, generators were introduced to the language in general, and that method was reimplemented as an iterator-generator method named iteritems(). The original remains for backwards compatibility.

One of Python 3's changes is that items() now return iterators, and a list is never fully built. The iteritems() method is also gone, since items() now works like iteritems() in Python 2.

Conclusion

We've learned how to use for loops to iterate through tuples,lists,strings, and dictionaries. It will be an important tool for us, so make sure you know it well and understood the above examples.

More resources

while loops

The **while** statement in Python is one of most general ways to perform iteration. A **while** statement will repeatedly execute a single statement or group of statements as long as the condition is true. The reason it is called a 'loop' is because the code statements are looped through over and over again until the condition is no longer met.

The general format of a while loop is:

```
while test:
    code statement
else:
    final code statements
```

Let's look at a few simple while loops in action.

In [1]:
```
x = 0

while x < 10:
    print('x is currently: '+str(x))
    print('x is still less than 10, adding 1 to x')
    x+=1
```

```
x is currently: 0
x is still less than 10, adding 1 to x
x is currently: 1
x is still less than 10, adding 1 to x
x is currently: 2
x is still less than 10, adding 1 to x
x is currently: 3
x is still less than 10, adding 1 to x
x is currently: 4
x is still less than 10, adding 1 to x
x is currently: 5
x is still less than 10, adding 1 to x
x is currently: 6
x is still less than 10, adding 1 to x
x is currently: 7
x is still less than 10, adding 1 to x
x is currently: 8
x is still less than 10, adding 1 to x
x is currently: 9
x is still less than 10, adding 1 to x
```

Notice how many times the print statements occurred and how the while loop kept going until the True condition was met, which occurred once x==10. Its important to note that once this occurred the code stopped. Lets see how we could add an else statement:

In [2]:
```
x = 0
```

```
while x < 10:
    print('x is currently: '+str(x))
    print('x is still less than 10, adding 1 to x')
    x+=1

else:
    print('All Done!')
```

```
x is currently: 0
x is still less than 10, adding 1 to x
x is currently: 1
x is still less than 10, adding 1 to x
x is currently: 2
x is still less than 10, adding 1 to x
x is currently: 3
x is still less than 10, adding 1 to x
x is currently: 4
x is still less than 10, adding 1 to x
x is currently: 5
x is still less than 10, adding 1 to x
x is currently: 6
x is still less than 10, adding 1 to x
x is currently: 7
x is still less than 10, adding 1 to x
x is currently: 8
x is still less than 10, adding 1 to x
x is currently: 9
x is still less than 10, adding 1 to x
All Done!
```

break, continue, pass

We can use break, continue, and pass statements in our loops to add additional functionality for various cases. The three statements are defined by:

```
break: Breaks out of the current closest enclosing loop.
continue: Goes to the top of the closest enclosing loop.
pass: Does nothing at all.
```

Thinking about **break** and **continue** statements, the general format of the while loop looks like this:

```
while test:
    code statement
    if test:
        break
    if test:
        continue
else:
```

break and **continue** statements can appear anywhere inside the loop's body,but we will usually put them further nested in conjunction with an **if** statement to perform an action based on some condition.

Lets go ahead and look at some examples!

In [4]:
```
x = 0

while x < 10:
    print('x is currently: '+str(x))
    print('x is still less than 10, adding 1 to x')
    x+=1
    if x ==3:
        print('x==3')
    else:
        print('continuing...')
        continue
```

```
x is currently: 0
x is still less than 10, adding 1 to x
continuing...
x is currently: 1
x is still less than 10, adding 1 to x
continuing...
x is currently: 2
x is still less than 10, adding 1 to x
x==3
x is currently: 3
x is still less than 10, adding 1 to x
continuing...
x is currently: 4
x is still less than 10, adding 1 to x
continuing...
x is currently: 5
x is still less than 10, adding 1 to x
continuing...
x is currently: 6
x is still less than 10, adding 1 to x
continuing...
x is currently: 7
x is still less than 10, adding 1 to x
continuing...
x is currently: 8
x is still less than 10, adding 1 to x
continuing...
x is currently: 9
x is still less than 10, adding 1 to x
continuing...
```

Note how we have a printed statement when x==3, and a continue being printed out as we continue through the outer while loop. Let's put in a break once x ==3 and see if the result makes sense:

In [5]:
```
x = 0

while x < 10:
```

```
    print('x is currently: ',x)
    print('x is still less than 10, adding 1 to x')
    x+=1
    if x ==3:
        print('Breaking because x==3')
        break
    else:
        print('continuing...')
        continue
```

```
x is currently:  0
x is still less than 10, adding 1 to x
continuing...
x is currently:  1
x is still less than 10, adding 1 to x
continuing...
x is currently:  2
x is still less than 10, adding 1 to x
Breaking because x==3
```

Note how the other else statement wasn't reached and continuing was never printed!

After these brief but simple examples, you should feel comfortable using while statements in you code.

A word of caution however! It is possible to create an infinitely running loop with while statements. For example:

DO NOT RUN THIS CODE!!!!

while True:

```
    print('Uh Oh infinite Loop!')
```

Functions

Introduction to Functions

This lecture will consist of explaining what a function is in Python and how to create one. Functions will be one of our main building blocks when we construct larger and larger amounts of code to solve problems.

So what is a function?

Formally, a function is a useful device that groups together a set of statements so they can be run more than once. They can also let us specify parameters that can serve as inputs to the functions.

On a more fundamental level, functions allow us to not have to repeatedly write the same code again and again. If you remember back to the lessons on strings and lists, remember that we used a function len() to get the length of a string. Since checking the length of a sequence is a common task you would want to write a function that can do this repeatedly at command.

Functions will be one of most basic levels of reusing code in Python, and it will also allow us to start thinking of program design (we will dive much deeper into the ideas of design when we learn about Object Oriented Programming).

def Statements

Let's see how to build out a function's syntax in Python. It has the following form:

In [3]:
```
def name_of_function(arg1,arg2):
    '''
    This is where the function's Document String (doc-string) goes
    '''
    # Do stuff here
    #return desired result
```

We begin with def then a space followed by the name of the function. Try to keep names relevant, for example len() is a good name for a length() function. Also be careful with names, you wouldn't want to call a function the same name as a built-in function in Python (such as len).

Next come a pair of parenthesis with a number of arguments separated by a comma. These arguments are the inputs for your function. You'll be able to use these inputs in your function and reference them. After this you put a colon.

Now here is the important step, you must indent to begin the code inside your function correctly. Python makes use of *whitespace* to organize code. Lots of other programing languages do not do this, so keep that in mind.

Next you'll see the doc-string, this is where you write a basic description of the function. Using

iPython and iPython Notebooks, you'll be ab;e to read these doc-strings by pressing Shift+Tab after a function name. Doc strings are not necessary for simple functions, but its good practice to put them in so you or other people can easily understand the code you write.

After all this you begin writing the code you wish to execute.

The best way to learn functions is by going through examples. So let's try to go through examples that relate back to the various objects and data structures we learned about before.

Example 1: A simple print 'hello' function

In [4]:
```
def say_hello():
    print('hello')
```

Call the function

In [5]:
```
say_hello()
```

```
hello
```

Example 2: A simple greeting function

Let's write a function that greets people with company name.

In [2]:
```
def greeting(name):
    print('Hello '+str(name))
```

In [3]:
```
greeting('DXC')
```

```
Hello DXC
```

Using return

Let's see some example that use a return statement. return allows a function to *return* a result that can then be stored as a variable, or used in whatever manner a user wants.

Example 3: Addition function

In [6]:
```
def add_num(num1,num2):
    return num1+num2
```

In [7]:
```
add_num(4,5)
```

Out[7]: 9

In [8]:
```
# Can also save as variable due to return
result = add_num(4,5)
```

In [11]:
```
print(result)
```

```
9
```

What happens if we input two strings?

In [10]:
```
print(add_num('one','two'))
```

```
onetwo
```

Note that because we don't declare variable types in Python, this function could be used to add numbers or sequences together! We'll later learn about adding in checks to make sure a user puts in the correct arguments into a function.

Lets also start using *break*,*continue*, and *pass* statements in our code. We introduced these during the while lecture.

Finally lets go over a full example of creating a function to check if a number is prime (a common interview exercise).

We know a number is prime if that number is only evenly divisible by 1 and itself. Let's write our first version of the function to check all the numbers from 1 to N and perform modulo checks.

In [12]:
```
def is_prime(num):
    '''
    Naive method of checking for primes.
    '''
    for n in range(2,num):
        if num % n == 0:
            print('not prime')
            break
    else: # If never mod zero, then prime
        print('prime')
```

In [14]:
```
is_prime(17)
```

```
prime
```

Note how we break the code after the print statement! We can actually improve this by only checking to the square root of the target number, also we can disregard all even numbers after checking for 2. We'll also switch to returning a boolean value to get an example of using return statements:

In [15]:
```
import math

def is_prime(num):
    '''
    Better method of checking for primes.
    '''
    if num % 2 == 0 and num > 2:
        return False
    for i in range(3, int(math.sqrt(num)) + 1, 2):
        if num % i == 0:
```

```
            return False
    return True
```

In [16]:
```
is_prime(14)
```

Out[16]:
```
False
```

Great! You should now have a basic understanding of creating your own functions to save yourself from repeatedly writing code!

Functions in Python

You use functions in programming to bundle a set of instructions that you want to use repeatedly.That means that a function is a piece of code written to carry out a specified task.

There are three types of functions in Python:

Built-in functions:

Built-in fucntions are functions that are already created for readymade use. Python has different type of built-in fucntions. There are some built-in fucntions to do the casting from one object type to other object type such as int() float(),str(),list(),set(),tuple(),dict() and some built-in helper functions such as help(),type(). And some are used for aggregate functions such as min(),max(),abs().

To know the complete list of available built-in functions go to https://docs.python.org/3/library/functions.html.

UDF(User Defined Functions)

User-Defined Functions (UDFs), which are functions that users create to help them out

Anonymous Functions(lambda)

The lambda operator or lambda function is a way to create small anonymous functions, i.e. functions without a name. These functions are throw-away functions, i.e. they are just needed where they have been created. Lambda functions are mainly used in combination with the functions filter(), map() and reduce().

Functions Vs Methods

A method refers to a function which is part of a class. You access it with an instance or object of the class. A function doesn't have this restriction: it just refers to a standalone function. This means that all methods are functions but not all functions are methods.

Consider this example, where you first define a function add_func() and then a Addition class with a add() method:

In [8]:
```
# User defined function

def add func(a,b):
```

```
    return a+b
```

In [9]:
```
# calling a function
add_func(1,2)
```

Out[9]: 3

In [10]:
```
#Defining class or object

class Addition(object):
    def add(self,a,b):
        self.result = a + b
        return self.result
```

If you now want to call the add() method that is part of the Addition class, you first need to define an instance or object of that class. So, let's define such an object:

In [12]:
```
# creating instance of class
AddtionInstance = Addition()
```

In [15]:
```
# Calling a method
AddtionInstance.add(1,2)
```

Out[15]: 3

Parameters Vs Arguments

Parameters are the names used when defining a function or a method, and into which arguments will be mapped. In other words, arguments are the things which are supplied to any function or method call, while the function or method code refers to the arguments by their parameter names.

In Simple words, Inside method or function it will be called as *parameter* and when you're passing to other functions as input that is called as *Arguments*.

How To Define A Function: User-Defined Functions (UDFs)

The four steps to defining a function in Python are the following:

1)Use the keyword def to declare the function and follow this up with the function name.

2)Add arguments to the function: they should be within the parentheses of the function. End your line with a colon.

3)Add statements that the functions should execute.

4)End your function with a return statement if the function should output something. Without the return statement, your function will return an object None.

Example

In [26]:
```
def hello(a,b):
  print("Hello World")
  return a+b
```

Of course, your functions will get more complex as you go along: you can add for loops, flow control, … and more to it to make it more finegrained:

In [1]:
```
def hello():
  name = str(input("What is your name buddy?: "))
  if name:
    print ("Hello " + str(name))
  else:
    print("Hello World")
  return

hello()
```

```
What is your name buddy?: Naveenkumar Murugan
Hello Naveenkumar Murugan
```

Return statement

1)Return statement can just return control back ex: return

2)Return statement can just return single value. ex: return a

3)Return statement can return multiple values through tuples : return (a,b). You would require to unpack the tuple. Have a look at example below

In [5]:
```
# Define `plus()`
def plus(a,b):
  sum = a + b
  return

plus()
```

How To Add Docstrings To A Python Function

Docstrings describe what your function does, such as the computations it performs or its return values. These descriptions serve as documentation for your function so that anyone who reads your function's docstring understands what your function does, without having to trace through all the code in the function definition.

In [7]:
```
def hello():

    """
    Function: Prints "Hello World".
```

```
    Returns:
        None
    """
    print("Hello World")
    return
```

In [36]:
```
hello()
```

```
Hello World
```

Function Arguments in Python

There are four types of arguments that Python UDFs can take:

1)Default arguments

2)Required arguments

3)Keyword arguments

4)Variable number of arguments

Default Arguments

Default arguments are those that take a default value if no argument value is passed during the function call. You can assign this default value by with the assignment operator =, just like in the following example:

In [1]:
```
# Define `plus()` function
def plus(a,b = 2):
  return a + b
```

In [41]:
```
# Call `plus()` with only `a` parameter
plus(a=1)
```

Out[41]: 3

In [42]:
```
# Call `plus()` with `a` and `b` parameters
plus(a=1, b=3)
```

Out[42]: 4

Required Arguments

As the name kind of gives away, the required arguments of a UDF are those that have to be in there. These arguments need to be passed during the function call and in exactly the right order, just like in the following example:

In [1]:
```
# Define `plus()` with required arguments
```

```
def plus(a,b):
  return a+b
```

In [2]:
```
# Note: You should pass elements in the order, changing the order will giv
e you different result based on the operation
plus(1,2)
```

Out[2]: 3

Keyword Arguments

If you want to make sure that you call all the parameters in the right order, you can use the keyword arguments in your function call.

In [7]:
```
# Note: You should pass elements in the order, changing the order will giv
e you different result based on the operation
plus(a=10,b=5)
```

Out[7]: 2.0

Variable Number of Arguments

In cases where you don't know the exact number of arguments that you want to pass to a function, you can use the following syntax with *args:

It can be any name like *var,* parm.

In [1]:
```
# Define `plus()` function to accept a variable number of arguments
def plus(*args):
  return sum(args)

# Calculate the sum
plus(1,4,5)
```

Out[1]: 10

In [11]:
```
# you can pass 0 to n numbers arguments to that function when the argument
s decalred with * infront
plus()
```

Out[11]: 0

Global vs Local Variables

In general, variables that are defined inside a function body have a local scope, and those defined outside have a global scope. That means that local variables are defined within a function block and can only be accessed inside that function, while global variables can be accessed by all functions that might be in your script:

In [2]:
```
name = 'This is a global name'
```

```
def greet():
    # Enclosing function
    name = 'Naveen'
    print('am inside greet func: '+ str(name))

    def hello():
        name = 'Kumar'
        print('am inside hello func: '+str(name))

    hello()

greet()

print('outside function: ' + str(name))
```

```
am inside greet func: Naveen
am inside hello func: Kumar
outside function: This is a global name
```

Anonymous Functions in Python

Anonymous functions are also called lambda functions in Python because instead of declaring them with the standard def keyword, you use the lambda keyword.

In [32]:
```
addition = lambda x : x**2
```

In [33]:
```
addition(2)
```

Out[33]: 4

Global Variables can be accessed anywhere in the module or program

In [1]:
```
var = 'I am global variable'

def func():
    print(var)

func()

# let us print from outside func()

print(var)
```

```
I am global variable
I am global variable
```

Local variables can be accessed only inside function , and not from outside

In [3]:
```
# This print(var1) will give you NameError, because local variable var1 do
esn't have visibility outside the function
def func():

    var1 = 'I am a local Variable'
    # But, func() is able to access that local variable, that's why it is
printed 'I am a local Variable'
    print(var1)

func()

# let us print from outside func()

print(var1)
```

```
I am a local Variable
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call last
)
<ipython-input-3-c04e336feb10> in <module>()
     10 # let us print from outside func()
     11
---> 12 print(var1)

NameError: name 'var1' is not defined
```

What will happen if i use the same variable inside and

outside the fucntion?

In [10]:
```
var = 'I am global variable'

def func():
    global var
    var = 'I am a local variable'
    print(var)

func()

print(var)
```

```
I am a local variable
I am global variable
```

In [34]:
```
# if you see the able example, when var printed inside the func() it print
ed 'I am a local variable'
# when var printed ouside it printed 'I am global variable',
#because whaterver the changes done to var isn't visible outside the funct
ion
```

What should i do, if i need to access the variable outside the function?

In [7]:
```
var = 'I am global variable'

def func():
    global var
    var = 'I am a local variable'
    print('The value of var inside func(): ' + str(var))

print('Before calling func() the values of var is : ' + str(var))

print('Let me call func() to modify the value of var')

func()

print('After calling func() the values of var is : ' + str(var))
```

```
Before calling func() the values of var is : I am global variable
Let me call func() to modify the value of var
The value of var inside func(): I am a local variable
After calling func() the values of var is : I am a local variable
```

Nested Functions

The function resides inside other function is called as 'Nested Functions'

In [14]:
```
def func():
    print('I am function inside func()')
```

```
    def nest_func():
        print('I am nested function inside nest_func()')
    nest_func()
```

In [16]:
```
# You call func() as like other normal function like below, but how will y
ou call nested function?
func()
```

```
I am function inside func()
I am nested function inside nest_func()
```

In [17]:
```
# When you try to execute a nested function outside the parent function, y
ou will run into an error
nest_func()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last
)
<ipython-input-17-5a2595be1d33> in <module>()
      1 # When you try to execute a nested function outside the parent fun
ction, you will run into an error
----> 2 nest_func()

NameError: name 'nest_func' is not defined
```

How you should call a nested function?

In [55]:
```
def func():
    print('I am function inside func()')
    def nest_func():
        print('I am nested function inside nest_func()')
    # You should call it from parent function like
    nest_func()
```

In [57]:
```
# so when you call parent func(), the nested function nest_func() also wil
l get executed
func()
```

```
I am function inside func()
I am nested function inside nest_func()
```

Understanding Global Vs Nonlocal Statements

In [23]:
```
var = 'I am global variable'

def func():

    var = 'I am inside func()'

    def nest_func():
        global  var
```

```
        var = 'I am inside nest_func()'

    print('Before calling nest_func(): ' + str(var))
    nest_func()
    print('After calling nest_func(): ' + str(var))

print('Before calling func() the values of var is : ' + str(var))


func()


print('After calling func() the values of var is : ' + str(var))
```

```
Before calling func() the values of var is : I am global variable
Before calling nest_func(): I am inside func()
After calling nest_func(): I am inside func()
After calling func() the values of var is : I am inside nest_func()
```

In [24]:

```
var = 'I am global variable'

def func():
    var = 'I am inside func()'

    def nest_func():
        nonlocal var
        var = 'I am inside nest_func()'

    print('Before calling nest_func(): ' + str(var))
    nest_func()
    print('After calling nest_func(): ' + str(var))

print('Before calling func() the values of var is : ' + str(var))


func()


print('After calling func() the values of var is : ' + str(var))
```

```
Before calling func() the values of var is : I am global variable
Before calling nest_func(): I am inside func()
After calling nest_func(): I am inside nest_func()
After calling func() the values of var is : I am global variable
```

Errors and Exception Handling

In this lecture we will learn about Errors and Exception Handling in Python. You've definitely already encountered errors by this point in the course. For example:

In [1]:
```
print('Hello
```

```
  File "<ipython-input-1-7c6c1c9377e7>", line 1
    print('Hello
                ^
SyntaxError: EOL while scanning string literal
```

Note how we get a SyntaxError, with the further description that it was an EOL (End of Line Error) while scanning the string literal. This is specific enough for us to see that we forgot a single quote at the end of the line. Understanding these various error types will help you debug your code much faster.

This type of error and description is known as an Exception. Even if a statement or expression is syntactically correct, it may cause an error when an attempt is made to execute it. Errors detected during execution are called exceptions and are not unconditionally fatal.

You can check out the full list of built-in exceptions here. now lets learn how to handle errors and exceptions in our own code.

try and except

The basic terminology and syntax used to handle errors in Python is the **try** and **except** statements. The code which can cause an exception to occue is put in the *try* block and the handling of the exception is the implemented in the *except* block of code. The syntax form is:

```
try:
   You do your operations here...
   ...
except ExceptionI:
   If there is ExceptionI, then execute this block.
except ExceptionII:
   If there is ExceptionII, then execute this block.
   ...
else:
   If there is no exception or try block executed sucessfully then e
xecute this block.
finally:
   I will execute all the time(exception or no exception)
```

We can also just check for any exception with just using except: To get a better understanding of all

this lets check out an example: We will look at some code that opens and writes a file:

Let's see some basic exceptions

Zero Division Error

In [1]:
```
8 + (1/0)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-1-10137a11588d> in <module>()
----> 1 8 + (1/0)

ZeroDivisionError: division by zero
```

TypeError

In [2]:
```
6 + 'string'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-2-37375bff2856> in <module>()
----> 1 6 + 'string'

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

NameError

In [4]:
```
4 + badvar
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-4-0978d57ed1b8> in <module>()
----> 1 4 + badvar

NameError: name 'badvar' is not defined
```

You can handle all these exceptions with Try and Except

In [5]:
```
try:
    8 + (1/0)
except ZeroDivisionError as e:
    print(e)
```

```
try:
    8 + 'String'
except TypeError as e:
    print(e)

try:
    8 + badvar
except NameError as e:
    print(e)
```

```
division by zero
unsupported operand type(s) for +: 'int' and 'str'
name 'badvar' is not defined
```

You can combine known exception together like below

In [8]:
```
try:
    8 + (1/0)
except (ZeroDivisionError,TypeError,NameError) as e:
    print(e)
```

```
division by zero
```

How you can handle exceptions generic way

In [6]:
```
try:
    8 + (1/0)
except Exception as e:
    print(e)

try:
    8 + 'String'
except Exception as e:
    print(e)

try:
    8 + badvar
except Exception as e:
    print(e)
```

```
division by zero
unsupported operand type(s) for +: 'int' and 'str'
name 'badvar' is not defined
```

Real time examples

In [12]:
```
f = open('XXX.txt','r')
```

```
---------------------------------------------------------------------
-
FileNotFoundError                         Traceback (most recent call last
```

```
)
<ipython-input-12-266cbba16655> in <module>()
----> 1 f = open('XXX.txt','r')

FileNotFoundError: [Errno 2] No such file or directory: 'XXX.txt'
```

In [13]:
```
try:
    f = open('XXX.txt','r')
except FileNotFoundError as e:
    print(e)
```

```
[Errno 2] No such file or directory: 'XXX.txt'
```

In [16]:
```
try:
    #Trying to read the file available
    f = open('testfile','r')
except FileNotFoundError as e:
    print(e)
else:
    # this block executes only when try block runs without exception
    print(f.read())
    f.close
```

```
Test write statement for exception handling!!
```

We can use this in conjunction with except. Lets see a new example that will take into account a user putting in the wrong input:

Another Example

In [17]:
```
while True:
    try:
        val = int(input("Please enter an integer: "))
    except:
        print("Looks like you did not enter an integer!")
        continue
    else:
        print('You Entered interver and the value is',val)
        break
```

```
Please enter an integer: e
Looks like you did not enter an integer!
Please enter an integer: am mad and giving string again
Looks like you did not enter an integer!
Please enter an integer: now i will give integer
Looks like you did not enter an integer!
Please enter an integer: 4
You Entered interver and the value is 4
```

In [19]:
```
### Raising  exceptions
### if you need to raise your own customized exception then you need to cr
eate exception object with calss keyword
```

In [18]:
```
while True:
    try:
        val = int(input("Please enter an integer: "))
        if val == 5:
            raise ValueError
    except ValueError:
        print("I don't like number 5, give me another number")
        continue
    else:
        print('You Entered interver and the value is',val)
        break
```

```
Please enter an integer: 5
I don't like number 5, give me another number
Please enter an integer: 4
You Entered interver and the value is 4
```

enumerate()

In this lecture we will learn about an extremely useful built-in function: enumerate(). Enumerate allows you to keep a count as you iterate through an object. It does this by returning a tuple in the form (count,element). The function itself is equivalent to:

```
def enumerate(sequence, start=0):
    n = start
    for elem in sequence:
        yield n, elem
        n += 1
```

Example

In [1]:
```
lst = ['a','b','c']

for number,item in enumerate(lst):
    print number
    print item
```

```
0
a
1
b
2
c
```

enumerate() becomes particularly useful when you have a case where you need to have some sort of tracker. For example:

In [3]:
```
for count,item in enumerate(lst):
    if count >= 2:
        break
    else:
        print item
```

```
a
b
```

Great! You should now have a good understanding of enumerate and its potential use cases.

lambda expressions

One of Pythons most useful (and for beginners, confusing) tools is the lambda expression. lambda expressions allow us to create "anonymous" functions. This basically means we can quickly make ad-hoc functions without needing to properly define a function using def.

Function objects returned by running lambda expressions work exactly the same as those created and assigned by defs. There is key difference that makes lambda useful in specialized roles:

lambda's body is a single expression, not a block of statements.

- The lambda's body is similar to what we would put in a def body's return statement. We simply type the result as an expression instead of explicitly returning it. Because it is limited to an expression, a lambda is less general that a def. We can only squeeze design, to limit program nesting. lambda is designed for coding simple functions, and def handles the larger tasks.

Lets slowly break down a lambda expression by deconstructing a function:

In [1]:
```
def square(num):
    result = num**2
    return result
```

In [2]:
```
square(2)
```

Out[2]: 4

Continuing the breakdown:

In [3]:
```
def square(num):
    return num**2
```

In [4]:
```
square(2)
```

Out[4]: 4

We can actually write this in one line (although it would be bad style to do so)

In [5]:
```
def square(num): return num**2
```

In [6]:
```
square(2)
```

Out[6]: 4

This is the form a function that a lambda expression intends to replicate. A lambda expression can then be written as:

In [7]:
```
lambda num: num**2
```

Out[7]:
```
<function __main__.<lambda>>
```

Note how we get a function back. We can assign this function to a label:

In [8]:
```
square = lambda num: num**2
```

In [9]:
```
square(2)
```

Out[9]:
```
4
```

And there you have it! The breakdown of a function into a lambda expression! Lets see a few more examples:

Example 1

Check it a number is even

In [13]:
```
even = lambda x: x%2==0
```

In [14]:
```
even(3)
```

Out[14]:
```
False
```

In [15]:
```
even(4)
```

Out[15]:
```
True
```

Example 2

Grab first character of a string:

In [22]:
```
first = lambda s: s[0]
```

In [23]:
```
first('hello')
```

Out[23]:
```
'h'
```

Example 3

Reverse a string:

In [24]:
```
rev = lambda s: s[::-1]
```

In [25]:
```
rev('hello')
```

Out[25]:
```
'olleh'
```

Example 4

Just like a normal function, we can accept more than one function into a lambda expression:

In [17]:
```
adder = lambda x,y : x+y
```

In [19]:
```
adder(2,3)
```

Out[19]:
```
5
```

Lambda functions are mainly used in conjunction with map(),reduce() (deprecatated in Python3),filter() and sorted() built in function

In [11]:
```
### What you should do when you are required to order the list of strings
based on last letter of the
### string?
```

In [6]:
```
last_letter=lambda x: x[-1]
```

In [9]:
```
# It takes last letter
last_letter('You there?')
```

Out[9]:
```
'?'
```

In [12]:
```
# Lets say i have a list of fruits
basket = ['pappaya','Orange','apple','mango','strawberry']
```

In [10]:
```
# Sorted buitin function takes One string from the list and takes last let
ter x[-1](Reads from right)
# and sorts based on last letter
# Note: You can sort based on any letter in a string
sorted(basket,key=lambda x: x[-1])
```

Out[10]:
```
['pappaya', 'Orange', 'apple', 'mango', 'strawberry']
```

Lambda with filter() function

Note: Your lambda should always return True or False to use filter function

In [15]:
```
even=lambda n:n%2 ==0
```

In [16]:
```
even(4)
```

Out[16]:
```
True
```

In [17]:
```
even(3)
```

Out[17]:
```
False
```

In [36]:
```
filter(lambda n:n%2 ==0,[1,2,3,34,4,499])
```

Out[36]:
```
<filter at 0x24f38500080>
```

In [1]:
```
### Note: In Python Object returned from filter() method aren't list objec
t. Its a filter object.
### so you need to cast it to a list.
list(filter(lambda n:n%2 ==0,[1,2,3,34,4,499]))
```

Out[1]:
```
[2, 34, 4]
```

Lambda with map() function

Map function takes list of elements and apply lambda function on each element of a list and returns map object.

In [23]:
```
map(lambda n:n**2,[1,2,3,4,5])
```

Out[23]:
```
<map at 0x24f3845cf98>
```

In [24]:
```
list(map(lambda n:n**2,[1,2,3,4,5]))
```

Out[24]:
```
[1, 4, 9, 16, 25]
```

Lambda with reduce() function

Reduce is a really useful function for performing some computation on a list and returning the result. It applies a rolling computation to sequential pairs of values in a list. For example, if you wanted to compute the product of a list of integers.

So the normal way you might go about doing this task in python is using a basic for loop:

In [37]:
```
product = 1
l = [1, 2, 3, 4]
for num in l:
    product = product * num

print(product)

# product = 24
```

```
24
```

In [28]:
```
# Reduce is depricated in Python 3. so you can import from functools
from functools import reduce
product = reduce((lambda x, y: x * y), [1, 2, 3, 4])
print(product)
```

```
24
```

lambda expressions really shine when used in conjunction with map(),filter() and reduce(). Each of those functions has its own lecture, so feel free to explore them if you're very interested in lambda.

I highly recommend reading this blog post at Python Conquers the Universe for a great breakdown on lambda expressions and some explanations of common confusions!

Files

Python uses file objects to interact with external files on your computer. These file objects can be any sort of file you have on your computer, whether it be an audio file, a text file, emails, Excel documents, etc. Note: You will probably need to install certain libraries or modules to interact with those various file types, but they are easily available. (We will cover downloading modules later on in the course).

The open Function

Before you can read or write a file, you have to open it using Python's built-in open() function. This function creates a file object, which would be utilized to call other support methods associated with it.

file object = open(file_name [, access_mode][, buffering])

Here are parameter details −

file_name − The file_name argument is a string value that contains the name of the file that you want to access.

access_mode − The access_mode determines the mode in which the file has to be opened, i.e., read, write, append, etc. A complete list of possible values is given below in the table. This is optional parameter and the default file access mode is read (r).

buffering − If the buffering value is set to 0, no buffering takes place. If the buffering value is 1, line buffering is performed while accessing a file. If you specify the buffering value as an integer greater than 1, then buffering action is performed with the indicated buffer size. If negative, the buffer size is the system default(default behavior).

Here is a list of the different modes of opening a file −

Python has a built-in open function that allows us to open and play with basic file types. First we will need a file though. We're going to use some iPython magic to create a text file!

iPython Writing a File

In [1]:
```
%%writefile test.txt
Hello, this is a quick test file
```

```
Writing test.txt
```

Python Opening a file

We can open a file with the open() function. The open function also takes in arguments (also called

parameters). Lets see how this is used:

In [2]:
```
# Open the text.txt we made earlier
my_file = open('test.txt')
```

In [3]:
```
# We can now read the file
my_file.read()
```

Out[3]:
```
'Hello, this is a quick test file'
```

In [4]:
```
# But what happens if we try to read it again?
my_file.read()
```

Out[4]:
```
''
```

This happens because you can imagine the reading "cursor" is at the end of the file after having read it. So there is nothing left to read. We can reset the "cursor" like this:

In [30]:
```
# Seek to the start of file (index 0)
my_file.seek(0)
```

Out[30]:
```
0
```

In [11]:
```
# Now read again
my_file.read()
```

Out[11]:
```
'Hello, this is a quick test file'
```

In order to not have to reset every time, we can also use the readlines method. Use caution with large files, since everything will be held in memory. We will learn how to iterate over large files later in the course.

In [31]:
```
# Readlines returns a list of the lines in the file.
my_file.readlines()
```

Out[31]:
```
['Hello, this is a quick test file']
```

In [33]:
```
my_file.seek(0)
# To check number of records in the file
len(my_file.readlines())
```

Out[33]:
```
1
```

In [35]:
```
my_file.seek(0)
# To check record length
len(my_file.readline())
```

Out[35]:
```
32
```

In [16]:
```
# To get the name of the file
my_file.name
```

Out[16]:
```
'test.txt'
```

In [18]:
```
# To know file mode
my_file.mode
```

Out[18]:
```
'r'
```

In [20]:
```
#To Check the status of the file ( Closed or not)
my_file.closed
```

Out[20]:
```
False
```

In [23]:
```
#To check whether the files readable or not
my_file.readable()
```

Out[23]:
```
True
```

In [26]:
```
#To Check whether the file is seekable or not
my_file.seekable()
```

Out[26]:
```
True
```

In [29]:
```
# To check current position of the cursor
my_file.tell()
```

Out[29]:
```
32
```

Writing to a File

By default, using the open() function will only allow us to read the file, we need to pass the argument 'w' to write over the file. For example:

In [39]:
```
# Add a second argument to the function, 'w' which stands for write
my_file = open('test.txt','w+')
```

In [40]:
```
# Write to the file
my_file.write('This is a new line')
```

In [43]:
```
# Read the file
my_file.read()
```

Out[43]:
```
'This is a new line'
```

Iterating through a File

Lets get a quick preview of a for loop by iterating over a text file. First let's make a new text file with some iPython Magic:

In [44]:
```
%%writefile test.txt
First Line
Second Line
```

```
Overwriting test.txt
```

Now we can use a little bit of flow to tell the program to for through every line of the file and do something:

In [45]:
```
for line in open('test.txt'):
    print line
```

```
First Line

Second Line
```

Don't worry about fully understanding this yet, for loops are coming up soon. But we'll break down what we did above. We said that for every line in this text file, go ahead and print that line. Its important to note a few things here:

```
1.) We could have called the 'line' object anything (see example bel
ow).
2.) By not calling .read() on the file, the whole text file was not
stored in memory.
3.) Notice the indent on the second line for print. This whitespace
is required in Python.
```

We'll learn a lot more about this later, but up next: Sets and Booleans!

In [46]:
```
# Pertaining to the first point above
for asdf in open('test.txt'):
    print asdf
```

```
First Line

Second Line
```

CSV handling in Python

In [1]:
```
# importing CSV module
import csv
```

Create file object and pass it to CSV reader

In [4]:
```
# Reading CSV file
with open('names.csv','r') as f:
    # Creating csv reader object
    csv_reader = csv.reader(f)
    # Print statement won't work csv reader is an generator object
    print(csv_reader)
```

```
<_csv.reader object at 0x000002CBB5B94800>
```

Iterating through generator Object

In [13]:
```
# Reading CSV file
with open('names.csv','r') as f:

    # Creating csv reader object
    csv_reader = csv.reader(f)

    # you can ignore the header using next()
    next(csv_reader)

    # Print statement won't work csv reader is an generator object
    for row in csv_reader:
        print(row)
        # You can filter only coulmns by print(row[0])
```

```
['John', 'Doe', 'john-doe@bogusemail.com']
['Mary', 'Smith-Robinson', 'maryjacobs@gmail.com']
['Dave', 'Smith', 'davesmith@gmail.com']
['Jane', 'Stuart', 'janestuart@gmail.com']
['Tom', 'Wright', 'tomwright@gmail.com']
['Steve', 'Robinson', 'steverobinson@gmail.com']
['Nicole', 'Jacobs', 'nicolejacobs@gmail.com']
['Jane', 'Wright', 'janewright@gmail.com']
['Jane', 'Doe', 'janedoe@gmail.com']
['Kurt', 'Wright', 'kurtwright@gmail.com']
['Kurt', 'Robinson', 'kurtrobinson@gmail.com']
['Jane', 'Jenkins', 'janejenkins@gmail.com']
['Neil', 'Robinson', 'neilrobinson@gmail.com']
['Tom', 'Patterson', 'tompatterson@gmail.com']
['Sam', 'Jenkins', 'samjenkins@gmail.com']
['Steve', 'Stuart', 'stevestuart@gmail.com']
```

```
['Maggie', 'Patterson', 'maggiepatterson@gmail.com']
['Maggie', 'Stuart', 'maggiestuart@gmail.com']
['Jane', 'Doe', 'janedoe@gmail.com']
['Steve', 'Patterson', 'stevepatterson@gmail.com']
['Dave', 'Smith', 'davesmith@gmail.com']
['Sam', 'Wilks', 'samwilks@gmail.com']
['Kurt', 'Jefferson', 'kurtjefferson@gmail.com']
['Sam', 'Stuart', 'samstuart@gmail.com']
['Jane', 'Stuart', 'janestuart@gmail.com']
['Dave', 'Davis', 'davedavis@gmail.com']
['Sam', 'Patterson', 'sampatterson@gmail.com']
['Tom', 'Jefferson', 'tomjefferson@gmail.com']
['Jane', 'Stuart', 'janestuart@gmail.com']
['Maggie', 'Jefferson', 'maggiejefferson@gmail.com']
['Mary', 'Wilks', 'marywilks@gmail.com']
['Neil', 'Patterson', 'neilpatterson@gmail.com']
['Corey', 'Davis', 'coreydavis@gmail.com']
['Steve', 'Jacobs', 'stevejacobs@gmail.com']
['Jane', 'Jenkins', 'janejenkins@gmail.com']
['John', 'Jacobs', 'johnjacobs@gmail.com']
['Neil', 'Smith', 'neilsmith@gmail.com']
['Corey', 'Wilks', 'coreywilks@gmail.com']
['Corey', 'Smith', 'coreysmith@gmail.com']
['Mary', 'Patterson', 'marypatterson@gmail.com']
['Jane', 'Stuart', 'janestuart@gmail.com']
['Travis', 'Arnold', 'travisarnold@gmail.com']
['John', 'Robinson', 'johnrobinson@gmail.com']
['Travis', 'Arnold', 'travisarnold@gmail.com']
```

Writing CSV file

In [11]:
```
# Reading CSV file
with open('names.csv','r') as f:
    # Creating csv reader object
    csv_reader = csv.reader(f)

    #Make sure create the file object with 'w'(write mode)
    with open('csvout.csv','w') as wf:
        csv_write = csv.writer(wf)
        for row in csv_reader:
            csv_write.writerow(row)
```

Reading and Writing with DictReader and Dict Writer Method

In [16]:
```
with open('names.csv','r') as f:
    csv_reader = csv.DictReader(f)
    next(csv_reader)
    for line in csv_reader:
        print(line)
```

```
OrderedDict([('first_name', 'Mary'), ('last_name', 'Smith-Robinson'), ('em
ail', 'maryjacobs@gmail.com')])
OrderedDict([('first_name', 'Dave'), ('last_name', 'Smith'), ('email', 'da
vesmith@gmail.com')])
```

```
OrderedDict([('first_name', 'Jane'), ('last_name', 'Stuart'), ('email', 'j
anestuart@gmail.com')])
OrderedDict([('first_name', 'Tom'), ('last_name', 'Wright'), ('email', 'to
mwright@gmail.com')])
OrderedDict([('first_name', 'Steve'), ('last_name', 'Robinson'), ('email',
 'steverobinson@gmail.com')])
OrderedDict([('first_name', 'Nicole'), ('last_name', 'Jacobs'), ('email',
'nicolejacobs@gmail.com')])
OrderedDict([('first_name', 'Jane'), ('last_name', 'Wright'), ('email', 'j
anewright@gmail.com')])
OrderedDict([('first_name', 'Jane'), ('last_name', 'Doe'), ('email', 'jane
doe@gmail.com')])
OrderedDict([('first_name', 'Kurt'), ('last_name', 'Wright'), ('email', 'k
urtwright@gmail.com')])
OrderedDict([('first_name', 'Kurt'), ('last_name', 'Robinson'), ('email',
'kurtrobinson@gmail.com')])
OrderedDict([('first_name', 'Jane'), ('last_name', 'Jenkins'), ('email', '
janejenkins@gmail.com')])
OrderedDict([('first_name', 'Neil'), ('last_name', 'Robinson'), ('email',
'neilrobinson@gmail.com')])
OrderedDict([('first_name', 'Tom'), ('last_name', 'Patterson'), ('email',
'tompatterson@gmail.com')])
OrderedDict([('first_name', 'Sam'), ('last_name', 'Jenkins'), ('email', 's
amjenkins@gmail.com')])
OrderedDict([('first_name', 'Steve'), ('last_name', 'Stuart'), ('email', '
stevestuart@gmail.com')])
OrderedDict([('first_name', 'Maggie'), ('last_name', 'Patterson'), ('email
', 'maggiepatterson@gmail.com')])
OrderedDict([('first_name', 'Maggie'), ('last_name', 'Stuart'), ('email',
'maggiestuart@gmail.com')])
OrderedDict([('first_name', 'Jane'), ('last_name', 'Doe'), ('email', 'jane
doe@gmail.com')])
OrderedDict([('first_name', 'Steve'), ('last_name', 'Patterson'), ('email'
, 'stevepatterson@gmail.com')])
OrderedDict([('first_name', 'Dave'), ('last_name', 'Smith'), ('email', 'da
vesmith@gmail.com')])
OrderedDict([('first_name', 'Sam'), ('last_name', 'Wilks'), ('email', 'sam
wilks@gmail.com')])
OrderedDict([('first_name', 'Kurt'), ('last_name', 'Jefferson'), ('email',
 'kurtjefferson@gmail.com')])
OrderedDict([('first_name', 'Sam'), ('last_name', 'Stuart'), ('email', 'sa
mstuart@gmail.com')])
OrderedDict([('first_name', 'Jane'), ('last_name', 'Stuart'), ('email', 'j
anestuart@gmail.com')])
OrderedDict([('first_name', 'Dave'), ('last_name', 'Davis'), ('email', 'da
vedavis@gmail.com')])
OrderedDict([('first_name', 'Sam'), ('last_name', 'Patterson'), ('email',
'sampatterson@gmail.com')])
OrderedDict([('first_name', 'Tom'), ('last_name', 'Jefferson'), ('email',
'tomjefferson@gmail.com')])
OrderedDict([('first_name', 'Jane'), ('last_name', 'Stuart'), ('email', 'j
anestuart@gmail.com')])
OrderedDict([('first_name', 'Maggie'), ('last_name', 'Jefferson'), ('email
', 'maggiejefferson@gmail.com')])
OrderedDict([('first_name', 'Mary'), ('last_name', 'Wilks'), ('email', 'ma
rywilks@gmail.com')])
OrderedDict([('first_name', 'Neil'), ('last_name', 'Patterson'), ('email',
```

```
 'neilpatterson@gmail.com')])
OrderedDict([('first_name', 'Corey'), ('last_name', 'Davis'), ('email', 'c
oreydavis@gmail.com')])
OrderedDict([('first_name', 'Steve'), ('last_name', 'Jacobs'), ('email', '
stevejacobs@gmail.com')])
OrderedDict([('first_name', 'Jane'), ('last_name', 'Jenkins'), ('email', '
janejenkins@gmail.com')])
OrderedDict([('first_name', 'John'), ('last_name', 'Jacobs'), ('email', 'j
ohnjacobs@gmail.com')])
OrderedDict([('first_name', 'Neil'), ('last_name', 'Smith'), ('email', 'ne
ilsmith@gmail.com')])
OrderedDict([('first_name', 'Corey'), ('last_name', 'Wilks'), ('email', 'c
oreywilks@gmail.com')])
OrderedDict([('first_name', 'Corey'), ('last_name', 'Smith'), ('email', 'c
oreysmith@gmail.com')])
OrderedDict([('first_name', 'Mary'), ('last_name', 'Patterson'), ('email',
 'marypatterson@gmail.com')])
OrderedDict([('first_name', 'Jane'), ('last_name', 'Stuart'), ('email', 'j
anestuart@gmail.com')])
OrderedDict([('first_name', 'Travis'), ('last_name', 'Arnold'), ('email',
'travisarnold@gmail.com')])
OrderedDict([('first_name', 'John'), ('last_name', 'Robinson'), ('email',
'johnrobinson@gmail.com')])
OrderedDict([('first_name', 'Travis'), ('last_name', 'Arnold'), ('email',
'travisarnold@gmail.com')])
```

In [14]:
```
with open('names.csv','r') as f:
    csv_reader = csv.DictReader(f)

    with open('dictcsvout.csv','w') as wf:

        # you must pass field names to write with DictWriter method
        fieldnames = ['first_name','last_name','email']

        csv_write = csv.DictWriter(wf,fieldnames=fieldnames)

        for line in csv_reader:
            # You can remove the column through del() func
            del line['email']
            csv_write.writerow(line)
```

Reading and writing JSON

In [18]:
```
import json
```

In [26]:
```
# Lets create file Object
f = open('movie.txt','r')
```

In [27]:
```
# Lets pass the file object created to load() method so that
# all the json notations will be converted to python dictionary
movie=json.load(f)
```

In [29]:
```
# if you notice all the true became True and null became none in python di
```

```
ct
print(movie)
```

```
{'title': 'Vikram Vedha', 'release_year': 2017, 'is_awesome': True, 'won_o
scar': False, 'actors': ['R. Madhavan', 'Vijay Sethupathi', 'Varalaxmi Sar
athkumar', 'Shraddha Srinath', 'Kathir'], 'budget': None, 'credit': {'dire
ctor': 'Pushkar-Gayathri', 'producer': 'S. Sashikanth', 'Music': 'Sam C. S
.'}}
```

In [30]:
```
# you can access any element by key
movie['title']
```

Out[30]:
```
'Vikram Vedha'
```

You can also load the json from string

In [32]:
```
json_string = """
{
 "title": "Vikram Vedha",
 "release_year":2017,
 "is_awesome":true,
 "won_oscar":false,
 "actors":["R. Madhavan","Vijay Sethupathi","Varalaxmi Sarathkumar","Shrad
dha Srinath","Kathir"],
 "budget":null,
 "credit":{
                        "director":"Pushkar-Gayathri",
           "producer":"S. Sashikanth",
                   "Music":"Sam C. S."
    }
}
"""
```

In [33]:
```
# loads() method used to create a dictionary from json string
movie=json.loads(json_string)
```

In [34]:
```
print(movie)
```

```
{'title': 'Vikram Vedha', 'release_year': 2017, 'is_awesome': True, 'won_o
scar': False, 'actors': ['R. Madhavan', 'Vijay Sethupathi', 'Varalaxmi Sar
athkumar', 'Shraddha Srinath', 'Kathir'], 'budget': None, 'credit': {'dire
ctor': 'Pushkar-Gayathri', 'producer': 'S. Sashikanth', 'Music': 'Sam C. S
.'}}
```

You can dump your dictionary to json through dump method

In [35]:
```
wf = open('json_dump.txt','w')
```

In [37]:
```
# this will create new json file in your current directory.
json.dump(movie,wf)
```

Iterators and Generators

In this section of the course we will be learning about the difference between iteration and generation in Python and how to construct our own Generators with the *yield* statement. Generators allow us to generate as we go along, instead of holding everything in memory.

We've touch on this topic in the past when discussing the range() function in Python 2 and the similar xrange(), with the difference being the xrange() was a generator.

Lets explore a little deeper. We've learned how to create functions with **def** and the **return** statement. Generator functions allow us to write a function that can send back a value and then later resume to pick up where it left off. This type of function is a generator in Python, allowing us to generate a sequence of values over time. The main difference in syntax will be the use of a **yield** statement.

In most aspects, a generator function will appear very similar to a normal function. The main difference is when a generator function is compiled they become an object that support an iteration protocol. That means when they are called in your code the don't actually return a value and then exit, the generator functions will automatically suspend and resume their execution and state around the last point of value generation. The main advantage here is that instead of having to compute an entire series of values upfront and the generator functions can be suspended, this feature is known as *state suspension*.

To start getting a better understanding of generators, lets go ahead and see how we can create some.

In [2]:
```
# Generator function for the cube of numbers (power of 3)
def gencubes(n):
    for num in range(n):
        yield num**3
```

In [3]:
```
for x in gencubes(10):
    print(x)
```

```
0
1
8
27
64
125
216
343
512
729
```

Great! Now since we have a generator function we don't have to keep track of every single cube we created.

Generators are best for calculating large sets of results (particularly in calculations that involve loops

themselves) in cases where we don't want to allocate the memory for all of the results at the same time.

As we've noted in previous lectures (such as range()) many Standard Library functions that return lists in Python 2 have been modified to return generators in Python 3 because generators.

Lets create another example generator which calculates fibonacci numbers:

In [4]:
```
def genfibon(n):
    '''
    Generate a fibonnaci sequence up to n
    '''
    a = 1
    b = 1
    for i in range(n):
        yield a
        a,b = b,a+b
```

In [5]:
```
for num in genfibon(10):
    print(num)
```

```
1
1
2
3
5
8
13
21
34
55
```

What is this was a normal function, what would it look like?

In [6]:
```
def fibon(n):
    a = 1
    b = 1
    output = []

    for i in range(n):
        output.append(a)
        a,b = b,a+b

    return output
```

In [7]:
```
fibon(10)
```

Out[7]:
```
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

In [8]:
```
# Standard way to create a list with squared numbers

def square_nums(numbers):
    result= []
    for x in numbers:
```

```
        result.append(x**2)
    return result
```

In [11]:
```
squared = square_nums([1,2,3,4,5])
```

In [12]:
```
print(squared)
```

```
[1, 4, 9, 16, 25]
```

Creating Iterator through Generator

In [14]:
```
# if you see yield inplace of return then that is a generator which will r
eturn one value at a time
def square_nums_gen(numbers):
    for x in numbers:
        yield x**2
```

In [16]:
```
squared_gen = square_nums_gen([1,2,3,4,5])
```

In [17]:
```
for num in suared_gen:
    print(num)
```

```
1
4
9
16
25
```

Let's check performance of generator

In [18]:
```
# Executing standard way
import time
import sys
t1=time.clock()
squared = square_nums([x**2 for x in range(100000)])
t2=time.clock()
print('Time took to execute standard way :', t2-t1)
print('Memory help by squared ',sys.getsizeof(squared))
```

```
Time took to execute standard way : 0.2017408271200484
Memory help by squared  824464
```

In [20]:
```
# Executing generator Object
import time
import sys
t1=time.clock()
squared_gen = square_nums_gen([x**2 for x in range(100000)])
t2=time.clock()
print('Time took to execute standard way :', t2-t1)
print('Memory help by squared ',sys.getsizeof(squared_gen))
```

```
Time took to execute standard way : 0.09732002317585398
Memory help by squared  88
```

Notice! that generator object took only 0.097 seconds to run and occupied on 88 bit

you can check this with generator comprehension also

In [27]:
```
import time
import sys
t1=time.clock()
squared_gen = [x**2 for x in range(100000)]
t2=time.clock()
print('Time took to execute standard way :', t2-t1)
print('Memory help by squared ',sys.getsizeof(squared_gen))

import time
import sys
t1=time.clock()
squared_gen = (x**2 for x in range(100000))
t2=time.clock()
print('Time took to execute standard way :', t2-t1)
print('Memory help by squared ',sys.getsizeof(squared_gen))
```

```
Time took to execute standard way : 0.09476354668731801
Memory help by squared  824464
Time took to execute standard way : 0.006299103352375823
Memory help by squared  88
```

next() and iter() built-in functions

A key to fully understanding generators is the next function() and the iter() function.

The next function allows us to access the next element in a sequence. Lets check it out:

In [21]:
```
def simple_gen():
    for x in range(3):
        yield x
```

In [22]:
```
# Assign simple_gen
g = simple_gen()
```

In [23]:
```
next(g)
```

Out[23]: 0

In [24]:
```
next(g)
```

Out[24]: 1

In [25]:
```
next(g)
```

Out[25]: 2

In [26]:
```
next(g)
```

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
<ipython-input-26-5f315c5de15b> in <module>()
----> 1 next(g)

StopIteration:
```

After yielding all the values next() caused a StopIteration error. What this error informs us of is that all the values have been yielded.

You might be wondering that why don't we get this error while using a for loop? The for loop automatically catches this error and stops calling next.

Lets go ahead and check out how to use iter(). You remember that strings are iterables:

In [26]:
```
s = 'hello'

#Iterate over string
for let in s:
    print let
```

```
h
e
l
l
o
```

But that doesn't mean the string itself is an *iterator*! We can check this with the next() function:

In [27]:
```
next(s)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-27-bc0566bea448> in <module>()
----> 1 next(s)

TypeError: str object is not an iterator
```

Interesting, this means that a string object supports iteration, but we can not directly iterate over it as we could with a generator function. The iter() function allows us to do just that!

In [28]:
```
s_iter = iter(s)
```

In [29]:
```
next(s_iter)
```

Out[29]: 'h'

In [30]:
```
next(s_iter)
```

Out[30]: 'e'

Great! Now you know how to convert objects that are iterable into iterators themselves!

The main takeaway from this lecture is that using the yield keyword at a function will cause the function to become a generator. This change can save you a lot of memory for large use cases. For more information on generators check out:

Stack Overflow Answer

Another StackOverflow Answer

filter

The function filter(function, list) offers a convenient way to filter out all the elements of an iterable, for which the function returns True.

The function filter(function(),l) needs a function as its first argument. The function needs to return a Boolean value (either True or False). This function will be applied to every element of the iterable. Only if the function returns True will the element of the iterable be included in the result.

Lets see some examples:

In [6]:
```
#First let's make a function
def even_check(num):
    if num%2 ==0:
        return True
    else:
        return False
```

Now let's filter a list of numbers. Note: putting the function into filter without any parenthesis might feel strange, but keep in mind that functions are objects as well.

In [7]:
```
lst =range(20)

list(filter(even_check,lst))
```

Out[7]:
```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

filter() is more commonly used with lambda functions, this because we usually use filter for a quick job where we don't want to write an entire function. Lets repeat the example above using a lambda expression:

In [5]:
```
list(filter(lambda x: x%2==0,lst))
```

Out[5]:
```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

Great! You should now have a solid understanding of filter() and how to apply it to your code!

map()

map() is a function that takes in two arguments: a function and a sequence iterable. In the form: map(function, sequence)

The first argument is the name of a function and the second a sequence (e.g. a list). map() applies the function to all the elements of the sequence. It returns a new list with the elements changed by function.

When we went over list comprehension we created a small expression to convert Fahrenheit to Celsius. Let's do the same here but use map.

We'll start with two functions:

How will you square all the elements of a list?

In [5]:
```
l = [1,2,3,4,5,6,7,8]
```

In [4]:
```
def sqrt(l):
    output=[]
    for element in l:
        output.append(element**2)
    return output
```

In [6]:
```
sqrt(l)
```

Out[6]:
```
[1, 4, 9, 16, 25, 36, 49, 64]
```

The simple way

In [8]:
```
l = [1,2,3,4,5,6,7,8]
```

In [9]:
```
def sqrt(l):
    return [x**2 for x in l]
```

In [10]:
```
sqrt(l)
```

Out[10]:
```
[1, 4, 9, 16, 25, 36, 49, 64]
```

How about even more simple solution?

In [12]:
```
list(map(lambda x: x**2,l))
```

Out[12]:
```
[1, 4, 9, 16, 25, 36, 49, 64]
```

Python 2 Alert

In python map() - returned list by default but in python it returns iterator object . so you may required to cast as a list

Great! We got the same result! Using map is much more commonly used with lambda expressions since the entire purpose of map() is to save effort on having to create manual for loops.

map() can be applied to more than one iterable. The iterables have to have the same length.

For instance, if we are working with two lists-map() will apply its lambda function to the elements of the argument lists, i.e. it first applies to the elements with the 0th index, then to the elements with the 1st index until the n-th index is reached.

For example lets map a lambda expression to two lists:

Can i pass more than one iterator object to map?

ofcourse ,you can do it!

In [17]:
```
a = [1,2,3,4]
b = [5,6,7,8]


list(map(lambda x,y:x+y,a,b))
```

Out[17]:
```
[6, 8, 10, 12]
```

if you see above example, it takes 1st element from list a and b and adds together, then it takes 2nd element of both a and b and adds up.

Note

The list a and b should have same number of elements , otherwise truncation will happen. What truncation? What do you mean?

For example: if a has 4 elements and b has 2 elements, then it will process only 2 elements for map processing. it will ignore 3,4 like below

In [19]:
```
a = [1,2,3,4]
b = [5,6,]

list((map(lambda x,y: x+y,a,b)))
```

Out[19]:
```
[6, 8]
```

In [15]:
```
a = [1,2,3,4]
b = [5,6,7,8]
c = [9,10,11,12]
```

In [13]:
```
# Now all three lists
map(lambda x,y,z:x+y+z, a,b,c)
```

Out[13]:
```
[15, 18, 21, 24]
```

We can see in the example above that the parameter x gets its values from the list a, while y gets its values from b and z from list c. Go ahead and play with your own example to make sure you fully understand mapping to more than one iterable.

Great job! You should now have a basic understanding of the map() function.

reduce()

Many times students have difficulty understanding reduce() so pay careful attention to this lecture. The function reduce(function, sequence) continually applies the function to the sequence. It then returns a single value.

If seq = [s1, s2, s3, ... , sn], calling reduce(function, sequence) works like this:

- At first the first two elements of seq will be applied to function, i.e. func(s1,s2)
- The list on which reduce() works looks now like this: [function(s1, s2), s3, ... , sn]
- In the next step the function will be applied on the previous result and the third element of the list, i.e. function(function(s1, s2),s3)
- The list looks like this now: [function(function(s1, s2),s3), ... , sn]
- It continues like this until just one element is left and return this element as the result of reduce()

Lets see some examples:

Finding maximum value of a list - The Regular Way

In [5]:
```
l=[43,2,1,23,88,0]
```

In [6]:
```
def maximum(l):
    count=0
    while count < len(l)-1:
        if l[count] > l[count+1]:
            maxx = l[count]
            count+=1
        else:
            maxx= l[count+1]
            count+=1
    return maxx
```

In [7]:
```
maximum(l)
```

Out[7]: 88

Simple Solution with reduce()

In [3]:
```
from functools import reduce
lst =[47,11,42,13]
reduce(lambda x,y: x if x>y else b,lst)
```

Out[3]: 47

Adding all the elements of a list

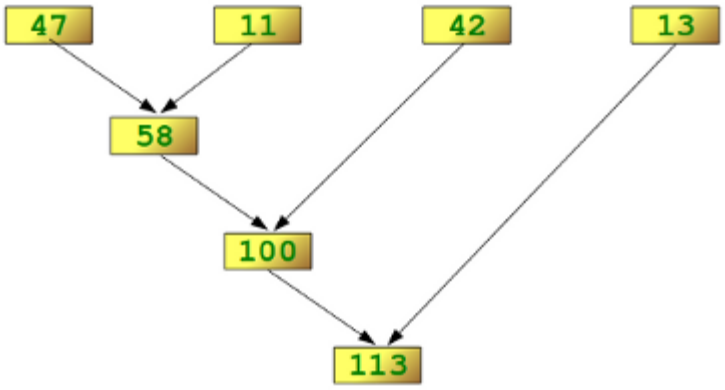
In [2]:
```
from functools import reduce
lst =[47,11,42,13]
reduce(lambda x,y: x+y,lst)
```

Out[2]: 113

Lets look at a diagram to get a better understanding of what is going on here:

In [9]:
```
from IPython.display import Image
Image('http://www.python-course.eu/images/reduce_diagram.png')
```

Out[9]:



Note how we keep reducing the sequence until a single final value is obtained. Lets see another example:

In [9]:
```
#Find the maximum of a sequence (This already exists as max())
max_find = lambda a,b: a if (a > b) else b
```

In [11]:
```
#Find max
reduce(max_find,lst)
```

Out[11]: 47

Hopefully you can see how useful reduce can be in various situations. Keep it in mind as you think about your code projects!

Object Oriented Programming

Object Oriented Programming (OOP) tends to be one of the major obstacles for beginners when they are first starting to learn Python.

There are many,many tutorials and lessons covering OOP so feel free to Google search other lessons, and I have also put some links to other useful tutorials online at the bottom of this Notebook.

For this lesson we will construct our knowledge of OOP in Python by building on the following topics:

- Objects
- Using the *class* keyword
- Creating class attributes
- Creating methods in a class
- Learning about Inheritance
- Learning about Multiple Inheritance
- Learning about Special Methods for classes

Lets start the lesson by remembering about the Basic Python Objects. For example:

In [1]:
```
l = [1,2,3]
```

Remember how we could call methods on a list?

In [3]:
```
l.count(2)
```

Out[3]:
```
1
```

What we will basically be doing in this lecture is exploring how we could create an Object type like a list. We've already learned about how to create functions. So lets explore Objects in general:

Objects

In Python, *everything is an object*. Remember from previous lectures we can use type() to check the type of object something is:

In [4]:
```
print type(1)
print type([])
print type(())
print type({})
```

```
<type 'int'>
<type 'list'>
<type 'tuple'>
<type 'dict'>
```

So we know all these things are objects, so how can we create our own Object types? That is where

the *class* keyword comes in.

class

The user defined objects are created using the class keyword. The class is a blueprint that defines a nature of a future object. From classes we can construct instances. An instance is a specific object created from a particular class. For example, above we created the object 'l' which was an instance of a list object.

Let see how we can use **class**:

In [2]:
```
# Create a new object type called Sample
class Sample(object):
    pass

# Instance of Sample
x = Sample()

print type(x)
```

```
<class '__main__.Sample'>
```

Note:

In Python 2.x you need pass in object parameter while creating class or new object, But for python 3.x you don't need to pass in object, internally when you create new object , the base class 'object' is included

Create a new object type called Sample

Removed Object parameter

In [35]:
```
class Sample:
    pass

# Instance of Sample
x = Sample()

print(type(x))
```

```
<class '__main__.Sample'>
```

By convention we give classes a name that starts with a capital letter. Note how x is now the reference to our new instance of a Sample class. In other words, we **instantiate** the Sample class.

Inside of the class we currently just have pass. But we can define class attributes and methods.

An **attribute** is a characteristic of an object. A **method** is an operation we can perform with the object.

For example we can create a class called Dog. An attribute of a dog may be its breed or its name, while a method of a dog may be defined by a .bark() method which returns a sound.

Let's get a better understanding of attributes through an example.

Attributes

The syntax for creating an attribute is:

```
self.attribute = something
```

There is a special method called:

```
__init__()
```

This method is used to initialize the attributes of an object. For example:

In [3]:
```
class Dog(object):
    def __init__(self,breed):
        self.breed = breed

sam = Dog(breed='Lab')
frank = Dog(breed='Huskie')
```

Lets break down what we have above.The special method

```
__init__()
```

is called automatically right after the object has been created:

```
def __init__(self, breed):
```

Each attribute in a class definition begins with a reference to the instance object. It is by convention named self. The breed is the argument. The value is passed during the class instantiation.

```
self.breed = breed
```

Let's create Doc string for an Object

In [27]:
```
class Dog(object):
    'Hey This is Just Dog Class or Dog Object'
    def __init__(self,breed):
        self.breed = breed

sam = Dog(breed='Lab')
frank = Dog(breed='Huskie')
```

Special methods available to check the characteristic of an Object

In [28]:
```
# To know the Docstring or Documentation of an Object
Dog.__doc__
```

Out[28]:
```
'Hey This is Just Dog Class or Dog Object'
```

In [29]:
```
# To know the name of an Object
Dog.__name__
```

Out[29]:
```
'Dog'
```

In [31]:
```
# To know the base classes for the derived classes

# It returns base classes list in tuple object format

Dog.__bases__
```

Out[31]:
```
(object,)
```

In [33]:
```
# Returns name of the module where the object got created
Dog.__module__
```

Out[33]:
```
'__main__'
```

Now we have created two instances of the Dog class. With two breed types, we can then access these attributes like this:

In [11]:
```
sam.breed
```

Out[11]:
```
'Lab'
```

In [9]:
```
frank.breed
```

Out[9]:
```
'Huskie'
```

Note how we don't have any parenthesis after breed, this is because it is an attribute and doesn't take any arguments.

In Python there are also *class object attributes*. These Class Object Attributes are the same for any instance of the class. For example, we could create the attribute *species* for the Dog class. Dogs (regardless of their breed,name, or other attributes will always be mammals. We apply this logic in the following manner:

In [3]:
```
class Dog(object):

    # Class Object Attribute
    species = 'mammal'

    def __init__(self,breed,name):
        self.breed = breed
        self.name = name
```

In [4]:
```
sam = Dog('Lab','Sam')
```

In [5]:
```
sam.name
```

Out[5]: 'Sam'

In [6]:
```
sam.species
```

Out[6]: 'mammal'

Note that the Class Object Attribute is defined outside of any methods in the class. Also by convention, we place them first before the init.

Built-in functions available for Object

1) getattr() - gets the value of attribute from Object

syntax: *getattr(Object Instance or Object, name of the attribute in string format)*

2) setattr() - sets the value for attribute if it is defined , else it will create new attribute and assigns the value you passed

syntax: *setattr(Object Instance or Object, name of the attribute in string format, Value for the attribute)*

3) hasattr() - Checks whether the attribute present in the object or not

syntax: *hasattr(Object Instance or Object, name of the attribute in string format)*

4) delattr() - Deletes the attribute

syntax: *delattr(Object Instance or Object, name of the attribute in string format)*

In [11]:
```
# get name attribute value

getattr(sam,'name')
```

Out[11]: 'Sam'

In [17]:
```
# Creating and assigning value to attribute

setattr(sam,'age',10)
```

In [18]:
```
# Modifying value of name attrubute

setattr(sam,'name','pepper')
```

In [15]:
```
# Checking whether age attibute present in sam object instance

# this function returns True or False

# True if the attribute available in object
hasattr(sam,'name')
```

Out[15]: True

In [22]:
```
# Help functions to know about Object and any Python objects

help(Dog)
```

```
Help on class Dog in module __main__:

class Dog(builtins.object)
 |  Methods defined here:
 |
 |  __init__(self, breed, name)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  species = 'mammal'
```

Methods

Methods are functions defined inside the body of a class. They are used to perform operations with the attributes of our objects. Methods are essential in encapsulation concept of the OOP paradigm. This is essential in dividing responsibilities in programming, especially in large applications.

You can basically think of methods as functions acting on an Object that take the Object itself into account through its *self* argument.

Lets go through an example of creating a Circle class:

In [6]:
```
class Circle(object):
    pi = 3.14

    # Circle get instantiated with a radius (default is 1)
    def __init__(self, radius=1):
        self.radius = radius

    # Area method calculates the area. Note the use of self.
    def area(self):
        return self.radius * self.radius * Circle.pi

    # Method for resetting Radius
    def setRadius(self, radius):
        self.radius = radius
```

```
    # Method for getting radius (Same as just calling .radius)
    def getRadius(self):
        return self.radius


c = Circle()

c.setRadius(2)
print('Radius is: ' +str(c.getRadius()))
print('Area is: '+str(c.area()))
```

```
Radius is: 2
Area is: 12.56
```

Great! Notice how we used self. notation to reference attributes of the class within the method calls. Review how the code above works and try creating your own method

Inheritance

Inheritance is a way to form new classes using classes that have already been defined. The newly formed classes are called derived classes, the classes that we derive from are called base classes. Important benefits of inheritance are code reuse and reduction of complexity of a program. The derived classes (descendants) override or extend the functionality of base classes (ancestors).

Lets see an example by incorporating our previous work on the Dog class:

In [5]:
```
class Animal(object):
    def __init__(self):
        print ("Animal created")

    def whoAmI(self):
        print ("Animal")

    def eat(self):
        print("Eating")


class Dog(Animal):
    def __init__(self):
        Animal.__init__(self)
        print("Dog created")

    def whoAmI(self):
        print("Dog")

    def bark(self):
        print("Woof!")
```

In [10]:
```
d = Dog()
```

```
Animal created
Dog created
```

In [25]:
```
d.whoAmI()
```

```
Dog
```

In [26]:
```
d.eat()
```

```
Eating
```

In [27]:
```
d.bark()
```

```
Woof!
```

In this example, we have two classes: Animal and Dog. The Animal is the base class, the Dog is the derived class.

The derived class inherits the functionality of the base class.

* It is shown by the eat() method.

The derived class modifies existing behavior of the base class.

- shown by the whoAmI() method.

Finally, the derived class extends the functionality of the base class, by defining a new bark() method.

Multiple Inheritance

In [36]:
```
class Martialart:
    def __init__(self,name):
        self.name = name
    def kunfu(self):
        print('i know kung fu')

    def karate(self):
        print('i know karate')
```

In [38]:
```
class Sports:
    def silambam(self):
        print('i know silambam')
```

In [40]:
```
# passing two base classes or parent classes

# Now the professional object can access all the methods and attributes av
ailable in Martialart and Sport classes

class professional(Martialart,Sports):
    pass
```

Special Methods

Finally lets go over special methods. Classes in Python can implement certain operations with special method names. These methods are not actually called directly but by Python specific language syntax. For example Lets create a Book class:

In [2]:
```
class Book(object):
    def __init__(self, title, author, pages):
        print ("A book is created")
        self.title = title
        self.author = author
        self.pages = pages

    def __str__(self):
        return ("Title:"+str(self.title)+" author:" +str(self.author) +" p
ages : " +str(self.pages))

    def __len__(self):
        return self.pages

    def __del__(self):
        print("A book is destroyed")
```

In [4]:
```
book = Book("The Power of Python", "Naveenkumar Murugan", 200)

#Special Methods
print (book)
print (len(book))
del book
```

```
A book is created
Title:The Power of Python author:Naveenkumar Murugan pages : 200
200
A book is destroyed
```

The __init__(), __str__(), __len__() and the __del__() methods.

These special methods are defined by their use of underscores. They allow us to use Python specific functions on objects created through our class.

Decorators

Decorators can be thought of as functions which modify the *functionality* of another function. They help to make your code shorter and more "Pythonic".

To properly explain decorators we will slowly build up from functions. Make sure to restart the Python and the Notebooks for this lecture to look the same on your own computer. So lets break down the steps:

Functions Review

In [1]:
```
def func():
    return 1
```

In [2]:
```
func()
```

Out[2]:
```
1
```

Scope Review

Remember from the nested statements lecture that Python uses Scope to know what a label is referring to. For example:

In [6]:
```
s = 'Global Variable'

def func():
    print locals()
```

Remember that Python functions create a new scope, meaning the function has its own namespace to find variable names when they are mentioned within the function. We can check for local variables and global variables with the local() and globals() functions. For example:

In [7]:
```
print globals()
```

```
{'_dh': [u'/Users/marci/Udemy-Complete-Python-Bootcamp'], '__': '', '_i':
u"s = 'Global Variable'\n\ndef func():\n    print locals()", 'quit': <IPyt
hon.core.autocall.ZMQExitAutocall object at 0x1037e0a10>, '__builtins__':
<module '__builtin__' (built-in)>, 's': 'Global Variable', '_ih': ['', u'd
ef func():\n    return 1', u'func()', u"s = 'Global Variable'\n\ndef func(
):\n    print locals()", u'print globals()', u'print globals().keys()', u"
s = 'Global Variable'\n\ndef func():\n    print locals()", u'print globals
()'], '__builtin__': <module '__builtin__' (built-in)>, '_2': 1, 'func': <
function func at 0x10445aa28>, '__name__': '__main__', '___': '', '_': 1,
'_sh': <module 'IPython.core.shadowns' from '//anaconda/lib/python2.7/site
-packages/IPython/core/shadowns.pyc'>, '_i7': u'print globals()', '_i6': u
"s = 'Global Variable'\n\ndef func():\n    print locals()", '_i5': u'print
 globals().keys()', '_i4': u'print globals()', '_i3': u"s = 'Global Variab
```

```
le'\n\ndef func():\n    print locals()", '_i2': u'func()', '_i1': u'def fu
nc():\n    return 1', '__doc__': 'Automatically created module for IPython
 interactive environment', '_iii': u'print globals()', 'exit': <IPython.co
re.autocall.ZMQExitAutocall object at 0x1037e0a10>, 'get_ipython': <bound
method ZMQInteractiveShell.get_ipython of <IPython.kernel.zmq.zmqshell.ZMQ
InteractiveShell object at 0x1037c3990>>, '_ii': u'print globals().keys()'
, 'In': ['', u'def func():\n    return 1', u'func()', u"s = 'Global Variab
le'\n\ndef func():\n    print locals()", u'print globals()', u'print globa
ls().keys()', u"s = 'Global Variable'\n\ndef func():\n    print locals()",
 u'print globals()'], '_oh': {2: 1}, 'Out': {2: 1}}
```

Here we get back a dictionary of all the global variables, many of them are predefined in Python. So let's go ahead and look at the keys:

In [8]:
```
print globals().keys()
```

```
['_dh', '__', '_i', 'quit', '__builtins__', 's', '_ih', '__builtin__', '_2
', 'func', '__name__', '___', '_', '_sh', '_i8', '_i7', '_i6', '_i5', '_i4
', '_i3', '_i2', '_i1', '__doc__', '_iii', 'exit', 'get_ipython', '_ii', '
In', '_oh', 'Out']
```

Note how **s** is there, the Global Variable we defined as a string:

In [10]:
```
globals()['s']
```

Out[10]:
```
'Global Variable'
```

Now lets run our function to check for any local variables in the func() (there shouldn't be any)

In [11]:
```
func()
```

```
{}
```

Great! Now lets continue with building out the logic of what a decorator is. Remember that in Python **everything is an object**. That means functions are objects which can be assigned labels and passed into other functions. Lets start with some simple examples:

In [1]:
```
def hello(name='Naveen'):
    return 'Hello '+name
```

In [2]:
```
hello()
```

Out[2]:
```
'Hello Naveen'
```

Assign a label to the function. Note that e are not using parentheses here because we are not calling the function hello, instead we are just putting it into the greet variable.

In [3]:
```
greet = hello
```

In [4]:
```
greet
```

Out[4]:
```
<function __main__.hello>
```

In [5]:
```
greet()
```

Out[5]:
```
'Hello Naveen'
```

This assignment is not attached to the original function:

In [6]:
```
del hello
```

In [7]:
```
hello()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-7-a803225a2f97> in <module>()
----> 1 hello()

NameError: name 'hello' is not defined
```

In [8]:
```
greet()
```

Out[8]:
```
'Hello Naveen'
```

Functions within functions

Great! So we've seen how we can treat functions as objects, now lets see how we can define functions inside of other functions:

In [19]:
```
def hello(name='Naveen'):
    print('The hello() function has been executed')

    def greet():
        return '\t This is inside the greet() function'

    def welcome():
        return "\t This is inside the welcome() function"

    print(greet())
    print(welcome())
    print("Now we are back inside the hello() function")
```

In [20]:
```
hello()
```

```
The hello() function has been executed
         This is inside the greet() function
         This is inside the welcome() function
Now we are back inside the hello() function
```

In [21]:
```
welcome()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last
```

```
)
<ipython-input-21-efaf77b113fd> in <module>()
----> 1 welcome()

NameError: name 'welcome' is not defined
```

Note how due to scope, the welcome() function is not defined outside of the hello() function. Now lets learn about returning functions from within functions:

Returning Functions

In [22]:
```
def hello(name='Naveen'):

    def greet():
        return '\t This is inside the greet() function'

    def welcome():
        return "\t This is inside the welcome() function"

    if name == 'Naveen':
        return greet
    else:
        return welcome
```

In [23]:
```
x = hello()
```

Now lets see what function is returned if we set x = hello(), note how the closed parenthesis means that name has been defined as Naveen.

In [33]:
```
x
```

Out[33]:
```
<function __main__.hello.<locals>.greet>
```

Great! Now we can see how x is pointing to the greet function inside of the hello function.

In [26]:
```
print(x())
```

```
        This is inside the greet() function
```

Lets take a quick look at the code again.

In the if/else clause we are returning greet and welcome, not greet() and welcome().

This is because when you put a pair of parentheses after it, the function gets executed; whereas if you don't put parenthesis after it, then it can be passed around and can be assigned to other variables without executing it.

When we write x = hello(), hello() gets executed and because the name is Naveen by default, the function greet is returned. If we change the statement to x = hello(name = "Sam") then the welcome function will be returned. We can also do print hello()() which outputs now you are in the greet() function.

Functions as Arguments

Now lets see how we can pass functions as arguments into other functions:

In [29]:
```
def hello():
    return 'Hi Naveen!'

def other(func):
    print('Other code would go here')
    print(func())
```

In [30]:
```
other(hello)
```

```
Other code would go here
Hi Naveen!
```

Great! Note how we can pass the functions as objects and then use them within other functions. Now we can get started with writing our first decorator:

Creating a Decorator

In the previous example we actually manually created a Decorator. Here we will modify it to make its use case clear:

In [46]:
```
def new_decorator(func):

    def wrap_func():
        print "Code would be here, before executing the func"

        func()

        print "Code here will execute after the func()"

    return wrap_func

def func_needs_decorator():
    print "This function is in need of a Decorator"
```

In [47]:
```
func_needs_decorator()
```

```
This function is in need of a Decorator
```

In [50]:
```
# Reassign func_needs_decorator
func_needs_decorator = new_decorator(func_needs_decorator)
```

In [51]:
```
func_needs_decorator()
```

```
Code would be here, before executing the func
This function is in need of a Decorator
Code here will execute after the func()
```

So what just happened here? A decorator simple wrapped the function and modified its behavior. Now lets understand how we can rewrite this code using the @ symbol, which is what Python uses for Decorators:

In [52]:
```
@new_decorator
def func_needs_decorator():
    print "This function is in need of a Decorator"
```

In [53]:
```
func_needs_decorator()
```

```
Code would be here, before executing the func
This function is in need of a Decorator
Code here will execute after the func()
```

*args vs kwargs

In [43]:
```
### When you're not sure how many variables or arguments your function or
method about to receive,
### you can pass *args (required argument) *kwargs (keyword arguement).
### so that your function will be generic to accept variable number of arg
uments
```

In [44]:
```
def simple(*args):
    for arg in args:
        print(arg)
```

In [46]:
```
#calling the simple function that accepts many number of arguments
simple(1,'string',[1,2,3,4])
```

```
1
string
[1, 2, 3, 4]
```

In [47]:
```
# It can also run without argument too
simple()
```

In [50]:
```
def simple_kwarg(**kwargs):
    for key in kwargs:
        print(kwargs[key])
```

In [51]:
```
simple_kwarg(name='Naveenkumar',company='DXC')
```

```
Naveenkumar
DXC
```

Real time Examples

In [38]:
```
### What you need to do if you need log each and every functions informati
on on a logfile?
```

In [35]:
```
def my logger(orig func):
```

```
    import logging
    logging.basicConfig(filename='{}.log'.format(orig_func.__name__), leve
l=logging.INFO)

    def wrapper(*args, **kwargs):
        logging.info(
            'Ran with args: {}, and kwargs: {}'.format(args, kwargs))
        orig_func(*args, **kwargs)

    return wrapper
```

In [36]:
```
@my_logger
def display_info(name,company):
    print(name,company)
```

In [37]:
```
display_info('naveen','DXC')
```

```
naveen DXC
```

In [40]:
```
### What you need to do if you need to know execution of all your function
s?
```

In [39]:
```
def my_timer(orig_func):
    import time


    def wrapper(*args, **kwargs):
        t1 = time.clock()
        orig_func(*args, **kwargs)
        t2 = time.clock() - t1
        print('{} ran in: {} sec'.format(orig_func.__name__, t2))


    return wrapper
```

In [41]:
```
@my_timer
def display_sleep():
    import time
    time.sleep(3)
    print('display func executed')
```

In [42]:
```
display_sleep()
```

```
display func executed
display_sleep ran in: 3.009298187847528 sec
```

In [52]:
```

```

Out[52]:
```
'C:\\Users\\murugnav\\Complete-Python-Bootcamp-master'
```

Great! You've now built a Decorator manually and then saw how we can use the @ symbol in Python to automate this and clean our code. You'll run into Decorators a lot if you begin using Python for Web Development, such as Flask or Django!

Regular Expressions

Regular expressions are text matching patterns described with a formal syntax. You'll often hear regular expressions referred to as 'regex' or 'regexp' in conversation. Regular expressions can include a variety of rules, fro finding repetition, to text-matching, and much more. As you advance in Python you'll see that a lot of your parsing problems can be solved with regular expressions (they're also a common interview question!).

If you're familiar with Perl, you'll notice that the syntax for regular expressions are very similar in Python. We will be using the re module with Python for this lecture.

Let's get started!

Searching for Patterns in Text

One of the most common uses for the re module is for finding patterns in text. Let's do a quick example of using the search method in the re module to find some text:

In [4]:
```
import re

# List of patterns to search for
patterns = [ 'term1', 'term2' ]

# Text to parse
text = 'This is a string with term1, but it does not have the other term.'

for pattern in patterns:
    print('Searching for {} in: {}'.format(pattern, text))

    #Check for match
    if re.search(pattern,  text):
        print('\n')
        print('Match was found. \n')
    else:
        print('\n')
        print('No Match was found.\n')
```

```
Searching for term1 in: This is a string with term1, but it does not have
the other term.


Match was found.

Searching for term2 in: This is a string with term1, but it does not have
the other term.


No Match was found.
```

Now we've seen that re.search() will take the pattern, scan the text, and then returns a **Match** object. If no pattern is found, a **None** is returned. To give a clearer picture of this match object, check out the cell below:

In [6]:
```
# List of patterns to search for
pattern = 'term1'

# Text to parse
text = 'This is a string with term1, but it does not have the other term.'

match = re.search(pattern,  text)

type(match)
```

Out[6]:
```
_sre.SRE_Match
```

This **Match** object returned by the search() method is more than just a Boolean or None, it contains information about the match, including the original input string, the regular expression that was used, and the location of the match. Let's see the methods we can use on the match object:

In [7]:
```
# Show start of match
match.start()
```

Out[7]:
```
22
```

In [8]:
```
# Show end
match.end()
```

Out[8]:
```
27
```

Split with regular expressions

Let's see how we can split with the re syntax. This should look similar to how you used the split() method with strings.

In [9]:
```
# Term to split on
split_term = '@'

phrase = 'What is the domain name of someone with the email: hello@gmail.com'

# Split the phrase
re.split(split_term,phrase)
```

Out[9]:
```
['What is the domain name of someone with the email: hello', 'gmail.com']
```

Note how re.split() returns a list with the term to spit on removed and the terms in the list are a split up version of the string. Create a couple of more examples for yourself to make sure you understand!

Finding all instances of a pattern

You can use re.findall() to find all the instances of a pattern in a string. For example:

In [10]:
```
# Returns a list of all matches
re.findall('match','test phrase match is in middle')
```

Out[10]:
```
['match']
```

Pattern re Syntax

This will be the bulk of this lecture on using re with Python. Regular expressions supports a huge variety of patterns the just simply finding where a single string occurred.

We can use *metacharacters* along with re to find specific types of patterns.

Since we will be testing multiple re syntax forms, let's create a function that will print out results given a list of various regular expressions and a phrase to parse:

In [12]:
```
def multi_re_find(patterns,phrase):
    '''
    Takes in a list of regex patterns
    Prints a list of all matches
    '''
    for pattern in patterns:
        print('Searching the phrase using the re check:',pattern)
        print(re.findall(pattern,phrase))
        print('\n')
```

Repetition Syntax

There are five ways to express repetition in a pattern:

```
1.) A pattern followed by the meta-character * is repeated zero or m
ore times.
2.) Replace the * with + and the pattern must appear at least once.
3.) Using ? means the pattern appears zero or one time.
4.) For a specific number of occurrences, use {m} after the pattern,
 where m is replaced with the number of times         the pattern sh
ould repeat.
5.) Use {m,n} where m is the minimum number of repetitions and n is
the maximum. Leaving out n ({m,}) means the           value appears
at least m times, with no maximum.
```

Now we will see an example of each of these using our multi_re_find function:

In [13]:
```
test_phrase = 'sdsd..sssddd...sdddsddd...dsds...dssssss...sdddd'

test_patterns = [ 'sd*',     # s followed by zero or more d's
                  'sd+',          # s followed by one or more d's
                  'sd?',          # s followed by zero or one d's
```

```
                'sd{3}',        # s followed by three d's
                'sd{2,3}',      # s followed by two to three d's
                ]

multi_re_find(test_patterns,test_phrase)
```

```
Searching the phrase using the re check: sd*
['sd', 'sd', 's', 's', 'sddd', 'sddd', 'sddd', 'sd', 's', 's', 's', 's', '
s', 's', 'sdddd']


Searching the phrase using the re check: sd+
['sd', 'sd', 'sddd', 'sddd', 'sddd', 'sd', 'sdddd']


Searching the phrase using the re check: sd?
['sd', 'sd', 's', 's', 'sd', 'sd', 'sd', 'sd', 's', 's', 's', 's', 's', 's
', 'sd']


Searching the phrase using the re check: sd{3}
['sddd', 'sddd', 'sddd', 'sddd']


Searching the phrase using the re check: sd{2,3}
['sddd', 'sddd', 'sddd', 'sddd']
```

Character Sets

Character sets are used when you wish to match any one of a group of characters at a point in the input. Brackets are used to construct character set inputs. For example: the input [ab] searches for occurrences of either a or b. Let's see some examples:

In [14]:
```
test_phrase = 'sdsd..sssddd...sdddsddd...dsds...dssss...sdddd'

test_patterns = [ '[sd]',    # either s or d
            's[sd]+']   # s followed by one or more s or d


multi_re_find(test_patterns,test_phrase)
```

```
Searching the phrase using the re check: [sd]
['s', 'd', 's', 'd', 's', 's', 's', 'd', 'd', 'd', 's', 'd', 'd', 'd', 's'
, 'd', 'd', 'd', 'd', 's', 'd', 's', 'd', 's', 's', 's', 's', 's', 's', 'd
', 'd', 'd', 'd']


Searching the phrase using the re check: s[sd]+
['sdsd', 'sssddd', 'sdddsddd', 'sds', 'sssss', 'sdddd']
```

It makes sense that the first [sd] returns every instance. Also the second input will just return any

thing starting with an s in this particular case of the test phrase input.

Exclusion

We can use ^ to exclude terms by incorporating it into the bracket syntax notation. For example: [^...] will match any single character not in the brackets. Let's see some examples:

In [ ]:
```
test_phrase = 'This is a string! But it has punctuation. How can we remove
 it?'
```

Use [^!.?] to check for matches that are not a !,.,?, or space. Add the + to check that the match appears at least once, this basically translate into finding the words.

In [56]:
```
re.findall('[^!.? ]+',test_phrase)
```

Out[56]:
```
['This',
 'is',
 'a',
 'string',
 'But',
 'it',
 'has',
 'punctutation',
 'How',
 'can',
 'we',
 'remove',
 'it']
```

Character Ranges

As character sets grow larger, typing every character that should (or should not) match could become very tedious. A more compact format using character ranges lets you define a character set to include all of the contiguous characters between a start and stop point. The format used is [start-end].

Common use cases are to search for a specific range of letters in the alphabet, such [a-f] would return matches with any instance of letters between a and f.

Let's walk through some examples:

In [65]:
```
test_phrase = 'This is an example sentence. Lets see if we can find some l
etters.'

test_patterns=[ '[a-z]+',      # sequences of lower case letters
                '[A-Z]+',      # sequences of upper case letters
                '[a-zA-Z]+',   # sequences of lower or upper case letters
                '[A-Z][a-z]+'] # one upper case letter followed by lower c
ase letters

multi_re_find(test_patterns,test_phrase)
```

```
Searching the phrase using the re check: '[a-z]+'
['his', 'is', 'an', 'example', 'sentence', 'ets', 'see', 'if', 'we', 'can'
, 'find', 'some', 'letters']


Searching the phrase using the re check: '[A-Z]+'
['T', 'L']


Searching the phrase using the re check: '[a-zA-Z]+'
['This', 'is', 'an', 'example', 'sentence', 'Lets', 'see', 'if', 'we', 'ca
n', 'find', 'some', 'letters']


Searching the phrase using the re check: '[A-Z][a-z]+'
['This', 'Lets']
```

Escape Codes

You can use special escape codes to find specific types of patterns in your data, such as digits, non-digits,whitespace, and more. For example:

Code	Meaning
\d	a digit
\D	a non-digit
\s	whitespace (tab, space, newline, etc.)
\S	non-whitespace
\w	alphanumeric
\W	non-alphanumeric

Escapes are indicated by prefixing the character with a backslash (). Unfortunately, a backslash must itself be escaped in normal Python strings, and that results in expressions that are difficult to read. Using raw strings, created by prefixing the literal value with r, for creating regular expressions eliminates this problem and maintains readability.

Personally, I think this use of r to escape a backslash is probably one of the things that block someone who is not familiar with regex in Python from being able to read regex code at first. Hopefully after seeing these examples this syntax will become clear.

In [68]:
```
test_phrase = 'This is a string with some numbers 1233 and a symbol #hasht
ag'

test_patterns=[ r'\d+', # sequence of digits
                r'\D+', # sequence of non-digits
                r'\s+', # sequence of whitespace
                r'\S+', # sequence of non-whitespace
                r'\w+', # alphanumeric characters
```

```
                r'\W+', # non-alphanumeric
                ]

multi_re_find(test_patterns,test_phrase)
```

```
Searching the phrase using the re check: '\\d+'
['1233']


Searching the phrase using the re check: '\\D+'
['This is a string with some numbers ', ' and a symbol #hashtag']


Searching the phrase using the re check: '\\s+'
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']


Searching the phrase using the re check: '\\S+'
['This', 'is', 'a', 'string', 'with', 'some', 'numbers', '1233', 'and', 'a
', 'symbol', '#hashtag']


Searching the phrase using the re check: '\\w+'
['This', 'is', 'a', 'string', 'with', 'some', 'numbers', '1233', 'and', 'a
', 'symbol', 'hashtag']


Searching the phrase using the re check: '\\W+'
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' #']
```

Among many other database Postgresql open source object-relational database is widely used along with Python applications as a backend for data storage and retrieval.

PostgreSQL is often viewed as more feature robust and stable when compared to MySQL, SQLServer and Oracle. All of those databases are reasonable choices. However, because PostgreSQL tends to be used by Python developers the drivers and example code for using the database tend to be better documented and contain fewer bugs for typical usage scenarios

Advantages of Postgresql:

```
1.User-defined types
2.Table inheritance
3.Sophisticated locking mechanism
4.Foreign key referential integrity
5.Views, rules, subquery
6.Nested transactions (savepoints)
7.Multi-version concurrency control (MVCC)
8.Asynchronous replication
```

Prerequisites:

1. You should have postgresql server up and running on local.

Download Postgresql

Once downlowded, double click on the file to start the installation wizard.

Once installation completed, you can to 'all programs' on windows and check the installation.

You can manage postgresql through:

- Graphical Database management tool : pgadmin 4
- Psql - shell for managing postgresql

Two Things to remember:

> It will ask for password for the superuser postgres, so remember the password as its required to connect to database.
>
> Enter the port for PostgreSQL. Make sure that no other applications are using this port. Leave it as default if you are unsure.
>
> if you still need the guidence for installing postgresql, then you can check out this article

2. You should install the PostgreSQL Database Adapter-psycopg2.

Many other postgres adapters are available for python but most developers prefer psycopg as he psycopg2 database adapter implemented in C as a libpq wrapper resulting in both fast and secure.

Go to Anaconda prompt and install psycopg2 adapter by : pip install psycopg2.

> You could also install through windows command prompt ,if you defined path for anaconda in evironment variable.
>
> Remember: if you're connected to HPE or DXC network it will throw you SSL exception. So, whenever you want install some pakacges make sure you're not connected to office network

3. Finally create your own python program to do database operations.

In [ ]:
```
# First, We should create a database through Pgadmin or psql

# You create a database just with the following comment

#On windows application search - type in psql.In that prompt execute this query

#Usually, superuser details will be fetched for this db such as user,password.

CREATE DATABASE SAMPLE;
```

Importing database adapter

In [1]:
```
# You can import the postgres adapter to connect python with postgresql database
import psycopg2
```

Pass in the following parameters to method connect() to create a database connection

- *dbname*: the database name
- *database*: the database name (only as keyword argument)
- *user*: user name used to authenticate
- *password*: password used to authenticate
- *host*: database host address (defaults to UNIX socket if not provided)
- *port*: connection port number (defaults to 5432 if not provided)

In [2]:
```
try:
    conn = psycopg2.connect("host=localhost dbname=sample user=postgres password=DXC@1989")

except psycopg2.DatabaseError as e:
    print('Unable to connect!\n{0}',(e))

else:
```

```
    print('Database connected')
```

```
Database connected
```

In [3]:
```
#We obtained connection object, now its time to create cusrsor to execute PostgreSQL command in a database session
cur = conn.cursor()

print(cur.connection)
```

```
<connection object at 0x0000024394E80EB8; dsn: 'user=postgres password=xxx
 dbname=sample host=localhost', closed: 0>
```

In [6]:
```
Create_Table ="""CREATE TABLE phonebook(phone bigint,
                                        firstname VARCHAR(32),
                                        lastname VARCHAR(32),
                                        address VARCHAR(64),
                                        join_date date)
"""
```

In [7]:
```
try:
    cur.execute(Create_Table)

except psycopg2.DatabaseError as e:
    print(e)
    conn.rollback()
else:
    conn.commit()
    print('Created table and commited')
```

```
Created table and commited
```

In [17]:
```
### Inserting a row after formatting using format method
```

In [8]:
```
Insert_one_row="""INSERT INTO phonebook(phone,firstname,lastname,address,join_date)
                  VALUES {}""".format(('9710410808','NAVEENKUMAR','MRUGUAN','CHENNAI','2015-11-04'))
```

In [10]:
```
try:
    cur.execute(Insert_one_row)
except psycopg2.DatabaseError as e:
    print(e)
    conn.rollback()
else:
    print('No of Rows commited',cur.rowcount)
    conn.commit()
```

```
No of Rows commited 1
```

In [27]:
```
# Execute query with bound vars.
```

In [11]:
```
query="""INSERT INTO phonebook(phone,firstname,lastname,address,join_date)
```

```
                VALUES (%s,%s,%s,%s,%s)"""
values = ['9791240058','RAJKUMAR','YOGA','SALEM','2015-11-8']

try:
    cur.execute(query,values)
except psycopg2.DatabaseError as e:
    print(e)
    conn.rollback()
else:
    print('No of Rows commited',cur.rowcount)
    conn.commit()
```

```
No of Rows commited 1
```

In [46]:
```
# Inserting Multiple Values
```

In [12]:
```
mass_insert =[('9710410808','NAVEENKUMAR','MURUGAN','CHENNAI','2015-11-04'
),
              ('9710419898','DINESHKUMAR','MURUGAN','CHENNAI','2015-11-18'
),
              ('9710417777','SURESHKUMAR','GANESH','BANGALORE','2015-11-29
'),
              ('9176617114','RAVIKUMAR','MAHESH','CHENNAI','2015-11-7'),
              ('9791240058','RAJKUMAR','YOGA','SALEM','2015-11-8')
             ]
try:
    cur.executemany("INSERT INTO phonebook(phone,firstname,lastname,addres
s,join_date) VALUES (%s,%s,%s,%s,%s)",mass_insert)
except psycopg2.DatabaseError as e:
    print(e)
    conn.rollback()
else:
    conn.commit()
    print('No of rows commited to table is :',cur.rowcount)
```

```
No of rows commited to table is : 5
```

In [47]:
```
#csvfaker -r 100 phone_number first_name last_name address date_time > pho
nebook.csv
```

In [ ]:
```
# Capturing Duplicate Key
```

In [52]:
```
try:

    cur.executemany("INSERT INTO phonebook(phone,firstname,lastname,addres
s,join_date) VALUES (%s,%s,%s,%s,%s)",mass_insert)

except psycopg2.DatabaseError as e:
    print(e)
    conn.rollback()

else:
    print('No of Rows inserted',cur.rowcount)
    conn.commit()
```

```
duplicate key value violates unique constraint "phonebook_pkey"
DETAIL:  Key (phone)=(9710410808) already exists.
```

In [16]:
```
Select = """SELECT * FROM phonebook """
```

In [ ]:
```
# Fetch All
```

In [19]:
```
try:
    cur.execute(Select)
except Exception as e:
    print(e)
    conn.rollback()
else:
    print("Select successfully")
    print(cur.rowcount)
    records = cur.fetchall()
```

```
Select successfully
8
(9710410808, 'NAVEENKUMAR', 'MRUGUAN', 'CHENNAI', datetime.date(2015, 11,
4))
(9710410808, 'NAVEENKUMAR', 'MRUGUAN', 'CHENNAI', datetime.date(2015, 11,
4))
(9791240058, 'RAJKUMAR', 'YOGA', 'SALEM', datetime.date(2015, 11, 8))
(9710410808, 'NAVEENKUMAR', 'MURUGAN', 'CHENNAI', datetime.date(2015, 11,
4))
(9710419898, 'DINESHKUMAR', 'MURUGAN', 'CHENNAI', datetime.date(2015, 11,
18))
(9710417777, 'SURESHKUMAR', 'GANESH', 'BANGALORE', datetime.date(2015, 11,
 29))
(9176617114, 'RAVIKUMAR', 'MAHESH', 'CHENNAI', datetime.date(2015, 11, 7))
(9791240058, 'RAJKUMAR', 'YOGA', 'SALEM', datetime.date(2015, 11, 8))
```

In [25]:
```
try:
    cur.execute(Select)
except Exception as e:
    print(e)
    conn.rollback()
else:
    print("Select successfully")
    print(cur.rowcount)
    print(cur.fetchmany(2))
```

```
Select successfully
8
[(9710410808, 'NAVEENKUMAR', 'MRUGUAN', 'CHENNAI', datetime.date(2015, 11,
 4)), (9710410808, 'NAVEENKUMAR', 'MRUGUAN', 'CHENNAI', datetime.date(2015
, 11, 4))]
```

In [24]:
```

```

Out[24]:
```
[]
```

In [ ]:
```
# Update
```

In [26]:
```
Update = "UPDATE phonebook set phone=%s where firstname=%s;"
```

In [28]:
```
try:
    cur.execute(Update,('9176617114','NAVEENKUMAR'))
except Exception as e:
    print(e)
    conn.rollback()
else:
    print("Update successfully")
    print(cur.rowcount)
```

```
Update successfully
3
```

In [29]:
```
conn.commit()
```

In [ ]:
```
# Delete
```

In [30]:
```
Delete = "DELETE FROM phonebook where phone=%s"
```

In [32]:
```
try:
    cur.execute(Delete,('9791240058',))
except Exception as e:
    print(e)

else:
    print("Delete successfully")
    print(cur.rowcount)
    conn.commit()
```

```
Delete successfully
2
```

In [21]:
```
conn.set_isolation_level(0)
```

In [71]:
```
import config
```

In [72]:
```
help(config)
```

```
Help on module config:

NAME
    config

FUNCTIONS
    config(filename='database.ini', section='postgresql')

FILE
    c:\users\murugnav\config.py
```

In [ ]:
```
# You can store the all the connection details in any initialization file
(ex datatabase.ini)
# and use the
```

In [35]:
```
import psycopg2
from config import config


def create_tables():
    """ create tables in the PostgreSQL database"""
    commands = (
        """
        CREATE TABLE vendors (
            vendor_id SERIAL PRIMARY KEY,
            vendor_name VARCHAR(255) NOT NULL
        )
        """,
        """ CREATE TABLE parts (
                part_id SERIAL PRIMARY KEY,
                part_name VARCHAR(255) NOT NULL
                )
        """,
        """
        CREATE TABLE part_drawings (
                part_id INTEGER PRIMARY KEY,
                file_extension VARCHAR(5) NOT NULL,
                drawing_data BYTEA NOT NULL,
                FOREIGN KEY (part_id)
                REFERENCES parts (part_id)
                ON UPDATE CASCADE ON DELETE CASCADE
        )
        """,
        """
        CREATE TABLE vendor_parts (
                vendor_id INTEGER NOT NULL,
                part_id INTEGER NOT NULL,
                PRIMARY KEY (vendor_id , part_id),
                FOREIGN KEY (vendor_id)
                    REFERENCES vendors (vendor_id)
                    ON UPDATE CASCADE ON DELETE CASCADE,
                FOREIGN KEY (part_id)
                    REFERENCES parts (part_id)
                    ON UPDATE CASCADE ON DELETE CASCADE
        )
        """)
    conn = None
    try:
        # read the connection parameters
        params = config()
        # connect to the PostgreSQL server
        conn = psycopg2.connect(**params)
        cur = conn.cursor()
        # create table one by one
        for command in commands:
```

```
            cur.execute(command)
        # close communication with the PostgreSQL database server
        cur.close()
        # commit the changes
        conn.commit()
    except (Exception, psycopg2.DatabaseError) as error:
        print(error)
    finally:
        if conn is not None:
            conn.close()
```

In [36]:
```
#calling the function defined
create_tables()
```

Web Scraping In Python

Web scraping is a computer software technique of extracting information from websites. This technique mostly focuses on the transformation of unstructured data (HTML format) on the web into structured data (database or spreadsheet).

Why Web Scraping?

Web scraping is used for contact scraping, and as a component of applications used for web indexing, web mining and data mining, online price change monitoring and price comparison, product review scraping (to watch the competition), gathering real estate listings, weather data monitoring, website change detection, research, tracking online presence and reputation, web mashup and, web data integration.

Real time Examples:

Google Crawls other websites for Indexing

DailyHunt - Scraps all news websites and reformats the content.

MySmartPrice - Scrapes all the ecommerce websites (price comparision)

Python Packages Required for Web Scraping

urllib - To fetch source code of URL

Parser - html.parser or lxml parser or html5lib parser

Beautifulsoup4 - Beautiful Soup is a Python library for pulling data out of HTML and XML files.

How to install Parsers to parse HTML or XML source?

html.parser - Included in python library itself so you don't need to install it separately.

lxml parser - pip install lxml

html5lib - pip install html5lib

This table summarizes the advantages and disadvantages of each parser library:

Parser	Typical usage	Advantages	Disadvantages
Python's html.parser	`BeautifulSoup(markup, "html.parser")`	• Batteries included • Decent speed • Lenient (as of Python 2.7.3 and 3.2.)	• Not very lenient (before Python 2.7.3 or 3.2.2)
lxml's HTML parser	`BeautifulSoup(markup, "lxml")`	• Very fast • Lenient	• External C dependency
lxml's XML parser	`BeautifulSoup(markup, "lxml-xml")` `BeautifulSoup(markup, "xml")`	• Very fast • The only currently supported XML parser	• External C dependency
html5lib	`BeautifulSoup(markup, "html5lib")`	• Extremely lenient • Parses pages the same way a web browser does • Creates valid HTML5	• Very slow • External Python dependency

How to install Beautifulsoup4

pip install beautifulsoup4

To know more about BeautifulSoup, check documention

In [1]:
```
import urllib.request as urllib2
```

In [2]:
```
html_doc = urllib2.urlopen('https://en.wikipedia.org/wiki/Web_scraping').read()
```

In [3]:
```
# Raw HTML Source fetched from URL using urllib
print(html_doc)
```

b'<!DOCTYPE html>\n<html class="client-nojs" lang="en" dir="ltr">\n<head>\n<meta charset="UTF-8"/>\n<title>Web scraping - Wikipedia</title>\n<script>document.documentElement.className = document.documentElement.className.replace(/(^|\\s)client-nojs(\\s|$)/, "$1client-js$2");</script>\n<script>(window.RLQ=window.RLQ||[]).push(function(){mw.config.set({"wgCanonicalNamespace":"","wgCanonicalSpecialPageName":false,"wgNamespaceNumber":0,"wgPageName":"Web_scraping","wgTitle":"Web scraping","wgCurRevisionId":825057514,"wgRevisionId":825057514,"wgArticleId":2696619,"wgIsArticle":true,"wgIsRedirect":false,"wgAction":"view","wgUserName":null,"wgUserGroups":["*"],"wgCategories":["CS1 Danish-language sources (da)","Articles needing additional references from June 2017","All articles needing additional references","Articles with limited geographic scope from October 2015","USA-centric","Pages using div col with deprecated parameters","Web scraping","World Wide Web","Spamming"],"wgBreakFrames":false,"wgPageContentLanguage":"en","wgPageContentModel":"wikitext","wgSeparatorTransformTable":["",""],"wgDigitTransformTable":["",""],"wgDefaultDateFormat":"dmy","wgMonthNames":["","January","February","March","April","May","June","July","August","September","October","November","December"],"wgMonthNamesShort":["","Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"],"wgRelevantPageName":"Web_scraping","wgRelevantArticleId":2696619,"wgRequestId":"WpOzcApAMFQAAKv1HOgAAABY","wgIsProbablyEditable":true,"wgRelevantPageIsProbablyEditable":true,"wgRestrictionEdit":[],"wgRestrictionMove":[],"wgFlaggedRevsParams":{"tags":{}},"wgStableRevisionId":null,"wgWikiEditorEnabledModules":[],"wgBetaFeaturesFeatures":[],"wgMediaViewerOnClick":true,"wgMediaViewerEnabledByDefault":true,"wgPopupsShouldSendModuleToUser":true,"wgPopupsConflictsWithNavPopupGadget":false,"wgVisualEditor":{"pageLanguageCode":"en","pageLanguageDir":"ltr","pageVariantFallbacks":"en","usePageImages":true,"usePageDescriptions":true},"wgPreferredVariant":"en","wgMFExpandAllSectionsUserOption":true,"wgMFEnableFontChanger":true,"wgMFDisplayWikibaseDescriptions":{"search":false,"nearby":false,"watchlist":false,"tagline":false},"wgRelatedArticles":null,"wgRelatedArticlesUseCirrusSearch":true,"wgRelatedArticlesOnlyUseCirrusSearch":false,"wgULSCurrentAutonym":"English","wgNoticeProject":"wikipedia","wgCentralNoticeCookiesToDelete":[],"wgCentralNoticeCategoriesUsingLegacy":["Fundraising","fundraising"],"wgCategoryTreePageCategoryOptions":"{\\"mode\\":0,\\"hideprefix\\":20,\\"showcount\\":true,\\"namespaces\\":false}","wgWikibaseItemId":"Q665452","wgScoreNoteLanguages":{"arabic":"\xd8\xa7\xd9\x84\xd8\xb9\xd8\xb1\xd8\xa8\xd9\x8a\xd8\xa9","catalan":"catal\xc3\xa0","deutsch":"Deutsch","english":"English","espanol":"espa\xc3\xb1ol","italiano":"italiano","nederlands":"Nederlands","norsk":"norsk","portugues":"portugu\xc3\xaas","suomi":"suomi","svenska":"svenska","vlaams":"West-Vlams"},"wgScoreDefaultNoteLanguage":"nederlands","wgCentralAuthMobileDomain":false,"wgCodeMirrorEna

bled":false,"wgVisualEditorToolbarScrollOffset":0,"wgVisualEditorUnsupportedEditParams":["undo","undoafter","veswitched"],"wgEditSubmitButtonLabelPublish":true});mw.loader.state({"ext.gadget.charinsert-styles":"ready","ext.globalCssJs.user.styles":"ready","ext.globalCssJs.site.styles":"ready","site.styles":"ready","noscript":"ready","user.styles":"ready","user":"ready","user.options":"ready","user.tokens":"loading","ext.cite.styles":"ready","wikibase.client.init":"ready","ext.visualEditor.desktopArticleTarget.noscript":"ready","ext.uls.interlanguage":"ready","ext.wikimediaBadges":"ready","mediawiki.legacy.shared":"ready","mediawiki.legacy.commonPrint":"ready","mediawiki.sectionAnchor":"ready","mediawiki.skinning.interface":"ready","skins.vector.styles":"ready","ext.globalCssJs.user":"ready","ext.globalCssJs.site":"ready"});mw.loader.implement("user.tokens@1dqfd7l",function($,jQuery,require,module){/*@nomin*/mw.user.tokens.set({"editToken":"+\\\\","patrolToken":"+\\\\","watchToken":"+\\\\","csrfToken":"+\\\\"});\n});mw.loader.load(["ext.cite.a11y","site","mediawiki.page.startup","mediawiki.user","mediawiki.hidpi","mediawiki.page.ready","mediawiki.toc","mediawiki.searchSuggest","ext.gadget.teahouse","ext.gadget.ReferenceTooltips","ext.gadget.watchlist-notice","ext.gadget.DRN-wizard","ext.gadget.charinsert","ext.gadget.refToolbar","ext.gadget.extra-toolbar-buttons","ext.gadget.switcher","ext.centralauth.centralautologin","mmv.head","mmv.bootstrap.autostart","ext.popups","ext.visualEditor.desktopArticleTarget.init","ext.visualEditor.targetLoader","ext.eventLogging.subscriber","ext.wikimediaEvents","ext.navigationTiming","ext.uls.eventlogger","ext.uls.init","ext.uls.interface","ext.3d","ext.centralNotice.geoIP","ext.centralNotice.startUp","skins.vector.js"]);});</script>\n<link rel="stylesheet" href="/w/load.php?debug=false&lang=en&modules=ext.cite.styles%7Cext.uls.interlanguage%7Cext.visualEditor.desktopArticleTarget.noscript%7Cext.wikimediaBadges%7Cmediawiki.legacy.commonPrint%2Cshared%7Cmediawiki.sectionAnchor%7Cmediawiki.skinning.interface%7Cskins.vector.styles%7Cwikibase.client.init&only=styles&skin=vector"/>\n<script async="" src="/w/load.php?debug=false&lang=en&modules=startup&only=scripts&skin=vector"></script>\n<meta name="ResourceLoaderDynamicStyles" content=""/>\n<link rel="stylesheet" href="/w/load.php?debug=false&lang=en&modules=ext.gadget.charinsert-styles&only=styles&skin=vector"/>\n<link rel="stylesheet" href="/w/load.php?debug=false&lang=en&modules=site.styles&only=styles&skin=vector"/>\n<meta name="generator" content="MediaWiki 1.31.0-wmf.22"/>\n<meta name="referrer" content="origin"/>\n<meta name="referrer" content="origin-when-crossorigin"/>\n<meta name="referrer" content="origin-when-cross-origin"/>\n<link rel="alternate" href="android-app://org.wikipedia/http/en.m.wikipedia.org/wiki/Web_scraping"/>\n<link rel="alternate" type="application/x-wiki" title="Edit this page" href="/w/index.php?title=Web_scraping&action=edit"/>\n<link rel="edit" title="Edit this page" href="/w/index.php?title=Web_scraping&action=edit"/>\n<link rel="apple-touch-icon" href="/static/apple-touch/wikipedia.png"/>\n<link rel="shortcut icon" href="/static/favicon/wikipedia.ico"/>\n<link rel="search" type="application/opensearchdescription+xml" href="/w/opensearch_desc.php" title="Wikipedia (en)"/>\n<link rel="EditURI" type="application/rsd+xml" href="//en.wikipedia.org/w/api.php?action=rsd"/>\n<link rel="license" href="//creativecommons.org/licenses/by-sa/3.0/"/>\n<link rel="canonical" href="https://en.wikipedia.org/wiki/Web_scraping"/>\n<link rel="dns-prefetch" href="//login.wikimedia.org"/>\n<link rel="dns-prefetch" href="//meta.wikimedia.org" />\n<!--[if lt IE 9]><script src="/w/load.php?debug=false&lang=en&modules=html5shiv&only=scripts&skin=vector&sync=1"></script><![endif]-->\n</head>\n<body class="mediawiki ltr sitedir-ltr mw-hide-empty-elt ns-0 ns-subject page-Web_scraping rootpage-Web_scraping skin-vector action-view">\t\t \n\t\t \n\t\t \n\t\t\t\n\t\t\t <!-- CentralNotice --> \n \n<h1 id="firstHeading" class="firstHeading" lang="en">Web scraping</h1>\t\t\t \n\t\t\t\t From Wikipedia, the free encyclopedia \t\t\t\t \n\t\t\t\t\t\t\t\t \n\t\t\t\t\tJump to:\t\t\t\tnavigation, \t\t\t\tsearch\n\t\t\t \n\t\t\t <table class="plainlinks metadata ambox ambox-content ambox-Refimprove" role="presentation">\n<tr>\n<td class="mbox-image">\n \n</td>\n<td class="mbox-text">\n This article needs additional citations for verification. Please help improve this article by adding citations to reliable s

ources. Unsourced material may be challenged and removed. <small>(June 2017)</small> <small class="hide-when-compact">(Learn how and when to remove this template message)</small> \n</td>\n</tr>\n</table>\n For a broader coverage related to this topic, see Data scraping. \n Web scraping, web harvesting, or web data extraction is data scraping used for extracting data from websites.^{[1]} Web scraping software may access the World Wide Web directly using the Hypertext Transfer Protocol, or through a web browser. While web scraping can be done manually by a software user, the term typically refers to automated processes implemented using a bot or web crawler. It is a form of copying, in which specific data is gathered and copied from the web, typically into a central local database or spreadsheet, for later retrieval or analysis. \n Web scraping a web page involves fetching it and extracting from it.^{[1]}^{[2]} Fetching is the downloading of a page (which a browser does when you view the page). Therefore, web crawling is a main component of web scraping, to fetch pages for later processing. Once fetched, then extraction can take place. The content of a page may be parsed, searched, reformatted, its data copied into a spreadsheet, and so on. Web scrapers typically take something out of a page, to make use of it for another purpose somewhere else. An example would be to find and copy names and phone numbers, or companies and their URLs, to a list (contact scraping). \n Web scraping is used for contact scraping, and as a component of applications used for web indexing, web mining and data mining, online price change monitoring and price comparison, product review scraping (to watch the competition), gathering real estate listings, weather data monitoring, website change detection, research, tracking online presence and reputation, web mashup and, web data integration. \n Web pages are built using text-based mark-up languages (HTML and XHTML), and frequently contain a wealth of useful data in text form. However, most web pages are designed for human end-users and not for ease of automated use. Because of this, tool kits that scrape web content were created. A web scraper is an Application Programming Interface (API) to extract data from a web site. Companies like Amazon AWS and Google provide web scraping tools, services and public data available free of cost to end users. \n Newer forms of web scraping involve listening to data feeds from web servers. For example, JSON is commonly used as a transport storage mechanism between the client and the web server. \n There are methods that some websites use to prevent web scraping, such as detecting and disallowing bots from crawling (viewing) their pages. In response, there are web scraping systems that rely on using techniques in DOM parsing, computer vision and natural language processing to simulate human browsing to enable gathering web page content for offline parsing. \n \n \n \n<h2>Contents</h2>\n \n
\n<li class="toclevel-1 tocsection-1">1 Techniques\n
\n<li class="toclevel-2 tocsection-2">1.1 Human copy-and-paste
\n<li class="toclevel-2 tocsection-3">1.2 Text pattern matching
\n<li class="toclevel-2 tocsection-4">1.3 HTTP programming
\n<li class="toclevel-2

tocsection-5">1.4 HTML parsing
\n<li class="toclevel-2 tocsection-6">1.5 DOM parsing
\n<li class="toclevel-2 tocsection-7">1.6 Vertical aggregation
\n<li class="toclevel-2 tocsection-8">1.7 Semantic annotation recognizing
\n<li class="toclevel-2 tocsection-9">1.8 Computer vision web-page analysis
\n
\n
\n<li class="toclevel-1 tocsection-10">2 Software\n
\n<li class="toclevel-2 tocsection-11">2.1 Example tools\n
\n<li class="toclevel-3 tocsection-12">2.1.1 Javascript tools
\n<li class="toclevel-3 tocsection-13">2.1.2 SaaS version
\n<li class="toclevel-3 tocsection-14">2.1.3 Web crawling frameworks
\n
\n
\n
\n
\n<li class="toclevel-1 tocsection-15">3 Legal issues\n
\n<li class="toclevel-2 tocsection-16">3.1 United States
\n<li class="toclevel-2 tocsection-17">3.2 The EU
\n<li class="toclevel-2 tocsection-18">3.3 Australia
\n
\n
\n<li class="toclevel-1 tocsection-19">4 Methods to prevent web scraping
\n<li class="toclevel-1 tocsection-20">5 See also
\n<li class="toclevel-1 tocsection-21">6 References
\n
\n \n \n<h2>Techniques[edit]</h2>\n Web scraping is the process of automatically mining data or collecting information from the World Wide Web. It is a field with active developments sharing a common goal with the semantic web vision, an ambitious initiative that still requires breakthroughs in text processing, semantic understanding, artificial intelligence and human-computer interactions. Current web scraping solutions range from the ad-hoc, requiring human effort, to fully automated systems that are able to convert entire web sites into structured information, with limitations. \n<h3>Human copy-and-paste[edit]</h3>\n Sometimes even the best web-scraping technology cannot replace a human\xe2\x80\x99s manual examination and copy-and-paste, and sometimes this may be the only workable solution when the websites for scraping explicitly set up barriers to prevent machine automation. \n<h3>Text pattern matching[edit]</h3>\n A simple yet powerful approach to extract information from web pages can be based on the UNIX grep command or regular expression-matching facilities of programming languages (for instance Perl or Python). \n<h3>HTTP programming[edit]</h3>\n Static and dynamic web pages can be retrieved by posting HTTP requests to the remote web server using socket programming. \n<h3>HTML parsing[

edit]</h3>\n Many websites have large collections of pages generated dynamically from an underlying structured source like a database. Data of the same category are typically encoded into similar pages by a common script or template. In data mining, a program that detects such templates in a particular information source, extracts its content and translates it into a relational form, is called a wrapper. Wrapper generation algorithms assume that input pages of a wrapper induction system conform to a common template and that they can be easily identified in terms of a URL common scheme.^{[3]} Moreover, some semi-structured data query languages, such as XQuery and the HTQL, can be used to parse HTML pages and to retrieve and transform page content. \n<h3>DOM parsing[edit]</h3>\n Further information: Document Object Model \n By embedding a full-fledged web browser, such as the Internet Explorer or the Mozilla browser control, programs can retrieve the dynamic content generated by client-side scripts. These browser controls also parse web pages into a DOM tree, based on which programs can retrieve parts of the pages. \n<h3>Vertical aggregation[edit]</h3>\n There are several companies that have developed vertical specific harvesting platforms. These platforms create and monitor a multitude of \xe2\x80\x9cbots\xe2\x80\x9d for specific verticals with no "man in the loop" (no direct human involvement), and no work related to a specific target site. The preparation involves establishing the knowledge base for the entire vertical and then the platform creates the bots automatically. The platform\'s robustness is measured by the quality of the information it retrieves (usually number of fields) and its scalability (how quick it can scale up to hundreds or thousands of sites). This scalability is mostly used to target the Long Tail of sites that common aggregators find complicated or too labor-intensive to harvest content from. \n<h3>Semantic annotation recognizing[edit]</h3>\n The pages being scraped may embrace metadata or semantic markups and annotations, which can be used to locate specific data snippets. If the annotations are embedded in the pages, as Microformat does, this technique can be viewed as a special case of DOM parsing. In another case, the annotations, organized into a semantic layer,^{[4]} are stored and managed separately from the web pages, so the scrapers can retrieve data schema and instructions from this layer before scraping the pages. \n<h3>Computer vision web-page analysis[edit]</h3>\n There are efforts using machine learning and computer vision that attempt to identify and extract information from web pages by interpreting pages visually as a human being might.^{[5]} \n<h2>Software[edit]</h2>\n There are many software tools available that can be used to customize web-scraping solutions. This software may attempt to automatically recognize the data structure of a page or provide a recording interface that removes the necessity to manually write web-scraping code, or some scripting functions that can be used to extract and transform content, and database interfaces that can store the scraped data in local databases. Some web scraping software can also be used to extract data from an API directly. \n<h3>Example tools[<a href="/w/index.php?title=Web_scra

ping&action=edit&section=11" title="Edit section: Example tools">edit]</h3>\n
\n
cURL \xe2\x80\x93 command line tool and library for transferring (including getting) data with URLs supporting a wide range of HTTP methods (GET, POST, cookies, etc.)
\n
Data Toolbar \xe2\x80\x93 web scraping add-on for Internet Explorer, Mozilla Firefox, and Google Chrome Web browsers that collects and converts structured data from web pages into a tabular format that can be loaded into a spreadsheet or database management program.
\n
Diffbot \xe2\x80\x93 uses computer vision and machine learning to automatically extract data from web pages by interpreting pages visually as a human being might.
\n
Heritrix \xe2\x80\x93 gets pages (lots of them). It is a web crawler designed for web archiving, written by the Internet Archive (see Wayback Machine).
\n
HtmlUnit \xe2\x80\x93 headless browser that can be used for retrieving web pages, web scraping, and more.
\n
HTTrack \xe2\x80\x93 free and open source Web crawler and offline browser, designed to download websites.
\n
iMacros \xe2\x80\x93a browser extension to record, code, share and replay browser automation (javascript)
\n
Kantu \xe2\x80\x93 uses screenshots and OCR for scraping
\n
Selenium (software) \xe2\x80\x93 a portable software-testing framework for web applications
\n
Jaxer
\n
nokogiri
\n
OutWit Hub \xe2\x80\x93 Web scraping application including built-in data, image, document extractors and editors for custom scrapers and automatic exploration and extraction jobs (free and paid versions)
\n
watir \xe2\x80\x93
\n
Wget \xe2\x80\x93 computer program that retrieves content from web servers. It is part of the GNU Project. It supports downloading via the HTTP, HTTPS, and FTP protocols.
\n
WSO2 Mashup Server \xe2\x80\x93
\n
Yahoo! Query Language (YQL) \xe2\x80\x93
\n
Data Scraping Studio \xe2\x80\x93 Stand alone windows desktop software to scrape data using CSS selectors and REGEX.
\n
\n<h4>Javascript tools[edit]</h4>\n
\n
Greasemonkey
\n
Node.js
\n
PhantomJS \xe2\x80\x93 scripted, headless browser used for automating web page interaction.
\n
jQuery
\n
\n<h4>SaaS version[edit]</h4>\n
\n
Agenty \xe2\x80\x93 SaaS solution, paid versions available from $29 (06/09/17)
\n
Apify \xe2\x80\x93 Web scraping and automation platform, free and paid versions available (10/10/17)
\n
dexi.io \xe2\x80\x93 SaaS solution, free and paid versions available from $119 USD (31/10/17)
\n
diggernaut.com \xe2\x80\x93 Turn websites into datasets, free and paid (from $9.99 USD) subscriptions available (02/05/18)
\n
fScraper \xe2\x80\x93 Facebook friendly scraper, SaaS solution, free and paid versions available
\n
Import.io \xe2\x80\x93 SaaS solution, paid versions available from $299 USD (06/09/17)
\n
Listly.io \xe2\x80\x93 HTML to Excel in seconds, free SaaS service (04/10/17)
\n
Mozenda \xe2\x80\x93 SaaS solution, is a web-based platform for web data extraction (01/22/18)
\n
uScraper \xe2\x80\x93 SaaS service, free and paid versions available. Functionality primarily for scraping email addresses.
\n
\n<h4>Web crawling frameworks[edit]</h4>\n These can be used to build web scrapers. \n
\n
Scrapy
\n
\n<h2>Legal issue

s[edit]</h2>\n<table class="plainlinks metadata ambox ambox-content ambox-globalize" role="presentation">\n<tr>\n<td class="mbox-image">\n \n</td>\n<td class="mbox-text">\n The examples and perspective in this article deal primarily with the United States and do not represent a worldwide view of the subject. You may improve this article, discuss the issue on the talk page, or create a new article, as appropriate. <small>(October 2015)</small> <small class="hide-when-compact">(Learn how and when to remove this template message)</small> \n</td>\n</tr>\n</table>\n The legality of web scraping varies across the world. In general, web scraping may be against the terms of use of some websites, but the enforceability of these terms is unclear.^{[6]} \n<h3>United States[edit]</h3>\n In the United States, website owners can use three major legal claims to prevent undesired web scraping: (1) copyright infringement (compilation), (2) violation of the Computer Fraud and Abuse Act (\xe2\x80\x9cCFAA\xe2\x80\x9d), and (3) trespass to chattel.^{[7]} However, the effectiveness of these claims relies upon meeting various criteria, and the case law is still evolving. For example, with regard to copyright, while outright duplication of original expression will in many cases be illegal, in the United States the courts ruled in Feist Publications v. Rural Telephone Service that duplication of facts is allowable. \n U.S. courts have acknowledged that users of "scrapers" or "robots" may be held liable for committing trespass to chattels,^{[8]} ^{[9]} which involves a computer system itself being considered personal property upon which the user of a scraper is trespassing. The best known of these cases, eBay v. Bidder\'s Edge, resulted in an injunction ordering Bidder\'s Edge to stop accessing, collecting, and indexing auctions from the eBay web site. This case involved automatic placing of bids, known as auction sniping. However, in order to succeed on a claim of trespass to chattels, the plaintiff must demonstrate that the defendant intentionally and without authorization interfered with the plaintiff\'s possessory interest in the computer system and that the defendant\'s unauthorized use caused damage to the plaintiff. Not all cases of web spidering brought before the courts have been considered trespass to chattels.^{[10]} \n One of the first major tests of screen scraping involved American Airlines (AA), and a firm called FareChase.^{[11]} AA successfully obtained an injunction from a Texas trial court, stopping FareChase from selling software that enables users to compare online fares if the software also searches AA\'s website. The airline argued that FareChase\'s web search software trespassed on AA\'s servers when it collected the publicly available data. FareChase filed an appeal in March 2003. By June, FareChase and AA agreed to settle and the appeal was dropped.^{[12]} \n Southwest Airlines has also challenged screen-scraping practices, and has involv

ed both FareChase and another firm, Outtask, in a legal claim. Southwest Airlines charged that the screen-scraping is Illegal since it is an example of "Computer Fraud and Abuse" and has led to "Damage and Loss" and "Unauthorized Access" of Southwest\'s site. It also constitutes "Interference with Business Relations", "Trespass", and "Harmful Access by Computer". They also claimed that screen-scraping constitutes what is legally known as "Misappropriation and Unjust Enrichment", as well as being a breach of the web site\'s user agreement. Outtask denied all these claims, claiming that the prevailing law in this case should be US Copyright law, and that under copyright, the pieces of information being scraped would not be subject to copyright protection. Although the cases were never resolved in the Supreme Court of the United States, FareChase was eventually shuttered by parent company Yahoo!, and Outtask was purchased by travel expense company Concur.^{[13]} In 2012, a startup called 3Taps scraped classified housing ads from Craigslist. Craigslist sent 3Taps a cease-and-desist letter and blocked their IP addresses and later sued, in Craigslist v. 3Taps. The court held that the cease-and-desist letter and IP blocking was sufficient for Craigslist to properly claim that 3Taps had violated the Computer Fraud and Abuse Act. \n Although these are early scraping decisions, and the theories of liability are not uniform, it is difficult to ignore a pattern emerging that the courts are prepared to protect proprietary content on commercial sites from uses which are undesirable to the owners of such sites. However, the degree of protection for such content is not settled, and will depend on the type of access made by the scraper, the amount of information accessed and copied, the degree to which the access adversely affects the site owner\xe2\x80\x99s system and the types and manner of prohibitions on such conduct.^{[14]} \n While the law in this area becomes more settled, entities contemplating using scraping programs to access a public web site should also consider whether such action is authorized by reviewing the terms of use and other terms or notices posted on or made available through the site. In a 2010 ruling in the Cvent, Inc. v. Eventbrite, Inc. In the United States district court for the eastern district of Virginia, the court ruled that the terms of use should be brought to the users\' attention In order for a browse wrap contract or license to be enforced.^{[15]} In a 2014, filed in the United States District Court for the Eastern District of Pennsylvania,^{[16]} e-commerce site QVC objected to the Pinterest-like shopping aggregator Resultly\xe2\x80\x99s `scraping of QVC\xe2\x80\x99s site for real-time pricing data. QVC alleges that Resultly \xe2\x80\x9cexcessively crawled\xe2\x80\x9d QVC\xe2\x80\x99s retail site (allegedly sending 200-300 search requests to QVC\xe2\x80\x99s website per minute, sometimes to up to 36,000 requests per minute) which caused QVC\'s site to crash for two days, resulting in lost sales for QVC.^{[17]} QVC\'s complaint alleges that the defendant disguised its web crawler to mask its source IP address and thus prevented QVC from quickly repairing the problem. This is a particularly interesting scraping case because QVC is seeking damages for the unavailability of their website, which QVC claims was caused by Resultly. \n In the plaintiff\'s web site during the period of this trial the terms of use link is displayed among all the links of the site, at the bottom of the page as most sites on the internet. This ruling contradicts the Irish ruling described below. The court also rejected the plaintiff\'s argument that the browse wrap restrictions were enforceable in view of Virginia\'s adoption of the Uniform Computer Information Transactions Act (UCITA)\xe2\x80\x94a uniform law that many believed was in favor on common browse wrap contracting practices.^{[18]} \n In Facebook, Inc. v. Power Ventures, Inc., a district court ruled in 2012 that Power Ventures could not scrape Facebook pages on behalf of a Facebook user. The case is on appeal, and the Electronic Frontier Foundation filed a brief in 2015 asking that it be overturned.^{[19]}^{[20]} In Associated Press v. Meltwater U.S. Holdings, Inc., a court in the US held Meltwater liable for scraping and republishing news information from the Associated Press, but a court in the United Kingdom held in favor of Meltwater. \n<h3><span class="mw-headline" id

="The_EU">The EU[edit]</h3>\n Outside of the United States, in February 2006, the Danish Maritime and Commercial Court (Copenhagen) ruled that systematic crawling, indexing, and deep linking by portal site ofir.dk of estate site Home.dk does not conflict with Danish law or the database directive of the European Union.^{[21]} \n In a February 2010 case complicated by matters of jurisdiction, Ireland\'s High Court delivered a verdict that illustrates the inchoate state of developing case law. In the case of Ryanair Ltd v Billigfluege.de GmbH, Ireland\'s High Court ruled Ryanair\'s "click-wrap" agreement to be legally binding. In contrast to the findings of the United States District Court Eastern District of Virginia and those of the Danish Maritime and Commercial Court, Mr. Justice Michael Hanna ruled that the hyperlink to Ryanair\'s terms and conditions was plainly visible, and that placing the onus on the user to agree to terms and conditions in order to gain access to online services is sufficient to comprise a contractual relationship. ^{[22]} The decision is under appeal in Ireland\'s Supreme Court. ^{[23]} \n<h3>Australia[edit]</h3>\n In Australia, the Spam Act 2003 outlaws some forms of web harvesting, although this only applies to email addresses.^{[24]}^{[25]} \n<h2>Methods to prevent web scraping[edit]</h2>\n The administrator of a website can use various measures to stop or slow a bot. Some techniques include: \n
\n
Blocking an IP address either manually or based on criteria such as geolocation and DNSRBL. This will also block all browsing from that address.
\n
Disabling any web service API that the website\'s system might expose.
\n
Bots sometimes declare who they are (using user agent strings) and can be blocked on that basis using robots.txt; \'googlebot\' is an example. Other bots make no distinction between themselves and a human using a browser.
\n
Bots can be blocked by monitoring excess traffic
\n
Bots can sometimes be blocked with tools to verify that it is a real person accessing the site, like a CAPTCHA. Bots are sometimes coded to explicitly break specific CAPTCHA patterns or may employ third-party services that utilize human labor to read and respond in real-time to CAPTCHA challenges.
\n
Commercial anti-bot services: Companies offer anti-bot and anti-scraping services for websites. A few web application firewalls have limited bot detection capabilities as well. However, many such solutions are not very effective^{[26]}.
\n
\n
\n
Locating bots with a honeypot or other method to identify the IP addresses of automated crawlers.
\n
Obfuscation using CSS sprites to display such data as phone numbers or email addresses, at the cost of accessibility to screen reader users.
\n<li>Because bots rely on consistency in the front-end code of a target website, adding small variations to the HTML/CSS surrounding important data and navigation elements would require more human involvement in the initial set up of a bot and if done effectively may render the target website too difficult to scrape due to the diminished ability to automate the scraping process.
\n
Websites can declare if crawling is allowed or not in the robots.txt file and allow partial access, limit the crawl rate, specify the optimal time to crawl and more.
\n
\n<h2>See also[<a href="/w/index.php?title=Web_scraping&action=edit

&section=20" title="Edit section: See also">edit]</h2>\n \n
\n
Archive.is
\n
Comparison of feed aggregators
\n
Data scraping
\n
Data wrangling
\n
Importer
\n
Job wrapping
\n
Knowledge extraction
\n
OpenSocial
\n
Scraper site
\n
Fake news website
\n
Blog scraping
\n
Spamdexing
\n
Domain name drop list
\n
Text corpus
\n
Web archiving
\n
Blog network
\n
Search Engine Scraping
\n
Web crawlers
\n
\n \n<h2>References[edit]</h2>\n \n<ol class="references">\n<li id="cite_note-Boeing2016JPER-1">^ ^a ^b <cite class="citation journal">Boeing, G.; Waddell, P. (2016). "New Insights into Rental Housing Markets across the United States: Web Scraping and Analyzing Craigslist Rental Listings". Journal of Planning Education and Research (0739456X16664789). doi:10.1177/0739456X16664789.</cite><span title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&rft.genre=article&rft.jtitle=Journal+of+Planning+Education+and+Research&rft.atitle=New+Insights+into+Rental+Housing+Markets+across+the+United+States%3A+Web+Scraping+and+Analyzing+Craigslist+Rental+Listings&rft.issue=0739456X16664789&rft.date=2016&rft_id=info%3Adoi%2F10.1177%2F0739456X16664789&rft.aulast=Boeing&rft.aufirst=G.&rft.au=Waddell%2C+P.&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping" class="Z3988"> \n<li id="cite_note-2">^ <cite class="citation journal">Vargiu & Urru (2013). "Exploiting web scraping in a collaborative filtering- based approach to web advertising". Artificial Intelligence Research. 2 (1). doi:10.5430/air.v2n1p44.</cite><span title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&rft.genre=article&rft.jtitle=Artificial+Intelligence+Research&rft.atitle=Exploiting+web+scraping+in+a+collaborative+filtering-+based+approach+to+web+advertising&rft.volume=2&rft.issue=1&rft.date=2013&rft_id=info%3Adoi%2F10.5430%2Fair.v2n1p44&rft.au=Vargiu+%26+Urru&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping" class="Z3988"> \n<li id="cite_note-3">^ <cite class="citation journal">Song, Ruihua; Microsoft Research (Sep 14, 2007). "Joint Optimization of Wrapper Generation and Template Detection" (PDF). The 13th International Conference on Knowledge Discovery and Data Mining.</cite><span title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&rft.genre=article&rft.jtitle=The+13th+International+Conference+on+Knowledge+Discovery+and+Data+Mining&rft.atitle=Joint+Optimization+of+Wrapper+Generation+and+Template+Detection&rft.date=2007-09-14&rft.aulast=Song&rft.aufirst=Ruihua&rft.au=Microsoft+Research&rft_id=https%3A%2F%2Fpdfs.semanticscholar.org%2F4fb4%2F3c5a212df751e84c3b2f8d29fabfe56c3616.pdf&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping" class="Z3988"> \n<li

id="cite_note-4">^ Semantic annotation based web scraping
\n<li id="cite_note-5">^ <cite class="citation web">Roush, Wade (2012-07-25). "Diffbot Is Using Computer Vision to Reinvent the Semantic Web". www.xconomy.com. Retrieved 2013-03-15.</cite><span title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=Diffbot+Is+Using+Computer+Vision+to+Reinvent+the+Semantic+Web&rft.pub=www.xconomy.com&rft.date=2012-07-25&rft.aulast=Roush&rft.aufirst=Wade&rft_id=http%3A%2F%2Fwww.xconomy.com%2Fsan-francisco%2F2012%2F07%2F25%2Fdiffbot-is-using-computer-vision-to-reinvent-the-semantic-web%2F&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping" class="Z3988"> \n<li id="cite_note-6">^ <cite class="citation web">"FAQ about linking \xe2\x80\x93 Are website terms of use binding contracts?". www.chillingeffects.org. 2007-08-20. Archived from the original on 2002-03-08. Retrieved 2007-08-20.</cite><span title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=FAQ+about+linking+%E2%80%93+Are+website+terms+of+use+binding+contracts%3F&rft.pub=www.chillingeffects.org&rft.date=2007-08-20&rft_id=http%3A%2F%2Fwww.chillingeffects.org%2Flinking%2Ffaq.cgi%23QID596&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping" class="Z3988"> \n<li id="cite_note-7">^ <cite class="citation journal">Kenneth, Hirschey, Jeffrey (2014-01-01). "Symbiotic Relationships: Pragmatic Acceptance of Data Scraping". Berkeley Technology Law Journal. 29 (4). doi:10.15779/Z38B39B. ISSN 1086-3818.</cite><span title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&rft.genre=article&rft.jtitle=Berkeley+Technology+Law+Journal&rft.atitle=Symbiotic+Relationships%3A+Pragmatic+Acceptance+of+Data+Scraping&rft.volume=29&rft.issue=4&rft.date=2014-01-01&rft_id=info%3Adoi%2F10.15779%2FZ38B39B&rft.issn=1086-3818&rft.aulast=Kenneth&rft.aufirst=Hirschey%2C+Jeffrey&rft_id=http%3A%2F%2Fscholarship.law.berkeley.edu%2Fbtlj%2Fvol29%2Fiss4%2F16%2F&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping" class="Z3988"> \n<li id="cite_note-8">^ <cite class="citation web">"Internet Law, Ch. 06: Trespass to Chattels". www.tomwbell.com. 2007-08-20. Retrieved 2007-08-20.</cite><span title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=Internet+Law%2C+Ch.+06%3A+Trespass+to+Chattels&rft.pub=www.tomwbell.com&rft.date=2007-08-20&rft_id=http%3A%2F%2Fwww.tomwbell.com%2FNetLaw%2FCh06.html&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping" class="Z3988"> \n<li id="cite_note-9">^ <cite class="citation web">"What are the "trespass to chattels" claims some companies or website owners have brought?". www.chillingeffects.org. 2007-08-20. Archived from the original on 2002-03-08. Retrieved 2007-08-20.</cite><span title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=What+are+the+%22trespass+to+chattels%22+claims+some+companies+or+website+owners+have+brought%3F&rft.pub=www.chillingeffects.org&rft.date=2007-08-20&rft_id=http%3A%2F%2Fwww.chilli

ngeffects.org%2Flinking%2Ffaq.cgi%23QID460&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping" class="Z3988"> \n<li id="cite_note-10">^ <cite class="citation web">"Ticketmaster Corp. v. Tickets.com, Inc." 2007-08-20. Retrieved 2007-08-20.</cite><span title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=Ticketmaster+Corp.+v.+Tickets.com%2C+Inc.&rft.date=2007-08-20&rft_id=http%3A%2F%2Fwww.tomwbell.com%2FNetLaw%2FCh07%2FTicketmaster.html&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping" class="Z3988"> \n<li id="cite_note-11">^ <cite class="citation web">"American Airlines v. FareChase" (PDF). 2007-08-20. Archived from the original (PDF) on 2011-07-23. Retrieved 2007-08-20.</cite><span title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=American+Airlines+v.+FareChase&rft.date=2007-08-20&rft_id=http%3A%2F%2Fwww.fornova.net%2Fdocuments%2FAAFareChase.pdf&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping" class="Z3988"> \n<li id="cite_note-12">^ <cite class="citation web">"American Airlines, FareChase Settle Suit". The Free Library. 2003-06-13. Retrieved 2012-02-26.</cite><span title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=American+Airlines%2C+FareChase+Settle+Suit.&rft.pub=The+Free+Library&rft.date=2003-06-13&rft_id=http%3A%2F%2Fwww.thefreelibrary.com%2FAmerican%2BAirlines%2C%2BFareChase%2BSettle%2BSuit.-a0103213546&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping" class="Z3988"> \n<li id="cite_note-impervawp2011-13">^ Imperva (2011). Detecting and Blocking Site Scraping Attacks. Imperva white paper..
\n<li id="cite_note-14">^ <cite class="citation web">Adler, Kenneth A. (2003-07-29). "Controversy Surrounds \'Screen Scrapers\': Software Helps Users Access Web Sites But Activity by Competitors Comes Under Scrutiny". Retrieved 2010-10-27.</cite><span title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=Controversy+Surrounds+%27Screen+Scrapers%27%3A+Software+Helps+Users+Access+Web+Sites+But+Activity+by+Competitors+Comes+Under+Scrutiny&rft.date=2003-07-29&rft.aulast=Adler&rft.aufirst=Kenneth+A.&rft_id=http%3A%2F%2Flibrary.findlaw.com%2F2003%2FJul%2F29%2F132944.html&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping" class="Z3988"> \n<li id="cite_note-15">^ <cite class="citation web">"QVC Inc. v. Resultly LLC, No. 14-06714 (E.D. Pa. filed Nov. 24, 2014)" (PDF). 2014-11-24. Retrieved 2015-11-05.</cite><span title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=QVC+Inc.+v.+Resultly+LLC%2C+No.+14-06714+%28E.D.+Pa.+filed+Nov.+24%2C+2014%29&rft.date=2014-11-24&rft_id=http%3A%2F%2Fwww.fornova.net%2Fdocuments%2FCvent.pdf&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping" class="Z3988"> \n<li id="cite_note-16">^ <cite class="citation web">"QVC Inc. v. Resultly LLC, No. 14-06714 (E.D. Pa. filed Nov. 24, 2014)". <i>United States District Court for the Easte

rn District of Pennsylvania. Retrieved 5 November 2015.</cite><span title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&rft.genre=unknown&rft.jtitle=United+States+District+Court+for+the+Eastern+District+of+Pennsylvania&rft.atitle=QVC+Inc.+v.+Resultly+LLC%2C+No.+14-06714+%28E.D.+Pa.+filed+Nov.+24%2C+2014%29&rft_id=https%3A%2F%2Fwww.scribd.com%2Fdoc%2F249068700%2FLinkedIn-v-Resultly-LLC-Complaint%3Fsecret_password%3DpEVKDbnvhQL52oKfdrmT&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping" class="Z3988"> \n<li id="cite_note-17">^ <cite class="citation journal">Neuburger, Jeffrey D (5 December 2014). "QVC Sues Shopping App for Web Scraping That Allegedly Triggered Site Outage". The National Law Review. Proskauer Rose LLP. Retrieved 5 November 2015.</cite><span title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&rft.genre=article&rft.jtitle=The+National+Law+Review&rft.atitle=QVC+Sues+Shopping+App+for+Web+Scraping+That+Allegedly+Triggered+Site+Outage&rft.date=2014-12-05&rft.aulast=Neuburger&rft.aufirst=Jeffrey+D&rft_id=http%3A%2F%2Fnewmedialaw.proskauer.com%2F2014%2F12%2F05%2Fqvc-sues-shopping-app-for-web-scraping-that-allegedly-triggered-site-outage%2F&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping" class="Z3988"> \n<li id="cite_note-18">^ <cite class="citation web">"Did Iqbal/Twombly Raise the Bar for Browsewrap Claims?" (PDF). 2010-09-17. Retrieved 2010-10-27.</cite><span title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=Did+Iqbal%2FTwombly+Raise+the+Bar+for+Browsewrap+Claims%3F&rft.date=2010-09-17&rft_id=http%3A%2F%2Fwww.fornova.net%2Fdocuments%2Fpblog-bna-com.pdf&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping" class="Z3988"> \n<li id="cite_note-19">^ <cite class="citation web">"Can Scraping Non-Infringing Content Become Copyright Infringement... Because Of How Scrapers Work? | Techdirt". Techdirt. 2009-06-10. Retrieved 2016-05-24.</cite><span title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&rft.genre=unknown&rft.jtitle=Techdirt.&rft.atitle=Can+Scraping+Non-Infringing+Content+Become+Copyright+Infringement...+Because+Of+How+Scrapers+Work%3F+%7C+Techdirt&rft.date=2009-06-10&rft_id=https%3A%2F%2Fwww.techdirt.com%2Farticles%2F20090605%2F2228205147.shtml&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping" class="Z3988"> \n<li id="cite_note-20">^ <cite class="citation web">"Facebook v. Power Ventures". Electronic Frontier Foundation. Retrieved 2016-05-24.</cite><span title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&rft.genre=unknown&rft.jtitle=Electronic+Frontier+Foundation&rft.atitle=Facebook+v.+Power+Ventures&rft_id=https%3A%2F%2Fwww.eff.org%2Fcases%2Ffacebook-v-power-ventures&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping" class="Z3988"> \n<li id="cite_note-21">^ <cite class="citation web">"UDSKRIFT AF S\xc3\x98- & HANDELSRETTENS DOMBOG" (PDF) (in Danish). bvhd.dk. 2006-02-24. Archived from the original (PDF) on 2007-10-12. Retrieved 2007-05-30.</cite><span title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=UDSKRIFT+AF+S%C3%98-+%26+HANDELSRETTENS+DOMBOG&rft.pub=bvhd.dk&rft.date=2006-02-24&rft_id=http%3A%2F%2Fwww.bvhd.dk%2Fuploads%2Ftx_moc

articles%2FS_-_og_Handelsrettens_afg_relse_i_Ofir-sagen.pdf&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping" class="Z3988"> \n<li id="cite_note-22">^ <cite class="citation web">"High Court of Ireland Decisions >> Ryanair Ltd -v- Billigfluege.de G MBH 2010 IEHC 47 (26 February 2010)". British and Irish Legal Information Institute. 2010-02-26. Retrieved 2012-04-19.</cite><span title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=High+Court+of+Ireland+Decisions+%3E%3E+Ryanair+Ltd+-v-+Billigfluege.de+GMBH+2010+IEHC+47+%2826+February+2010%29&rft.pub=British+and+Irish+Legal+Information+Institute&rft.date=2010-02-26&rft_id=http%3A%2F%2Fwww.bailii.org%2Fie%2Fcases%2FIEHC%2F2010%2FH47.html&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping" class="Z3988"> \n<li id="cite_note-23">^ <cite class="citation web">Matthews, \xc3\x81ine (June 2010). "Intellectual Property: Website Terms of Use". Issue 26: June 2010. LK Shields Solicitors Update. p. 03. Retrieved 2012-04-19.</cite><span title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&rft.genre=unknown&rft.jtitle=Issue+26%3A+June+2010&rft.atitle=Intellectual+Property%3A+Website+Terms+of+Use&rft.pages=03&rft.date=2010-06&rft.aulast=Matthews&rft.aufirst=%C3%81ine&rft_id=http%3A%2F%2Fwww.lkshields.ie%2Fhtmdocs%2Fpublications%2Fnewsletters%2Fupdate26%2Fupdate26_03.htm&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping" class="Z3988"> \n<li id="cite_note-24">^ <cite class="citation web">National Office for the Information Economy (February 2004). "Spam Act 2003: An overview for business". Australian Communications Authority. p. 6. Retrieved 2017-12-07.</cite><span title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=Spam+Act+2003%3A+An+overview+for+business&rft.pages=6&rft.pub=Australian+Communications+Authority&rft.date=2004-02&rft.au=National+Office+for+the+Information+Economy&rft_id=https%3A%2F%2Fwww.lloyds.com%2F~%2Fmedia%2F5880dae185914b2487bed7bd63b96286.ashx&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping" class="Z3988"> \n<li id="cite_note-25">^ <cite class="citation web">National Office for the Information Economy (February 2004). "Spam Act 2003: A practical guide for business" (PDF). Australian Communications Authority. p. 20. Retrieved 2017-12-07.</cite><span title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=Spam+Act+2003%3A+A+practical+guide+for+business&rft.pages=20&rft.pub=Australian+Communications+Authority&rft.date=2004-02&rft.au=National+Office+for+the+Information+Economy&rft_id=http%3A%2F%2Fwww.webstartdesign.com.au%2Fspam_business_practical_guide.pdf&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping" class="Z3988"> \n<li id="cite_note-26">^ Mayank Dhiman Breaking Fraud & Bot Detection Solutions OWASP AppSec Cali\' 2018 Retrieved February 10, 2018.
\n
\n \n\n\n<!-- \nNewPP limit report\nParsed by mw1249\nCached time: 20180226071249\nCache expiry: 1900800\nDynamic content: false\nCPU time usage: 0.228 seconds\nReal time usage: 0.285 seconds\nPreprocessor visited node count: 1447/1000000\nPreprocessor generated node count: 0/1500000\nPost\xe2\x80\x90expand include size: 49571/2097152 bytes\nTemplate argument size: 320/2097152 bytes\nHighest expansion depth: 10/40\nExpensive parser function count: 3/500\nLua time usage: 0.119/10.000 seconds\nLua memory usage: 4.53 MB/50 MB\n-->\n<!--\nTransclusion expansion time report (%,ms,calls,template)\n100.00% 240.818   1 -total\n 59.86% 144.146   1 Template:Reflist\n 28.76%   69.247   18 Template:Cite_web\n 19.28%   46.421   5 Template:Cite_journal\n 17.92%   43.157   1 Template:More_citations_needed\n 13.68%   32.949   2 Template:Ambox\n 4.68

```
%   11.269      1 Template:US-centric\n  4.39%   10.578      1 Template:Further_information\n  3.99%    9.620
  1 Template:Colbegin\n  3.20%    7.700      1 Template:Globalize\n-->\n</div>\n<!-- Saved in parser cache wit
h key enwiki:pcache:idhash:2696619-0!canonical and timestamp 20180226071248 and revision id 825057514\n
 -->\n<noscript><img src="//en.wikipedia.org/wiki/Special:CentralAutoLogin/start?type=1x1" alt="" title="" wi
dth="1" height="1" style="border: none; position: absolute;" /></noscript></div>\t\t\t\t\t<div class="printfooter
">\n\t\t\t\t\tRetrieved from "<a dir="ltr" href="https://en.wikipedia.org/w/index.php?title=Web_scraping&amp
;oldid=825057514">https://en.wikipedia.org/w/index.php?title=Web_scraping&amp;oldid=825057514</a>"\t\t\
t\t\t</div>\n\t\t\t\t<div id="catlinks" class="catlinks" data-mw="interface"><div id="mw-normal-catlinks" class
="mw-normal-catlinks"><a href="/wiki/Help:Category" title="Help:Category">Categories</a>: <ul><li><a hre
f="/wiki/Category:Web_scraping" title="Category:Web scraping">Web scraping</a></li><li><a href="/wiki/C
ategory:World_Wide_Web" title="Category:World Wide Web">World Wide Web</a></li><li><a href="/wiki/
Category:Spamming" title="Category:Spamming">Spamming</a></li></ul></div><div id="mw-hidden-catlin
ks" class="mw-hidden-catlinks mw-hidden-cats-hidden">Hidden categories: <ul><li><a href="/wiki/Category:
CS1_Danish-language_sources_(da)" title="Category:CS1 Danish-language sources (da)">CS1 Danish-languag
e sources (da)</a></li><li><a href="/wiki/Category:Articles_needing_additional_references_from_June_2017"
 title="Category:Articles needing additional references from June 2017">Articles needing additional references
from June 2017</a></li><li><a href="/wiki/Category:All_articles_needing_additional_references" title="Categ
ory:All articles needing additional references">All articles needing additional references</a></li><li><a href="
/wiki/Category:Articles_with_limited_geographic_scope_from_October_2015" title="Category:Articles with li
mited geographic scope from October 2015">Articles with limited geographic scope from October 2015</a></l
i><li><a href="/wiki/Category:USA-centric" title="Category:USA-centric">USA-centric</a></li><li><a href=
"/wiki/Category:Pages_using_div_col_with_deprecated_parameters" title="Category:Pages using div col with d
eprecated parameters">Pages using div col with deprecated parameters</a></li></ul></div></div>\t\t\t\t<div cl
ass="visualClear"></div>\n\t\t\t\t\t\t</div>\n\t\t</div>\n\t\t<div id="mw-navigation">\n\t\t\t<h2>Navigation
menu</h2>\n\t\t\t<div id="mw-head">\n\t\t\t\t\t\t\t\t\t<div id="p-personal" role="navigation" class="" aria-label
ledby="p-personal-label">\n\t\t\t\t\t\t<h3 id="p-personal-label">Personal tools</h3>\n\t\t\t\t\t\t<ul>\n\t\t\t\t\t\t\t
<li id="pt-anonuserpage">Not logged in</li><li id="pt-anontalk"><a href="/wiki/Special:MyTalk" title="Discu
ssion about edits from this IP address [n]" accesskey="n">Talk</a></li><li id="pt-anoncontribs"><a href="/wi
ki/Special:MyContributions" title="A list of edits made from this IP address [y]" accesskey="y">Contributions
</a></li><li id="pt-createaccount"><a href="/w/index.php?title=Special:CreateAccount&amp;returnto=Web+s
craping" title="You are encouraged to create an account and log in; however, it is not mandatory">Create accou
nt</a></li><li id="pt-login"><a href="/w/index.php?title=Special:UserLogin&amp;returnto=Web+scraping" tit
le="You&#039;re encouraged to log in; however, it&#039;s not mandatory. [o]" accesskey="o">Log in</a></li
>\t\t\t\t\t\t</ul>\n\t\t\t\t\t</div>\n\t\t\t\t\t\t\t\t\t<div id="left-navigation">\n\t\t\t\t\t\t\t\t\t\t<div id="p-namespaces"
 role="navigation" class="vectorTabs" aria-labelledby="p-namespaces-label">\n\t\t\t\t\t\t<h3 id="p-namespaces
-label">Namespaces</h3>\n\t\t\t\t\t\t<ul>\n\t\t\t\t\t\t\t<li id="ca-nstab-main" class="selected"><span><a href="/
wiki/Web_scraping" title="View the content page [c]" accesskey="c">Article</a></span></li><li id="ca-talk"
><span><a href="/wiki/Talk:Web_scraping" rel="discussion" title="Discussion about the content page [t]" acce
sskey="t">Talk</a></span></li>\t\t\t\t\t\t</ul>\n\t\t\t\t\t</div>\n\t\t\t\t\t\t\t\t\t\t<div id="p-variants" role="navig
ation" class="vectorMenu emptyPortlet" aria-labelledby="p-variants-label">\n\t\t\t\t\t\t\t\t\t\t\t\t<input type="ch
eckbox" class="vectorMenuCheckbox" aria-labelledby="p-variants-label" />\n\t\t\t\t\t\t<h3 id="p-variants-label
">\n\t\t\t\t\t\t\t<span>Variants</span>\n\t\t\t\t\t\t</h3>\n\t\t\t\t\t\t<div class="menu">\n\t\t\t\t\t\t\t<ul>\n\t\t\t\t\t\t\t
\t\t\t\t\t\t\t\t</ul>\n\t\t\t\t\t\t</div>\n\t\t\t\t\t</div>\n\t\t\t\t\t\t\t\t\t</div>\n\t\t\t\t<div id="right-navigation">\n\t\t
\t\t\t\t\t\t\t<div id="p-views" role="navigation" class="vectorTabs" aria-labelledby="p-views-label">\n\t\t\t\t\t\
t<h3 id="p-views-label">Views</h3>\n\t\t\t\t\t\t<ul>\n\t\t\t\t\t\t\t<li id="ca-view" class="collapsible selected">
<span><a href="/wiki/Web_scraping">Read</a></span></li><li id="ca-edit" class="collapsible"><span><a hr
ef="/w/index.php?title=Web_scraping&amp;action=edit" title="Edit this page [e]" accesskey="e">Edit</a></sp
an></li><li id="ca-history" class="collapsible"><span><a href="/w/index.php?title=Web_scraping&amp;actio
n=history" title="Past revisions of this page [h]" accesskey="h">View history</a></span></li>\t\t\t\t\t\t</ul>\n\
t\t\t\t\t</div>\n\t\t\t\t\t\t\t\t\t\t<div id="p-cactions" role="navigation" class="vectorMenu emptyPortlet" aria-labe
lledby="p-cactions-label">\n\t\t\t\t\t\t<input type="checkbox" class="vectorMenuCheckbox" aria-labelledby="p
-cactions-label" />\n\t\t\t\t\t\t<h3 id="p-cactions-label"><span>More</span></h3>\n\t\t\t\t\t\t<div class="menu"
>\n\t\t\t\t\t\t\t<ul>\n\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t</ul>\n\t\t\t\t\t\t</div>\n\t\t\t\t\t</div>\n\t\t\t\t\t\t\t\t\t\t<div id="p-searc
h" role="search">\n\t\t\t\t\t\t<h3>\n\t\t\t\t\t\t\t<label for="searchInput">Search</label>\n\t\t\t\t\t\t</h3>\n\t\t\t\t\t\t
\t<form action="/w/index.php" id="searchform">\n\t\t\t\t\t\t\t<div id="simpleSearch">\n\t\t\t\t\t\t\t\t<input type
="search" name="search" placeholder="Search Wikipedia" title="Search Wikipedia [f]" accesskey="f" id="sear
chInput"/><input type="hidden" value="Special:Search" name="title"/><input type="submit" name="fulltext"
```

value="Search" title="Search Wikipedia for this text" id="mw-searchButton" class="searchButton mw-fallback SearchButton"/><input type="submit" name="go" value="Go" title="Go to a page with this exact name if it exi sts" id="searchButton" class="searchButton"/>\t\t\t\t\t\t\t \n\t\t\t\t\t\t</form>\n\t\t\t\t\t \n\t\t\t\t\t\t\t\t\t\t \n\t\t\t \n\t\t\t \n\t\t\t\t \n\t\t\t\t\t\t<div class="portal" role="navigat ion" id=\'p-navigation\' aria-labelledby=\'p-navigation-label\'>\n\t\t\t<h3 id=\'p-navigation-label\'>Navigation</ h3>\n\t\t\t \n\t\t\t\t\t\t\t\t
\n\t\t\t\t\t<li id="n-mainpage-description">Main page
<li id="n-contents">Contents
<li id="n-featuredcontent"><a href="/ wiki/Portal:Featured_content" title="Featured content \xe2\x80\x93 the best of Wikipedia">Featured content
<li id="n-currentevents">Current events
<li id="n-randompage">Random article
<li id="n-sitesupport">Donate to Wikipedia
<li id="n-shop link">Wikipedia store
\t\t\t\t
\n\t \t\t\t\t\t\t \n\t\t \n\t\t\t<div class="portal" role="navigation" id=\'p-interaction\' aria-labelledby=\'p-int eraction-label\'>\n\t\t\t<h3 id=\'p-interaction-label\'>Interaction</h3>\n\t\t\t \n\t\t\t\t\t\t\t\t
\n\t\t\t\t\t<li id="n-help"> Help
<li id="n-aboutsite">About Wikipedia
<li id="n-portal">Community portal
<li id="n-recentchanges">Recent changes
<li i d="n-contactpage">Contact page
\t\t\t\t
\n\t\t\t\t\t\t\t \n\t\t \n\t\t\t<div class="portal" role="navigation" id =\'p-tb\' aria-labelledby=\'p-tb-label\'>\n\t\t\t<h3 id=\'p-tb-label\'>Tools</h3>\n\t\t\t \n\t\t\t\t\t\t\t\t
\n\t\t\t\t\t<li id="t-whatlinkshere">What links here
<li id="t -recentchangeslinked">Related changes
<li id="t-upload">Upload file
<li i d="t-specialpages"> Special pages
<li id="t-permalink">Permanent link
<li id="t-info">Page information< /li><li id="t-wikibase">Wikidata item
<li id="t-cite">Cite this page
\t\t\t\t
\n\t\t\t\t\t\t\t \n\t\t \n\t\t\t<div class="portal" role="navi gation" id=\'p-coll-print_export\' aria-labelledby=\'p-coll-print_export-label\'>\n\t\t\t<h3 id=\'p-coll-print_export -label\'>Print/export</h3>\n\t\t\t \n\t\t\t\t\t\t\t\t
\n\t\t\t\t\t<li id="coll-create_a_book">Create a b ook
<li id="coll-download-as-rdf2latex">Download as PDF
<li id="t-print"> Printable version
\t\t\t\t
\n\t\t\t\t\t\t\t \n\t\t \n\t\t\t<div class="portal" role="navigation" id=\'p-lang\' aria-labelledby=\'p-lang-label\'>\n\t\t\t<h3 id=\'p-lang-label\'>Languages</h3>\n\t\t\t \n\t\t\t\t\t\t\t\t
\n\t\t\t\t\t<li class="interlanguage-link interwiki-ar">\xd8\xa7\xd9\x84\xd8\xb9\xd8\xb1\xd8\xa8\xd9\x8a\xd8\xa9
<li class="interl anguage-link interwiki-ca">Catal\xc3\xa0
<li class="int erlanguage-link interwiki-de">Deutsch
<li class=" interlanguage-link interwiki-es"><a href="https://es.wikipedia.org/wiki/Web_scraping" title="Web scraping \xe

2\x80\x93 Spanish" lang="es" hreflang="es" class="interlanguage-link-target">Espa\xc3\xb1ol
<li class="interlanguage-link interwiki-eu">Euskara
<li class="interlanguage-link interwiki-fr">Fran\xc3\xa7ais
<li class="interlanguage-link interwiki-is">\xc3\x8dslenska
<li class="interlanguage-link interwiki-it">Italiano
<li class="interlanguage-link interwiki-lv">Latvie\xc5\xa1u
<li class="interlanguage-link interwiki-nl">Nederlands
<li class="interlanguage-link interwiki-ja">\xe6\x97\xa5\xe6\x9c\xac\xe8\xaa\x9e
<li class="interlanguage-link interwiki-sr">\xd0\xa1\xd1\x80\xd0\xbf\xd1\x81\xd0\xba\xd0\xb8 / srpski
<li class="interlanguage-link interwiki-tr">T\xc3\xbcrk\xc3\xa7e
<li class="interlanguage-link interwiki-uk">\xd0\xa3\xd0\xba\xd1\x80\xd0\xb0\xd1\x97\xd0\xbd\xd1\x81\xd1\x8c\xd0\xba\xd0\xb0
<li class="interlanguage-link interwiki-zh">\xe4\xb8\xad\xe6\x96\x87
\t\t\t\t
\n\t\t\t\t Edit links \t\t\t \n\t\t \n\t\t\t\t \n\t\t \n\t\t\t\t \n\t\t\t\t\t\t<ul id="footer-info">\n\t\t\t\t\t\t\t\t<li id="footer-info-lastmod"> This page was last edited on 11 February 2018, at 06:38.
\n\t\t\t\t\t\t\t\t<li id="footer-info-copyright">Text is available under the Creative Commons Attribution-ShareAlike License;\nadditional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia\xc2\xae is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.
\n\t\t\t\t\t\t
\n\t\t\t\t\t\t<ul id="footer-places">\n\t\t\t\t\t\t\t<li id="footer-places-privacy">Privacy policy
\n\t\t\t\t\t\t\t<li id="footer-places-about">About Wikipedia
\n\t\t\t\t\t\t\t<li id="footer-places-disclaimer">Disclaimers
\n\t\t\t\t\t\t\t<li id="footer-places-contact">Contact Wikipedia
\n\t\t\t\t\t\t\t<li id="footer-places-developers">Developers
\n\t\t\t\t\t\t\t<li id="footer-places-cookiestatement">Cookie statement
\n\t\t\t\t\t\t\t<li id="footer-places-mobileview">Mobile view
\n\t\t\t\t\t\t
\n\t\t\t\t\t\t<ul id="footer-icons" class="noprint">\n\t\t\t\t\t\t\t<li id="footer-copyrightico">\n\t\t\t\t\t\t\t\t\t\t
\n\t\t\t\t\t\t\t<li id="footer-poweredbyico">\n\t\t\t\t\t\t\t\t\t\t
\n\t\t\t\t\t\t
\n\t\t\t\t\t \n\t\t \n\t\t<script>(wind

ow.RLQ=window.RLQ||[]).push(function(){mw.config.set({"wgPageParseReport":{"limitreport":{"cputime":"
0.228","walltime":"0.285","ppvisitednodes":{"value":1447,"limit":1000000},"ppgeneratednodes":{"value":0,"li
mit":1500000},"postexpandincludesize":{"value":49571,"limit":2097152},"templateargumentsize":{"value":32
0,"limit":2097152},"expansiondepth":{"value":10,"limit":40},"expensivefunctioncount":{"value":3,"limit":500
},"entityaccesscount":{"value":0,"limit":400},"timingprofile":["100.00% 240.818   1 -total"," 59.86% 144.1
46   1 Template:Reflist"," 28.76%   69.247   18 Template:Cite_web"," 19.28%   46.421   5 Template:Cite_j
ournal"," 17.92%   43.157   1 Template:More_citations_needed"," 13.68%   32.949   2 Template:Ambox","
4.68%   11.269   1 Template:US-centric","   4.39%   10.578   1 Template:Further_information","   3.99%   9.
620   1 Template:Colbegin","   3.20%   7.700   1 Template:Globalize"]},"scribunto":{"limitreport-timeusage
":{"value":"0.119","limit":"10.000"},"limitreport-memusage":{"value":4752818,"limit":52428800}},"cacherep
ort":{"origin":"mw1249","timestamp":"20180226071249","ttl":1900800,"transientcontent":false}}});});</script
><script>(window.RLQ=window.RLQ||[]).push(function(){mw.config.set({"wgBackendResponseTime":374,"
wgHostname":"mw1249"});});</script>\n\t</body>\n</html>\n'

In [4]:
```
# importing beautiful soup for formatting and etracting data out of raw HT
ML

# Bs4 uses parsers to format the HTML

from bs4 import BeautifulSoup
soup = BeautifulSoup(html_doc, 'html.parser')
```

In [5]:
```
# compare print(html_doc) vs print(soup). You'll see better in soup becaus
e it used parser to format HTML properly
print(soup)
```

<!DOCTYPE html>

<html class="client-nojs" dir="ltr" lang="en">
<head>
<meta charset="utf-8"/>
<title>Web scraping - Wikipedia</title>
<script>document.documentElement.className = document.documentElement.className.replace(/(^|\s)client
-nojs(\s|$)/, "$1client-js$2");</script>
<script>(window.RLQ=window.RLQ||[]).push(function(){mw.config.set({"wgCanonicalNamespace":"","wgCa
nonicalSpecialPageName":false,"wgNamespaceNumber":0,"wgPageName":"Web_scraping","wgTitle":"Web s
craping","wgCurRevisionId":825057514,"wgRevisionId":825057514,"wgArticleId":2696619,"wgIsArticle":tru
e,"wgIsRedirect":false,"wgAction":"view","wgUserName":null,"wgUserGroups":["*"],"wgCategories":["CS1
Danish-language sources (da)","Articles needing additional references from June 2017","All articles needing ad
ditional references","Articles with limited geographic scope from October 2015","USA-centric","Pages using di
v col with deprecated parameters","Web scraping","World Wide Web","Spamming"],"wgBreakFrames":false,"
wgPageContentLanguage":"en","wgPageContentModel":"wikitext","wgSeparatorTransformTable":["",""],"wg
DigitTransformTable":["",""],"wgDefaultDateFormat":"dmy","wgMonthNames":["","January","February","Ma
rch","April","May","June","July","August","September","October","November","December"],"wgMonthName
sShort":["","Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"],"wgRelevantPageNa
me":"Web_scraping","wgRelevantArticleId":2696619,"wgRequestId":"WpOzcApAMFQAAKv1HOgAAABY"
,"wgIsProbablyEditable":true,"wgRelevantPageIsProbablyEditable":true,"wgRestrictionEdit":[],"wgRestriction
Move":[],"wgFlaggedRevsParams":{"tags":{}},"wgStableRevisionId":null,"wgWikiEditorEnabledModules":[],
"wgBetaFeaturesFeatures":[],"wgMediaViewerOnClick":true,"wgMediaViewerEnabledByDefault":true,"wgPo
pupsShouldSendModuleToUser":true,"wgPopupsConflictsWithNavPopupGadget":false,"wgVisualEditor":{"pa
geLanguageCode":"en","pageLanguageDir":"ltr","pageVariantFallbacks":"en","usePageImages":true,"usePage
Descriptions":true},"wgPreferredVariant":"en","wgMFExpandAllSectionsUserOption":true,"wgMFEnableFont
Changer":true,"wgMFDisplayWikibaseDescriptions":{"search":false,"nearby":false,"watchlist":false,"tagline":f
alse},"wgRelatedArticles":null,"wgRelatedArticlesUseCirrusSearch":true,"wgRelatedArticlesOnlyUseCirrusSe
arch":false,"wgULSCurrentAutonym":"English","wgNoticeProject":"wikipedia","wgCentralNoticeCookiesToD
elete":[],"wgCentralNoticeCategoriesUsingLegacy":["Fundraising","fundraising"],"wgCategoryTreePageCateg

oryOptions":"{\"mode\":0,\"hideprefix\":20,\"showcount\":true,\"namespaces\":false}","wgWikibaseItemId":"Q665452","wgScoreNoteLanguages":{"arabic":"العربية","catalan":"català","deutsch":"Deutsch","english":"English","espanol":"español","italiano":"italiano","nederlands":"Nederlands","norsk":"norsk","portugues":"português","suomi":"suomi","svenska":"svenska","vlaams":"West-Vlams"},"wgScoreDefaultNoteLanguage":"nederlands","wgCentralAuthMobileDomain":false,"wgCodeMirrorEnabled":false,"wgVisualEditorToolbarScrollOffset":0,"wgVisualEditorUnsupportedEditParams":["undo","undoafter","veswitched"],"wgEditSubmitButtonLabelPublish":true});mw.loader.state({"ext.gadget.charinsert-styles":"ready","ext.globalCssJs.user.styles":"ready","ext.globalCssJs.site.styles":"ready","site.styles":"ready","noscript":"ready","user.styles":"ready","user":"ready","user.options":"ready","user.tokens":"loading","ext.cite.styles":"ready","wikibase.client.init":"ready","ext.visualEditor.desktopArticleTarget.noscript":"ready","ext.uls.interlanguage":"ready","ext.wikimediaBadges":"ready","mediawiki.legacy.shared":"ready","mediawiki.legacy.commonPrint":"ready","mediawiki.sectionAnchor":"ready","mediawiki.skinning.interface":"ready","skins.vector.styles":"ready","ext.globalCssJs.user":"ready","ext.globalCssJs.site":"ready"});mw.loader.implement("user.tokens@1dqfd7l",function($,jQuery,require,module){/*@nomin*/mw.user.tokens.set({"editToken":"+\\","patrolToken":"+\\","watchToken":"+\\","csrfToken":"+\\"});});mw.loader.load(["ext.cite.a11y","site","mediawiki.page.startup","mediawiki.user","mediawiki.hidpi","mediawiki.page.ready","mediawiki.toc","mediawiki.searchSuggest","ext.gadget.teahouse","ext.gadget.ReferenceTooltips","ext.gadget.watchlist-notice","ext.gadget.DRN-wizard","ext.gadget.charinsert","ext.gadget.refToolbar","ext.gadget.extra-toolbar-buttons","ext.gadget.switcher","ext.centralauth.centralautologin","mmv.head","mmv.bootstrap.autostart","ext.popups","ext.visualEditor.desktopArticleTarget.init","ext.visualEditor.targetLoader","ext.eventLogging.subscriber","ext.wikimediaEvents","ext.navigationTiming","ext.uls.eventlogger","ext.uls.init","ext.uls.interface","ext.3d","ext.centralNotice.geoIP","ext.centralNotice.startUp","skins.vector.js"]);});</script>
<link href="/w/load.php?debug=false&lang=en&modules=ext.cite.styles%7Cext.uls.interlanguage%7Cext.visualEditor.desktopArticleTarget.noscript%7Cext.wikimediaBadges%7Cmediawiki.legacy.commonPrint%2Cshared%7Cmediawiki.sectionAnchor%7Cmediawiki.skinning.interface%7Cskins.vector.styles%7Cwikibase.client.init&only=styles&skin=vector" rel="stylesheet"/>
<script async="" src="/w/load.php?debug=false&lang=en&modules=startup&only=scripts&skin=vector"></script>
<meta content="" name="ResourceLoaderDynamicStyles"/>
<link href="/w/load.php?debug=false&lang=en&modules=ext.gadget.charinsert-styles&only=styles&skin=vector" rel="stylesheet"/>
<link href="/w/load.php?debug=false&lang=en&modules=site.styles&only=styles&skin=vector" rel="stylesheet"/>
<meta content="MediaWiki 1.31.0-wmf.22" name="generator"/>
<meta content="origin" name="referrer"/>
<meta content="origin-when-crossorigin" name="referrer"/>
<meta content="origin-when-cross-origin" name="referrer"/>
<link href="android-app://org.wikipedia/http/en.m.wikipedia.org/wiki/Web_scraping" rel="alternate"/>
<link href="/w/index.php?title=Web_scraping&action=edit" rel="alternate" title="Edit this page" type="application/x-wiki"/>
<link href="/w/index.php?title=Web_scraping&action=edit" rel="edit" title="Edit this page"/>
<link href="/static/apple-touch/wikipedia.png" rel="apple-touch-icon"/>
<link href="/static/favicon/wikipedia.ico" rel="shortcut icon"/>
<link href="/w/opensearch_desc.php" rel="search" title="Wikipedia (en)" type="application/opensearchdescription+xml"/>
<link href="//en.wikipedia.org/w/api.php?action=rsd" rel="EditURI" type="application/rsd+xml"/>
<link href="//creativecommons.org/licenses/by-sa/3.0/" rel="license"/>
<link href="https://en.wikipedia.org/wiki/Web_scraping" rel="canonical"/>
<link href="//login.wikimedia.org" rel="dns-prefetch"/>
<link href="//meta.wikimedia.org" rel="dns-prefetch"/>
<!--[if lt IE 9]><script src="/w/load.php?debug=false&lang=en&modules=html5shiv&only=scripts&skin=vector&sync=1"></script><![endif]-->
</head>
<body class="mediawiki ltr sitedir-ltr mw-hide-empty-elt ns-0 ns-subject page-Web_scraping rootpage-Web_scraping skin-vector action-view">

<!-- CentralNotice -->

<h1 class="firstHeading" id="firstHeading" lang="en">Web scraping</h1>
From Wikipedia, the free encyclopedia

Jump to: navigation,
search

<table class="plainlinks metadata ambox ambox-content ambox-Refimprove" role="presentation">
<tr>
<td class="mbox-image">

</td>
<td class="mbox-text">
This article needs additional citations for verification. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed. <small>(June 2017)</small> <small class="hide-when-compact">(Learn how and when to remove this template message)</small>
</td>
</tr>
</table>
For a broader coverage related to this topic, see Data scraping.
Web scraping, web harvesting, or web data extraction is data scraping used for extracting data from websites.^{[1]} Web scraping software may access the World Wide Web directly using the Hypertext Transfer Protocol, or through a web browser. While web scraping can be done manually by a software user, the term typically refers to automated processes implemented using a bot or web crawler. It is a form of copying, in which specific data is gathered and copied from the web, typically into a central local database or spreadsheet, for later retrieval or analysis.
Web scraping a web page involves fetching it and extracting from it.^{[1]}^{[2]} Fetching is the downloading of a page (which a browser does when you view the page). Therefore, web crawling is a main component of web scraping, to fetch pages for later processing. Once fetched, then extraction can take place. The content of a page may be parsed, searched, reformatted, its data copied into a spreadsheet, and so on. Web scrapers typically take something out of a page, to make use of it for another purpose somewhere else. An example would be to find and copy names and phone numbers, or companies and their URLs, to a list (contact scraping).
Web scraping is used for contact scraping, and as a component of applications used for web indexing, web mining and <a href="/wiki/Data_mining" tit

le="Data mining">data mining, online price change monitoring and price comparison, product review scraping (to watch the competition), gathering real estate listings, weather data monitoring, website change detection, research, tracking online presence and reputation, web mashup and, web data integration.
Web pages are built using text-based mark-up languages (HTML and XHTML), and frequently contain a wealth of useful data in text form. However, most web pages are designed for human end-users and not for ease of automated use. Because of this, tool kits that scrape web content were created. A web scraper is an Application Programming Interface (API) to extract data from a web site. Companies like Amazon AWS and Google provide web scraping tools, services and public data available free of cost to end users.
Newer forms of web scraping involve listening to data feeds from web servers. For example, JSON is commonly used as a transport storage mechanism between the client and the web server.
There are methods that some websites use to prevent web scraping, such as detecting and disallowing bots from crawling (viewing) their pages. In response, there are web scraping systems that rely on using techniques in DOM parsing, computer vision and natural language processing to simulate human browsing to enable gathering web page content for offline parsing.

<h2>Contents</h2>

<li class="toclevel-1 tocsection-1">1 Techniques

<li class="toclevel-2 tocsection-2">1.1 Human copy-and-paste
<li class="toclevel-2 tocsection-3">1.2 Text pattern matching
<li class="toclevel-2 tocsection-4">1.3 HTTP programming
<li class="toclevel-2 tocsection-5">1.4 HTML parsing
<li class="toclevel-2 tocsection-6">1.5 DOM parsing
<li class="toclevel-2 tocsection-7">1.6 Vertical aggregation
<li class="toclevel-2 tocsection-8">1.7 Semantic annotation recognizing
<li class="toclevel-2 tocsection-9">1.8 Computer vision web-page analysis

<li class="toclevel-1 tocsection-10">2 Software

<li class="toclevel-2 tocsection-11">2.1 Example tools

<li class="toclevel-3 tocsection-12">2.1.1 Javascript tools
<li class="toclevel-3 tocsection-13">2.1.2 SaaS version
<li class="toclevel-3 tocsection-14">2.1.3 Web crawling frameworks

<li class="toclevel-1 tocsection-15">3 Legal issues

<li class="toclevel-2 tocsection-16">3.1 United States
<li class="toclevel-2 tocsection-17">3.2 The EU
<li class="toclevel-2 tocsection-18">3.3 Australia

<li class="toclevel-1 tocsection-19">4 Methods to prevent web scraping
<li class="toclevel-1 tocsection-20">5 See also
<li class="toclevel-1 tocsection-21">6 References

<h2>Techniques[edit]</h2>
Web scraping is the process of automatically mining data or collecting information from the World Wide Web. It is a field with active developments sharing a common goal with the semantic web vision, an ambitious initiative that still requires breakthroughs in text processing, semantic understanding, artificial intelligence and human-computer interactions. Current web scraping solutions range from the ad-hoc, requiring human effort, to fully automated systems that are able to convert entire web sites into structured information, with limitations.
<h3>Human copy-and-paste[edit]</h3>
Sometimes even the best web-scraping technology cannot replace a human’s manual examination and copy-and-paste, and sometimes this may be the only workable solution when the websites for scraping explicitly set up barriers to prevent machine automation.
<h3>Text pattern matching[edit]</h3>
A simple yet powerful approach to extract information from web pages can be based on the UNIX grep command or regular expression-matching facilities of programming languages (for instance Perl or <a href="/wiki/Python_(programming_language)" title="Python (programming language)"

>Python).
<h3>HTTP programming[edit]</h3>
Static and dynamic web pages can be retrieved by posting HTTP requests to the remote web server using socket programming.
<h3>HTML parsing[edit]</h3>
Many websites have large collections of pages generated dynamically from an underlying structured source like a database. Data of the same category are typically encoded into similar pages by a common script or template. In data mining, a program that detects such templates in a particular information source, extracts its content and translates it into a relational form, is called a wrapper. Wrapper generation algorithms assume that input pages of a wrapper induction system conform to a common template and that they can be easily identified in terms of a URL common scheme.^{[3]} Moreover, some semi-structured data query languages, such as XQuery and the HTQL, can be used to parse HTML pages and to retrieve and transform page content.
<h3>DOM parsing[edit]</h3>
Further information: Document Object Model
By embedding a full-fledged web browser, such as the Internet Explorer or the Mozilla browser control, programs can retrieve the dynamic content generated by client-side scripts. These browser controls also parse web pages into a DOM tree, based on which programs can retrieve parts of the pages.
<h3>Vertical aggregation[edit]</h3>
There are several companies that have developed vertical specific harvesting platforms. These platforms create and monitor a multitude of “bots” for specific verticals with no "man in the loop" (no direct human involvement), and no work related to a specific target site. The preparation involves establishing the knowledge base for the entire vertical and then the platform creates the bots automatically. The platform's robustness is measured by the quality of the information it retrieves (usually number of fields) and its scalability (how quick it can scale up to hundreds or thousands of sites). This scalability is mostly used to target the Long Tail of sites that common aggregators find complicated or too labor-intensive to harvest content from.
<h3>Semantic annotation recognizing[edit]</h3>
The pages being scraped may embrace metadata or semantic markups and annotations, which can be used to locate specific data snippets. If the annotations are embedded in the pages, as Microformat does, this technique can be viewed as a special case of DOM parsing. In another case, the annotations, organized into a semantic layer,^{[4]} are stored and managed separately from the web pages, so the scrapers can retrieve data schema and instructions from this layer before scraping the pages.
<h3><span class="mw-headline" id="Computer_vision_web-page_analys

is[edit]</h3>
There are efforts using machine learning and computer vision that attempt to identify and extract information from web pages by interpreting pages visually as a human being might.^{[5]}
<h2>Software[edit]</h2>
There are many software tools available that can be used to customize web-scraping solutions. This software may attempt to automatically recognize the data structure of a page or provide a recording interface that removes the necessity to manually write web-scraping code, or some scripting functions that can be used to extract and transform content, and database interfaces that can store the scraped data in local databases. Some web scraping software can also be used to extract data from an API directly.
<h3>Example tools[edit]</h3>

cURL – command line tool and library for transferring (including getting) data with URLs supporting a wide range of HTTP methods (GET, POST, cookies, etc.)
Data Toolbar – web scraping add-on for Internet Explorer, Mozilla Firefox, and Google Chrome Web browsers that collects and converts structured data from web pages into a tabular format that can be loaded into a spreadsheet or database management program.
Diffbot – uses computer vision and machine learning to automatically extract data from web pages by interpreting pages visually as a human being might.
Heritrix – gets pages (lots of them). It is a web crawler designed for web archiving, written by the Internet Archive (see Wayback Machine).
HtmlUnit – headless browser that can be used for retrieving web pages, web scraping, and more.
HTTrack – free and open source Web crawler and offline browser, designed to download websites.
iMacros –a browser extension to record, code, share and replay browser automation (javascript)
Kantu – uses screenshots and OCR for scraping
Selenium (software) – a portable software-testing framework for web applications
Jaxer
nokogiri
OutWit Hub – Web scraping application including built-in data, image, document extractors and editors for custom scrapers and automatic exploration and extraction jobs (free and paid versions)
watir –
Wget – computer program that retrieves content from web servers. It is part of the GNU Project. It supports downloading via the HTTP, HTTPS, and FTP protocols.
WSO2 Mashup Server –
Yahoo! Query Language (YQL) –
Data Scraping Studio – Stand alone windows desktop software to scrape data using CSS selectors and REGEX.

<h4>Javascript tools[edit]</h4>

Greasemonkey
Node.js
PhantomJS – scripted, headless browser used for automating web page interaction.
jQuery

<h4>SaaS version[edit]</h4>

Agenty – SaaS solution, paid versions available from $29 (06/09/17)
Apify – Web scraping and automation platform, free and paid versions available (10/10/17)
dexi.io – SaaS solution, free and paid versions available from $119 USD (31/10/17)
diggernaut.com – Turn websites into datasets, free and paid (from $9.99 USD) subscriptions available (02/05/18)
fScraper – Facebook friendly scraper, SaaS solution, free and paid versions available
Import.io – SaaS solution, paid versions available from $299 USD (06/09/17)
Listly.io – HTML to Excel in seconds, free SaaS service (04/10/17)
Mozenda – SaaS solution, is a web-based platform for web data extraction (01/22/18)
uScraper – SaaS service, free and paid versions available. Functionality primarily for scraping email addresses.

<h4>Web crawling frameworks[edit]</h4>
These can be used to build web scrapers.

Scrapy

<h2>Legal issues[edit]</h2>
<table class="plainlinks metadata ambox ambox-content ambox-globalize" role="presentation">
<tr>
<td class="mbox-image">

</td>
<td class="mbox-text">
The examples and perspective in this article deal primarily with the United States and do not represent a worldwide view of the subject. You may improve this article, discuss the issue on the <a href="/wiki/Talk:Web_scraping" title="Talk:Web

scraping">talk page, or create a new article, as appropriate. <small>(October 2015)</small> <small class="hide-when-compact">(Learn how and when to remove this template message)</small>
</td>
</tr>
</table>
The legality of web scraping varies across the world. In general, web scraping may be against the terms of use of some websites, but the enforceability of these terms is unclear.^{[6]}
<h3>United States[edit]</h3>
In the United States, website owners can use three major legal claims to prevent undesired web scraping: (1) copyright infringement (compilation), (2) violation of the Computer Fraud and Abuse Act (“CFAA”), and (3) trespass to chattel.^{[7]} However, the effectiveness of these claims relies upon meeting various criteria, and the case law is still evolving. For example, with regard to copyright, while outright duplication of original expression will in many cases be illegal, in the United States the courts ruled in Feist Publications v. Rural Telephone Service that duplication of facts is allowable.
U.S. courts have acknowledged that users of "scrapers" or "robots" may be held liable for committing trespass to chattels,^{[8]}^{[9]} which involves a computer system itself being considered personal property upon which the user of a scraper is trespassing. The best known of these cases, eBay v. Bidder's Edge, resulted in an injunction ordering Bidder's Edge to stop accessing, collecting, and indexing auctions from the eBay web site. This case involved automatic placing of bids, known as auction sniping. However, in order to succeed on a claim of trespass to chattels, the plaintiff must demonstrate that the defendant intentionally and without authorization interfered with the plaintiff's possessory interest in the computer system and that the defendant's unauthorized use caused damage to the plaintiff. Not all cases of web spidering brought before the courts have been considered trespass to chattels.^{[10]}
One of the first major tests of screen scraping involved American Airlines (AA), and a firm called FareChase.^{[11]} AA successfully obtained an injunction from a Texas trial court, stopping FareChase from selling software that enables users to compare online fares if the software also searches AA's website. The airline argued that FareChase's websearch software trespassed on AA's servers when it collected the publicly available data. FareChase filed an appeal in March 2003. By June, FareChase and AA agreed to settle and the appeal was dropped.^{[12]}
Southwest Airlines has also challenged screen-scraping practices, and has involved both FareChase and another firm, Outtask, in a legal claim. Southwest Airlines charged that the screen-scraping is Illegal since it is an example of "Computer Fraud and Abuse" and has led to "Damage and Loss" and "Unauthorized Access" of Southwest's site. It also constitutes "Interference with Business Relations", "Trespass", and "Harmful Access by Computer". They also claimed that screen-scraping constitutes what is legally known as "Misappropriation and Unjust Enrichment", as well as being a breach of the web site's user agreement. Outtask denied all these claims, claiming that the prevailing law in this case should be US Copyright law, and that under copyright, the pieces of information being scraped would not be subject to copyright protecti

on. Although the cases were never resolved in the Supreme Court of the United States, FareChase was eventually shuttered by parent company Yahoo!, and Outtask was purchased by travel expense company Concur.^{[13]} In 2012, a startup called 3Taps scraped classified housing ads from Craigslist. Craigslist sent 3Taps a cease-and-desist letter and blocked their IP addresses and later sued, in Craigslist v. 3Taps. The court held that the cease-and-desist letter and IP blocking was sufficient for Craigslist to properly claim that 3Taps had violated the Computer Fraud and Abuse Act.

Although these are early scraping decisions, and the theories of liability are not uniform, it is difficult to ignore a pattern emerging that the courts are prepared to protect proprietary content on commercial sites from uses which are undesirable to the owners of such sites. However, the degree of protection for such content is not settled, and will depend on the type of access made by the scraper, the amount of information accessed and copied, the degree to which the access adversely affects the site owner's system and the types and manner of prohibitions on such conduct.^{[14]}

While the law in this area becomes more settled, entities contemplating using scraping programs to access a public web site should also consider whether such action is authorized by reviewing the terms of use and other terms or notices posted on or made available through the site. In a 2010 ruling in the Cvent, Inc. v. Eventbrite, Inc. In the United States district court for the eastern district of Virginia, the court ruled that the terms of use should be brought to the users' attention In order for a browse wrap contract or license to be enforced.^{[15]} In a 2014, filed in the United States District Court for the Eastern District of Pennsylvania,^{[16]} e-commerce site QVC objected to the Pinterest-like shopping aggregator Resultly's `scraping of QVC's site for real-time pricing data. QVC alleges that Resultly "excessively crawled" QVC's retail site (allegedly sending 200-300 search requests to QVC's website per minute, sometimes to up to 36,000 requests per minute) which caused QVC's site to crash for two days, resulting in lost sales for QVC.^{[17]} QVC's complaint alleges that the defendant disguised its web crawler to mask its source IP address and thus prevented QVC from quickly repairing the problem. This is a particularly interesting scraping case because QVC is seeking damages for the unavailability of their website, which QVC claims was caused by Resultly.

In the plaintiff's web site during the period of this trial the terms of use link is displayed among all the links of the site, at the bottom of the page as most sites on the internet. This ruling contradicts the Irish ruling described below. The court also rejected the plaintiff's argument that the browse wrap restrictions were enforceable in view of Virginia's adoption of the Uniform Computer Information Transactions Act (UCITA)—a uniform law that many believed was in favor on common browse wrap contracting practices.^{[18]}

In Facebook, Inc. v. Power Ventures, Inc., a district court ruled in 2012 that Power Ventures could not scrape Facebook pages on behalf of a Facebook user. The case is on appeal, and the Electronic Frontier Foundation filed a brief in 2015 asking that it be overturned.^{[19]}^{[20]} In Associated Press v. Meltwater U.S. Holdings, Inc., a court in the US held Meltwater liable for scraping and republishing news information from the Associated Press, but a court in the United Kingdom held in favor of Meltwater.

<h3>The EU[edit]</h3>

Outside of the United States, in February 2006, the Danish Maritime and Commercial Court (Copenhagen) ruled that systematic crawling, indexing, and deep linking by portal site ofir.dk of estate site Home.dk does not conflict with Danish law or the database directive of the European Union.<sup class="reference" id="cite_ref-21

">[21]</sup>
In a February 2010 case complicated by matters of jurisdiction, Ireland's High Court delivered a verdict that illustrates the inchoate state of developing case law. In the case of Ryanair Ltd v Billigfluege.de GmbH, Ireland's High Court ruled Ryanair's "click-wrap" agreement to be legally binding. In contrast to the findings of the United States District Court Eastern District of Virginia and those of the Danish Maritime and Commercial Court, Mr. Justice Michael Hanna ruled that the hyperlink to Ryanair's terms and conditions was plainly visible, and that placing the onus on the user to agree to terms and conditions in order to gain access to online services is sufficient to comprise a contractual relationship. ^{[22]} The decision is under appeal in Ireland's Supreme Court.^{[23]}
<h3>Australia[edit]</h3>
In Australia, the Spam Act 2003 outlaws some forms of web harvesting, although this only applies to email addresses.^{[24]}^{[25]}
<h2>Methods to prevent web scraping[edit]</h2>
The administrator of a website can use various measures to stop or slow a bot. Some techniques include:

Blocking an IP address either manually or based on criteria such as geolocation and DNSRBL. This will also block all browsing from that address.
Disabling any web service API that the website's system might expose.
Bots sometimes declare who they are (using user agent strings) and can be blocked on that basis using robots.txt; 'googlebot' is an example. Other bots make no distinction between themselves and a human using a browser.
Bots can be blocked by monitoring excess traffic
Bots can sometimes be blocked with tools to verify that it is a real person accessing the site, like a CAPTCHA. Bots are sometimes coded to explicitly break specific CAPTCHA patterns or may employ third-party services that utilize human labor to read and respond in real-time to CAPTCHA challenges.
Commercial anti-bot services: Companies offer anti-bot and anti-scraping services for websites. A few web application firewalls have limited bot detection capabilities as well. However, many such solutions are not very effective^{[26]}.

Locating bots with a honeypot or other method to identify the IP addresses of automated crawlers.
Obfuscation using CSS sprites to display such data as phone numbers or email addresses, at the cost of accessibility to screen reader users.
Because bots rely on consistency in the front-end code of a target website, adding small variations to the HTML/CSS surrounding important data and navigation elements would require more human involvement in the initial set up of a bot and if done effectively may render the target website too difficult to scrape due to the diminished ability to automate the scraping process.

Websites can declare if crawling is allowed or not in the robots.txt file and allow partial access, limit the crawl rate, specify the optimal time to crawl and more.

<h2>See also[edit]</h2>

Archive.is
Comparison of feed aggregators
Data scraping
Data wrangling
Importer
Job wrapping
Knowledge extraction
OpenSocial
Scraper site
Fake news website
Blog scraping
Spamdexing
Domain name drop list
Text corpus
Web archiving
Blog network
Search Engine Scraping
Web crawlers

<h2>References[edit]</h2>

<ol class="references">
<li id="cite_note-Boeing2016JPER-1">^ ^a ^b <cite class="citation journal">Boeing, G.; Waddell, P. (2016). "New Insights into Rental Housing Markets across the United States: Web Scraping and Analyzing Craigslist Rental Listings". Journal of Planning Education and Research (0739456X16664789). doi:10.1177/0739456X16664789.</cite><span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&rft.genre=article&rft.jtitle=Journal+of+Planning+Education+and+Research&rft.atitle=New+Insights+into+Rental+Housing+Markets+across+the+United+States%3A+Web+Scraping+and+Analyzing+Craigslist+Rental+Listings&rft.issue=0739456X16664789&rft.date=2016&rft_id=info%3Adoi%2F10.1177%2F0739456X16664789&rft.aulast=Boeing&rft.aufirst=G.&rft.au=Waddell%2C+P.&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<li id="cite_note-2">^ <cite class="citation journal">Vargiu & Urru (2013). "Exploiting web scraping in a collaborative filtering- based approach to web advertising". Artificial Intelligence Research. 2 (1). doi:10.5430/air.v2n1p44.</cite><span class="Z3988" title

="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&rft.genre=article&rft.jtitle=Artificial+Intelligence+Research&rft.atitle=Exploiting+web+scraping+in+a+collaborative+filtering-+based+approach+to+web+advertising&rft.volume=2&rft.issue=1&rft.date=2013&rft_id=info%3Adoi%2F10.5430%2Fair.v2n1p44&rft.au=Vargiu+%26+Urru&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<li id="cite_note-3">^ <cite class="citation journal">Song, Ruihua; Microsoft Research (Sep 14, 2007). "Joint Optimization of Wrapper Generation and Template Detection" (PDF). The 13th International Conference on Knowledge Discovery and Data Mining.</cite><span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&rft.genre=article&rft.jtitle=The+13th+International+Conference+on+Knowledge+Discovery+and+Data+Mining&rft.atitle=Joint+Optimization+of+Wrapper+Generation+and+Template+Detection&rft.date=2007-09-14&rft.aulast=Song&rft.aufirst=Ruihua&rft.au=Microsoft+Research&rft_id=https%3A%2F%2Fpdfs.semanticscholar.org%2F4fb4%2F3c5a212df751e84c3b2f8d29fabfe56c3616.pdf&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<li id="cite_note-4">^ Semantic annotation based web scraping
<li id="cite_note-5">^ <cite class="citation web">Roush, Wade (2012-07-25). "Diffbot Is Using Computer Vision to Reinvent the Semantic Web". www.xconomy.com . Retrieved 2013-03-15.</cite><span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=Diffbot+Is+Using+Computer+Vision+to+Reinvent+the+Semantic+Web&rft.pub=www.xconomy.com&rft.date=2012-07-25&rft.aulast=Roush&rft.aufirst=Wade&rft_id=http%3A%2F%2Fwww.xconomy.com%2Fsan-francisco%2F2012%2F07%2F25%2Fdiffbot-is-using-computer-vision-to-reinvent-the-semantic-web%2F&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<li id="cite_note-6">^ <cite class="citation web">"FAQ about linking – Are website terms of use binding contracts?". www.chillingeffects.org. 2007-08-20. Archived from the original on 2002-03-08. Retrieved 2007-08-20.</cite><span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=FAQ+about+linking+%E2%80%93+Are+website+terms+of+use+binding+contracts%3F&rft.pub=www.chillingeffects.org&rft.date=2007-08-20&rft_id=http%3A%2F%2Fwww.chillingeffects.org%2Flinking%2Ffaq.cgi%23QID596&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<li id="cite_note-7">^ <cite class="citation journal">Kenneth, Hirschey, Jeffrey (2014-01-01). "Symbiotic Relationships: Pragmatic Acceptance of Data Scraping". Berkeley Technology Law Journal. 29 (4). doi:10.15779/Z38B39B. ISSN 1086-3818.</cite><span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&rft.genre=article&rft.jtitle=Berkeley+Technology+Law+Journal&rft.atitle=Symbiotic+Relationships%3A+Pragmatic+Acceptance+of+Data+Scraping&rft.volume=29&rft.issue=4&rft.date=2014-01-01&rft_id=info%3Adoi%2F10.15779%2FZ38B39B&rft.issn=1086-3818&rft.aulast=Kenneth&rft.aufirst=Hirschey%2C+Jeffrey&rft_id=http%3A%2F%2Fscholarship.law.berkeley.edu%2Fbtlj%2Fvol29%2Fiss4%2F16%2F&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">

<li id="cite_note-8">^ <cite class="citation web">"Internet Law, Ch. 06: Trespass to Chattels". www.tomwbell.com. 2007-08-20. Retrieved 2007-08-20.</cite><span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=Internet+Law%2C+Ch.+06%3A+Trespass+to+Chattels&rft.pub=www.tomwbell.com&rft.date=2007-08-20&rft_id=http%3A%2F%2Fwww.tomwbell.com%2FNetLaw%2FCh06.html&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<li id="cite_note-9">^ <cite class="citation web">"What are the "trespass to chattels" claims some companies or website owners have brought?". www.chillingeffects.org. 2007-08-20. Archived from the original on 2002-03-08. Retrieved 2007-08-20.</cite><span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=What+are+the+%22trespass+to+chattels%22+claims+some+companies+or+website+owners+have+brought%3F&rft.pub=www.chillingeffects.org&rft.date=2007-08-20&rft_id=http%3A%2F%2Fwww.chillingeffects.org%2Flinking%2Ffaq.cgi%23QID460&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<li id="cite_note-10">^ <cite class="citation web">"Ticketmaster Corp. v. Tickets.com, Inc." 2007-08-20. Retrieved 2007-08-20.</cite><span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=Ticketmaster+Corp.+v.+Tickets.com%2C+Inc.&rft.date=2007-08-20&rft_id=http%3A%2F%2Fwww.tomwbell.com%2FNetLaw%2FCh07%2FTicketmaster.html&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<li id="cite_note-11">^ <cite class="citation web">"American Airlines v. FareChase" (PDF). 2007-08-20. Archived from the original (PDF) on 2011-07-23. Retrieved 2007-08-20.</cite><span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=American+Airlines+v.+FareChase&rft.date=2007-08-20&rft_id=http%3A%2F%2Fwww.fornova.net%2Fdocuments%2FAAFareChase.pdf&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<li id="cite_note-12">^ <cite class="citation web">"American Airlines, FareChase Settle Suit". The Free Library. 2003-06-13. Retrieved 2012-02-26.</cite><span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=American+Airlines%2C+FareChase+Settle+Suit.&rft.pub=The+Free+Library&rft.date=2003-06-13&rft_id=http%3A%2F%2Fwww.thefreelibrary.com%2FAmerican%2BAirlines%2C%2BFareChase%2BSettle%2BSuit.-a0103213546&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<li id="cite_note-impervawp2011-13">^ Imperva (2011). Detecting and Blocking Site Scraping Attacks. Imperva white paper..
<li id="cite_note-14">^ <cite class="citation web">Adler, Kenneth A. (2003-07-29). <a class="external text" href

="http://library.findlaw.com/2003/Jul/29/132944.html" rel="nofollow">"Controversy Surrounds 'Screen Scrapers': Software Helps Users Access Web Sites But Activity by Competitors Comes Under Scrutiny". Retrieved 2010-10-27.</cite><span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=Controversy+Surrounds+%27Screen+Scrapers%27%3A+Software+Helps+Users+Access+Web+Sites+But+Activity+by+Competitors+Comes+Under+Scrutiny&rft.date=2003-07-29&rft.aulast=Adler&rft.aufirst=Kenneth+A.&rft_id=http%3A%2F%2Flibrary.findlaw.com%2F2003%2FJul%2F29%2F132944.html&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<li id="cite_note-15">^ <cite class="citation web">"QVC Inc. v. Resultly LLC, No. 14-06714 (E.D. Pa. filed Nov. 24, 2014)" (PDF). 2014-11-24. Retrieved 2015-11-05.</cite><span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=QVC+Inc.+v.+Resultly+LLC%2C+No.+14-06714+%28E.D.+Pa.+filed+Nov.+24%2C+2014%29&rft.date=2014-11-24&rft_id=http%3A%2F%2Fwww.fornova.net%2Fdocuments%2FCvent.pdf&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<li id="cite_note-16">^ <cite class="citation web">"QVC Inc. v. Resultly LLC, No. 14-06714 (E.D. Pa. filed Nov. 24, 2014)". United States District Court for the Eastern District of Pennsylvania. Retrieved 5 November 2015.</cite><span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&rft.genre=unknown&rft.jtitle=United+States+District+Court+for+the+Eastern+District+of+Pennsylvania&rft.atitle=QVC+Inc.+v.+Resultly+LLC%2C+No.+14-06714+%28E.D.+Pa.+filed+Nov.+24%2C+2014%29&rft_id=https%3A%2F%2Fwww.scribd.com%2Fdoc%2F249068700%2FLinkedIn-v-Resultly-LLC-Complaint%3Fsecret_password%3DpEVKDbnvhQL52oKfdrmT&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<li id="cite_note-17">^ <cite class="citation journal">Neuburger, Jeffrey D (5 December 2014). "QVC Sues Shopping App for Web Scraping That Allegedly Triggered Site Outage". The National Law Review. Proskauer Rose LLP. Retrieved 5 November 2015.</cite><span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&rft.genre=article&rft.jtitle=The+National+Law+Review&rft.atitle=QVC+Sues+Shopping+App+for+Web+Scraping+That+Allegedly+Triggered+Site+Outage&rft.date=2014-12-05&rft.aulast=Neuburger&rft.aufirst=Jeffrey+D&rft_id=http%3A%2F%2Fnewmedialaw.proskauer.com%2F2014%2F12%2F05%2Fqvc-sues-shopping-app-for-web-scraping-that-allegedly-triggered-site-outage%2F&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<li id="cite_note-18">^ <cite class="citation web">"Did Iqbal/Twombly Raise the Bar for Browsewrap Claims?" (PDF). 2010-09-17. Retrieved 2010-10-27.</cite><span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=Did+Iqbal%2FTwombly+Raise+the+Bar+for+Browsewrap+Claims%3F&rft.date=2010-09-17&rft_id=http%3A%2F%2Fwww.fornova.net%2Fdocuments%2Fpblog-bna-com.pdf&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<li id="cite_note-19">^ <cite class="citation web">"Can Scraping Non-Infringing Content Become Copyright Infringement... Because Of How Scrapers Work? | Techdirt". Techdirt. 2009-06-10. Retrieved 2016-05-24.</cite><span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&rft.genre

=unknown&rft.jtitle=Techdirt.&rft.atitle=Can+Scraping+Non-Infringing+Content+Become+Copyright+Infringement...+Because+Of+How+Scrapers+Work%3F+%7C+Techdirt&rft.date=2009-06-10&rft_id=https%3A%2F%2Fwww.techdirt.com%2Farticles%2F20090605%2F2228205147.shtml&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<li id="cite_note-20">^ <cite class="citation web">"Facebook v. Power Ventures". Electronic Frontier Foundation. Retrieved 2016-05-24.</cite><span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&rft.genre=unknown&rft.jtitle=Electronic+Frontier+Foundation&rft.atitle=Facebook+v.+Power+Ventures&rft_id=https%3A%2F%2Fwww.eff.org%2Fcases%2Ffacebook-v-power-ventures&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<li id="cite_note-21">^ <cite class="citation web">"UDSKRIFT AF SØ- & HANDELSRETTENS DOMBOG" (PDF) (in Danish). bvhd.dk. 2006-02-24. Archived from the original (PDF) on 2007-10-12. Retrieved 2007-05-30.</cite><span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=UDSKRIFT+AF+S%C3%98-+%26+HANDELSRETTENS+DOMBOG&rft.pub=bvhd.dk&rft.date=2006-02-24&rft_id=http%3A%2F%2Fwww.bvhd.dk%2Fuploads%2Ftx_mocarticles%2FS_-_og_Handelsrettens_afg_relse_i_Ofir-sagen.pdf&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<li id="cite_note-22">^ <cite class="citation web">"High Court of Ireland Decisions >> Ryanair Ltd -v- Billigfluege.de GMBH 2010 IEHC 47 (26 February 2010)". British and Irish Legal Information Institute. 2010-02-26. Retrieved 2012-04-19.</cite><span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=High+Court+of+Ireland+Decisions+%3E%3E+Ryanair+Ltd+-v-+Billigfluege.de+GMBH+2010+IEHC+47+%2826+February+2010%29&rft.pub=British+and+Irish+Legal+Information+Institute&rft.date=2010-02-26&rft_id=http%3A%2F%2Fwww.bailii.org%2Fie%2Fcases%2FIEHC%2F2010%2FH47.html&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<li id="cite_note-23">^ <cite class="citation web">Matthews, Áine (June 2010). "Intellectual Property: Website Terms of Use". Issue 26: June 2010. LK Shields Solicitors Update. p. 03. Retrieved 2012-04-19.</cite><span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&rft.genre=unknown&rft.jtitle=Issue+26%3A+June+2010&rft.atitle=Intellectual+Property%3A+Website+Terms+of+Use&rft.pages=03&rft.date=2010-06&rft.aulast=Matthews&rft.aufirst=%C3%81ine&rft_id=http%3A%2F%2Fwww.lkshields.ie%2Fhtmdocs%2Fpublications%2Fnewsletters%2Fupdate26%2Fupdate26_03.htm&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<li id="cite_note-24">^ <cite class="citation web">National Office for the Information Economy (February 2004). "Spam Act 2003: An overview for business". Australian Communications Authority. p. 6. Retrieved 2017-12-07.</cite><span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=Spam+Act+2003%3A+An+overview+for+business&rft.pages=6&rft.pub=Australian+Communications+Authority&rft.date=2004-02&rft.au=National+Office+for+the+Information+Economy&rft_id=https%3A%2F%2Fwww.lloyds.com%2F~%2Fmedia%2F5880d

ae185914b2487bed7bd63b96286.ashx&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<li id="cite_note-25">^ <cite class="citation web">National Office for the Information Economy (February 2004). "Spam Act 2003: A practical guide for business" (PDF). Australian Communications Authority. p. 20. Retrieved 2017-12-07.</cite><span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=Spam+Act+2003%3A+A+practical+guide+for+business&rft.pages=20&rft.pub=Australian+Communications+Authority&rft.date=2004-02&rft.au=National+Office+for+the+Information+Economy&rft_id=http%3A%2F%2Fwww.webstartdesign.com.au%2Fspam_business_practical_guide.pdf&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<li id="cite_note-26">^ Mayank Dhiman Breaking Fraud & Bot Detection Solutions OWASP AppSec Cali' 2018 Retrieved February 10, 2018.

<!--
NewPP limit report
Parsed by mw1249
Cached time: 20180226071249
Cache expiry: 1900800
Dynamic content: false
CPU time usage: 0.228 seconds
Real time usage: 0.285 seconds
Preprocessor visited node count: 1447/1000000
Preprocessor generated node count: 0/1500000
Post-expand include size: 49571/2097152 bytes
Template argument size: 320/2097152 bytes
Highest expansion depth: 10/40
Expensive parser function count: 3/500
Lua time usage: 0.119/10.000 seconds
Lua memory usage: 4.53 MB/50 MB
-->
<!--
Transclusion expansion time report (%,ms,calls,template)
100.00% 240.818   1 -total
 59.86% 144.146   1 Template:Reflist
 28.76%   69.247   18 Template:Cite_web
 19.28%   46.421   5 Template:Cite_journal
 17.92%   43.157   1 Template:More_citations_needed
 13.68%   32.949   2 Template:Ambox
  4.68%   11.269   1 Template:US-centric
  4.39%   10.578   1 Template:Further_information
  3.99%   9.620   1 Template:Colbegin
  3.20%   7.700   1 Template:Globalize
-->

<!-- Saved in parser cache with key enwiki:pcache:idhash:2696619-0!canonical and timestamp 20180226071248 and revision id 825057514
 -->
<noscript></noscript>
Retrieved from "https://en.wikipedia.org/w/index.php?title=Web_scraping&oldid=82

5057514"
<div class="mw-normal-catlinks" id="mw-normal-cat
links">Categories: <a href="/wiki/Categor
y:Web_scraping" title="Category:Web scraping">Web scraping
<a href="/wiki/Category:World_
Wide_Web" title="Category:World Wide Web">World Wide Web
<a href="/wiki/Category:Spam
ming" title="Category:Spamming">Spamming <div class="mw-hidden-catlinks mw-hidde
n-cats-hidden" id="mw-hidden-catlinks">Hidden categories: <a href="/wiki/Category:CS1_Danish-lan
guage_sources_(da)" title="Category:CS1 Danish-language sources (da)">CS1 Danish-language sources (da)</
a>
<a href="/wiki/Category:Articles_needing_additional_references_from_June_2017" title="Category
:Articles needing additional references from June 2017">Articles needing additional references from June 2017
<a href="/wiki/Category:All_articles_needing_additional_references" title="Category:All articles
needing additional references">All articles needing additional references
<a href="/wiki/Category:
Articles_with_limited_geographic_scope_from_October_2015" title="Category:Articles with limited geographi
c scope from October 2015">Articles with limited geographic scope from October 2015
<a href="/
wiki/Category:USA-centric" title="Category:USA-centric">USA-centric
<a href="/wiki/Category:
Pages_using_div_col_with_deprecated_parameters" title="Category:Pages using div col with deprecated param
eters">Pages using div col with deprecated parameters </
div>

<h2>Navigation menu</h2>

<h3 id="p-personal-label">Personal tools</h3>

<li id="pt-anonuserpage">Not logged in
<li id="pt-anontalk"><a accesskey="n" href="/wiki/Special:MyTa
lk" title="Discussion about edits from this IP address [n]">Talk
<li id="pt-anoncontribs"><a accesskey
="y" href="/wiki/Special:MyContributions" title="A list of edits made from this IP address [y]">Contributions<
/a>
<li id="pt-createaccount"><a href="/w/index.php?title=Special:CreateAccount&returnto=Web+scr
aping" title="You are encouraged to create an account and log in; however, it is not mandatory">Create account
<li id="pt-login"><a accesskey="o" href="/w/index.php?title=Special:UserLogin&returnto=Web
+scraping" title="You're encouraged to log in; however, it's not mandatory. [o]">Log in

<h3 id="p-namespaces-label">Namespaces</h3>

<li class="selected" id="ca-nstab-main"><a accesskey="c" href="/wiki/Web_scraping" title="View the
content page [c]">Article
<li id="ca-talk"><a accesskey="t" href="/wiki/Talk:Web_scr
aping" rel="discussion" title="Discussion about the content page [t]">Talk

<input aria-labelledby="p-variants-label" class="vectorMenuCheckbox" type="checkbox"/>
<h3 id="p-variants-label">
Variants
</h3>

<h3 id="p-views-label">Views</h3>

<li class="collapsible selected" id="ca-view">Read
<li class="collapsible" id="ca-edit">Edit
<li class="collapsible" id="ca-history">View history

<input aria-labelledby="p-cactions-label" class="vectorMenuCheckbox" type="checkbox"/>
<h3 id="p-cactions-label">More</h3>

<h3>
<label for="searchInput">Search</label>
</h3>
<form action="/w/index.php" id="searchform">

<input accesskey="f" id="searchInput" name="search" placeholder="Search Wikipedia" title="Search Wikipedia [f]" type="search"/><input name="title" type="hidden" value="Special:Search"/><input class="searchButton mw-fallbackSearchButton" id="mw-searchButton" name="fulltext" title="Search Wikipedia for this text" type="submit" value="Search"/><input class="searchButton" id="searchButton" name="go" title="Go to a page with this exact name if it exists" type="submit" value="Go"/>
</form>

<h3 id="p-navigation-label">Navigation</h3>

<li id="n-mainpage-description">Main page
<li id="n-contents">Contents
<li id="n-featuredcontent">Featured content
<li id="n-currentevents">Current events
<li id="n-randompage">Random article
<li id="n-sitesupport">Donate to Wikipedia
<li id="n-shoplink">Wikipedia store

<h3 id="p-interaction-label">Interaction</h3>

<li id="n-help">Help
<li id="n-aboutsite">About Wikipedia
<li id="n-portal">Community portal
<li id="n-recentchanges">Recent changes
<li id="n-contac

tpage">Contact page

<h3 id="p-tb-label">Tools</h3>

<li id="t-whatlinkshere">What links here
<li id="t-recentchangeslinked">Related changes
<li id="t-upload">Upload file
<li id="t-specialpages">Special pages
<li id="t-permalink">Permanent link
<li id="t-info">Page information
<li id="t-wikibase">Wikidata item
<li id="t-cite">Cite this page

<h3 id="p-coll-print_export-label">Print/export</h3>

<li id="coll-create_a_book">Create a book
<li id="coll-download-as-rdf2latex">Download as PDF
<li id="t-print">Printable version

<h3 id="p-lang-label">Languages</h3>

<li class="interlanguage-link interwiki-ar">العربية
<li class="interlanguage-link interwiki-ca">Català
<li class="interlanguage-link interwiki-de">Deutsch
<li class="interlanguage-link interwiki-es">Español
<li class="interlanguage-link interwiki-eu">Euskara
<li class="interlanguage-link interwiki-fr">Français
<li class="interlanguage-link interwiki-is">Íslenska
<li class="interlanguage-link interwiki-it">Italiano
<li class="interlanguage-link interwiki-lv">Latviešu
<li class="interlanguage-link interwiki-nl"><a class="interlanguage-link-target" href="https://nl.wikipedia.org/wiki/Scrape

n" hreflang="nl" lang="nl" title="Scrapen – Dutch">Nederlands
<li class="interlanguage-link interwiki-ja">日本語
<li class="interlanguage-link interwiki-sr">Српски / srpski
<li class="interlanguage-link interwiki-tr">Türkçe
<li class="interlanguage-link interwiki-uk">Українська
<li class="interlanguage-link interwiki-zh">中文
Edit links

<ul id="footer-info">
<li id="footer-info-lastmod"> This page was last edited on 11 February 2018, at 06:38.
<li id="footer-info-copyright">Text is available under the Creative Commons Attribution-ShareAlike License;
additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.

<ul id="footer-places">
<li id="footer-places-privacy">Privacy policy
<li id="footer-places-about">About Wikipedia
<li id="footer-places-disclaimer">Disclaimers
<li id="footer-places-contact">Contact Wikipedia
<li id="footer-places-developers">Developers
<li id="footer-places-cookiestatement">Cookie statement
<li id="footer-places-mobileview">Mobile view

<ul class="noprint" id="footer-icons">
<li id="footer-copyrightico">

<li id="footer-poweredbyico">


```
</div>
<script>(window.RLQ=window.RLQ||[]).push(function(){mw.config.set({"wgPageParseReport":{"limitreport"
:{"cputime":"0.228","walltime":"0.285","ppvisitednodes":{"value":1447,"limit":1000000},"ppgeneratednodes":
{"value":0,"limit":1500000},"postexpandincludesize":{"value":49571,"limit":2097152},"templateargumentsize
":{"value":320,"limit":2097152},"expansiondepth":{"value":10,"limit":40},"expensivefunctioncount":{"value":
3,"limit":500},"entityaccesscount":{"value":0,"limit":400},"timingprofile":["100.00%  240.818      1 -total","
59.86%  144.146      1 Template:Reflist","  28.76%   69.247     18 Template:Cite_web","  19.28%   46.421
5 Template:Cite_journal","  17.92%   43.157      1 Template:More_citations_needed","  13.68%   32.949
2 Template:Ambox","   4.68%   11.269      1 Template:US-centric","   4.39%   10.578      1 Template:Further_information"
","   3.99%    9.620      1 Template:Colbegin","   3.20%    7.700      1 Template:Globalize"]},"scribunto":{"limitrep
ort-timeusage":{"value":"0.119","limit":"10.000"},"limitreport-memusage":{"value":4752818,"limit":52428800
}},"cachereport":{"origin":"mw1249","timestamp":"20180226071249","ttl":1900800,"transientcontent":false}}
});});</script><script>(window.RLQ=window.RLQ||[]).push(function(){mw.config.set({"wgBackendResponse
Time":374,"wgHostname":"mw1249"});});</script>
</body>
</html>
```


Now, you have created soup object and can call method avalaible for the object

How to format HTML even more cleaner?

In [9]:
```
print(soup.prettify())
```


```
<!DOCTYPE html>
<html class="client-nojs" dir="ltr" lang="en">
 <head>
  <meta charset="utf-8"/>
  <title>
   Web scraping - Wikipedia
  </title>
  <script>
   document.documentElement.className = document.documentElement.className.replace( /(^|\s)client-nojs(\
s|$)/, "$1client-js$2" );
  </script>
  <script>
   (window.RLQ=window.RLQ||[]).push(function(){mw.config.set({"wgCanonicalNamespace":"","wgCanonica
lSpecialPageName":false,"wgNamespaceNumber":0,"wgPageName":"Web_scraping","wgTitle":"Web scraping
","wgCurRevisionId":825057514,"wgRevisionId":825057514,"wgArticleId":2696619,"wgIsArticle":true,"wgIs
Redirect":false,"wgAction":"view","wgUserName":null,"wgUserGroups":["*"],"wgCategories":["CS1 Danish-l
anguage sources (da)","Articles needing additional references from June 2017","All articles needing additional r
eferences","Articles with limited geographic scope from October 2015","USA-centric","Pages using div col wit
h deprecated parameters","Web scraping","World Wide Web","Spamming"],"wgBreakFrames":false,"wgPageC
ontentLanguage":"en","wgPageContentModel":"wikitext","wgSeparatorTransformTable":["",""],"wgDigitTrans
formTable":["",""],"wgDefaultDateFormat":"dmy","wgMonthNames":["","January","February","March","April
","May","June","July","August","September","October","November","December"],"wgMonthNamesShort":["",
"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"],"wgRelevantPageName":"Web_s
craping","wgRelevantArticleId":2696619,"wgRequestId":"WpOzcApAMFQAAKv1HOgAAABY","wgIsProba
blyEditable":true,"wgRelevantPageIsProbablyEditable":true,"wgRestrictionEdit":[],"wgRestrictionMove":[],"w
gFlaggedRevsParams":{"tags":{}},"wgStableRevisionId":null,"wgWikiEditorEnabledModules":[],"wgBetaFeat
uresFeatures":[],"wgMediaViewerOnClick":true,"wgMediaViewerEnabledByDefault":true,"wgPopupsShouldS
endModuleToUser":true,"wgPopupsConflictsWithNavPopupGadget":false,"wgVisualEditor":{"pageLanguage
Code":"en","pageLanguageDir":"ltr","pageVariantFallbacks":"en","usePageImages":true,"usePageDescriptions
":true},"wgPreferredVariant":"en","wgMFExpandAllSectionsUserOption":true,"wgMFEnableFontChanger":tru
e,"wgMFDisplayWikibaseDescriptions":{"search":false,"nearby":false,"watchlist":false,"tagline":false},"wgRel
```

atedArticles":null,"wgRelatedArticlesUseCirrusSearch":true,"wgRelatedArticlesOnlyUseCirrusSearch":false,"wgULSCurrentAutonym":"English","wgNoticeProject":"wikipedia","wgCentralNoticeCookiesToDelete":[],"wgCentralNoticeCategoriesUsingLegacy":["Fundraising","fundraising"],"wgCategoryTreePageCategoryOptions":"{\"mode\":0,\"hideprefix\":20,\"showcount\":true,\"namespaces\":false}","wgWikibaseItemId":"Q665452","wgScoreNoteLanguages":{"arabic":"العربية","catalan":"català","deutsch":"Deutsch","english":"English","espanol":"español","italiano":"italiano","nederlands":"Nederlands","norsk":"norsk","portugues":"português","suomi":"suomi","svenska":"svenska","vlaams":"West-Vlams"},"wgScoreDefaultNoteLanguage":"nederlands","wgCentralAuthMobileDomain":false,"wgCodeMirrorEnabled":false,"wgVisualEditorToolbarScrollOffset":0,"wgVisualEditorUnsupportedEditParams":["undo","undoafter","veswitched"],"wgEditSubmitButtonLabelPublish":true});mw.loader.state({"ext.gadget.charinsert-styles":"ready","ext.globalCssJs.user.styles":"ready","ext.globalCssJs.site.styles":"ready","site.styles":"ready","noscript":"ready","user.styles":"ready","user":"ready","user.options":"ready","user.tokens":"loading","ext.cite.styles":"ready","wikibase.client.init":"ready","ext.visualEditor.desktopArticleTarget.noscript":"ready","ext.uls.interlanguage":"ready","ext.wikimediaBadges":"ready","mediawiki.legacy.shared":"ready","mediawiki.legacy.commonPrint":"ready","mediawiki.sectionAnchor":"ready","mediawiki.skinning.interface":"ready","skins.vector.styles":"ready","ext.globalCssJs.user":"ready","ext.globalCssJs.site":"ready"});mw.loader.implement("user.tokens@1dqfd7l",function($,jQuery,require,module){/*@nomin*/mw.user.tokens.set({"editToken":"+\\","patrolToken":"+\\","watchToken":"+\\","csrfToken":"+\\"});
});mw.loader.load(["ext.cite.a11y","site","mediawiki.page.startup","mediawiki.user","mediawiki.hidpi","mediawiki.page.ready","mediawiki.toc","mediawiki.searchSuggest","ext.gadget.teahouse","ext.gadget.ReferenceTooltips","ext.gadget.watchlist-notice","ext.gadget.DRN-wizard","ext.gadget.charinsert","ext.gadget.refToolbar","ext.gadget.extra-toolbar-buttons","ext.gadget.switcher","ext.centralauth.centralautologin","mmv.head","mmv.bootstrap.autostart","ext.popups","ext.visualEditor.desktopArticleTarget.init","ext.visualEditor.targetLoader","ext.eventLogging.subscriber","ext.wikimediaEvents","ext.navigationTiming","ext.uls.eventlogger","ext.uls.init","ext.uls.interface","ext.3d","ext.centralNotice.geoIP","ext.centralNotice.startUp","skins.vector.js"]);});
</script>
<link href="/w/load.php?debug=false&lang=en&modules=ext.cite.styles%7Cext.uls.interlanguage%7Cext.visualEditor.desktopArticleTarget.noscript%7Cext.wikimediaBadges%7Cmediawiki.legacy.commonPrint%2Cshared%7Cmediawiki.sectionAnchor%7Cmediawiki.skinning.interface%7Cskins.vector.styles%7Cwikibase.client.init&only=styles&skin=vector" rel="stylesheet"/>
<script async="" src="/w/load.php?debug=false&lang=en&modules=startup&only=scripts&skin=vector">
</script>
<meta content="" name="ResourceLoaderDynamicStyles"/>
<link href="/w/load.php?debug=false&lang=en&modules=ext.gadget.charinsert-styles&only=styles&skin=vector" rel="stylesheet"/>
<link href="/w/load.php?debug=false&lang=en&modules=site.styles&only=styles&skin=vector" rel="stylesheet"/>
<meta content="MediaWiki 1.31.0-wmf.22" name="generator"/>
<meta content="origin" name="referrer"/>
<meta content="origin-when-crossorigin" name="referrer"/>
<meta content="origin-when-cross-origin" name="referrer"/>
<link href="android-app://org.wikipedia/http/en.m.wikipedia.org/wiki/Web_scraping" rel="alternate"/>
<link href="/w/index.php?title=Web_scraping&action=edit" rel="alternate" title="Edit this page" type="application/x-wiki"/>
<link href="/w/index.php?title=Web_scraping&action=edit" rel="edit" title="Edit this page"/>
<link href="/static/apple-touch/wikipedia.png" rel="apple-touch-icon"/>
<link href="/static/favicon/wikipedia.ico" rel="shortcut icon"/>
<link href="/w/opensearch_desc.php" rel="search" title="Wikipedia (en)" type="application/opensearchdescription+xml"/>
<link href="//en.wikipedia.org/w/api.php?action=rsd" rel="EditURI" type="application/rsd+xml"/>
<link href="//creativecommons.org/licenses/by-sa/3.0/" rel="license"/>
<link href="https://en.wikipedia.org/wiki/Web_scraping" rel="canonical"/>
<link href="//login.wikimedia.org" rel="dns-prefetch"/>
<link href="//meta.wikimedia.org" rel="dns-prefetch"/>
<!--[if lt IE 9]><script src="/w/load.php?debug=false&lang=en&modules=html5shiv&only=scripts&skin=vector&sync=1"></script><![endif]-->
</head>

```
 <body class="mediawiki ltr sitedir-ltr mw-hide-empty-elt ns-0 ns-subject page-Web_scraping rootpage-Web_s
craping skin-vector action-view">
  <div class="noprint" id="mw-page-base">
  </div>
  <div class="noprint" id="mw-head-base">
  </div>
  <div class="mw-body" id="content" role="main">
   <a id="top">
   </a>
   <div class="mw-body-content" id="siteNotice">
    <!-- CentralNotice -->
   </div>
   <div class="mw-indicators mw-body-content">
   </div>
   <h1 class="firstHeading" id="firstHeading" lang="en">
    Web scraping
   </h1>
   <div class="mw-body-content" id="bodyContent">
    <div class="noprint" id="siteSub">
     From Wikipedia, the free encyclopedia
    </div>
    <div id="contentSub">
    </div>
    <div class="mw-jump" id="jump-to-nav">
     Jump to:
     <a href="#mw-head">
      navigation
     </a>
     ,
     <a href="#p-search">
      search
     </a>
    </div>
    <div class="mw-content-ltr" dir="ltr" id="mw-content-text" lang="en">
     <div class="mw-parser-output">
      <table class="plainlinks metadata ambox ambox-content ambox-Refimprove" role="presentation">
       <tr>
        <td class="mbox-image">
         <div style="width:52px">
          <a class="image" href="/wiki/File:Question_book-new.svg">
           <img alt="" data-file-height="399" data-file-width="512" height="39" src="//upload.wikimedia.org/wiki
pedia/en/thumb/9/99/Question_book-new.svg/50px-Question_book-new.svg.png" srcset="//upload.wikimedia.o
rg/wikipedia/en/thumb/9/99/Question_book-new.svg/75px-Question_book-new.svg.png 1.5x, //upload.wikimed
ia.org/wikipedia/en/thumb/9/99/Question_book-new.svg/100px-Question_book-new.svg.png 2x" width="50"/>
          </a>
         </div>
        </td>
        <td class="mbox-text">
         <div class="mbox-text-span">
          This article
          <b>
           needs additional citations for
           <a href="/wiki/Wikipedia:Verifiability" title="Wikipedia:Verifiability">
            verification
           </a>
          </b>
          .
```

```
          <span class="hide-when-compact">
           Please help
           <a class="external text" href="//en.wikipedia.org/w/index.php?title=Web_scraping&amp;action=edit">
            improve this article
           </a>
           by
           <a href="/wiki/Help:Introduction_to_referencing_with_Wiki_Markup/1" title="Help:Introduction to refe
rencing with Wiki Markup/1">
            adding citations to reliable sources
           </a>
           . Unsourced material may be challenged and removed.
          </span>
          <small>
           <i>
            (June 2017)
           </i>
          </small>
          <small class="hide-when-compact">
           <i>
            (
            <a href="/wiki/Help:Maintenance_template_removal" title="Help:Maintenance template removal">
             Learn how and when to remove this template message
            </a>
            )
           </i>
          </small>
         </div>
        </td>
       </tr>
      </table>
      <div class="hatnote navigation-not-searchable" role="note">
       For a broader coverage related to this topic, see
       <a href="/wiki/Data_scraping" title="Data scraping">
        Data scraping
       </a>
       .
      </div>
      <p>
       <b>
        Web scraping
       </b>
       ,
       <b>
        web harvesting
       </b>
       , or
       <b>
        web data extraction
       </b>
       is
       <a href="/wiki/Data_scraping" title="Data scraping">
        data scraping
       </a>
       used for
       <a href="/wiki/Data_extraction" title="Data extraction">
        extracting data
       </a>
```

from

websites

.
<sup class="reference" id="cite_ref-Boeing2016JPER_1-0">

[1]

</sup>
Web scraping software may access the World Wide Web directly using the

Hypertext Transfer Protocol

, or through a web browser. While web scraping can be done manually by a software user, the term typicall y refers to automated processes implemented using a

bot

or

web crawler

. It is a form of copying, in which specific data is gathered and copied from the web, typically into a central local

database

or spreadsheet, for later

retrieval

or

analysis

.

Web scraping a web page involves fetching it and extracting from it.
<sup class="reference" id="cite_ref-Boeing2016JPER_1-1">

[1]

</sup>
<sup class="reference" id="cite_ref-2">

[2]

</sup>
Fetching is the downloading of a page (which a browser does when you view the page). Therefore, web cra wling is a main component of web scraping, to fetch pages for later processing. Once fetched, then extraction ca n take place. The content of a page may be

parsed

, searched, reformatted, its data copied into a spreadsheet, and so on. Web scrapers typically take something

```
 out of a page, to make use of it for another purpose somewhere else. An example would be to find and copy na
mes and phone numbers, or companies and their URLs, to a list (contact scraping).
    </p>
    <p>
     Web scraping is used for
     <a href="/wiki/Contact_scraping" title="Contact scraping">
      contact scraping
     </a>
     , and as a component of applications used for
     <a href="/wiki/Web_indexing" title="Web indexing">
      web indexing
     </a>
     ,
     <a href="/wiki/Web_mining" title="Web mining">
      web mining
     </a>
     and
     <a href="/wiki/Data_mining" title="Data mining">
      data mining
     </a>
     , online price change monitoring and
     <a href="/wiki/Comparison_shopping_website" title="Comparison shopping website">
      price comparison
     </a>
     , product review scraping (to watch the competition), gathering real estate listings, weather data monitoring,
     <a href="/wiki/Change_detection_and_notification" title="Change detection and notification">
      website change detection
     </a>
     , research, tracking online presence and reputation,
     <a class="mw-redirect" href="/wiki/Web_mashup" title="Web mashup">
      web mashup
     </a>
     and, web data integration.
    </p>
    <p>
     <a href="/wiki/Web_page" title="Web page">
      Web pages
     </a>
     are built using text-based mark-up languages (
     <a href="/wiki/HTML" title="HTML">
      HTML
     </a>
     and
     <a href="/wiki/XHTML" title="XHTML">
      XHTML
     </a>
     ), and frequently contain a wealth of useful data in text form. However, most web pages are designed for hu
man
     <a class="mw-redirect" href="/wiki/End-user_(computer_science)" title="End-user (computer science)">
      end-users
     </a>
     and not for ease of automated use. Because of this, tool kits that scrape web content were created. A web sc
raper is an
     <a class="mw-redirect" href="/wiki/Application_Programming_Interface" title="Application Programming
 Interface">
      Application Programming Interface
     </a>
```

(API) to extract data from a web site. Companies like

Amazon AWS

and

Google

provide web scraping tools, services and public data available free of cost to end users.

Newer forms of web scraping involve listening to data feeds from web servers. For example,

JSON

is commonly used as a transport storage mechanism between the client and the web server.

There are methods that some websites use to prevent web scraping, such as detecting and disallowing bots f
rom crawling (viewing) their pages. In response, there are web scraping systems that rely on using techniques in

DOM

parsing,

computer vision

and

natural language processing

to simulate human browsing to enable gathering web page content for offline parsing.

<h2>
Contents
</h2>

<li class="toclevel-1 tocsection-1">

1

Techniques

<li class="toclevel-2 tocsection-2">
<a href="#Human_copy-and-paste">

1.1

```
 <span class="toctext">
  Human copy-and-paste
 </span>
</a>
</li>
<li class="toclevel-2 tocsection-3">
 <a href="#Text_pattern_matching">
  <span class="tocnumber">
   1.2
  </span>
  <span class="toctext">
   Text pattern matching
  </span>
 </a>
</li>
<li class="toclevel-2 tocsection-4">
 <a href="#HTTP_programming">
  <span class="tocnumber">
   1.3
  </span>
  <span class="toctext">
   HTTP programming
  </span>
 </a>
</li>
<li class="toclevel-2 tocsection-5">
 <a href="#HTML_parsing">
  <span class="tocnumber">
   1.4
  </span>
  <span class="toctext">
   HTML parsing
  </span>
 </a>
</li>
<li class="toclevel-2 tocsection-6">
 <a href="#DOM_parsing">
  <span class="tocnumber">
   1.5
  </span>
  <span class="toctext">
   DOM parsing
  </span>
 </a>
</li>
<li class="toclevel-2 tocsection-7">
 <a href="#Vertical_aggregation">
  <span class="tocnumber">
   1.6
  </span>
  <span class="toctext">
   Vertical aggregation
  </span>
 </a>
</li>
<li class="toclevel-2 tocsection-8">
 <a href="#Semantic_annotation_recognizing">
```

```
<span class="tocnumber">
 1.7
</span>
<span class="toctext">
 Semantic annotation recognizing
</span>
</a>
</li>
<li class="toclevel-2 tocsection-9">
<a href="#Computer_vision_web-page_analysis">
<span class="tocnumber">
 1.8
</span>
<span class="toctext">
 Computer vision web-page analysis
</span>
</a>
</li>
</ul>
</li>
<li class="toclevel-1 tocsection-10">
<a href="#Software">
<span class="tocnumber">
 2
</span>
<span class="toctext">
 Software
</span>
</a>
<ul>
<li class="toclevel-2 tocsection-11">
<a href="#Example_tools">
<span class="tocnumber">
 2.1
</span>
<span class="toctext">
 Example tools
</span>
</a>
<ul>
<li class="toclevel-3 tocsection-12">
<a href="#Javascript_tools">
<span class="tocnumber">
 2.1.1
</span>
<span class="toctext">
 Javascript tools
</span>
</a>
</li>
<li class="toclevel-3 tocsection-13">
<a href="#SaaS_version">
<span class="tocnumber">
 2.1.2
</span>
<span class="toctext">
 SaaS version
```

```
      </span>
     </a>
    </li>
    <li class="toclevel-3 tocsection-14">
     <a href="#Web_crawling_frameworks">
      <span class="tocnumber">
       2.1.3
      </span>
      <span class="toctext">
       Web crawling frameworks
      </span>
     </a>
    </li>
   </ul>
  </li>
 </ul>
</li>
<li class="toclevel-1 tocsection-15">
 <a href="#Legal_issues">
  <span class="tocnumber">
   3
  </span>
  <span class="toctext">
   Legal issues
  </span>
 </a>
 <ul>
  <li class="toclevel-2 tocsection-16">
   <a href="#United_States">
    <span class="tocnumber">
     3.1
    </span>
    <span class="toctext">
     United States
    </span>
   </a>
  </li>
  <li class="toclevel-2 tocsection-17">
   <a href="#The_EU">
    <span class="tocnumber">
     3.2
    </span>
    <span class="toctext">
     The EU
    </span>
   </a>
  </li>
  <li class="toclevel-2 tocsection-18">
   <a href="#Australia">
    <span class="tocnumber">
     3.3
    </span>
    <span class="toctext">
     Australia
    </span>
   </a>
  </li>
```

```
</ul>
</li>
<li class="toclevel-1 tocsection-19">
<a href="#Methods_to_prevent_web_scraping">
<span class="tocnumber">
4
</span>
<span class="toctext">
Methods to prevent web scraping
</span>
</a>
</li>
<li class="toclevel-1 tocsection-20">
<a href="#See_also">
<span class="tocnumber">
5
</span>
<span class="toctext">
See also
</span>
</a>
</li>
<li class="toclevel-1 tocsection-21">
<a href="#References">
<span class="tocnumber">
6
</span>
<span class="toctext">
References
</span>
</a>
</li>
</ul>
</div>
<p>
</p>
<h2>
<span class="mw-headline" id="Techniques">
Techniques
</span>
<span class="mw-editsection">
<span class="mw-editsection-bracket">
[
</span>
<a href="/w/index.php?title=Web_scraping&amp;action=edit&amp;section=1" title="Edit section: Techni
ques">
edit
</a>
<span class="mw-editsection-bracket">
]
</span>
</span>
</h2>
<p>
Web scraping is the process of automatically mining data or collecting information from the World Wide
Web. It is a field with active developments sharing a common goal with the
<a class="mw-redirect" href="/wiki/Semantic_web" title="Semantic web">
```

```
 semantic web
</a>
vision, an ambitious initiative that still requires breakthroughs in text processing, semantic understanding, a
rtificial intelligence and
<a class="mw-redirect" href="/wiki/Human-computer_interaction" title="Human-computer interaction">
 human-computer interactions
</a>
. Current web scraping solutions range from the ad-hoc, requiring human effort, to fully automated systems
that are able to convert entire web sites into structured information, with limitations.
</p>
<h3>
 <span class="mw-headline" id="Human_copy-and-paste">
  Human copy-and-paste
 </span>
 <span class="mw-editsection">
  <span class="mw-editsection-bracket">
   [
  </span>
  <a href="/w/index.php?title=Web_scraping&amp;action=edit&amp;section=2" title="Edit section: Human
 copy-and-paste">
   edit
  </a>
  <span class="mw-editsection-bracket">
   ]
  </span>
 </span>
</h3>
<p>
 Sometimes even the best web-scraping technology cannot replace a human’s manual examination and copy
-and-paste, and sometimes this may be the only workable solution when the websites for scraping explicitly set
up barriers to prevent machine automation.
</p>
<h3>
 <span class="mw-headline" id="Text_pattern_matching">
  Text pattern matching
 </span>
 <span class="mw-editsection">
  <span class="mw-editsection-bracket">
   [
  </span>
  <a href="/w/index.php?title=Web_scraping&amp;action=edit&amp;section=3" title="Edit section: Text pa
ttern matching">
   edit
  </a>
  <span class="mw-editsection-bracket">
   ]
  </span>
 </span>
</h3>
<p>
 A simple yet powerful approach to extract information from web pages can be based on the UNIX
 <a href="/wiki/Grep" title="Grep">
  grep
 </a>
 command or
 <a href="/wiki/Regular_expression" title="Regular expression">
  regular expression
```

```
</a>
-matching facilities of programming languages (for instance
<a href="/wiki/Perl" title="Perl">
 Perl
</a>
or
<a href="/wiki/Python_(programming_language)" title="Python (programming language)">
 Python
</a>
).
</p>
<h3>
 <span class="mw-headline" id="HTTP_programming">
  HTTP programming
 </span>
 <span class="mw-editsection">
  <span class="mw-editsection-bracket">
   [
  </span>
  <a href="/w/index.php?title=Web_scraping&amp;action=edit&amp;section=4" title="Edit section: HTTP
programming">
   edit
  </a>
  <span class="mw-editsection-bracket">
   ]
  </span>
 </span>
</h3>
<p>
 <a href="/wiki/Static_web_page" title="Static web page">
  Static
 </a>
 and
 <a href="/wiki/Dynamic_web_page" title="Dynamic web page">
  dynamic web pages
 </a>
 can be retrieved by posting HTTP requests to the remote web server using
 <a class="mw-redirect" href="/wiki/Socket_programming" title="Socket programming">
  socket programming
 </a>
 .
</p>
<h3>
 <span class="mw-headline" id="HTML_parsing">
  HTML parsing
 </span>
 <span class="mw-editsection">
  <span class="mw-editsection-bracket">
   [
  </span>
  <a href="/w/index.php?title=Web_scraping&amp;action=edit&amp;section=5" title="Edit section: HTML
parsing">
   edit
  </a>
  <span class="mw-editsection-bracket">
   ]
  </span>
```

```
 </span>
</h3>
<p>
 Many websites have large collections of pages generated dynamically from an underlying structured source like a database. Data of the same category are typically encoded into similar pages by a common script or template. In data mining, a program that detects such templates in a particular information source, extracts its content and translates it into a relational form, is called a
 <a href="/wiki/Wrapper_(data_mining)" title="Wrapper (data mining)">
  wrapper
 </a>
 . Wrapper generation algorithms assume that input pages of a wrapper induction system conform to a common template and that they can be easily identified in terms of a URL common scheme.
 <sup class="reference" id="cite_ref-3">
  <a href="#cite_note-3">
   [3]
  </a>
 </sup>
 Moreover, some
 <a href="/wiki/Semi-structured_data" title="Semi-structured data">
  semi-structured data
 </a>
 query languages, such as
 <a href="/wiki/XQuery" title="XQuery">
  XQuery
 </a>
 and the HTQL, can be used to parse HTML pages and to retrieve and transform page content.
</p>
<h3>
 <span class="mw-headline" id="DOM_parsing">
  DOM parsing
 </span>
 <span class="mw-editsection">
  <span class="mw-editsection-bracket">
   [
  </span>
  <a href="/w/index.php?title=Web_scraping&amp;action=edit&amp;section=6" title="Edit section: DOM parsing">
   edit
  </a>
  <span class="mw-editsection-bracket">
   ]
  </span>
 </span>
</h3>
<div class="hatnote navigation-not-searchable" role="note">
 Further information:
 <a href="/wiki/Document_Object_Model" title="Document Object Model">
  Document Object Model
 </a>
</div>
<p>
 By embedding a full-fledged web browser, such as the
 <a href="/wiki/Internet_Explorer" title="Internet Explorer">
  Internet Explorer
 </a>
 or the
 <a href="/wiki/Mozilla" title="Mozilla">
```

Mozilla

browser control, programs can retrieve the dynamic content generated by client-side scripts. These browser controls also parse web pages into a DOM tree, based on which programs can retrieve parts of the pages.

<h3>

Vertical aggregation

[

edit

]

</h3>

There are several companies that have developed vertical specific harvesting platforms. These platforms create and monitor a multitude of “bots” for specific verticals with no "man in the loop" (no direct human involvement), and no work related to a specific target site. The preparation involves establishing the knowledge base for the entire vertical and then the platform creates the bots automatically. The platform's robustness is measured by the quality of the information it retrieves (usually number of fields) and its scalability (how quick it can scale up to hundreds or thousands of sites). This scalability is mostly used to target the

Long Tail

of sites that common aggregators find complicated or too labor-intensive to harvest content from.

<h3>

Semantic annotation recognizing

[

edit

]

</h3>

The pages being scraped may embrace

metadata

or semantic markups and annotations, which can be used to locate specific data snippets. If the annotations

are embedded in the pages, as

Microformat

does, this technique can be viewed as a special case of DOM parsing. In another case, the annotations, organized into a semantic layer,
<sup class="reference" id="cite_ref-4">

[4]

</sup>
are stored and managed separately from the web pages, so the scrapers can retrieve data schema and instructions from this layer before scraping the pages.

<h3>

Computer vision web-page analysis

[

edit

]

</h3>

There are efforts using

machine learning

and

computer vision

that attempt to identify and extract information from web pages by interpreting pages visually as a human being might.
<sup class="reference" id="cite_ref-5">

[5]

</sup>

<h2>

Software

[

<a href="/w/index.php?title=Web_scraping&action=edit&section=10" title="Edit section: Softw

are">
edit

]

</h2>

There are many software tools available that can be used to customize web-scraping solutions. This softwar e may attempt to automatically recognize the data structure of a page or provide a recording interface that remo ves the necessity to manually write web-scraping code, or some scripting functions that can be used to extract a nd transform content, and database interfaces that can store the scraped data in local databases. Some web scrap ing software can also be used to extract data from an API directly.

<h3>

Example tools

[

edit

]

</h3>

cURL

– command line tool and library for transferring (including getting) data with URLs supporting a wide rang e of HTTP methods (GET, POST, cookies, etc.)

Data Toolbar

– web scraping add-on for Internet Explorer, Mozilla Firefox, and Google Chrome Web browsers that colle cts and converts structured data from web pages into a tabular format that can be loaded into a spreadsheet or da tabase management program.

Diffbot

– uses computer vision and machine learning to automatically extract data from web pages by interpreting pages visually as a human being might.

<a href="/wiki/Heritrix" title="Heritrix">

```
 Heritrix
</a>
– gets pages (lots of them). It is a web crawler designed for web archiving, written by the Internet Archive (see
<a href="/wiki/Wayback_Machine" title="Wayback Machine">
 Wayback Machine
</a>
).
</li>
<li>
 <a href="/wiki/HtmlUnit" title="HtmlUnit">
  HtmlUnit
 </a>
 – headless browser that can be used for retrieving web pages, web scraping, and more.
</li>
<li>
 <a href="/wiki/HTTrack" title="HTTrack">
  HTTrack
 </a>
 – free and open source Web crawler and offline browser, designed to download websites.
</li>
<li>
 <a href="/wiki/IMacros" title="IMacros">
  iMacros
 </a>
 –a browser extension to record, code, share and replay browser automation (javascript)
</li>
<li>
 Kantu – uses screenshots and OCR for scraping
</li>
<li>
 <a href="/wiki/Selenium_(software)" title="Selenium (software)">
  Selenium (software)
 </a>
 – a portable software-testing framework for web applications
</li>
<li>
 <a href="/wiki/Aptana#Aptana_Jaxer" title="Aptana">
  Jaxer
 </a>
</li>
<li>
 <a href="/wiki/Nokogiri_(software)" title="Nokogiri (software)">
  nokogiri
 </a>
</li>
<li>
 <a href="/wiki/OutWit_Hub" title="OutWit Hub">
  OutWit Hub
 </a>
 – Web scraping application including built-in data, image, document extractors and editors for custom scrapers and automatic exploration and extraction jobs (free and paid versions)
</li>
<li>
 <a href="/wiki/Watir" title="Watir">
  watir
 </a>
```

```
 –
</li>
<li>
 <a href="/wiki/Wget" title="Wget">
  Wget
 </a>
 – computer program that retrieves content from web servers. It is part of the GNU Project. It supports dow
nloading via the HTTP, HTTPS, and FTP protocols.
</li>
<li>
 <a href="/wiki/WSO2_Mashup_Server" title="WSO2 Mashup Server">
  WSO2 Mashup Server
 </a>
 –
</li>
<li>
 <a href="/wiki/Yahoo!_Query_Language" title="Yahoo! Query Language">
  Yahoo! Query Language
 </a>
 (YQL) –
</li>
<li>
 Data Scraping Studio – Stand alone windows desktop software to scrape data using CSS selectors and RE
GEX.
</li>
</ul>
<h4>
 <span class="mw-headline" id="Javascript_tools">
  Javascript tools
 </span>
 <span class="mw-editsection">
  <span class="mw-editsection-bracket">
   [
  </span>
  <a href="/w/index.php?title=Web_scraping&amp;amp;action=edit&amp;amp;section=12" title="Edit section: Javasc
ript tools">
   edit
  </a>
  <span class="mw-editsection-bracket">
   ]
  </span>
 </span>
</h4>
<ul>
 <li>
  <a href="/wiki/Greasemonkey" title="Greasemonkey">
   Greasemonkey
  </a>
 </li>
 <li>
  <a href="/wiki/Node.js" title="Node.js">
   Node.js
  </a>
 </li>
 <li>
  <a href="/wiki/PhantomJS" title="PhantomJS">
   PhantomJS
```

```
        </a>
        – scripted,
        <a href="/wiki/Headless_browser" title="Headless browser">
         headless browser
        </a>
        used for automating web page interaction.
       </li>
       <li>
        <a href="/wiki/JQuery" title="JQuery">
         jQuery
        </a>
       </li>
      </ul>
      <h4>
       <span class="mw-headline" id="SaaS_version">
        SaaS version
       </span>
       <span class="mw-editsection">
        <span class="mw-editsection-bracket">
         [
        </span>
        <a href="/w/index.php?title=Web_scraping&amp;action=edit&amp;section=13" title="Edit section: SaaS
version">
         edit
        </a>
        <span class="mw-editsection-bracket">
         ]
        </span>
       </span>
      </h4>
      <ul>
       <li>
        Agenty – SaaS solution, paid versions available from $29 (06/09/17)
       </li>
       <li>
        Apify – Web scraping and automation platform, free and paid versions available (10/10/17)
       </li>
       <li>
        dexi.io – SaaS solution, free and paid versions available from $119 USD (31/10/17)
       </li>
       <li>
        <a class="external text" href="https://www.diggernaut.com/" rel="nofollow">
         diggernaut.com
        </a>
        – Turn websites into datasets, free and paid (from $9.99 USD) subscriptions available (02/05/18)
       </li>
       <li>
        <a class="new" href="/w/index.php?title=FScraper&amp;action=edit&amp;redlink=1" title="FScraper (pa
ge does not exist)">
         fScraper
        </a>
        – Facebook friendly scraper, SaaS solution, free and paid versions available
       </li>
       <li>
        <a href="/wiki/Import.io" title="Import.io">
         Import.io
        </a>
```

```
 – SaaS solution, paid versions available from $299 USD (06/09/17)
</li>
<li>
 <a class="external text" href="https://listly.io/" rel="nofollow">
  Listly.io
 </a>
 – HTML to Excel in seconds, free SaaS service (04/10/17)
</li>
<li>
 <a href="/wiki/Mozenda" title="Mozenda">
  Mozenda
 </a>
 – SaaS solution, is a web-based platform for web data extraction (01/22/18)
</li>
<li>
 <a class="new" href="/w/index.php?title=UScraper&amp;action=edit&amp;redlink=1" title="UScraper (page does not exist)">
  uScraper
 </a>
 – SaaS service, free and paid versions available. Functionality primarily for scraping email addresses.
</li>
</ul>
<h4>
 <span class="mw-headline" id="Web_crawling_frameworks">
  Web crawling frameworks
 </span>
 <span class="mw-editsection">
  <span class="mw-editsection-bracket">
   [
  </span>
  <a href="/w/index.php?title=Web_scraping&amp;action=edit&amp;section=14" title="Edit section: Web crawling frameworks">
   edit
  </a>
  <span class="mw-editsection-bracket">
   ]
  </span>
 </span>
</h4>
<p>
 These can be used to build web scrapers.
</p>
<ul>
 <li>
  <a href="/wiki/Scrapy" title="Scrapy">
   Scrapy
  </a>
 </li>
</ul>
<h2>
 <span class="mw-headline" id="Legal_issues">
  Legal issues
 </span>
 <span class="mw-editsection">
  <span class="mw-editsection-bracket">
   [
  </span>
```

```
<a href="/w/index.php?title=Web_scraping&amp;action=edit&amp;section=15" title="Edit section: Legal issues">
 edit
</a>
<span class="mw-editsection-bracket">
 ]
</span>
</span>
</h2>
<table class="plainlinks metadata ambox ambox-content ambox-globalize" role="presentation">
<tr>
<td class="mbox-image">
<div style="width:52px">
<img alt="Globe icon." data-file-height="290" data-file-width="350" height="40" src="//upload.wikimedia.org/wikipedia/commons/thumb/b/bd/Ambox_globe_content.svg/48px-Ambox_globe_content.svg.png" srcset="//upload.wikimedia.org/wikipedia/commons/thumb/b/bd/Ambox_globe_content.svg/73px-Ambox_globe_content.svg.png 1.5x, //upload.wikimedia.org/wikipedia/commons/thumb/b/bd/Ambox_globe_content.svg/97px-Ambox_globe_content.svg.png 2x" width="48"/>
</div>
</td>
<td class="mbox-text">
<div class="mbox-text-span">
The examples and perspective in this article
<b>
deal primarily with the United States and do not represent a
<a href="/wiki/Wikipedia:WikiProject_Countering_systemic_bias" title="Wikipedia:WikiProject Countering systemic bias">
worldwide view
</a>
of the subject
</b>
.
<span class="hide-when-compact">
You may
<a class="external text" href="//en.wikipedia.org/w/index.php?title=Web_scraping&amp;action=edit">
improve this article
</a>
, discuss the issue on the
<a href="/wiki/Talk:Web_scraping" title="Talk:Web scraping">
talk page
</a>
, or
<a href="/wiki/Wikipedia:Article_wizard" title="Wikipedia:Article wizard">
create a new article
</a>
, as appropriate.
</span>
<small>
<i>
(October 2015)
</i>
</small>
<small class="hide-when-compact">
<i>
(
<a href="/wiki/Help:Maintenance_template_removal" title="Help:Maintenance template removal">
Learn how and when to remove this template message
```

```
      </a>
      )
     </i>
    </small>
   </div>
  </td>
 </tr>
</table>
<p>
 The legality of web scraping varies across the world. In general, web scraping may be against the
 <a class="mw-redirect" href="/wiki/Terms_of_use" title="Terms of use">
  terms of use
 </a>
 of some websites, but the enforceability of these terms is unclear.
 <sup class="reference" id="cite_ref-6">
  <a href="#cite_note-6">
   [6]
  </a>
 </sup>
</p>
<h3>
 <span class="mw-headline" id="United_States">
  United States
 </span>
 <span class="mw-editsection">
  <span class="mw-editsection-bracket">
   [
  </span>
  <a href="/w/index.php?title=Web_scraping&amp;action=edit&amp;section=16" title="Edit section: Unite
d States">
   edit
  </a>
  <span class="mw-editsection-bracket">
   ]
  </span>
 </span>
</h3>
<p>
 In the United States, website owners can use three major
 <a href="/wiki/Cause_of_action" title="Cause of action">
  legal claims
 </a>
 to prevent undesired web scraping: (1) copyright infringement (compilation), (2) violation of the
 <a href="/wiki/Computer_Fraud_and_Abuse_Act" title="Computer Fraud and Abuse Act">
  Computer Fraud and Abuse Act
 </a>
 (“CFAA”), and (3)
 <a href="/wiki/Trespass_to_chattels" title="Trespass to chattels">
  trespass to chattel
 </a>
 .
 <sup class="reference" id="cite_ref-7">
  <a href="#cite_note-7">
   [7]
  </a>
 </sup>
 However, the effectiveness of these claims relies upon meeting various criteria, and the case law is still evol
```

ving. For example, with regard to copyright, while outright duplication of original expression will in many case s be illegal, in the United States the courts ruled in

Feist Publications v. Rural Telephone Service

that duplication of facts is allowable.

U.S. courts have acknowledged that users of "scrapers" or "robots" may be held liable for committing

trespass to chattels

,
<sup class="reference" id="cite_ref-8">

[8]

</sup>
<sup class="reference" id="cite_ref-9">

[9]

</sup>

which involves a computer system itself being considered personal property upon which the user of a scrap er is trespassing. The best known of these cases,

eBay v. Bidder's Edge

, resulted in an injunction ordering Bidder's Edge to stop accessing, collecting, and indexing auctions from t he eBay web site. This case involved automatic placing of bids, known as

auction sniping

. However, in order to succeed on a claim of trespass to

chattels

, the

plaintiff

must demonstrate that the

defendant

intentionally and without authorization interfered with the plaintiff's possessory interest in the computer sys tem and that the defendant's unauthorized use caused damage to the plaintiff. Not all cases of web spidering bro ught before the courts have been considered trespass to chattels.

<sup class="reference" id="cite_ref-10">

[10]


```
</sup>
</p>
<p>
One of the first major tests of
<a class="mw-redirect" href="/wiki/Screen_scraping" title="Screen scraping">
 screen scraping
</a>
involved
<a href="/wiki/American_Airlines" title="American Airlines">
 American Airlines
</a>
(AA), and a firm called FareChase.
<sup class="reference" id="cite_ref-11">
 <a href="#cite_note-11">
  [11]
 </a>
</sup>
AA successfully obtained an
<a href="/wiki/Injunction" title="Injunction">
 injunction
</a>
```

from a Texas trial court, stopping FareChase from selling software that enables users to compare online fares if the software also searches AA's website. The airline argued that FareChase's websearch software trespassed on AA's servers when it collected the publicly available data. FareChase filed an appeal in March 2003. By June, FareChase and AA agreed to settle and the appeal was dropped.

```
<sup class="reference" id="cite_ref-12">
 <a href="#cite_note-12">
  [12]
 </a>
</sup>
</p>
<p>
<a href="/wiki/Southwest_Airlines" title="Southwest Airlines">
 Southwest Airlines
</a>
```

has also challenged screen-scraping practices, and has involved both FareChase and another firm, Outtask, in a legal claim. Southwest Airlines charged that the screen-scraping is Illegal since it is an example of "Computer Fraud and Abuse" and has led to "Damage and Loss" and "Unauthorized Access" of Southwest's site. It also constitutes "Interference with Business Relations", "Trespass", and "Harmful Access by Computer". They also claimed that screen-scraping constitutes what is legally known as "Misappropriation and Unjust Enrichment", as well as being a breach of the web site's user agreement. Outtask denied all these claims, claiming that the prevailing law in this case should be

```
<a class="mw-redirect" href="/wiki/US_Copyright_law" title="US Copyright law">
 US Copyright law
</a>
```

, and that under copyright, the pieces of information being scraped would not be subject to copyright protection. Although the cases were never resolved in the

```
<a href="/wiki/Supreme_Court_of_the_United_States" title="Supreme Court of the United States">
 Supreme Court of the United States
</a>
```

, FareChase was eventually shuttered by parent company

```
<a href="/wiki/Yahoo!" title="Yahoo!">
 Yahoo!
</a>
```

, and Outtask was purchased by travel expense company Concur.

```
<sup class="reference" id="cite_ref-impervawp2011_13-0">
 <a href="#cite_note-impervawp2011-13">
```

[13]

</sup>
In 2012, a startup called 3Taps scraped classified housing ads from Craigslist. Craigslist sent 3Taps a cease-and-desist letter and blocked their IP addresses and later sued, in

Craigslist v. 3Taps

. The court held that the cease-and-desist letter and IP blocking was sufficient for Craigslist to properly claim that 3Taps had violated the

Computer Fraud and Abuse Act

.

Although these are early scraping decisions, and the theories of liability are not uniform, it is difficult to ignore a pattern emerging that the courts are prepared to protect proprietary content on commercial sites from uses which are undesirable to the owners of such sites. However, the degree of protection for such content is not settled, and will depend on the type of access made by the scraper, the amount of information accessed and copied, the degree to which the access adversely affects the site owner's system and the types and manner of prohibitions on such conduct.
<sup class="reference" id="cite_ref-14">

[14]

</sup>

While the law in this area becomes more settled, entities contemplating using scraping programs to access a public web site should also consider whether such action is authorized by reviewing the terms of use and other terms or notices posted on or made available through the site. In a 2010 ruling in the

Cvent, Inc.

v.

Eventbrite, Inc.

In the United States district court for the eastern district of Virginia, the court ruled that the terms of use should be brought to the users' attention In order for a

browse wrap

contract or license to be enforced.
<sup class="reference" id="cite_ref-15">

[15]

</sup>
In a 2014, filed in the

United States District Court for the Eastern District of Pennsylvania

,
<sup class="reference" id="cite_ref-16">

[16]

</sup>
e-commerce site

QVC

objected to the Pinterest-like shopping aggregator Resultly's `scraping of QVC's site for real-time pricing d ata. QVC alleges that Resultly "excessively crawled" QVC's retail site (allegedly sending 200-300 search reque sts to QVC's website per minute, sometimes to up to 36,000 requests per minute) which caused QVC's site to cr ash for two days, resulting in lost sales for QVC.
<sup class="reference" id="cite_ref-17">

[17]

</sup>
QVC's complaint alleges that the defendant disguised its web crawler to mask its source IP address and thus prevented QVC from quickly repairing the problem. This is a particularly interesting scraping case because QV C is seeking damages for the unavailability of their website, which QVC claims was caused by Resultly.

In the plaintiff's web site during the period of this trial the terms of use link is displayed among all the links of the site, at the bottom of the page as most sites on the internet. This ruling contradicts the Irish ruling describ ed below. The court also rejected the plaintiff's argument that the browse wrap restrictions were enforceable in v iew of Virginia's adoption of the Uniform Computer Information Transactions Act (UCITA)—a uniform law th at many believed was in favor on common browse wrap contracting practices.
<sup class="reference" id="cite_ref-18">

[18]

</sup>

In

Facebook, Inc. v. Power Ventures, Inc.

, a district court ruled in 2012 that Power Ventures could not scrape Facebook pages on behalf of a Faceboo k user. The case is on appeal, and the

Electronic Frontier Foundation

filed a brief in 2015 asking that it be overturned.
<sup class="reference" id="cite_ref-19">

[19]

</sup>
<sup class="reference" id="cite_ref-20">
<a href="#cite_note-20">

```
 [20]
 </a>
</sup>
In
<i>
 <a href="/wiki/Associated_Press_v._Meltwater_U.S._Holdings,_Inc." title="Associated Press v. Meltwate
r U.S. Holdings, Inc.">
  Associated Press v. Meltwater U.S. Holdings, Inc.
 </a>
</i>
, a court in the US held Meltwater liable for scraping and republishing news information from the Associate
d Press, but a court in the United Kingdom held in favor of Meltwater.
</p>
<h3>
 <span class="mw-headline" id="The_EU">
  The EU
 </span>
 <span class="mw-editsection">
  <span class="mw-editsection-bracket">
   [
  </span>
  <a href="/w/index.php?title=Web_scraping&amp;action=edit&amp;section=17" title="Edit section: The E
U">
   edit
  </a>
  <span class="mw-editsection-bracket">
   ]
  </span>
 </span>
</h3>
<p>
 Outside of the United States, in February 2006, the Danish Maritime and Commercial Court (Copenhagen)
ruled that systematic crawling, indexing, and deep linking by portal site ofir.dk of estate site Home.dk does not
conflict with Danish law or the database directive of the European Union.
 <sup class="reference" id="cite_ref-21">
  <a href="#cite_note-21">
   [21]
  </a>
 </sup>
</p>
<p>
 In a February 2010 case complicated by matters of jurisdiction, Ireland's High Court delivered a verdict that
 illustrates the
 <a class="new" href="/w/index.php?title=Inchoate&amp;action=edit&amp;redlink=1" title="Inchoate (pag
e does not exist)">
  inchoate
 </a>
 state of developing case law. In the case of
 <i>
  Ryanair Ltd v Billigfluege.de GmbH
 </i>
 , Ireland's High Court ruled Ryanair's "click-wrap" agreement to be legally binding. In contrast to the findin
gs of the United States District Court Eastern District of Virginia and those of the Danish Maritime and Comme
rcial Court, Mr. Justice Michael Hanna ruled that the hyperlink to Ryanair's terms and conditions was plainly vi
sible, and that placing the onus on the user to agree to terms and conditions in order to gain access to online serv
ices is sufficient to comprise a contractual relationship.
 <sup class="reference" id="cite_ref-22">
```

```
<a href="#cite_note-22">
 [22]
 </a>
</sup>
The decision is under appeal in Ireland's Supreme Court.
<sup class="reference" id="cite_ref-23">
 <a href="#cite_note-23">
  [23]
 </a>
</sup>
</p>
<h3>
 <span class="mw-headline" id="Australia">
  Australia
 </span>
 <span class="mw-editsection">
  <span class="mw-editsection-bracket">
   [
  </span>
  <a href="/w/index.php?title=Web_scraping&amp;action=edit&amp;section=18" title="Edit section: Austr
alia">
   edit
  </a>
  <span class="mw-editsection-bracket">
   ]
  </span>
 </span>
</h3>
<p>
 In Australia, the
 <a href="/wiki/Spam_Act_2003" title="Spam Act 2003">
  Spam Act 2003
 </a>
 outlaws some forms of web harvesting, although this only applies to email addresses.
 <sup class="reference" id="cite_ref-24">
  <a href="#cite_note-24">
   [24]
  </a>
 </sup>
 <sup class="reference" id="cite_ref-25">
  <a href="#cite_note-25">
   [25]
  </a>
 </sup>
</p>
<h2>
 <span class="mw-headline" id="Methods_to_prevent_web_scraping">
  Methods to prevent web scraping
 </span>
 <span class="mw-editsection">
  <span class="mw-editsection-bracket">
   [
  </span>
  <a href="/w/index.php?title=Web_scraping&amp;action=edit&amp;section=19" title="Edit section: Metho
ds to prevent web scraping">
   edit
  </a>
```

```
 <span class="mw-editsection-bracket">
  ]
 </span>
</span>
</h2>
<p>
 The administrator of a website can use various measures to stop or slow a bot. Some techniques include:
</p>
<ul>
 <li>
  Blocking an
  <a href="/wiki/IP_address" title="IP address">
   IP address
  </a>
  either manually or based on criteria such as
  <a href="/wiki/Geolocation" title="Geolocation">
   geolocation
  </a>
  and
  <a href="/wiki/DNSBL" title="DNSBL">
   DNSRBL
  </a>
  . This will also block all browsing from that address.
 </li>
 <li>
  Disabling any
  <a href="/wiki/Web_service" title="Web service">
   web service
  </a>
  <a href="/wiki/Application_programming_interface" title="Application programming interface">
   API
  </a>
  that the website's system might expose.
 </li>
 <li>
  Bots sometimes declare who they are (using
  <a href="/wiki/User_agent" title="User agent">
   user agent
  </a>
  <a href="/wiki/String_(computer_science)" title="String (computer science)">
   strings
  </a>
  ) and can be blocked on that basis using
  <a href="/wiki/Robots_exclusion_standard" title="Robots exclusion standard">
   robots.txt
  </a>
  ; '
  <a href="/wiki/Googlebot" title="Googlebot">
   googlebot
  </a>
  ' is an example. Other bots make no distinction between themselves and a human using a browser.
 </li>
 <li>
  Bots can be blocked by monitoring excess traffic
 </li>
 <li>
  Bots can sometimes be blocked with tools to verify that it is a real person accessing the site, like a
```


CAPTCHA

. Bots are sometimes coded to explicitly break specific CAPTCHA patterns or may employ third-party services that utilize human labor to read and respond in real-time to CAPTCHA challenges.

Commercial anti-bot services: Companies offer anti-bot and anti-scraping services for websites. A few web

application firewalls

have limited bot detection capabilities as well. However, many such solutions are not very effective
<sup class="reference" id="cite_ref-26">

[26]

</sup>
.

Locating bots with a

honeypot

or other method to identify the IP addresses of automated crawlers.

Obfuscation

using

CSS sprites

to display such data as phone numbers or email addresses, at the cost of

accessibility

to

screen reader

users.

Because bots rely on consistency in the front-end code of a target website, adding small variations to the HTML/CSS surrounding important data and navigation elements would require more human involvement in the initial set up of a bot and if done effectively may render the target website too difficult to scrape due to the diminished ability to automate the scraping process.

Websites can declare if crawling is allowed or not in the
<a class="external text" href="https://www.promptcloud.com/blog/how-to-read-and-respect-robots-file" rel="nofollow">
robots.txt

```
        </a>
        file and allow partial access, limit the crawl rate, specify the optimal time to crawl and more.
       </li>
      </ul>
      <h2>
       <span class="mw-headline" id="See_also">
        See also
       </span>
       <span class="mw-editsection">
        <span class="mw-editsection-bracket">
         [
        </span>
        <a href="/w/index.php?title=Web_scraping&amp;action=edit&amp;section=20" title="Edit section: See al
so">
         edit
        </a>
        <span class="mw-editsection-bracket">
         ]
        </span>
       </span>
      </h2>
      <div class="div-col columns column-count column-count-3" style="-moz-column-count: 3; -webkit-column
-count: 3; column-count: 3;">
       <ul>
        <li>
         <a href="/wiki/Archive.is" title="Archive.is">
          Archive.is
         </a>
        </li>
        <li>
         <a href="/wiki/Comparison_of_feed_aggregators" title="Comparison of feed aggregators">
          Comparison of feed aggregators
         </a>
        </li>
        <li>
         <a href="/wiki/Data_scraping" title="Data scraping">
          Data scraping
         </a>
        </li>
        <li>
         <a href="/wiki/Data_wrangling" title="Data wrangling">
          Data wrangling
         </a>
        </li>
        <li>
         <a href="/wiki/Importer_(computing)" title="Importer (computing)">
          Importer
         </a>
        </li>
        <li>
         <a href="/wiki/Job_wrapping" title="Job wrapping">
          Job wrapping
         </a>
        </li>
        <li>
         <a href="/wiki/Knowledge_extraction" title="Knowledge extraction">
          Knowledge extraction
```

```
 </a>
</li>
<li>
 <a href="/wiki/OpenSocial" title="OpenSocial">
  OpenSocial
 </a>
</li>
<li>
 <a href="/wiki/Scraper_site" title="Scraper site">
  Scraper site
 </a>
</li>
<li>
 <a href="/wiki/Fake_news_website" title="Fake news website">
  Fake news website
 </a>
</li>
<li>
 <a href="/wiki/Blog_scraping" title="Blog scraping">
  Blog scraping
 </a>
</li>
<li>
 <a href="/wiki/Spamdexing" title="Spamdexing">
  Spamdexing
 </a>
</li>
<li>
 <a href="/wiki/Domain_name_drop_list" title="Domain name drop list">
  Domain name drop list
 </a>
</li>
<li>
 <a href="/wiki/Text_corpus" title="Text corpus">
  Text corpus
 </a>
</li>
<li>
 <a href="/wiki/Web_archiving" title="Web archiving">
  Web archiving
 </a>
</li>
<li>
 <a class="mw-redirect" href="/wiki/Blog_network" title="Blog network">
  Blog network
 </a>
</li>
<li>
 <a class="mw-redirect" href="/wiki/Search_Engine_Scraping" title="Search Engine Scraping">
  Search Engine Scraping
 </a>
</li>
<li>
 <a href="/wiki/Category:Web_crawlers" title="Category:Web crawlers">
  Web crawlers
 </a>
</li>
```

```
      </ul>
     </div>
     <h2>
      <span class="mw-headline" id="References">
       References
      </span>
      <span class="mw-editsection">
       <span class="mw-editsection-bracket">
        [
       </span>
       <a href="/w/index.php?title=Web_scraping&amp;action=edit&amp;section=21" title="Edit section: Refer
ences">
        edit
       </a>
       <span class="mw-editsection-bracket">
        ]
       </span>
      </span>
     </h2>
     <div class="reflist columns references-column-width" style="-moz-column-width: 20em; -webkit-column-
width: 20em; column-width: 20em; list-style-type: decimal;">
      <ol class="references">
       <li id="cite_note-Boeing2016JPER-1">
        <span class="mw-cite-backlink">
         ^
         <a href="#cite_ref-Boeing2016JPER_1-0">
          <sup>
           <i>
            <b>
             a
            </b>
           </i>
          </sup>
         </a>
         <a href="#cite_ref-Boeing2016JPER_1-1">
          <sup>
           <i>
            <b>
             b
            </b>
           </i>
          </sup>
         </a>
        </span>
        <span class="reference-text">
         <cite class="citation journal">
          Boeing, G.; Waddell, P. (2016). "New Insights into Rental Housing Markets across the United States: W
eb Scraping and Analyzing Craigslist Rental Listings".
          <i>
           Journal of Planning Education and Research
          </i>
          (0739456X16664789).
          <a href="/wiki/Digital_object_identifier" title="Digital object identifier">
           doi
          </a>
          :
          <a class="external text" href="//doi.org/10.1177%2F0739456X16664789" rel="nofollow">
```

10.1177/0739456X16664789

.
</cite>
<span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&rft.genre=article&rft.jtitle=Journal+of+Planning+Education+and+Research&rft.atitle=New+Insights+into+Rental+Housing+Markets+across+the+United+States%3A+Web+Scraping+and+Analyzing+Craigslist+Rental+Listings&rft.issue=0739456X16664789&rft.date=2016&rft_id=info%3Adoi%2F10.1177%2F0739456X16664789&rft.aulast=Boeing&rft.aufirst=G.&rft.au=Waddell%2C+P.&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">

<li id="cite_note-2">

^

<cite class="citation journal">
Vargiu & Urru (2013). "Exploiting web scraping in a collaborative filtering- based approach to web advertising".

Artificial Intelligence Research

.

2

(1).

doi

:

10.5430/air.v2n1p44

.
</cite>
<span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&rft.genre=article&rft.jtitle=Artificial+Intelligence+Research&rft.atitle=Exploiting+web+scraping+in+a+collaborative+filtering-+based+approach+to+web+advertising&rft.volume=2&rft.issue=1&rft.date=2013&rft_id=info%3Adoi%2F10.5430%2Fair.v2n1p44&rft.au=Vargiu+%26+Urru&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">

<li id="cite_note-3">

```
<a href="#cite_ref-3">
^
</a>
</b>
</span>
<span class="reference-text">
<cite class="citation journal">
Song, Ruihua; Microsoft Research (Sep 14, 2007).
<a class="external text" href="https://pdfs.semanticscholar.org/4fb4/3c5a212df751e84c3b2f8d29fabfe56c3616.pdf" rel="nofollow">
"Joint Optimization of Wrapper Generation and Template Detection"
</a>
<span style="font-size:85%;">
(PDF)
</span>
.
<i>
The 13th International Conference on Knowledge Discovery and Data Mining
</i>
.
</cite>
<span class="Z3988" title="ctx_ver=Z39.88-2004&amp;rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&amp;rft.genre=article&amp;rft.jtitle=The+13th+International+Conference+on+Knowledge+Discovery+and+Data+Mining&amp;rft.atitle=Joint+Optimization+of+Wrapper+Generation+and+Template+Detection&amp;rft.date=2007-09-14&amp;rft.aulast=Song&amp;rft.aufirst=Ruihua&amp;rft.au=Microsoft+Research&amp;rft_id=https%3A%2F%2Fpdfs.semanticscholar.org%2F4fb4%2F3c5a212df751e84c3b2f8d29fabfe56c3616.pdf&amp;rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<span style="display:none;">
</span>
</span>
</span>
</li>
<li id="cite_note-4">
<span class="mw-cite-backlink">
<b>
<a href="#cite_ref-4">
^
</a>
</b>
</span>
<span class="reference-text">
<a class="external text" href="http://www.gooseeker.com/en/node/knowledgebase/freeformat" rel="nofollow">
Semantic annotation based web scraping
</a>
</span>
</li>
<li id="cite_note-5">
<span class="mw-cite-backlink">
<b>
<a href="#cite_ref-5">
^
</a>
</b>
</span>
<span class="reference-text">
<cite class="citation web">
```

Roush, Wade (2012-07-25).

"Diffbot Is Using Computer Vision to Reinvent the Semantic Web"

. www.xconomy.com

. Retrieved

2013-03-15

.
</cite>
<span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=Diffbot+Is+Using+Computer+Vision+to+Reinvent+the+Semantic+Web&rft.pub=www.xconomy.com&rft.date=2012-07-25&rft.aulast=Roush&rft.aufirst=Wade&rft_id=http%3A%2F%2Fwww.xconomy.com%2Fsan-francisco%2F2012%2F07%2F25%2Fdiffbot-is-using-computer-vision-to-reinvent-the-semantic-web%2F&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">

<li id="cite_note-6">

^

<cite class="citation web">

"FAQ about linking – Are website terms of use binding contracts?"

. www.chillingeffects.org. 2007-08-20. Archived from

the original

on 2002-03-08

. Retrieved

2007-08-20

.
</cite>
<span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=FAQ+about+linking+%E2%80%93+Are+website+terms+of+use+binding+contracts%3F&rft.pub=www.chillingeffects.org&rft.date=2007-08-20&rft_id=http%3A%2F%2Fwww.chillingeffects.org%2Flinking%2Ffaq.cgi%23QID596&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">

```
 <span style="display:none;">
 </span>
 </span>
 </span>
</li>
<li id="cite_note-7">
 <span class="mw-cite-backlink">
 <b>
 <a href="#cite_ref-7">
 ^
 </a>
 </b>
 </span>
 <span class="reference-text">
 <cite class="citation journal">
 Kenneth, Hirschey, Jeffrey (2014-01-01).
 <a class="external text" href="http://scholarship.law.berkeley.edu/btlj/vol29/iss4/16/" rel="nofollow">
 "Symbiotic Relationships: Pragmatic Acceptance of Data Scraping"
 </a>
 .
 <i>
 Berkeley Technology Law Journal
 </i>
 .
 <b>
 29
 </b>
 (4).
 <a href="/wiki/Digital_object_identifier" title="Digital object identifier">
 doi
 </a>
 :
 <a class="external text" href="//doi.org/10.15779%2FZ38B39B" rel="nofollow">
 10.15779/Z38B39B
 </a>
 .
 <a href="/wiki/International_Standard_Serial_Number" title="International Standard Serial Number">
 ISSN
 </a>
 <a class="external text" href="//www.worldcat.org/issn/1086-3818" rel="nofollow">
 1086-3818
 </a>
 .
 </cite>
 <span class="Z3988" title="ctx_ver=Z39.88-2004&amp;rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&amp;rft.genre=article&amp;rft.jtitle=Berkeley+Technology+Law+Journal&amp;rft.atitle=Symbiotic+Relationships%3A+Pragmatic+Acceptance+of+Data+Scraping&amp;rft.volume=29&amp;rft.issue=4&amp;rft.date=2014-01-01&amp;rft_id=info%3Adoi%2F10.15779%2FZ38B39B&amp;rft.issn=1086-3818&amp;rft.aulast=Kenneth&amp;rft.aufirst=Hirschey%2C+Jeffrey&amp;rft_id=http%3A%2F%2Fscholarship.law.berkeley.edu%2Fbtlj%2Fvol29%2Fiss4%2F16%2F&amp;rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
 <span style="display:none;">
 </span>
 </span>
 </span>
</li>
<li id="cite_note-8">
```

```
<span class="mw-cite-backlink">
 <b>
  <a href="#cite_ref-8">
   ^
  </a>
 </b>
</span>
<span class="reference-text">
 <cite class="citation web">
  <a class="external text" href="http://www.tomwbell.com/NetLaw/Ch06.html" rel="nofollow">
   "Internet Law, Ch. 06: Trespass to Chattels"
  </a>
  . www.tomwbell.com. 2007-08-20
  <span class="reference-accessdate">
   . Retrieved
   <span class="nowrap">
    2007-08-20
   </span>
  </span>
  .
 </cite>
 <span class="Z3988" title="ctx_ver=Z39.88-2004&amp;rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&amp;rft.genre=unknown&amp;rft.btitle=Internet+Law%2C+Ch.+06%3A+Trespass+to+Chattels&amp;rft.pub=www.tomwbell.com&amp;rft.date=2007-08-20&amp;rft_id=http%3A%2F%2Fwww.tomwbell.com%2FNetLaw%2FCh06.html&amp;rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
  <span style="display:none;">
  </span>
 </span>
</span>
</li>
<li id="cite_note-9">
 <span class="mw-cite-backlink">
  <b>
   <a href="#cite_ref-9">
    ^
   </a>
  </b>
 </span>
 <span class="reference-text">
  <cite class="citation web">
   <a class="external text" href="https://web.archive.org/web/20020308222536/http://www.chillingeffects.org/linking/faq.cgi#QID460" rel="nofollow">
    "What are the "trespass to chattels" claims some companies or website owners have brought?"
   </a>
   . www.chillingeffects.org. 2007-08-20. Archived from
   <a class="external text" href="http://www.chillingeffects.org/linking/faq.cgi#QID460" rel="nofollow">
    the original
   </a>
   on 2002-03-08
   <span class="reference-accessdate">
    . Retrieved
    <span class="nowrap">
     2007-08-20
    </span>
   </span>
   .
  </cite>
```

```
        <span class="Z3988" title="ctx_ver=Z39.88-2004&amp;rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&amp;rft.genre=unknown&amp;rft.btitle=What+are+the+%22trespass+to+chattels%22+claims+some+companies+or+website+owners+have+brought%3F&amp;rft.pub=www.chillingeffects.org&amp;rft.date=2007-08-20&amp;rft_id=http%3A%2F%2Fwww.chillingeffects.org%2Flinking%2Ffaq.cgi%23QID460&amp;rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
         <span style="display:none;">
         </span>
        </span>
       </span>
      </li>
      <li id="cite_note-10">
       <span class="mw-cite-backlink">
        <b>
         <a href="#cite_ref-10">
          ^
         </a>
        </b>
       </span>
       <span class="reference-text">
        <cite class="citation web">
         <a class="external text" href="http://www.tomwbell.com/NetLaw/Ch07/Ticketmaster.html" rel="nofollow">
          "Ticketmaster Corp. v. Tickets.com, Inc."
         </a>
         2007-08-20
         <span class="reference-accessdate">
          . Retrieved
          <span class="nowrap">
           2007-08-20
          </span>
         </span>
         .
        </cite>
        <span class="Z3988" title="ctx_ver=Z39.88-2004&amp;rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&amp;rft.genre=unknown&amp;rft.btitle=Ticketmaster+Corp.+v.+Tickets.com%2C+Inc.&amp;rft.date=2007-08-20&amp;rft_id=http%3A%2F%2Fwww.tomwbell.com%2FNetLaw%2FCh07%2FTicketmaster.html&amp;rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
         <span style="display:none;">
         </span>
        </span>
       </span>
      </li>
      <li id="cite_note-11">
       <span class="mw-cite-backlink">
        <b>
         <a href="#cite_ref-11">
          ^
         </a>
        </b>
       </span>
       <span class="reference-text">
        <cite class="citation web">
         <a class="external text" href="https://web.archive.org/web/20110723131832/http://www.fornova.net/documents/AAFareChase.pdf" rel="nofollow">
          "American Airlines v. FareChase"
         </a>
         <span style="font-size:85%;">
```

(PDF)

. 2007-08-20. Archived from

the original

(PDF)

on 2011-07-23

. Retrieved

2007-08-20

.
</cite>
<span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=American+Airlines+v.+FareChase&rft.date=2007-08-20&rft_id=http%3A%2F%2Fwww.fornova.net%2Fdocuments%2FAAFareChase.pdf&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">

<li id="cite_note-12">

^

<cite class="citation web">

"American Airlines, FareChase Settle Suit"

. The Free Library. 2003-06-13

. Retrieved

2012-02-26

.
</cite>
<span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=American+Airlines%2C+FareChase+Settle+Suit.&rft.pub=The+Free+Library&rft.date=2003-06-13&rft_id=http%3A%2F%2Fwww.thefreelibrary.com%2FAmerican%2BAirlines%2C%2BFareChase%2BSettle%2BSuit.-a0103213546&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">

```
        </span>
       </span>
      </li>
      <li id="cite_note-impervawp2011-13">
       <span class="mw-cite-backlink">
        <b>
         <a href="#cite_ref-impervawp2011_13-0">
          ^
         </a>
        </b>
       </span>
       <span class="reference-text">
        Imperva (2011).
        <a class="external text" href="http://www.imperva.com/docs/WP_Detecting_and_Blocking_Site_Scraping_Attacks.pdf" rel="nofollow">
         Detecting and Blocking Site Scraping Attacks
        </a>
        . Imperva white paper..
       </span>
      </li>
      <li id="cite_note-14">
       <span class="mw-cite-backlink">
        <b>
         <a href="#cite_ref-14">
          ^
         </a>
        </b>
       </span>
       <span class="reference-text">
        <cite class="citation web">
         Adler, Kenneth A. (2003-07-29).
         <a class="external text" href="http://library.findlaw.com/2003/Jul/29/132944.html" rel="nofollow">
          "Controversy Surrounds 'Screen Scrapers': Software Helps Users Access Web Sites But Activity by Competitors Comes Under Scrutiny"
         </a>
         <span class="reference-accessdate">
          . Retrieved
          <span class="nowrap">
           2010-10-27
          </span>
         </span>
         .
        </cite>
        <span class="Z3988" title="ctx_ver=Z39.88-2004&amp;rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&amp;rft.genre=unknown&amp;rft.btitle=Controversy+Surrounds+%27Screen+Scrapers%27%3A+Software+Helps+Users+Access+Web+Sites+But+Activity+by+Competitors+Comes+Under+Scrutiny&amp;rft.date=2003-07-29&amp;rft.aulast=Adler&amp;rft.aufirst=Kenneth+A.&amp;rft_id=http%3A%2F%2Flibrary.findlaw.com%2F2003%2FJul%2F29%2F132944.html&amp;rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
         <span style="display:none;">
         </span>
        </span>
       </span>
      </li>
      <li id="cite_note-15">
       <span class="mw-cite-backlink">
        <b>
```


^

<cite class="citation web">

"QVC Inc. v. Resultly LLC, No. 14-06714 (E.D. Pa. filed Nov. 24, 2014)"

(PDF)

. 2014-11-24

. Retrieved

2015-11-05

.
</cite>
<span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=QVC+Inc.+v.+Resultly+LLC%2C+No.+14-06714+%28E.D.+Pa.+filed+Nov.+24%2C+2014%29&rft.date=2014-11-24&rft_id=http%3A%2F%2Fwww.fornova.net%2Fdocuments%2FCvent.pdf&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">

<li id="cite_note-16">

^

<cite class="citation web">

"QVC Inc. v. Resultly LLC, No. 14-06714 (E.D. Pa. filed Nov. 24, 2014)"

.

United States District Court for the Eastern District of Pennsylvania

. Retrieved

5 November

2015

.

```
        </cite>
        <span class="Z3988" title="ctx_ver=Z39.88-2004&amp;rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&amp;rft.genre=unknown&amp;rft.jtitle=United+States+District+Court+for+the+Eastern+District+of+Pennsylvania&amp;rft.atitle=QVC+Inc.+v.+Resultly+LLC%2C+No.+14-06714+%28E.D.+Pa.+filed+Nov.+24%2C+2014%29&amp;rft_id=https%3A%2F%2Fwww.scribd.com%2Fdoc%2F249068700%2FLinkedIn-v-Resultly-LLC-Complaint%3Fsecret_password%3DpEVKDbnvhQL52oKfdrmT&amp;rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
         <span style="display:none;">
         </span>
        </span>
       </span>
      </li>
      <li id="cite_note-17">
       <span class="mw-cite-backlink">
        <b>
         <a href="#cite_ref-17">
          ^
         </a>
        </b>
       </span>
       <span class="reference-text">
        <cite class="citation journal">
         Neuburger, Jeffrey D (5 December 2014).
         <a class="external text" href="http://newmedialaw.proskauer.com/2014/12/05/qvc-sues-shopping-app-for-web-scraping-that-allegedly-triggered-site-outage/" rel="nofollow">
          "QVC Sues Shopping App for Web Scraping That Allegedly Triggered Site Outage"
         </a>
         .
         <i>
          The National Law Review
         </i>
         . Proskauer Rose LLP
         <span class="reference-accessdate">
          . Retrieved
          <span class="nowrap">
           5 November
          </span>
          2015
         </span>
         .
        </cite>
        <span class="Z3988" title="ctx_ver=Z39.88-2004&amp;rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&amp;rft.genre=article&amp;rft.jtitle=The+National+Law+Review&amp;rft.atitle=QVC+Sues+Shopping+App+for+Web+Scraping+That+Allegedly+Triggered+Site+Outage&amp;rft.date=2014-12-05&amp;rft.aulast=Neuburger&amp;rft.aufirst=Jeffrey+D&amp;rft_id=http%3A%2F%2Fnewmedialaw.proskauer.com%2F2014%2F12%2F05%2Fqvc-sues-shopping-app-for-web-scraping-that-allegedly-triggered-site-outage%2F&amp;rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
         <span style="display:none;">
         </span>
        </span>
       </span>
      </li>
      <li id="cite_note-18">
       <span class="mw-cite-backlink">
        <b>
         <a href="#cite_ref-18">
          ^
```

```
</a>
</b>
</span>
<span class="reference-text">
<cite class="citation web">
<a class="external text" href="http://www.fornova.net/documents/pblog-bna-com.pdf" rel="nofollow">
"Did Iqbal/Twombly Raise the Bar for Browsewrap Claims?"
</a>
<span style="font-size:85%;">
(PDF)
</span>
. 2010-09-17
<span class="reference-accessdate">
. Retrieved
<span class="nowrap">
2010-10-27
</span>
</span>
.
</cite>
<span class="Z3988" title="ctx_ver=Z39.88-2004&amp;rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&amp;rft.genre=unknown&amp;rft.btitle=Did+Iqbal%2FTwombly+Raise+the+Bar+for+Browsewrap+Claims%3F&amp;rft.date=2010-09-17&amp;rft_id=http%3A%2F%2Fwww.fornova.net%2Fdocuments%2Fpblog-bna-com.pdf&amp;rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
<span style="display:none;">
</span>
</span>
</span>
</li>
<li id="cite_note-19">
<span class="mw-cite-backlink">
<b>
<a href="#cite_ref-19">
^
</a>
</b>
</span>
<span class="reference-text">
<cite class="citation web">
<a class="external text" href="https://www.techdirt.com/articles/20090605/2228205147.shtml" rel="nofollow">
"Can Scraping Non-Infringing Content Become Copyright Infringement... Because Of How Scrapers Work? | Techdirt"
</a>
.
<i>
Techdirt
</i>
. 2009-06-10
<span class="reference-accessdate">
. Retrieved
<span class="nowrap">
2016-05-24
</span>
</span>
.
</cite>
```

```
<span class="Z3988" title="ctx_ver=Z39.88-2004&amp;rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&amp;rft.genre=unknown&amp;rft.jtitle=Techdirt.&amp;rft.atitle=Can+Scraping+Non-Infringing+Content+Become+Copyright+Infringement...+Because+Of+How+Scrapers+Work%3F+%7C+Techdirt&amp;rft.date=2009-06-10&amp;rft_id=https%3A%2F%2Fwww.techdirt.com%2Farticles%2F20090605%2F2228205147.shtml&amp;rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
 <span style="display:none;">
 </span>
 </span>
</span>
</li>
<li id="cite_note-20">
 <span class="mw-cite-backlink">
  <b>
   <a href="#cite_ref-20">
    ^
   </a>
  </b>
 </span>
 <span class="reference-text">
  <cite class="citation web">
   <a class="external text" href="https://www.eff.org/cases/facebook-v-power-ventures" rel="nofollow">
    "Facebook v. Power Ventures"
   </a>
   .
   <i>
    Electronic Frontier Foundation
   </i>
   <span class="reference-accessdate">
    . Retrieved
    <span class="nowrap">
     2016-05-24
    </span>
   </span>
   .
  </cite>
  <span class="Z3988" title="ctx_ver=Z39.88-2004&amp;rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&amp;rft.genre=unknown&amp;rft.jtitle=Electronic+Frontier+Foundation&amp;rft.atitle=Facebook+v.+Power+Ventures&amp;rft_id=https%3A%2F%2Fwww.eff.org%2Fcases%2Ffacebook-v-power-ventures&amp;rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
   <span style="display:none;">
   </span>
  </span>
 </span>
</li>
<li id="cite_note-21">
 <span class="mw-cite-backlink">
  <b>
   <a href="#cite_ref-21">
    ^
   </a>
  </b>
 </span>
 <span class="reference-text">
  <cite class="citation web">
   <a class="external text" href="https://web.archive.org/web/20071012005033/http://www.bvhd.dk/uploads/tx_mocarticles/S_-_og_Handelsrettens_afg_relse_i_Ofir-sagen.pdf" rel="nofollow">
    "UDSKRIFT AF SØ- &amp; HANDELSRETTENS DOMBOG"
```

```
          </a>
          <span style="font-size:85%;">
           (PDF)
          </span>
          (in Danish). bvhd.dk. 2006-02-24. Archived from
          <a class="external text" href="http://www.bvhd.dk/uploads/tx_mocarticles/S_-_og_Handelsrettens_afg_relse_i_Ofir-sagen.pdf" rel="nofollow">
           the original
          </a>
          <span style="font-size:85%;">
           (PDF)
          </span>
          on 2007-10-12
          <span class="reference-accessdate">
           . Retrieved
           <span class="nowrap">
            2007-05-30
           </span>
          </span>
          .
         </cite>
         <span class="Z3988" title="ctx_ver=Z39.88-2004&amp;rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&amp;rft.genre=unknown&amp;rft.btitle=UDSKRIFT+AF+S%C3%98-+%26+HANDELSRETTENS+DOMBOG&amp;rft.pub=bvhd.dk&amp;rft.date=2006-02-24&amp;rft_id=http%3A%2F%2Fwww.bvhd.dk%2Fuploads%2Ftx_mocarticles%2FS_-_og_Handelsrettens_afg_relse_i_Ofir-sagen.pdf&amp;rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">
          <span style="display:none;">
          </span>
         </span>
        </li>
        <li id="cite_note-22">
         <span class="mw-cite-backlink">
          <b>
           <a href="#cite_ref-22">
            ^
           </a>
          </b>
         </span>
         <span class="reference-text">
          <cite class="citation web">
           <a class="external text" href="http://www.bailii.org/ie/cases/IEHC/2010/H47.html" rel="nofollow">
            "High Court of Ireland Decisions &gt;&gt; Ryanair Ltd -v- Billigfluege.de GMBH 2010 IEHC 47 (26 February 2010)"
           </a>
           . British and Irish Legal Information Institute. 2010-02-26
           <span class="reference-accessdate">
            . Retrieved
            <span class="nowrap">
             2012-04-19
            </span>
           </span>
           .
          </cite>
          <span class="Z3988" title="ctx_ver=Z39.88-2004&amp;rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&amp;rft.genre=unknown&amp;rft.btitle=High+Court+of+Ireland+Decisions+%3E%3E+Ryanair+Ltd+-v-+Billigfluege.de+GMBH+2010+IEHC+47+%2826+February+2010%29&amp;rft.pub=British+and+Iris
```

h+Legal+Information+Institute&rft.date=2010-02-26&rft_id=http%3A%2F%2Fwww.bailii.org%2Fie%2Fcases%2FIEHC%2F2010%2FH47.html&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">

<li id="cite_note-23">

^

<cite class="citation web">
Matthews, Áine (June 2010).

"Intellectual Property: Website Terms of Use"

.

Issue 26: June 2010

. LK Shields Solicitors Update. p. 03

. Retrieved

2012-04-19

.
</cite>
<span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Ajournal&rft.genre=unknown&rft.jtitle=Issue+26%3A+June+2010&rft.atitle=Intellectual+Property%3A+Website+Terms+of+Use&rft.pages=03&rft.date=2010-06&rft.aulast=Matthews&rft.aufirst=%C3%81ine&rft_id=http%3A%2F%2Fwww.lkshields.ie%2Fhtmdocs%2Fpublications%2Fnewsletters%2Fupdate26%2Fupdate26_03.htm&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">

<li id="cite_note-24">

^

<cite class="citation web">

National Office for the Information Economy (February 2004).

"Spam Act 2003: An overview for business"

. Australian Communications Authority. p. 6

. Retrieved

2017-12-07

.
</cite>
<span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=Spam+Act+2003%3A+An+overview+for+business&rft.pages=6&rft.pub=Australian+Communications+Authority&rft.date=2004-02&rft.au=National+Office+for+the+Information+Economy&rft_id=https%3A%2F%2Fwww.lloyds.com%2F~%2Fmedia%2F5880dae185914b2487bed7bd63b96286.ashx&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">

<li id="cite_note-25">

^

<cite class="citation web">
National Office for the Information Economy (February 2004).

"Spam Act 2003: A practical guide for business"

(PDF)

. Australian Communications Authority. p. 20

. Retrieved

2017-12-07

.
</cite>
<span class="Z3988" title="ctx_ver=Z39.88-2004&rft_val_fmt=info%3Aofi%2Ffmt%3Akev%3Amtx%3Abook&rft.genre=unknown&rft.btitle=Spam+Act+2003%3A+A+practical+guide+for+business&rft.pages=20&rft.pub=Australian+Communications+Authority&rft.date=2004-02&rft.au=National+Office+for+the+Information+Economy&rft_id=http%3A%2F%2Fwww.webstartdesign.com.au%2Fspam_business_practical_guide.pdf&rfr_id=info%3Asid%2Fen.wikipedia.org%3AWeb+scraping">

```
          <span style="display:none;">
          </span>
         </span>
        </span>
       </li>
       <li id="cite_note-26">
        <span class="mw-cite-backlink">
         <b>
          <a href="#cite_ref-26">
           ^
          </a>
         </b>
        </span>
        <span class="reference-text">
         Mayank Dhiman
         <a class="external text" href="https://s3.us-west-2.amazonaws.com/research-papers-mynk/Breaking-Frau
d-And-Bot-Detection-Solutions.pdf" rel="nofollow">
          Breaking Fraud &amp;amp; Bot Detection Solutions
         </a>
         <i>
          OWASP AppSec Cali' 2018
         </i>
         Retrieved February 10, 2018.
        </span>
       </li>
      </ol>
     </div>
     <!--
NewPP limit report
Parsed by mw1249
Cached time: 20180226071249
Cache expiry: 1900800
Dynamic content: false
CPU time usage: 0.228 seconds
Real time usage: 0.285 seconds
Preprocessor visited node count: 1447/1000000
Preprocessor generated node count: 0/1500000
Post-expand include size: 49571/2097152 bytes
Template argument size: 320/2097152 bytes
Highest expansion depth: 10/40
Expensive parser function count: 3/500
Lua time usage: 0.119/10.000 seconds
Lua memory usage: 4.53 MB/50 MB
-->
     <!--
Transclusion expansion time report (%,ms,calls,template)
100.00%  240.818      1 -total
 59.86%  144.146      1 Template:Reflist
 28.76%   69.247     18 Template:Cite_web
 19.28%   46.421      5 Template:Cite_journal
 17.92%   43.157      1 Template:More_citations_needed
 13.68%   32.949      2 Template:Ambox
  4.68%   11.269      1 Template:US-centric
  4.39%   10.578      1 Template:Further_information
  3.99%    9.620      1 Template:Colbegin
  3.20%    7.700      1 Template:Globalize
-->
```

```
    </div>
    <!-- Saved in parser cache with key enwiki:pcache:idhash:2696619-0!canonical and timestamp 2018022607
1248 and revision id 825057514
 -->
    <noscript>
     <img alt="" height="1" src="//en.wikipedia.org/wiki/Special:CentralAutoLogin/start?type=1x1" style="bor
der: none; position: absolute;" title="" width="1"/>
    </noscript>
   </div>
   <div class="printfooter">
    Retrieved from "
    <a dir="ltr" href="https://en.wikipedia.org/w/index.php?title=Web_scraping&amp;oldid=825057514">
     https://en.wikipedia.org/w/index.php?title=Web_scraping&amp;oldid=825057514
    </a>
    "
   </div>
   <div class="catlinks" data-mw="interface" id="catlinks">
    <div class="mw-normal-catlinks" id="mw-normal-catlinks">
     <a href="/wiki/Help:Category" title="Help:Category">
      Categories
     </a>
     :
     <ul>
      <li>
       <a href="/wiki/Category:Web_scraping" title="Category:Web scraping">
        Web scraping
       </a>
      </li>
      <li>
       <a href="/wiki/Category:World_Wide_Web" title="Category:World Wide Web">
        World Wide Web
       </a>
      </li>
      <li>
       <a href="/wiki/Category:Spamming" title="Category:Spamming">
        Spamming
       </a>
      </li>
     </ul>
    </div>
    <div class="mw-hidden-catlinks mw-hidden-cats-hidden" id="mw-hidden-catlinks">
     Hidden categories:
     <ul>
      <li>
       <a href="/wiki/Category:CS1_Danish-language_sources_(da)" title="Category:CS1 Danish-language sour
ces (da)">
        CS1 Danish-language sources (da)
       </a>
      </li>
      <li>
       <a href="/wiki/Category:Articles_needing_additional_references_from_June_2017" title="Category:Articl
es needing additional references from June 2017">
        Articles needing additional references from June 2017
       </a>
      </li>
      <li>
       <a href="/wiki/Category:All_articles_needing_additional_references" title="Category:All articles needing
```

```
additional references">
        All articles needing additional references
       </a>
      </li>
      <li>
       <a href="/wiki/Category:Articles_with_limited_geographic_scope_from_October_2015" title="Category:
Articles with limited geographic scope from October 2015">
        Articles with limited geographic scope from October 2015
       </a>
      </li>
      <li>
       <a href="/wiki/Category:USA-centric" title="Category:USA-centric">
        USA-centric
       </a>
      </li>
      <li>
       <a href="/wiki/Category:Pages_using_div_col_with_deprecated_parameters" title="Category:Pages using
div col with deprecated parameters">
        Pages using div col with deprecated parameters
       </a>
      </li>
     </ul>
    </div>
   </div>
   <div class="visualClear">
   </div>
  </div>
 </div>
 <div id="mw-navigation">
  <h2>
   Navigation menu
  </h2>
  <div id="mw-head">
   <div aria-labelledby="p-personal-label" class="" id="p-personal" role="navigation">
    <h3 id="p-personal-label">
     Personal tools
    </h3>
    <ul>
     <li id="pt-anonuserpage">
      Not logged in
     </li>
     <li id="pt-anontalk">
      <a accesskey="n" href="/wiki/Special:MyTalk" title="Discussion about edits from this IP address [n]">
       Talk
      </a>
     </li>
     <li id="pt-anoncontribs">
      <a accesskey="y" href="/wiki/Special:MyContributions" title="A list of edits made from this IP address [y]
">
       Contributions
      </a>
     </li>
     <li id="pt-createaccount">
      <a href="/w/index.php?title=Special:CreateAccount&amp;returnto=Web+scraping" title="You are encoura
ged to create an account and log in; however, it is not mandatory">
       Create account
      </a>
```

```
      </li>
      <li id="pt-login">
       <a accesskey="o" href="/w/index.php?title=Special:UserLogin&amp;returnto=Web+scraping" title="You're encouraged to log in; however, it's not mandatory. [o]">
        Log in
       </a>
      </li>
     </ul>
    </div>
    <div id="left-navigation">
     <div aria-labelledby="p-namespaces-label" class="vectorTabs" id="p-namespaces" role="navigation">
      <h3 id="p-namespaces-label">
       Namespaces
      </h3>
      <ul>
       <li class="selected" id="ca-nstab-main">
        <span>
         <a accesskey="c" href="/wiki/Web_scraping" title="View the content page [c]">
          Article
         </a>
        </span>
       </li>
       <li id="ca-talk">
        <span>
         <a accesskey="t" href="/wiki/Talk:Web_scraping" rel="discussion" title="Discussion about the content page [t]">
          Talk
         </a>
        </span>
       </li>
      </ul>
     </div>
     <div aria-labelledby="p-variants-label" class="vectorMenu emptyPortlet" id="p-variants" role="navigation">
      <input aria-labelledby="p-variants-label" class="vectorMenuCheckbox" type="checkbox"/>
      <h3 id="p-variants-label">
       <span>
        Variants
       </span>
      </h3>
      <div class="menu">
       <ul>
       </ul>
      </div>
     </div>
    </div>
    <div id="right-navigation">
     <div aria-labelledby="p-views-label" class="vectorTabs" id="p-views" role="navigation">
      <h3 id="p-views-label">
       Views
      </h3>
      <ul>
       <li class="collapsible selected" id="ca-view">
        <span>
         <a href="/wiki/Web_scraping">
          Read
         </a>
```

```
        </span>
       </li>
       <li class="collapsible" id="ca-edit">
        <span>
         <a accesskey="e" href="/w/index.php?title=Web_scraping&amp;action=edit" title="Edit this page [e]">
          Edit
         </a>
        </span>
       </li>
       <li class="collapsible" id="ca-history">
        <span>
         <a accesskey="h" href="/w/index.php?title=Web_scraping&amp;action=history" title="Past revisions of t
his page [h]">
          View history
         </a>
        </span>
       </li>
      </ul>
     </div>
     <div aria-labelledby="p-cactions-label" class="vectorMenu emptyPortlet" id="p-cactions" role="navigation"
>
      <input aria-labelledby="p-cactions-label" class="vectorMenuCheckbox" type="checkbox"/>
      <h3 id="p-cactions-label">
       <span>
        More
       </span>
      </h3>
      <div class="menu">
       <ul>
       </ul>
      </div>
     </div>
     <div id="p-search" role="search">
      <h3>
       <label for="searchInput">
        Search
       </label>
      </h3>
      <form action="/w/index.php" id="searchform">
       <div id="simpleSearch">
        <input accesskey="f" id="searchInput" name="search" placeholder="Search Wikipedia" title="Search Wik
ipedia [f]" type="search"/>
        <input name="title" type="hidden" value="Special:Search"/>
        <input class="searchButton mw-fallbackSearchButton" id="mw-searchButton" name="fulltext" title="Sear
ch Wikipedia for this text" type="submit" value="Search"/>
        <input class="searchButton" id="searchButton" name="go" title="Go to a page with this exact name if it e
xists" type="submit" value="Go"/>
       </div>
      </form>
     </div>
    </div>
   </div>
   <div id="mw-panel">
    <div id="p-logo" role="banner">
     <a class="mw-wiki-logo" href="/wiki/Main_Page" title="Visit the main page">
     </a>
    </div>
```

```
<div aria-labelledby="p-navigation-label" class="portal" id="p-navigation" role="navigation">
 <h3 id="p-navigation-label">
  Navigation
 </h3>
 <div class="body">
  <ul>
   <li id="n-mainpage-description">
    <a accesskey="z" href="/wiki/Main_Page" title="Visit the main page [z]">
     Main page
    </a>
   </li>
   <li id="n-contents">
    <a href="/wiki/Portal:Contents" title="Guides to browsing Wikipedia">
     Contents
    </a>
   </li>
   <li id="n-featuredcontent">
    <a href="/wiki/Portal:Featured_content" title="Featured content – the best of Wikipedia">
     Featured content
    </a>
   </li>
   <li id="n-currentevents">
    <a href="/wiki/Portal:Current_events" title="Find background information on current events">
     Current events
    </a>
   </li>
   <li id="n-randompage">
    <a accesskey="x" href="/wiki/Special:Random" title="Load a random article [x]">
     Random article
    </a>
   </li>
   <li id="n-sitesupport">
    <a href="https://donate.wikimedia.org/wiki/Special:FundraiserRedirector?utm_source=donate&amp;utm_
medium=sidebar&amp;utm_campaign=C13_en.wikipedia.org&amp;uselang=en" title="Support us">
     Donate to Wikipedia
    </a>
   </li>
   <li id="n-shoplink">
    <a href="//shop.wikimedia.org" title="Visit the Wikipedia store">
     Wikipedia store
    </a>
   </li>
  </ul>
 </div>
</div>
<div aria-labelledby="p-interaction-label" class="portal" id="p-interaction" role="navigation">
 <h3 id="p-interaction-label">
  Interaction
 </h3>
 <div class="body">
  <ul>
   <li id="n-help">
    <a href="/wiki/Help:Contents" title="Guidance on how to use and edit Wikipedia">
     Help
    </a>
   </li>
   <li id="n-aboutsite">
```

```
        <a href="/wiki/Wikipedia:About" title="Find out about Wikipedia">
         About Wikipedia
        </a>
       </li>
       <li id="n-portal">
        <a href="/wiki/Wikipedia:Community_portal" title="About the project, what you can do, where to find thi
ngs">
         Community portal
        </a>
       </li>
       <li id="n-recentchanges">
        <a accesskey="r" href="/wiki/Special:RecentChanges" title="A list of recent changes in the wiki [r]">
         Recent changes
        </a>
       </li>
       <li id="n-contactpage">
        <a href="//en.wikipedia.org/wiki/Wikipedia:Contact_us" title="How to contact Wikipedia">
         Contact page
        </a>
       </li>
      </ul>
     </div>
    </div>
    <div aria-labelledby="p-tb-label" class="portal" id="p-tb" role="navigation">
     <h3 id="p-tb-label">
      Tools
     </h3>
     <div class="body">
      <ul>
       <li id="t-whatlinkshere">
        <a accesskey="j" href="/wiki/Special:WhatLinksHere/Web_scraping" title="List of all English Wikipedia
pages containing links to this page [j]">
         What links here
        </a>
       </li>
       <li id="t-recentchangeslinked">
        <a accesskey="k" href="/wiki/Special:RecentChangesLinked/Web_scraping" rel="nofollow" title="Recent
 changes in pages linked from this page [k]">
         Related changes
        </a>
       </li>
       <li id="t-upload">
        <a accesskey="u" href="/wiki/Wikipedia:File_Upload_Wizard" title="Upload files [u]">
         Upload file
        </a>
       </li>
       <li id="t-specialpages">
        <a accesskey="q" href="/wiki/Special:SpecialPages" title="A list of all special pages [q]">
         Special pages
        </a>
       </li>
       <li id="t-permalink">
        <a href="/w/index.php?title=Web_scraping&amp;amp;oldid=825057514" title="Permanent link to this revision
 of the page">
         Permanent link
        </a>
       </li>
```

```
      <li id="t-info">
       <a href="/w/index.php?title=Web_scraping&amp;action=info" title="More information about this page">
        Page information
       </a>
      </li>
      <li id="t-wikibase">
       <a accesskey="g" href="https://www.wikidata.org/wiki/Special:EntityPage/Q665452" title="Link to conne
cted data repository item [g]">
        Wikidata item
       </a>
      </li>
      <li id="t-cite">
       <a href="/w/index.php?title=Special:CiteThisPage&amp;page=Web_scraping&amp;id=825057514" title=
"Information on how to cite this page">
        Cite this page
       </a>
      </li>
     </ul>
    </div>
   </div>
   <div aria-labelledby="p-coll-print_export-label" class="portal" id="p-coll-print_export" role="navigation">
    <h3 id="p-coll-print_export-label">
     Print/export
    </h3>
    <div class="body">
     <ul>
      <li id="coll-create_a_book">
       <a href="/w/index.php?title=Special:Book&amp;bookcmd=book_creator&amp;referer=Web+scraping">
        Create a book
       </a>
      </li>
      <li id="coll-download-as-rdf2latex">
       <a href="/w/index.php?title=Special:ElectronPdf&amp;page=Web+scraping&amp;action=show-download
-screen">
        Download as PDF
       </a>
      </li>
      <li id="t-print">
       <a accesskey="p" href="/w/index.php?title=Web_scraping&amp;printable=yes" title="Printable version of
 this page [p]">
        Printable version
       </a>
      </li>
     </ul>
    </div>
   </div>
   <div aria-labelledby="p-lang-label" class="portal" id="p-lang" role="navigation">
    <h3 id="p-lang-label">
     Languages
    </h3>
    <div class="body">
     <ul>
      <li class="interlanguage-link interwiki-ar">
       <a class="interlanguage-link-target" href="https://ar.wikipedia.org/wiki/%D8%A5%D8%B3%D8%AA%
D8%AE%D9%84%D8%A7%D8%B5_%D8%A7%D9%84%D9%85%D9%88%D8%A7%D9%82%D8%B9" h
reflang="ar" lang="ar" title="إستخلاص المواقع – Arabic">
        العربية
```

```
        </a>
       </li>
       <li class="interlanguage-link interwiki-ca">
        <a class="interlanguage-link-target" href="https://ca.wikipedia.org/wiki/Web_scraping" hreflang="ca" lan
g="ca" title="Web scraping – Catalan">
         Català
        </a>
       </li>
       <li class="interlanguage-link interwiki-de">
        <a class="interlanguage-link-target" href="https://de.wikipedia.org/wiki/Screen_Scraping" hreflang="de" l
ang="de" title="Screen Scraping – German">
         Deutsch
        </a>
       </li>
       <li class="interlanguage-link interwiki-es">
        <a class="interlanguage-link-target" href="https://es.wikipedia.org/wiki/Web_scraping" hreflang="es" lan
g="es" title="Web scraping – Spanish">
         Español
        </a>
       </li>
       <li class="interlanguage-link interwiki-eu">
        <a class="interlanguage-link-target" href="https://eu.wikipedia.org/wiki/Web_scraping" hreflang="eu" lan
g="eu" title="Web scraping – Basque">
         Euskara
        </a>
       </li>
       <li class="interlanguage-link interwiki-fr">
        <a class="interlanguage-link-target" href="https://fr.wikipedia.org/wiki/Web_scraping" hreflang="fr" lang
="fr" title="Web scraping – French">
         Français
        </a>
       </li>
       <li class="interlanguage-link interwiki-is">
        <a class="interlanguage-link-target" href="https://is.wikipedia.org/wiki/Vefs%C3%B6fnun" hreflang="is"
lang="is" title="Vefsöfnun – Icelandic">
         Íslenska
        </a>
       </li>
       <li class="interlanguage-link interwiki-it">
        <a class="interlanguage-link-target" href="https://it.wikipedia.org/wiki/Web_scraping" hreflang="it" lang
="it" title="Web scraping – Italian">
         Italiano
        </a>
       </li>
       <li class="interlanguage-link interwiki-lv">
        <a class="interlanguage-link-target" href="https://lv.wikipedia.org/wiki/Rasmo%C5%A1ana" hreflang="lv
" lang="lv" title="Rasmošana – Latvian">
         Latviešu
        </a>
       </li>
       <li class="interlanguage-link interwiki-nl">
        <a class="interlanguage-link-target" href="https://nl.wikipedia.org/wiki/Scrapen" hreflang="nl" lang="nl"
title="Scrapen – Dutch">
         Nederlands
        </a>
       </li>
       <li class="interlanguage-link interwiki-ja">
```

```
      <a class="interlanguage-link-target" href="https://ja.wikipedia.org/wiki/%E3%82%A6%E3%82%A7%E3
%83%96%E3%82%B9%E3%82%AF%E3%83%AC%E3%82%A4%E3%83%94%E3%83%B3%E3%82%B0"
hreflang="ja" lang="ja" title="ウェブスクレイピング – Japanese">
       日本語
      </a>
     </li>
     <li class="interlanguage-link interwiki-sr">
      <a class="interlanguage-link-target" href="https://sr.wikipedia.org/wiki/Web_scraping" hreflang="sr" lang
="sr" title="Web scraping – Serbian">
       Српски / srpski
      </a>
     </li>
     <li class="interlanguage-link interwiki-tr">
      <a class="interlanguage-link-target" href="https://tr.wikipedia.org/wiki/Web_kaz%C4%B1ma" hreflang="
tr" lang="tr" title="Web kazıma – Turkish">
       Türkçe
      </a>
     </li>
     <li class="interlanguage-link interwiki-uk">
      <a class="interlanguage-link-target" href="https://uk.wikipedia.org/wiki/Web_scraping" hreflang="uk" lan
g="uk" title="Web scraping – Ukrainian">
       Українська
      </a>
     </li>
     <li class="interlanguage-link interwiki-zh">
      <a class="interlanguage-link-target" href="https://zh.wikipedia.org/wiki/%E7%BD%91%E9%A1%B5%E
6%8A%93%E5%8F%96" hreflang="zh" lang="zh" title="网页抓取 – Chinese">
       中文
      </a>
     </li>
    </ul>
    <div class="after-portlet after-portlet-lang">
     <span class="wb-langlinks-edit wb-langlinks-link">
      <a class="wbc-editpage" href="https://www.wikidata.org/wiki/Special:EntityPage/Q665452#sitelinks-wiki
pedia" title="Edit interlanguage links">
       Edit links
      </a>
     </span>
    </div>
   </div>
  </div>
 </div>
 <div id="footer" role="contentinfo">
  <ul id="footer-info">
   <li id="footer-info-lastmod">
    This page was last edited on 11 February 2018, at 06:38.
   </li>
   <li id="footer-info-copyright">
    Text is available under the
    <a href="//en.wikipedia.org/wiki/Wikipedia:Text_of_Creative_Commons_Attribution-ShareAlike_3.0_Unp
orted_License" rel="license">
     Creative Commons Attribution-ShareAlike License
    </a>
    <a href="//creativecommons.org/licenses/by-sa/3.0/" rel="license" style="display:none;">
    </a>
    ;
```

```
additional terms may apply.  By using this site, you agree to the
    <a href="//wikimediafoundation.org/wiki/Terms_of_Use">
     Terms of Use
    </a>
    and
    <a href="//wikimediafoundation.org/wiki/Privacy_policy">
     Privacy Policy
    </a>
    . Wikipedia® is a registered trademark of the
    <a href="//www.wikimediafoundation.org/">
     Wikimedia Foundation, Inc.
    </a>
    , a non-profit organization.
   </li>
  </ul>
  <ul id="footer-places">
   <li id="footer-places-privacy">
    <a class="extiw" href="https://wikimediafoundation.org/wiki/Privacy_policy" title="wmf:Privacy policy">
     Privacy policy
    </a>
   </li>
   <li id="footer-places-about">
    <a href="/wiki/Wikipedia:About" title="Wikipedia:About">
     About Wikipedia
    </a>
   </li>
   <li id="footer-places-disclaimer">
    <a href="/wiki/Wikipedia:General_disclaimer" title="Wikipedia:General disclaimer">
     Disclaimers
    </a>
   </li>
   <li id="footer-places-contact">
    <a href="//en.wikipedia.org/wiki/Wikipedia:Contact_us">
     Contact Wikipedia
    </a>
   </li>
   <li id="footer-places-developers">
    <a href="https://www.mediawiki.org/wiki/Special:MyLanguage/How_to_contribute">
     Developers
    </a>
   </li>
   <li id="footer-places-cookiestatement">
    <a href="https://wikimediafoundation.org/wiki/Cookie_statement">
     Cookie statement
    </a>
   </li>
   <li id="footer-places-mobileview">
    <a class="noprint stopMobileRedirectToggle" href="//en.m.wikipedia.org/w/index.php?title=Web_scraping
&amp;mobileaction=toggle_view_mobile">
     Mobile view
    </a>
   </li>
  </ul>
  <ul class="noprint" id="footer-icons">
   <li id="footer-copyrightico">
    <a href="https://wikimediafoundation.org/">
     <img alt="Wikimedia Foundation" height="31" src="/static/images/wikimedia-button.png" srcset="/static/i
```

mages/wikimedia-button-1.5x.png 1.5x, /static/images/wikimedia-button-2x.png 2x" width="88"/>

<li id="footer-poweredbyico">

<script>
(window.RLQ=window.RLQ||[]).push(function(){mw.config.set({"wgPageParseReport":{"limitreport":{"cputime":"0.228","walltime":"0.285","ppvisitednodes":{"value":1447,"limit":1000000},"ppgeneratednodes":{"value":0,"limit":1500000},"postexpandincludesize":{"value":49571,"limit":2097152},"templateargumentsize":{"value":320,"limit":2097152},"expansiondepth":{"value":10,"limit":40},"expensivefunctioncount":{"value":3,"limit":500},"entityaccesscount":{"value":0,"limit":400},"timingprofile":["100.00% 240.818   1 -total"," 59.86% 144.146   1 Template:Reflist"," 28.76%   69.247   18 Template:Cite_web"," 19.28%   46.421   5 Template:Cite_journal"," 17.92%   43.157   1 Template:More_citations_needed"," 13.68%   32.949   2 Template:Ambox","   4.68%   11.269   1 Template:US-centric","   4.39%   10.578   1 Template:Further_information","   3.99%   9.620   1 Template:Colbegin","   3.20%   7.700   1 Template:Globalize"]},"scribunto":{"limitreport-timeusage":{"value":"0.119","limit":"10.000"},"limitreport-memusage":{"value":4752818,"limit":52428800}},"cachereport":{"origin":"mw1249","timestamp":"20180226071249","ttl":1900800,"transientcontent":false}}});});
</script>
<script>
(window.RLQ=window.RLQ||[]).push(function(){mw.config.set({"wgBackendResponseTime":374,"wgHostname":"mw1249"});});
</script>
</body>
</html>

How to get title of a webpage?

In [6]:
```
soup.title
```

Out[6]:
```
<title>Web scraping - Wikipedia</title>
```

How to get all the links found on a webpage?

In [8]:
```
# This will return all the anchor tag or links in a list
soup.find_all('a')
```

Out[8]:
```
[<a id="top"></a>,
 <a href="#mw-head">navigation</a>,
 <a href="#p-search">search</a>,
 <a class="image" href="/wiki/File:Question_book-new.svg"><img alt="" data-file-height="399" data-file-width="512" height="39" src="//upload.wikimedia.org/wikipedia/en/thumb/9/99/Question_book-new.svg/50px-Question_book-new.svg.png" srcset="//upload.wikimedia.org/wikipedia/en/thumb/9/99/Question_book-new.svg/75px-Question_book-new.svg.png 1.5x, //upload.wikimedia.org/wikipedia/en/thumb/9/99/Question_book-new.svg/100px-Question_book-new.svg.png 2x" width="50"/></a>,
```

verification,
improve this article,
adding citations to reliable sources,
Learn how and when to remove this template message,
Data scraping,
data scraping,
extracting data,
websites,
[1],
Hypertext Transfer Protocol,
bot,
web crawler,
database,
retrieval,
analysis,
[1],
[2],
parsed,
contact scraping,
web indexing,
web mining,
data mining,
price comparison,
website change detection,
web mashup,
Web pages,
HTML,
XHTML,
end-users,
Application Programming Interface,
Amazon AWS,
Google,
JSON,
DOM,
computer vision,
natural language processing,
1 Techniques,
1.1 Human copy-and-paste,
1.2 Text pattern matching,
1.3 HTTP programming,
1.4 HTML parsing,
1.5 DOM parsing,
1.6 Vertical aggregation,

1.7 Semantic annotation recognizing,
1.8 Computer vision web-page analysis,
2 Software,
2.1 Example tools,
2.1.1 Javascript tools,
2.1.2 SaaS version,
2.1.3 Web crawling frameworks,
3 Legal issues,
3.1 United States,
3.2 The EU,
3.3 Australia,
4 Methods to prevent web scraping,
5 See also,
6 References,
edit,
semantic web,
human-computer interactions,
edit,
edit,
grep,
regular expression,
Perl,
Python,
edit,
Static,
dynamic web pages,
socket programming,
edit,
wrapper,
[3],
semi-structured data,
XQuery,
edit,
Document Object Model,
Internet Explorer,
Mozilla,
edit,
Long Tail,
<a href="/w/index.php?title=Web_scraping&action=edit&section=8" title="Edit section: Semantic a

nnotation recognizing">edit,
metadata,
Microformat,
[4],
edit,
machine learning,
computer vision,
[5],
edit,
edit,
cURL,
Data Toolbar,
Diffbot,
Heritrix,
Wayback Machine,
HtmlUnit,
HTTrack,
iMacros,
Selenium (software),
Jaxer,
nokogiri,
OutWit Hub,
watir,
Wget,
WSO2 Mashup Server,
Yahoo! Query Language,
edit,
Greasemonkey,
Node.js,
PhantomJS,
headless browser,
jQuery,
edit,
diggernaut.com,
fScraper,
Import.io,
Listly.io,
Mozenda,
uScraper,
edit,
Scrapy,
edit,
worldwide view,
improve this article,
talk page,
create a new article,

Learn how and when to remove this template message,
terms of use,
[6],
edit,
legal claims,
Computer Fraud and Abuse Act,
trespass to chattel,
[7],
Feist Publications v. Rural Telephone Service,
trespass to chattels,
[8],
[9],
eBay v. Bidder's Edge,
auction sniping,
chattels,
plaintiff,
defendant,
[10],
screen scraping,
American Airlines,
[11],
injunction,
[12],
Southwest Airlines,
US Copyright law,
Supreme Court of the United States,
Yahoo!,
[13],
Craigslist v. 3Taps,
Computer Fraud and Abuse Act,
[14],
Cvent, Inc.,
Eventbrite, Inc.,
browse wrap,
[15],
United States District Court for the Eastern District of Pennsylvania,
[16],
QVC,
[17],
[18],
Facebook, Inc. v. Power Ventures, Inc.,
Electronic Frontier Foundation,
[19],
[20],
Associated Press v. Meltwater U.S. Holdings, Inc.,

edit,
[21],
inchoate,
[22],
[23],
edit,
Spam Act 2003,
[24],
[25],
edit,
IP address,
geolocation,
DNSRBL,
web service,
API,
user agent,
strings,
robots.txt,
googlebot,
CAPTCHA,
application firewalls,
[26],
honeypot,
Obfuscation,
CSS sprites,
accessibility,
screen reader,
robots.txt,
edit,
Archive.is,
Comparison of feed aggregators,
Data scraping,
Data wrangling,
Importer,
Job wrapping,
Knowledge extraction,
OpenSocial,
Scraper site,
Fake news website,
Blog scraping,
Spamdexing,
Domain name drop list,
Text corpus,
Web archiving,
Blog network,
Search Engine Scraping,
Web crawlers,
edit,
^a,

^b,
doi:
10.1177/0739456X16664789,
^,
doi:
10.5430/air.v2n1p44,
^,
"Joint Optimization of Wrapper Generation and Template Detection",
^,
Semantic annotation based web scraping,
^,
"Diffbot Is Using Computer Vision to Reinvent the Semantic Web",
^,
"FAQ about linking – Are website terms of use binding contracts?",
the original,
^,
"Symbiotic Relationships: Pragmatic Acceptance of Data Scraping",
doi:
10.15779/Z38B39B,
ISSN,
1086-3818,
^,
"Internet Law, Ch. 06: Trespass to Chattels",
^,
"What are the "trespass to chattels" claims some companies or website owners have brought?",
the original,
^,
"Ticketmaster Corp. v. Tickets.com, Inc.",
^,
"American Airlines v. FareChase",
the original,
^,
"American Airlines, FareChase Settle Suit",
^,
Detecting and Blocking Site Scraping Attacks,
^,
"Controversy Surrounds 'Screen Scrapers': Software Helps Users Access Web Sites But Activity by Competitors Comes Under Scrutiny",

^,
"QVC Inc. v. Resultly LLC, No. 14-06714 (E.D. Pa. filed Nov. 24, 2014)",
^,
"QVC Inc. v. Resultly LLC, No. 14-06714 (E.D. Pa. filed Nov. 24, 2014)",
^,
"QVC Sues Shopping App for Web Scraping That Allegedly Triggered Site Outage",
^,
"Did Iqbal/Twombly Raise the Bar for Browsewrap Claims?",
^,
"Can Scraping Non-Infringing Content Become Copyright Infringement... Because Of How Scrapers Work? | Techdirt",
^,
"Facebook v. Power Ventures",
^,
"UDSKRIFT AF SØ- & HANDELSRETTENS DOMBOG",
the original,
^,
"High Court of Ireland Decisions >> Ryanair Ltd -v- Billigfluege.de GMBH 2010 IEHC 47 (26 February 2010)",
^,
"Intellectual Property: Website Terms of Use",
^,
"Spam Act 2003: An overview for business",
^,
"Spam Act 2003: A practical guide for business",
^,
Breaking Fraud & Bot Detection Solutions,
https://en.wikipedia.org/w/index.php?title=Web_scraping&oldid=825057514,
Categories,
Web scraping,
World Wide Web,
Spamming,
CS1 Danish-language sources (da),
Articles needing additional references from June 2017,
All articles needing additional references,
Articles with limited geographic scope from October 201

5,
USA-centric,
Pages using div col with deprecated parameters,
Talk,
Contributions,
Create account,
Log in,
Article,
Talk,
Read,
Edit,
View history,
,
Main page,
Contents,
Featured content,
Current events,
Random article,
Donate to Wikipedia,
Wikipedia store,
Help,
About Wikipedia,
Community portal,
Recent changes,
Contact page,
What links here,
Related changes,
Upload file,
Special pages,
Permanent link,
Page information,
Wikidata item,
Cite this page,
Create a book,
Download as PDF,

Printable version,
العربية,
Català,
Deutsch,
Español,
Euskara,
Français,
Íslenska,
Italiano,
Latviešu,
Nederlands,
日本語,
Српски / srpski,
Türkçe,
Українська,
中文,
Edit links,
Creative Commons Attribution-ShareAlike License,
,
Terms of Use,
Privacy Policy,
Wikimedia Foundation, Inc.,
Privacy policy,
About Wikipedia,
Disclaimers,
Contact Wikipedia,
Developers,
Cookie statement,
Mobile view,
,
]

How to get actual link ?

In [9]:
```
for url in soup.find_all('a'):
    print(url.get('href'))
```

```
None
#mw-head
#p-search
/wiki/File:Question_book-new.svg
/wiki/Wikipedia:Verifiability
//en.wikipedia.org/w/index.php?title=Web_scraping&action=edit
/wiki/Help:Introduction_to_referencing_with_Wiki_Markup/1
/wiki/Help:Maintenance_template_removal
/wiki/Data_scraping
/wiki/Data_scraping
/wiki/Data_extraction
/wiki/Website
#cite_note-Boeing2016JPER-1
/wiki/Hypertext_Transfer_Protocol
/wiki/Internet_bot
/wiki/Web_crawler
/wiki/Database
/wiki/Data_retrieval
/wiki/Data_analysis
#cite_note-Boeing2016JPER-1
#cite_note-2
/wiki/Parsing
/wiki/Contact_scraping
/wiki/Web_indexing
/wiki/Web_mining
/wiki/Data_mining
/wiki/Comparison_shopping_website
/wiki/Change_detection_and_notification
/wiki/Web_mashup
/wiki/Web_page
/wiki/HTML
/wiki/XHTML
/wiki/End-user_(computer_science)
/wiki/Application_Programming_Interface
/wiki/Amazon_AWS
/wiki/Google
/wiki/JSON
/wiki/Document_Object_Model
/wiki/Computer_vision
/wiki/Natural_language_processing
#Techniques
#Human_copy-and-paste
#Text_pattern_matching
#HTTP_programming
#HTML_parsing
#DOM_parsing
#Vertical_aggregation
#Semantic_annotation_recognizing
#Computer_vision_web-page_analysis
```

#Software
#Example_tools
#Javascript_tools
#SaaS_version
#Web_crawling_frameworks
#Legal_issues
#United_States
#The_EU
#Australia
#Methods_to_prevent_web_scraping
#See_also
#References
/w/index.php?title=Web_scraping&action=edit&section=1
/wiki/Semantic_web
/wiki/Human-computer_interaction
/w/index.php?title=Web_scraping&action=edit&section=2
/w/index.php?title=Web_scraping&action=edit&section=3
/wiki/Grep
/wiki/Regular_expression
/wiki/Perl
/wiki/Python_(programming_language)
/w/index.php?title=Web_scraping&action=edit&section=4
/wiki/Static_web_page
/wiki/Dynamic_web_page
/wiki/Socket_programming
/w/index.php?title=Web_scraping&action=edit&section=5
/wiki/Wrapper_(data_mining)
#cite_note-3
/wiki/Semi-structured_data
/wiki/XQuery
/w/index.php?title=Web_scraping&action=edit&section=6
/wiki/Document_Object_Model
/wiki/Internet_Explorer
/wiki/Mozilla
/w/index.php?title=Web_scraping&action=edit&section=7
/wiki/Long_Tail
/w/index.php?title=Web_scraping&action=edit&section=8
/wiki/Metadata
/wiki/Microformat
#cite_note-4
/w/index.php?title=Web_scraping&action=edit&section=9
/wiki/Machine_learning
/wiki/Computer_vision
#cite_note-5
/w/index.php?title=Web_scraping&action=edit&section=10
/w/index.php?title=Web_scraping&action=edit&section=11
/wiki/CURL
/wiki/Data_Toolbar
/wiki/Diffbot
/wiki/Heritrix
/wiki/Wayback_Machine
/wiki/HtmlUnit
/wiki/HTTrack
/wiki/IMacros
/wiki/Selenium_(software)
/wiki/Aptana#Aptana_Jaxer
/wiki/Nokogiri_(software)

/wiki/OutWit_Hub
/wiki/Watir
/wiki/Wget
/wiki/WSO2_Mashup_Server
/wiki/Yahoo!_Query_Language
/w/index.php?title=Web_scraping&action=edit&section=12
/wiki/Greasemonkey
/wiki/Node.js
/wiki/PhantomJS
/wiki/Headless_browser
/wiki/JQuery
/w/index.php?title=Web_scraping&action=edit&section=13
https://www.diggernaut.com/
/w/index.php?title=FScraper&action=edit&redlink=1
/wiki/Import.io
https://listly.io/
/wiki/Mozenda
/w/index.php?title=UScraper&action=edit&redlink=1
/w/index.php?title=Web_scraping&action=edit&section=14
/wiki/Scrapy
/w/index.php?title=Web_scraping&action=edit&section=15
/wiki/Wikipedia:WikiProject_Countering_systemic_bias
//en.wikipedia.org/w/index.php?title=Web_scraping&action=edit
/wiki/Talk:Web_scraping
/wiki/Wikipedia:Article_wizard
/wiki/Help:Maintenance_template_removal
/wiki/Terms_of_use
#cite_note-6
/w/index.php?title=Web_scraping&action=edit&section=16
/wiki/Cause_of_action
/wiki/Computer_Fraud_and_Abuse_Act
/wiki/Trespass_to_chattels
#cite_note-7
/wiki/Feist_Publications,_Inc.,_v._Rural_Telephone_Service_Co.
/wiki/Trespass_to_chattels
#cite_note-8
#cite_note-9
/wiki/EBay_v._Bidder%27s_Edge
/wiki/Auction_sniping
/wiki/Personal_property
/wiki/Plaintiff
/wiki/Defendant
#cite_note-10
/wiki/Screen_scraping
/wiki/American_Airlines
#cite_note-11
/wiki/Injunction
#cite_note-12
/wiki/Southwest_Airlines
/wiki/US_Copyright_law
/wiki/Supreme_Court_of_the_United_States
/wiki/Yahoo!
#cite_note-impervawp2011-13
/wiki/Craigslist_v._3Taps
/wiki/Computer_Fraud_and_Abuse_Act
#cite_note-14
/wiki/Cvent,_Inc.

/wiki/Eventbrite
/wiki/Browse_wrap
#cite_note-15
/wiki/United_States_District_Court_for_the_Eastern_District_of_Pennsylvania
#cite_note-16
/wiki/QVC
#cite_note-17
#cite_note-18
/wiki/Facebook,_Inc._v._Power_Ventures,_Inc.
/wiki/Electronic_Frontier_Foundation
#cite_note-19
#cite_note-20
/wiki/Associated_Press_v._Meltwater_U.S._Holdings,_Inc.
/w/index.php?title=Web_scraping&action=edit&section=17
#cite_note-21
/w/index.php?title=Inchoate&action=edit&redlink=1
#cite_note-22
#cite_note-23
/w/index.php?title=Web_scraping&action=edit&section=18
/wiki/Spam_Act_2003
#cite_note-24
#cite_note-25
/w/index.php?title=Web_scraping&action=edit&section=19
/wiki/IP_address
/wiki/Geolocation
/wiki/DNSBL
/wiki/Web_service
/wiki/Application_programming_interface
/wiki/User_agent
/wiki/String_(computer_science)
/wiki/Robots_exclusion_standard
/wiki/Googlebot
/wiki/CAPTCHA
/wiki/Application_firewall
#cite_note-26
/wiki/Honeypot_(computing)
/wiki/Obfuscation
/wiki/CSS_sprite
/wiki/Web_accessibility
/wiki/Screen_reader
https://www.promptcloud.com/blog/how-to-read-and-respect-robots-file
/w/index.php?title=Web_scraping&action=edit&section=20
/wiki/Archive.is
/wiki/Comparison_of_feed_aggregators
/wiki/Data_scraping
/wiki/Data_wrangling
/wiki/Importer_(computing)
/wiki/Job_wrapping
/wiki/Knowledge_extraction
/wiki/OpenSocial
/wiki/Scraper_site
/wiki/Fake_news_website
/wiki/Blog_scraping
/wiki/Spamdexing
/wiki/Domain_name_drop_list
/wiki/Text_corpus
/wiki/Web_archiving

/wiki/Blog_network
/wiki/Search_Engine_Scraping
/wiki/Category:Web_crawlers
/w/index.php?title=Web_scraping&action=edit&section=21
#cite_ref-Boeing2016JPER_1-0
#cite_ref-Boeing2016JPER_1-1
/wiki/Digital_object_identifier
//doi.org/10.1177%2F0739456X16664789
#cite_ref-2
/wiki/Digital_object_identifier
//doi.org/10.5430%2Fair.v2n1p44
#cite_ref-3
https://pdfs.semanticscholar.org/4fb4/3c5a212df751e84c3b2f8d29fabfe56c3616.pdf
#cite_ref-4
http://www.gooseeker.com/en/node/knowledgebase/freeformat
#cite_ref-5
http://www.xconomy.com/san-francisco/2012/07/25/diffbot-is-using-computer-vision-to-reinvent-the-semantic-web/
#cite_ref-6
https://web.archive.org/web/20020308222536/http://www.chillingeffects.org/linking/faq.cgi#QID596
http://www.chillingeffects.org/linking/faq.cgi#QID596
#cite_ref-7
http://scholarship.law.berkeley.edu/btlj/vol29/iss4/16/
/wiki/Digital_object_identifier
//doi.org/10.15779%2FZ38B39B
/wiki/International_Standard_Serial_Number
//www.worldcat.org/issn/1086-3818
#cite_ref-8
http://www.tomwbell.com/NetLaw/Ch06.html
#cite_ref-9
https://web.archive.org/web/20020308222536/http://www.chillingeffects.org/linking/faq.cgi#QID460
http://www.chillingeffects.org/linking/faq.cgi#QID460
#cite_ref-10
http://www.tomwbell.com/NetLaw/Ch07/Ticketmaster.html
#cite_ref-11
https://web.archive.org/web/20110723131832/http://www.fornova.net/documents/AAFareChase.pdf
http://www.fornova.net/documents/AAFareChase.pdf
#cite_ref-12
http://www.thefreelibrary.com/American+Airlines,+FareChase+Settle+Suit.-a0103213546
#cite_ref-impervawp2011_13-0
http://www.imperva.com/docs/WP_Detecting_and_Blocking_Site_Scraping_Attacks.pdf
#cite_ref-14
http://library.findlaw.com/2003/Jul/29/132944.html
#cite_ref-15
http://www.fornova.net/documents/Cvent.pdf
#cite_ref-16
https://www.scribd.com/doc/249068700/LinkedIn-v-Resultly-LLC-Complaint?secret_password=pEVKDbnvhQL52oKfdrmT
#cite_ref-17
http://newmedialaw.proskauer.com/2014/12/05/qvc-sues-shopping-app-for-web-scraping-that-allegedly-triggered-site-outage/
#cite_ref-18
http://www.fornova.net/documents/pblog-bna-com.pdf
#cite_ref-19
https://www.techdirt.com/articles/20090605/2228205147.shtml
#cite_ref-20
https://www.eff.org/cases/facebook-v-power-ventures

#cite_ref-21
https://web.archive.org/web/20071012005033/http://www.bvhd.dk/uploads/tx_mocarticles/S_-_og_Handelsrettens_afg_relse_i_Ofir-sagen.pdf
http://www.bvhd.dk/uploads/tx_mocarticles/S_-_og_Handelsrettens_afg_relse_i_Ofir-sagen.pdf
#cite_ref-22
http://www.bailii.org/ie/cases/IEHC/2010/H47.html
#cite_ref-23
http://www.lkshields.ie/htmdocs/publications/newsletters/update26/update26_03.htm
#cite_ref-24
https://www.lloyds.com/~/media/5880dae185914b2487bed7bd63b96286.ashx
#cite_ref-25
http://www.webstartdesign.com.au/spam_business_practical_guide.pdf
#cite_ref-26
https://s3.us-west-2.amazonaws.com/research-papers-mynk/Breaking-Fraud-And-Bot-Detection-Solutions.pdf
https://en.wikipedia.org/w/index.php?title=Web_scraping&oldid=825057514
/wiki/Help:Category
/wiki/Category:Web_scraping
/wiki/Category:World_Wide_Web
/wiki/Category:Spamming
/wiki/Category:CS1_Danish-language_sources_(da)
/wiki/Category:Articles_needing_additional_references_from_June_2017
/wiki/Category:All_articles_needing_additional_references
/wiki/Category:Articles_with_limited_geographic_scope_from_October_2015
/wiki/Category:USA-centric
/wiki/Category:Pages_using_div_col_with_deprecated_parameters
/wiki/Special:MyTalk
/wiki/Special:MyContributions
/w/index.php?title=Special:CreateAccount&returnto=Web+scraping
/w/index.php?title=Special:UserLogin&returnto=Web+scraping
/wiki/Web_scraping
/wiki/Talk:Web_scraping
/wiki/Web_scraping
/w/index.php?title=Web_scraping&action=edit
/w/index.php?title=Web_scraping&action=history
/wiki/Main_Page
/wiki/Main_Page
/wiki/Portal:Contents
/wiki/Portal:Featured_content
/wiki/Portal:Current_events
/wiki/Special:Random
https://donate.wikimedia.org/wiki/Special:FundraiserRedirector?utm_source=donate&utm_medium=sidebar&utm_campaign=C13_en.wikipedia.org&uselang=en
//shop.wikimedia.org
/wiki/Help:Contents
/wiki/Wikipedia:About
/wiki/Wikipedia:Community_portal
/wiki/Special:RecentChanges
//en.wikipedia.org/wiki/Wikipedia:Contact_us
/wiki/Special:WhatLinksHere/Web_scraping
/wiki/Special:RecentChangesLinked/Web_scraping
/wiki/Wikipedia:File_Upload_Wizard
/wiki/Special:SpecialPages
/w/index.php?title=Web_scraping&oldid=825057514
/w/index.php?title=Web_scraping&action=info
https://www.wikidata.org/wiki/Special:EntityPage/Q665452
/w/index.php?title=Special:CiteThisPage&page=Web_scraping&id=825057514
/w/index.php?title=Special:Book&bookcmd=book_creator&referer=Web+scraping

/w/index.php?title=Special:ElectronPdf&page=Web+scraping&action=show-download-screen
/w/index.php?title=Web_scraping&printable=yes
https://ar.wikipedia.org/wiki/%D8%A5%D8%B3%D8%AA%D8%AE%D9%84%D8%A7%D8%B5_%D8%A7%D9%84%D9%85%D9%88%D8%A7%D9%82%D8%B9
https://ca.wikipedia.org/wiki/Web_scraping
https://de.wikipedia.org/wiki/Screen_Scraping
https://es.wikipedia.org/wiki/Web_scraping
https://eu.wikipedia.org/wiki/Web_scraping
https://fr.wikipedia.org/wiki/Web_scraping
https://is.wikipedia.org/wiki/Vefs%C3%B6fnun
https://it.wikipedia.org/wiki/Web_scraping
https://lv.wikipedia.org/wiki/Rasmo%C5%A1ana
https://nl.wikipedia.org/wiki/Scrapen
https://ja.wikipedia.org/wiki/%E3%82%A6%E3%82%A7%E3%83%96%E3%82%B9%E3%82%AF%E3%83%AC%E3%82%A4%E3%83%94%E3%83%B3%E3%82%B0
https://sr.wikipedia.org/wiki/Web_scraping
https://tr.wikipedia.org/wiki/Web_kaz%C4%B1ma
https://uk.wikipedia.org/wiki/Web_scraping
https://zh.wikipedia.org/wiki/%E7%BD%91%E9%A1%B5%E6%8A%93%E5%8F%96
https://www.wikidata.org/wiki/Special:EntityPage/Q665452#sitelinks-wikipedia
//en.wikipedia.org/wiki/Wikipedia:Text_of_Creative_Commons_Attribution-ShareAlike_3.0_Unported_License
//creativecommons.org/licenses/by-sa/3.0/
//wikimediafoundation.org/wiki/Terms_of_Use
//wikimediafoundation.org/wiki/Privacy_policy
//www.wikimediafoundation.org/
https://wikimediafoundation.org/wiki/Privacy_policy
/wiki/Wikipedia:About
/wiki/Wikipedia:General_disclaimer
//en.wikipedia.org/wiki/Wikipedia:Contact_us
https://www.mediawiki.org/wiki/Special:MyLanguage/How_to_contribute
https://wikimediafoundation.org/wiki/Cookie_statement
//en.m.wikipedia.org/w/index.php?title=Web_scraping&mobileaction=toggle_view_mobile
https://wikimediafoundation.org/
//www.mediawiki.org/

How to filter links based on some parameter?

In [10]:
```
soup.find_all('a',title='Data scraping')
```

Out[10]:
```
[<a href="/wiki/Data_scraping" title="Data scraping">Data scraping</a>,
 <a href="/wiki/Data_scraping" title="Data scraping">data scraping</a>,
 <a href="/wiki/Data_scraping" title="Data scraping">Data scraping</a>]
```

In [11]:
```
for url in soup.find_all('a',title='Data scraping'):
    print(url.get('href'))
```

```
/wiki/Data_scraping
/wiki/Data_scraping
/wiki/Data_scraping
```

In [12]:
```
one_url=soup.find('a',title='Data scraping')
print(one_url)
one_url.get('href')
```

```
<a href="/wiki/Data_scraping" title="Data scraping">Data scraping</a>
```

Out[12]: '/wiki/Data_scraping'

In [13]:
```
one_url=soup.find('a',title='Data scraping')
one_url.text
```

Out[13]: 'Data scraping'

How can i do the same in list comprehension?

In [14]:
```
[url.get('href') for url in soup.find_all('a')]
```

Out[14]:
```
[None,
 '#mw-head',
 '#p-search',
 '/wiki/File:Question_book-new.svg',
 '/wiki/Wikipedia:Verifiability',
 '//en.wikipedia.org/w/index.php?title=Web_scraping&action=edit',
 '/wiki/Help:Introduction_to_referencing_with_Wiki_Markup/1',
 '/wiki/Help:Maintenance_template_removal',
 '/wiki/Data_scraping',
 '/wiki/Data_scraping',
 '/wiki/Data_extraction',
 '/wiki/Website',
 '#cite_note-Boeing2016JPER-1',
 '/wiki/Hypertext_Transfer_Protocol',
 '/wiki/Internet_bot',
 '/wiki/Web_crawler',
 '/wiki/Database',
 '/wiki/Data_retrieval',
 '/wiki/Data_analysis',
 '#cite_note-Boeing2016JPER-1',
 '#cite_note-2',
 '/wiki/Parsing',
 '/wiki/Contact_scraping',
 '/wiki/Web_indexing',
 '/wiki/Web_mining',
 '/wiki/Data_mining',
 '/wiki/Comparison_shopping_website',
 '/wiki/Change_detection_and_notification',
 '/wiki/Web_mashup',
 '/wiki/Web_page',
 '/wiki/HTML',
 '/wiki/XHTML',
 '/wiki/End-user_(computer_science)',
 '/wiki/Application_Programming_Interface',
 '/wiki/Amazon_AWS',
 '/wiki/Google',
 '/wiki/JSON',
 '/wiki/Document_Object_Model',
 '/wiki/Computer_vision',
 '/wiki/Natural_language_processing',
 '#Techniques',
 '#Human_copy-and-paste',
 '#Text_pattern_matching',
```

'#HTTP_programming',
'#HTML_parsing',
'#DOM_parsing',
'#Vertical_aggregation',
'#Semantic_annotation_recognizing',
'#Computer_vision_web-page_analysis',
'#Software',
'#Example_tools',
'#Javascript_tools',
'#SaaS_version',
'#Web_crawling_frameworks',
'#Legal_issues',
'#United_States',
'#The_EU',
'#Australia',
'#Methods_to_prevent_web_scraping',
'#See_also',
'#References',
'/w/index.php?title=Web_scraping&action=edit&section=1',
'/wiki/Semantic_web',
'/wiki/Human-computer_interaction',
'/w/index.php?title=Web_scraping&action=edit&section=2',
'/w/index.php?title=Web_scraping&action=edit&section=3',
'/wiki/Grep',
'/wiki/Regular_expression',
'/wiki/Perl',
'/wiki/Python_(programming_language)',
'/w/index.php?title=Web_scraping&action=edit&section=4',
'/wiki/Static_web_page',
'/wiki/Dynamic_web_page',
'/wiki/Socket_programming',
'/w/index.php?title=Web_scraping&action=edit&section=5',
'/wiki/Wrapper_(data_mining)',
'#cite_note-3',
'/wiki/Semi-structured_data',
'/wiki/XQuery',
'/w/index.php?title=Web_scraping&action=edit&section=6',
'/wiki/Document_Object_Model',
'/wiki/Internet_Explorer',
'/wiki/Mozilla',
'/w/index.php?title=Web_scraping&action=edit&section=7',
'/wiki/Long_Tail',
'/w/index.php?title=Web_scraping&action=edit&section=8',
'/wiki/Metadata',
'/wiki/Microformat',
'#cite_note-4',
'/w/index.php?title=Web_scraping&action=edit&section=9',
'/wiki/Machine_learning',
'/wiki/Computer_vision',
'#cite_note-5',
'/w/index.php?title=Web_scraping&action=edit&section=10',
'/w/index.php?title=Web_scraping&action=edit&section=11',
'/wiki/CURL',
'/wiki/Data_Toolbar',
'/wiki/Diffbot',
'/wiki/Heritrix',
'/wiki/Wayback_Machine',

```
'/wiki/HtmlUnit',
'/wiki/HTTrack',
'/wiki/IMacros',
'/wiki/Selenium_(software)',
'/wiki/Aptana#Aptana_Jaxer',
'/wiki/Nokogiri_(software)',
'/wiki/OutWit_Hub',
'/wiki/Watir',
'/wiki/Wget',
'/wiki/WSO2_Mashup_Server',
'/wiki/Yahoo!_Query_Language',
'/w/index.php?title=Web_scraping&action=edit&section=12',
'/wiki/Greasemonkey',
'/wiki/Node.js',
'/wiki/PhantomJS',
'/wiki/Headless_browser',
'/wiki/JQuery',
'/w/index.php?title=Web_scraping&action=edit&section=13',
'https://www.diggernaut.com/',
'/w/index.php?title=FScraper&action=edit&redlink=1',
'/wiki/Import.io',
'https://listly.io/',
'/wiki/Mozenda',
'/w/index.php?title=UScraper&action=edit&redlink=1',
'/w/index.php?title=Web_scraping&action=edit&section=14',
'/wiki/Scrapy',
'/w/index.php?title=Web_scraping&action=edit&section=15',
'/wiki/Wikipedia:WikiProject_Countering_systemic_bias',
'//en.wikipedia.org/w/index.php?title=Web_scraping&action=edit',
'/wiki/Talk:Web_scraping',
'/wiki/Wikipedia:Article_wizard',
'/wiki/Help:Maintenance_template_removal',
'/wiki/Terms_of_use',
'#cite_note-6',
'/w/index.php?title=Web_scraping&action=edit&section=16',
'/wiki/Cause_of_action',
'/wiki/Computer_Fraud_and_Abuse_Act',
'/wiki/Trespass_to_chattels',
'#cite_note-7',
'/wiki/Feist_Publications,_Inc.,_v._Rural_Telephone_Service_Co.',
'/wiki/Trespass_to_chattels',
'#cite_note-8',
'#cite_note-9',
'/wiki/EBay_v._Bidder%27s_Edge',
'/wiki/Auction_sniping',
'/wiki/Personal_property',
'/wiki/Plaintiff',
'/wiki/Defendant',
'#cite_note-10',
'/wiki/Screen_scraping',
'/wiki/American_Airlines',
'#cite_note-11',
'/wiki/Injunction',
'#cite_note-12',
'/wiki/Southwest_Airlines',
'/wiki/US_Copyright_law',
'/wiki/Supreme_Court_of_the_United_States',
```

```
'/wiki/Yahoo!',
'#cite_note-impervawp2011-13',
'/wiki/Craigslist_v._3Taps',
'/wiki/Computer_Fraud_and_Abuse_Act',
'#cite_note-14',
'/wiki/Cvent,_Inc.',
'/wiki/Eventbrite',
'/wiki/Browse_wrap',
'#cite_note-15',
'/wiki/United_States_District_Court_for_the_Eastern_District_of_Pennsylvania',
'#cite_note-16',
'/wiki/QVC',
'#cite_note-17',
'#cite_note-18',
'/wiki/Facebook,_Inc._v._Power_Ventures,_Inc.',
'/wiki/Electronic_Frontier_Foundation',
'#cite_note-19',
'#cite_note-20',
'/wiki/Associated_Press_v._Meltwater_U.S._Holdings,_Inc.',
'/w/index.php?title=Web_scraping&action=edit&section=17',
'#cite_note-21',
'/w/index.php?title=Inchoate&action=edit&redlink=1',
'#cite_note-22',
'#cite_note-23',
'/w/index.php?title=Web_scraping&action=edit&section=18',
'/wiki/Spam_Act_2003',
'#cite_note-24',
'#cite_note-25',
'/w/index.php?title=Web_scraping&action=edit&section=19',
'/wiki/IP_address',
'/wiki/Geolocation',
'/wiki/DNSBL',
'/wiki/Web_service',
'/wiki/Application_programming_interface',
'/wiki/User_agent',
'/wiki/String_(computer_science)',
'/wiki/Robots_exclusion_standard',
'/wiki/Googlebot',
'/wiki/CAPTCHA',
'/wiki/Application_firewall',
'#cite_note-26',
'/wiki/Honeypot_(computing)',
'/wiki/Obfuscation',
'/wiki/CSS_sprite',
'/wiki/Web_accessibility',
'/wiki/Screen_reader',
'https://www.promptcloud.com/blog/how-to-read-and-respect-robots-file',
'/w/index.php?title=Web_scraping&action=edit&section=20',
'/wiki/Archive.is',
'/wiki/Comparison_of_feed_aggregators',
'/wiki/Data_scraping',
'/wiki/Data_wrangling',
'/wiki/Importer_(computing)',
'/wiki/Job_wrapping',
'/wiki/Knowledge_extraction',
'/wiki/OpenSocial',
'/wiki/Scraper_site',
```

```
 '/wiki/Fake_news_website',
 '/wiki/Blog_scraping',
 '/wiki/Spamdexing',
 '/wiki/Domain_name_drop_list',
 '/wiki/Text_corpus',
 '/wiki/Web_archiving',
 '/wiki/Blog_network',
 '/wiki/Search_Engine_Scraping',
 '/wiki/Category:Web_crawlers',
 '/w/index.php?title=Web_scraping&action=edit&section=21',
 '#cite_ref-Boeing2016JPER_1-0',
 '#cite_ref-Boeing2016JPER_1-1',
 '/wiki/Digital_object_identifier',
 '//doi.org/10.1177%2F0739456X16664789',
 '#cite_ref-2',
 '/wiki/Digital_object_identifier',
 '//doi.org/10.5430%2Fair.v2n1p44',
 '#cite_ref-3',
 'https://pdfs.semanticscholar.org/4fb4/3c5a212df751e84c3b2f8d29fabfe56c3616.pdf',
 '#cite_ref-4',
 'http://www.gooseeker.com/en/node/knowledgebase/freeformat',
 '#cite_ref-5',
 'http://www.xconomy.com/san-francisco/2012/07/25/diffbot-is-using-computer-vision-to-reinvent-the-semantic
-web/',
 '#cite_ref-6',
 'https://web.archive.org/web/20020308222536/http://www.chillingeffects.org/linking/faq.cgi#QID596',
 'http://www.chillingeffects.org/linking/faq.cgi#QID596',
 '#cite_ref-7',
 'http://scholarship.law.berkeley.edu/btlj/vol29/iss4/16/',
 '/wiki/Digital_object_identifier',
 '//doi.org/10.15779%2FZ38B39B',
 '/wiki/International_Standard_Serial_Number',
 '//www.worldcat.org/issn/1086-3818',
 '#cite_ref-8',
 'http://www.tomwbell.com/NetLaw/Ch06.html',
 '#cite_ref-9',
 'https://web.archive.org/web/20020308222536/http://www.chillingeffects.org/linking/faq.cgi#QID460',
 'http://www.chillingeffects.org/linking/faq.cgi#QID460',
 '#cite_ref-10',
 'http://www.tomwbell.com/NetLaw/Ch07/Ticketmaster.html',
 '#cite_ref-11',
 'https://web.archive.org/web/20110723131832/http://www.fornova.net/documents/AAFareChase.pdf',
 'http://www.fornova.net/documents/AAFareChase.pdf',
 '#cite_ref-12',
 'http://www.thefreelibrary.com/American+Airlines,+FareChase+Settle+Suit.-a0103213546',
 '#cite_ref-impervawp2011_13-0',
 'http://www.imperva.com/docs/WP_Detecting_and_Blocking_Site_Scraping_Attacks.pdf',
 '#cite_ref-14',
 'http://library.findlaw.com/2003/Jul/29/132944.html',
 '#cite_ref-15',
 'http://www.fornova.net/documents/Cvent.pdf',
 '#cite_ref-16',
 'https://www.scribd.com/doc/249068700/LinkedIn-v-Resultly-LLC-Complaint?secret_password=pEVKDbnvh
QL52oKfdrmT',
 '#cite_ref-17',
 'http://newmedialaw.proskauer.com/2014/12/05/qvc-sues-shopping-app-for-web-scraping-that-allegedly-trigger
ed-site-outage/',
```

'#cite_ref-18',
'http://www.fornova.net/documents/pblog-bna-com.pdf',
'#cite_ref-19',
'https://www.techdirt.com/articles/20090605/2228205147.shtml',
'#cite_ref-20',
'https://www.eff.org/cases/facebook-v-power-ventures',
'#cite_ref-21',
'https://web.archive.org/web/20071012005033/http://www.bvhd.dk/uploads/tx_mocarticles/S_-_og_Handelsrettens_afg_relse_i_Ofir-sagen.pdf',
'http://www.bvhd.dk/uploads/tx_mocarticles/S_-_og_Handelsrettens_afg_relse_i_Ofir-sagen.pdf',
'#cite_ref-22',
'http://www.bailii.org/ie/cases/IEHC/2010/H47.html',
'#cite_ref-23',
'http://www.lkshields.ie/htmdocs/publications/newsletters/update26/update26_03.htm',
'#cite_ref-24',
'https://www.lloyds.com/~/media/5880dae185914b2487bed7bd63b96286.ashx',
'#cite_ref-25',
'http://www.webstartdesign.com.au/spam_business_practical_guide.pdf',
'#cite_ref-26',
'https://s3.us-west-2.amazonaws.com/research-papers-mynk/Breaking-Fraud-And-Bot-Detection-Solutions.pdf',
'https://en.wikipedia.org/w/index.php?title=Web_scraping&oldid=825057514',
'/wiki/Help:Category',
'/wiki/Category:Web_scraping',
'/wiki/Category:World_Wide_Web',
'/wiki/Category:Spamming',
'/wiki/Category:CS1_Danish-language_sources_(da)',
'/wiki/Category:Articles_needing_additional_references_from_June_2017',
'/wiki/Category:All_articles_needing_additional_references',
'/wiki/Category:Articles_with_limited_geographic_scope_from_October_2015',
'/wiki/Category:USA-centric',
'/wiki/Category:Pages_using_div_col_with_deprecated_parameters',
'/wiki/Special:MyTalk',
'/wiki/Special:MyContributions',
'/w/index.php?title=Special:CreateAccount&returnto=Web+scraping',
'/w/index.php?title=Special:UserLogin&returnto=Web+scraping',
'/wiki/Web_scraping',
'/wiki/Talk:Web_scraping',
'/wiki/Web_scraping',
'/w/index.php?title=Web_scraping&action=edit',
'/w/index.php?title=Web_scraping&action=history',
'/wiki/Main_Page',
'/wiki/Main_Page',
'/wiki/Portal:Contents',
'/wiki/Portal:Featured_content',
'/wiki/Portal:Current_events',
'/wiki/Special:Random',
'https://donate.wikimedia.org/wiki/Special:FundraiserRedirector?utm_source=donate&utm_medium=sidebar&utm_campaign=C13_en.wikipedia.org&uselang=en',
'//shop.wikimedia.org',
'/wiki/Help:Contents',
'/wiki/Wikipedia:About',
'/wiki/Wikipedia:Community_portal',
'/wiki/Special:RecentChanges',
'//en.wikipedia.org/wiki/Wikipedia:Contact_us',
'/wiki/Special:WhatLinksHere/Web_scraping',
'/wiki/Special:RecentChangesLinked/Web_scraping',

'/wiki/Wikipedia:File_Upload_Wizard',
'/wiki/Special:SpecialPages',
'/w/index.php?title=Web_scraping&oldid=825057514',
'/w/index.php?title=Web_scraping&action=info',
'https://www.wikidata.org/wiki/Special:EntityPage/Q665452',
'/w/index.php?title=Special:CiteThisPage&page=Web_scraping&id=825057514',
'/w/index.php?title=Special:Book&bookcmd=book_creator&referer=Web+scraping',
'/w/index.php?title=Special:ElectronPdf&page=Web+scraping&action=show-download-screen',
'/w/index.php?title=Web_scraping&printable=yes',
'https://ar.wikipedia.org/wiki/%D8%A5%D8%B3%D8%AA%D8%AE%D9%84%D8%A7%D8%B5_%D8%A7%D9%84%D9%85%D9%88%D8%A7%D9%82%D8%B9',
'https://ca.wikipedia.org/wiki/Web_scraping',
'https://de.wikipedia.org/wiki/Screen_Scraping',
'https://es.wikipedia.org/wiki/Web_scraping',
'https://eu.wikipedia.org/wiki/Web_scraping',
'https://fr.wikipedia.org/wiki/Web_scraping',
'https://is.wikipedia.org/wiki/Vefs%C3%B6fnun',
'https://it.wikipedia.org/wiki/Web_scraping',
'https://lv.wikipedia.org/wiki/Rasmo%C5%A1ana',
'https://nl.wikipedia.org/wiki/Scrapen',
'https://ja.wikipedia.org/wiki/%E3%82%A6%E3%82%A7%E3%83%96%E3%82%B9%E3%82%AF%E3%83%AC%E3%82%A4%E3%83%94%E3%83%B3%E3%82%B0',
'https://sr.wikipedia.org/wiki/Web_scraping',
'https://tr.wikipedia.org/wiki/Web_kaz%C4%B1ma',
'https://uk.wikipedia.org/wiki/Web_scraping',
'https://zh.wikipedia.org/wiki/%E7%BD%91%E9%A1%B5%E6%8A%93%E5%8F%96',
'https://www.wikidata.org/wiki/Special:EntityPage/Q665452#sitelinks-wikipedia',
'//en.wikipedia.org/wiki/Wikipedia:Text_of_Creative_Commons_Attribution-ShareAlike_3.0_Unported_License',
'//creativecommons.org/licenses/by-sa/3.0/',
'//wikimediafoundation.org/wiki/Terms_of_Use',
'//wikimediafoundation.org/wiki/Privacy_policy',
'//www.wikimediafoundation.org/',
'https://wikimediafoundation.org/wiki/Privacy_policy',
'/wiki/Wikipedia:About',
'/wiki/Wikipedia:General_disclaimer',
'//en.wikipedia.org/wiki/Wikipedia:Contact_us',
'https://www.mediawiki.org/wiki/Special:MyLanguage/How_to_contribute',
'https://wikimediafoundation.org/wiki/Cookie_statement',
'//en.m.wikipedia.org/w/index.php?title=Web_scraping&mobileaction=toggle_view_mobile',
'https://wikimediafoundation.org/',
'//www.mediawiki.org/']

How to find all H1 tags?

In [15]:
```
soup.find_all('h1')
```

Out[15]: [<h1 class="firstHeading" id="firstHeading" lang="en">Web scraping</h1>]

How to filter out H1 Tag by class attribute?

In [16]:
```
soup.find_all('h1',class_='firstHeading',id="firstHeading")
```

Out[16]: [<h1 class="firstHeading" id="firstHeading" lang="en">Web scraping</h1>]

How to get all the Paragraph Text in the Website?

In [17]:
```
for paragraph in soup.find_all('p'):
    print(paragraph.text)
```

Web scraping, web harvesting, or web data extraction is data scraping used for extracting data from websites.[1] Web scraping software may access the World Wide Web directly using the Hypertext Transfer Protocol, or thro ugh a web browser. While web scraping can be done manually by a software user, the term typically refers to au tomated processes implemented using a bot or web crawler. It is a form of copying, in which specific data is gat hered and copied from the web, typically into a central local database or spreadsheet, for later retrieval or analys is.
Web scraping a web page involves fetching it and extracting from it.[1][2] Fetching is the downloading of a pag e (which a browser does when you view the page). Therefore, web crawling is a main component of web scrapi ng, to fetch pages for later processing. Once fetched, then extraction can take place. The content of a page may be parsed, searched, reformatted, its data copied into a spreadsheet, and so on. Web scrapers typically take some thing out of a page, to make use of it for another purpose somewhere else. An example would be to find and cop y names and phone numbers, or companies and their URLs, to a list (contact scraping).
Web scraping is used for contact scraping, and as a component of applications used for web indexing, web mini ng and data mining, online price change monitoring and price comparison, product review scraping (to watch th e competition), gathering real estate listings, weather data monitoring, website change detection, research, tracki ng online presence and reputation, web mashup and, web data integration.
Web pages are built using text-based mark-up languages (HTML and XHTML), and frequently contain a wealth of useful data in text form. However, most web pages are designed for human end-users and not for ease of aut omated use. Because of this, tool kits that scrape web content were created. A web scraper is an Application Pro gramming Interface (API) to extract data from a web site. Companies like Amazon AWS and Google provide w eb scraping tools, services and public data available free of cost to end users.
Newer forms of web scraping involve listening to data feeds from web servers. For example, JSON is commonl y used as a transport storage mechanism between the client and the web server.
There are methods that some websites use to prevent web scraping, such as detecting and disallowing bots from crawling (viewing) their pages. In response, there are web scraping systems that rely on using techniques in DO M parsing, computer vision and natural language processing to simulate human browsing to enable gathering w eb page content for offline parsing.

Web scraping is the process of automatically mining data or collecting information from the World Wide Web. It is a field with active developments sharing a common goal with the semantic web vision, an ambitious initiati ve that still requires breakthroughs in text processing, semantic understanding, artificial intelligence and human-computer interactions. Current web scraping solutions range from the ad-hoc, requiring human effort, to fully au tomated systems that are able to convert entire web sites into structured information, with limitations.
Sometimes even the best web-scraping technology cannot replace a human's manual examination and copy-and -paste, and sometimes this may be the only workable solution when the websites for scraping explicitly set up b arriers to prevent machine automation.
A simple yet powerful approach to extract information from web pages can be based on the UNIX grep comma nd or regular expression-matching facilities of programming languages (for instance Perl or Python).
Static and dynamic web pages can be retrieved by posting HTTP requests to the remote web server using socket programming.
Many websites have large collections of pages generated dynamically from an underlying structured source like a database. Data of the same category are typically encoded into similar pages by a common script or template.
In data mining, a program that detects such templates in a particular information source, extracts its content and translates it into a relational form, is called a wrapper. Wrapper generation algorithms assume that input pages o f a wrapper induction system conform to a common template and that they can be easily identified in terms of a URL common scheme.[3] Moreover, some semi-structured data query languages, such as XQuery and the HTQ L, can be used to parse HTML pages and to retrieve and transform page content.
By embedding a full-fledged web browser, such as the Internet Explorer or the Mozilla browser control, progra

ms can retrieve the dynamic content generated by client-side scripts. These browser controls also parse web pag es into a DOM tree, based on which programs can retrieve parts of the pages.

There are several companies that have developed vertical specific harvesting platforms. These platforms create and monitor a multitude of “bots” for specific verticals with no "man in the loop" (no direct human involvement), and no work related to a specific target site. The preparation involves establishing the knowledge base for the entire vertical and then the platform creates the bots automatically. The platform's robustness is measured by the quality of the information it retrieves (usually number of fields) and its scalability (how quick it can scale up to hundreds or thousands of sites). This scalability is mostly used to target the Long Tail of sites that common aggr egators find complicated or too labor-intensive to harvest content from.

The pages being scraped may embrace metadata or semantic markups and annotations, which can be used to loc ate specific data snippets. If the annotations are embedded in the pages, as Microformat does, this technique can be viewed as a special case of DOM parsing. In another case, the annotations, organized into a semantic layer,[4] are stored and managed separately from the web pages, so the scrapers can retrieve data schema and instructi ons from this layer before scraping the pages.

There are efforts using machine learning and computer vision that attempt to identify and extract information fr om web pages by interpreting pages visually as a human being might.[5]

There are many software tools available that can be used to customize web-scraping solutions. This software ma y attempt to automatically recognize the data structure of a page or provide a recording interface that removes th e necessity to manually write web-scraping code, or some scripting functions that can be used to extract and tra nsform content, and database interfaces that can store the scraped data in local databases. Some web scraping so ftware can also be used to extract data from an API directly.

These can be used to build web scrapers.

The legality of web scraping varies across the world. In general, web scraping may be against the terms of use o f some websites, but the enforceability of these terms is unclear.[6]

In the United States, website owners can use three major legal claims to prevent undesired web scraping: (1) co pyright infringement (compilation), (2) violation of the Computer Fraud and Abuse Act (“CFAA”), and (3) tres pass to chattel.[7] However, the effectiveness of these claims relies upon meeting various criteria, and the case l aw is still evolving. For example, with regard to copyright, while outright duplication of original expression will in many cases be illegal, in the United States the courts ruled in Feist Publications v. Rural Telephone Service t hat duplication of facts is allowable.

U.S. courts have acknowledged that users of "scrapers" or "robots" may be held liable for committing trespass t o chattels,[8][9] which involves a computer system itself being considered personal property upon which the us er of a scraper is trespassing. The best known of these cases, eBay v. Bidder's Edge, resulted in an injunction or dering Bidder's Edge to stop accessing, collecting, and indexing auctions from the eBay web site. This case invo lved automatic placing of bids, known as auction sniping. However, in order to succeed on a claim of trespass to chattels, the plaintiff must demonstrate that the defendant intentionally and without authorization interfered wit h the plaintiff's possessory interest in the computer system and that the defendant's unauthorized use caused da mage to the plaintiff. Not all cases of web spidering brought before the courts have been considered trespass to chattels.[10]

One of the first major tests of screen scraping involved American Airlines (AA), and a firm called FareChase.[1 1] AA successfully obtained an injunction from a Texas trial court, stopping FareChase from selling software th at enables users to compare online fares if the software also searches AA's website. The airline argued that Fare Chase's websearch software trespassed on AA's servers when it collected the publicly available data. FareChase filed an appeal in March 2003. By June, FareChase and AA agreed to settle and the appeal was dropped.[12]

Southwest Airlines has also challenged screen-scraping practices, and has involved both FareChase and another firm, Outtask, in a legal claim. Southwest Airlines charged that the screen-scraping is Illegal since it is an exam ple of "Computer Fraud and Abuse" and has led to "Damage and Loss" and "Unauthorized Access" of Southwes t's site. It also constitutes "Interference with Business Relations", "Trespass", and "Harmful Access by Compute r". They also claimed that screen-scraping constitutes what is legally known as "Misappropriation and Unjust E nrichment", as well as being a breach of the web site's user agreement. Outtask denied all these claims, claiming that the prevailing law in this case should be US Copyright law, and that under copyright, the pieces of informa tion being scraped would not be subject to copyright protection. Although the cases were never resolved in the Supreme Court of the United States, FareChase was eventually shuttered by parent company Yahoo!, and Outta sk was purchased by travel expense company Concur.[13] In 2012, a startup called 3Taps scraped classified hou sing ads from Craigslist. Craigslist sent 3Taps a cease-and-desist letter and blocked their IP addresses and later s ued, in Craigslist v. 3Taps. The court held that the cease-and-desist letter and IP blocking was sufficient for Crai gslist to properly claim that 3Taps had violated the Computer Fraud and Abuse Act.

Although these are early scraping decisions, and the theories of liability are not uniform, it is difficult to ignore a pattern emerging that the courts are prepared to protect proprietary content on commercial sites from uses which are undesirable to the owners of such sites. However, the degree of protection for such content is not settled, and will depend on the type of access made by the scraper, the amount of information accessed and copied, the degree to which the access adversely affects the site owner's system and the types and manner of prohibitions on such conduct.[14]

While the law in this area becomes more settled, entities contemplating using scraping programs to access a public web site should also consider whether such action is authorized by reviewing the terms of use and other terms or notices posted on or made available through the site. In a 2010 ruling in the Cvent, Inc. v. Eventbrite, Inc. In the United States district court for the eastern district of Virginia, the court ruled that the terms of use should be brought to the users' attention In order for a browse wrap contract or license to be enforced.[15] In a 2014, filed in the United States District Court for the Eastern District of Pennsylvania,[16] e-commerce site QVC objected to the Pinterest-like shopping aggregator Resultly's `scraping of QVC's site for real-time pricing data. QVC alleges that Resultly "excessively crawled" QVC's retail site (allegedly sending 200-300 search requests to QVC's website per minute, sometimes to up to 36,000 requests per minute) which caused QVC's site to crash for two days, resulting in lost sales for QVC.[17] QVC's complaint alleges that the defendant disguised its web crawler to mask its source IP address and thus prevented QVC from quickly repairing the problem. This is a particularly interesting scraping case because QVC is seeking damages for the unavailability of their website, which QVC claims was caused by Resultly.

In the plaintiff's web site during the period of this trial the terms of use link is displayed among all the links of the site, at the bottom of the page as most sites on the internet. This ruling contradicts the Irish ruling described below. The court also rejected the plaintiff's argument that the browse wrap restrictions were enforceable in view of Virginia's adoption of the Uniform Computer Information Transactions Act (UCITA)—a uniform law that many believed was in favor on common browse wrap contracting practices.[18]

In Facebook, Inc. v. Power Ventures, Inc., a district court ruled in 2012 that Power Ventures could not scrape Facebook pages on behalf of a Facebook user. The case is on appeal, and the Electronic Frontier Foundation filed a brief in 2015 asking that it be overturned.[19][20] In Associated Press v. Meltwater U.S. Holdings, Inc., a court in the US held Meltwater liable for scraping and republishing news information from the Associated Press, but a court in the United Kingdom held in favor of Meltwater.

Outside of the United States, in February 2006, the Danish Maritime and Commercial Court (Copenhagen) ruled that systematic crawling, indexing, and deep linking by portal site ofir.dk of estate site Home.dk does not conflict with Danish law or the database directive of the European Union.[21]

In a February 2010 case complicated by matters of jurisdiction, Ireland's High Court delivered a verdict that illustrates the inchoate state of developing case law. In the case of Ryanair Ltd v Billigfluege.de GmbH, Ireland's High Court ruled Ryanair's "click-wrap" agreement to be legally binding. In contrast to the findings of the United States District Court Eastern District of Virginia and those of the Danish Maritime and Commercial Court, Mr. Justice Michael Hanna ruled that the hyperlink to Ryanair's terms and conditions was plainly visible, and that placing the onus on the user to agree to terms and conditions in order to gain access to online services is sufficient to comprise a contractual relationship. [22] The decision is under appeal in Ireland's Supreme Court.[23]

In Australia, the Spam Act 2003 outlaws some forms of web harvesting, although this only applies to email addresses.[24][25]

The administrator of a website can use various measures to stop or slow a bot. Some techniques include:

How to get all the text in a webpage?

In [18]:
```
text=soup.get_text(strip=True)
print(text)
```

Web scraping - Wikipediadocument.documentElement.className = document.documentElement.className.replace(/(^|\s)client-nojs(\s|$)/, "$1client-js$2");(window.RLQ=window.RLQ||[]).push(function(){mw.config.set({"wgCanonicalNamespace":"","wgCanonicalSpecialPageName":false,"wgNamespaceNumber":0,"wgPageName":"Web_scraping","wgTitle":"Web scraping","wgCurRevisionId":825057514,"wgRevisionId":825057514,"wgArticleId":2696619,"wgIsArticle":true,"wgIsRedirect":false,"wgAction":"view","wgUserName":null,"wgUserGroups":["*"],"wgCategories":["CS1 Danish-language sources (da)","Articles needing additional references from June 2017","All articles needing additional references","Articles with limited geographic scope from Octob

er 2015","USA-centric","Pages using div col with deprecated parameters","Web scraping","World Wide Web","Spamming"],"wgBreakFrames":false,"wgPageContentLanguage":"en","wgPageContentModel":"wikitext","wgSeparatorTransformTable":["",""],"wgDigitTransformTable":["",""],"wgDefaultDateFormat":"dmy","wgMonthNames":["","January","February","March","April","May","June","July","August","September","October","November","December"],"wgMonthNamesShort":["","Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"],"wgRelevantPageName":"Web_scraping","wgRelevantArticleId":2696619,"wgRequestId":"WpOzcApAMFQAAKv1HOgAAABY","wgIsProbablyEditable":true,"wgRelevantPageIsProbablyEditable":true,"wgRestrictionEdit":[],"wgRestrictionMove":[],"wgFlaggedRevsParams":{"tags":{}},"wgStableRevisionId":null,"wgWikiEditorEnabledModules":[],"wgBetaFeaturesFeatures":[],"wgMediaViewerOnClick":true,"wgMediaViewerEnabledByDefault":true,"wgPopupsShouldSendModuleToUser":true,"wgPopupsConflictsWithNavPopupGadget":false,"wgVisualEditor":{"pageLanguageCode":"en","pageLanguageDir":"ltr","pageVariantFallbacks":"en","usePageImages":true,"usePageDescriptions":true},"wgPreferredVariant":"en","wgMFExpandAllSectionsUserOption":true,"wgMFEnableFontChanger":true,"wgMFDisplayWikibaseDescriptions":{"search":false,"nearby":false,"watchlist":false,"tagline":false},"wgRelatedArticles":null,"wgRelatedArticlesUseCirrusSearch":true,"wgRelatedArticlesOnlyUseCirrusSearch":false,"wgULSCurrentAutonym":"English","wgNoticeProject":"wikipedia","wgCentralNoticeCookiesToDelete":[],"wgCentralNoticeCategoriesUsingLegacy":["Fundraising","fundraising"],"wgCategoryTreePageCategoryOptions":"{\"mode\":0,\"hideprefix\":20,\"showcount\":true,\"namespaces\":false}","wgWikibaseItemId":"Q665452","wgScoreNoteLanguages":{"arabic":"العربية","catalan":"català","deutsch":"Deutsch","english":"English","espanol":"español","italiano":"italiano","nederlands":"Nederlands","norsk":"norsk","portugues":"português","suomi":"suomi","svenska":"svenska","vlaams":"West-Vlams"},"wgScoreDefaultNoteLanguage":"nederlands","wgCentralAuthMobileDomain":false,"wgCodeMirrorEnabled":false,"wgVisualEditorToolbarScrollOffset":0,"wgVisualEditorUnsupportedEditParams":["undo","undoafter","veswitched"],"wgEditSubmitButtonLabelPublish":true});mw.loader.state({"ext.gadget.charinsert-styles":"ready","ext.globalCssJs.user.styles":"ready","ext.globalCssJs.site.styles":"ready","site.styles":"ready","noscript":"ready","user.styles":"ready","user":"ready","user.options":"ready","user.tokens":"loading","ext.cite.styles":"ready","wikibase.client.init":"ready","ext.visualEditor.desktopArticleTarget.noscript":"ready","ext.uls.interlanguage":"ready","ext.wikimediaBadges":"ready","mediawiki.legacy.shared":"ready","mediawiki.legacy.commonPrint":"ready","mediawiki.sectionAnchor":"ready","mediawiki.skinning.interface":"ready","skins.vector.styles":"ready","ext.globalCssJs.user":"ready","ext.globalCssJs.site":"ready"});mw.loader.implement("user.tokens@1dqfd7l",function($,jQuery,require,module){/*@nomin*/mw.user.tokens.set({"editToken":"+\\","patrolToken":"+\\","watchToken":"+\\","csrfToken":"+\\"});
});mw.loader.load(["ext.cite.a11y","site","mediawiki.page.startup","mediawiki.user","mediawiki.hidpi","mediawiki.page.ready","mediawiki.toc","mediawiki.searchSuggest","ext.gadget.teahouse","ext.gadget.ReferenceTooltips","ext.gadget.watchlist-notice","ext.gadget.DRN-wizard","ext.gadget.charinsert","ext.gadget.refToolbar","ext.gadget.extra-toolbar-buttons","ext.gadget.switcher","ext.centralauth.centralautologin","mmv.head","mmv.bootstrap.autostart","ext.popups","ext.visualEditor.desktopArticleTarget.init","ext.visualEditor.targetLoader","ext.eventLogging.subscriber","ext.wikimediaEvents","ext.navigationTiming","ext.uls.eventlogger","ext.uls.init","ext.uls.interface","ext.3d","ext.centralNotice.geoIP","ext.centralNotice.startUp","skins.vector.js"]);});Web scrapingFrom Wikipedia, the free encyclopediaJump to:navigation,searchThis articleneeds additional citations forverification.Please helpimprove this articlebyadding citations to reliable sources. Unsourced material may be challenged and removed.(June 2017)(Learn how and when to remove this template message)For a broader coverage related to this topic, seeData scraping.Web scraping,web harvesting, orweb data extractionisdata scrapingused forextracting datafromwebsites.[1]Web scraping software may access the World Wide Web directly using theHypertext Transfer Protocol, or through a web browser. While web scraping can be done manually by a software user, the term typically refers to automated processes implemented using abotorweb crawler. It is a form of copying, in which specific data is gathered and copied from the web, typically into a central localdatabaseor spreadsheet, for laterretrievaloranalysis.Web scraping a web page involves fetching it and extracting from it.[1][2]Fetching is the downloading of a page (which a browser does when you view the page). Therefore, web crawling is a main component of web scraping, to fetch pages for later processing. Once fetched, then extraction can take place. The content of a page may beparsed, searched, reformatted, its data copied into a spreadsheet, and so on. Web scrapers typically take something out of a page, to make use of it for another purpose somewhere else. An example would be to find and copy names and phone numbers, or companies and their URLs, to a list (contact scraping).Web scraping is used forcontact scraping, and as a component of applications used forweb indexing,web mininganddata mining, online price change monitoring andprice comparison, product review scraping (to watch the competition), gathering real estate listings, weather data monitoring,website change detection, research, tracking online presence and reputation,web mashupand, web data integration.Web pagesare built using text-based mark-up languages (HTMLandXHTML), and frequently contain a wealth of useful data in text form. However, m

ost web pages are designed for humanend-usersand not for ease of automated use. Because of this, tool kits that scrape web content were created. A web scraper is anApplication Programming Interface(API) to extract data fr om a web site. Companies likeAmazon AWSandGoogleprovide web scraping tools, services and public data av ailable free of cost to end users.Newer forms of web scraping involve listening to data feeds from web servers. For example,JSONis commonly used as a transport storage mechanism between the client and the web server.T here are methods that some websites use to prevent web scraping, such as detecting and disallowing bots from c rawling (viewing) their pages. In response, there are web scraping systems that rely on using techniques inDOM parsing,computer visionandnatural language processingto simulate human browsing to enable gathering web pa ge content for offline parsing.Contents1Techniques1.1Human copy-and-paste1.2Text pattern matching1.3HTTP programming1.4HTML parsing1.5DOM parsing1.6Vertical aggregation1.7Semantic annotation recognizing1.8 Computer vision web-page analysis2Software2.1Example tools2.1.1Javascript tools2.1.2SaaS version2.1.3Web crawling frameworks3Legal issues3.1United States3.2The EU3.3Australia4Methods to prevent web scraping5S ee also6ReferencesTechniques[edit]Web scraping is the process of automatically mining data or collecting infor mation from the World Wide Web. It is a field with active developments sharing a common goal with theseman tic webvision, an ambitious initiative that still requires breakthroughs in text processing, semantic understandin g, artificial intelligence andhuman-computer interactions. Current web scraping solutions range from the ad-hoc , requiring human effort, to fully automated systems that are able to convert entire web sites into structured infor mation, with limitations.Human copy-and-paste[edit]Sometimes even the best web-scraping technology cannot replace a human’s manual examination and copy-and-paste, and sometimes this may be the only workable soluti on when the websites for scraping explicitly set up barriers to prevent machine automation.Text pattern matchin g[edit]A simple yet powerful approach to extract information from web pages can be based on the UNIXgrepco mmand orregular expression-matching facilities of programming languages (for instancePerlorPython).HTTP pr ogramming[edit]Staticanddynamic web pagescan be retrieved by posting HTTP requests to the remote web serv er usingsocket programming.HTML parsing[edit]Many websites have large collections of pages generated dyna mically from an underlying structured source like a database. Data of the same category are typically encoded in to similar pages by a common script or template. In data mining, a program that detects such templates in a parti cular information source, extracts its content and translates it into a relational form, is called awrapper. Wrapper generation algorithms assume that input pages of a wrapper induction system conform to a common template a nd that they can be easily identified in terms of a URL common scheme.[3]Moreover, somesemi-structured data query languages, such asXQueryand the HTQL, can be used to parse HTML pages and to retrieve and transfor m page content.DOM parsing[edit]Further information:Document Object ModelBy embedding a full-fledged w eb browser, such as theInternet Exploreror theMozillabrowser control, programs can retrieve the dynamic conte nt generated by client-side scripts. These browser controls also parse web pages into a DOM tree, based on whic h programs can retrieve parts of the pages.Vertical aggregation[edit]There are several companies that have deve loped vertical specific harvesting platforms. These platforms create and monitor a multitude of “bots” for specif ic verticals with no "man in the loop" (no direct human involvement), and no work related to a specific target sit e. The preparation involves establishing the knowledge base for the entire vertical and then the platform creates the bots automatically. The platform's robustness is measured by the quality of the information it retrieves (usua lly number of fields) and its scalability (how quick it can scale up to hundreds or thousands of sites). This scalab ility is mostly used to target theLong Tailof sites that common aggregators find complicated or too labor-intensi ve to harvest content from.Semantic annotation recognizing[edit]The pages being scraped may embracemetadat aor semantic markups and annotations, which can be used to locate specific data snippets. If the annotations are embedded in the pages, asMicroformatdoes, this technique can be viewed as a special case of DOM parsing. In another case, the annotations, organized into a semantic layer,[4]are stored and managed separately from the we b pages, so the scrapers can retrieve data schema and instructions from this layer before scraping the pages.Com puter vision web-page analysis[edit]There are efforts usingmachine learningandcomputer visionthat attempt to i dentify and extract information from web pages by interpreting pages visually as a human being might.[5]Softw are[edit]There are many software tools available that can be used to customize web-scraping solutions. This soft ware may attempt to automatically recognize the data structure of a page or provide a recording interface that re moves the necessity to manually write web-scraping code, or some scripting functions that can be used to extrac t and transform content, and database interfaces that can store the scraped data in local databases. Some web scr aping software can also be used to extract data from an API directly.Example tools[edit]cURL– command line t ool and library for transferring (including getting) data with URLs supporting a wide range of HTTP methods (GET, POST, cookies, etc.)Data Toolbar– web scraping add-on for Internet Explorer, Mozilla Firefox, and Goog le Chrome Web browsers that collects and converts structured data from web pages into a tabular format that ca n be loaded into a spreadsheet or database management program.Diffbot– uses computer vision and machine lea rning to automatically extract data from web pages by interpreting pages visually as a human being might.Heritr

ix– gets pages (lots of them). It is a web crawler designed for web archiving, written by the Internet Archive (se eWayback Machine).HtmlUnit– headless browser that can be used for retrieving web pages, web scraping, and more.HTTrack– free and open source Web crawler and offline browser, designed to download websites.iMacros –a browser extension to record, code, share and replay browser automation (javascript)Kantu – uses screenshots and OCR for scrapingSelenium (software)– a portable software-testing framework for web applicationsJaxerno kogiriOutWit Hub– Web scraping application including built-in data, image, document extractors and editors fo r custom scrapers and automatic exploration and extraction jobs (free and paid versions)watir–Wget– computer program that retrieves content from web servers. It is part of the GNU Project. It supports downloading via the HTTP, HTTPS, and FTP protocols.WSO2 Mashup Server–Yahoo! Query Language(YQL) –Data Scraping Stud io – Stand alone windows desktop software to scrape data using CSS selectors and REGEX.Javascript tools[edit]GreasemonkeyNode.jsPhantomJS– scripted,headless browserused for automating web page interaction.jQuery SaaS version[edit]Agenty – SaaS solution, paid versions available from $29 (06/09/17)Apify – Web scraping an d automation platform, free and paid versions available (10/10/17)dexi.io – SaaS solution, free and paid version s available from $119 USD (31/10/17)diggernaut.com– Turn websites into datasets, free and paid (from $9.99 U SD) subscriptions available (02/05/18)fScraper– Facebook friendly scraper, SaaS solution, free and paid version s availableImport.io– SaaS solution, paid versions available from $299 USD (06/09/17)Listly.io– HTML to Exc el in seconds, free SaaS service (04/10/17)Mozenda– SaaS solution, is a web-based platform for web data extrac tion (01/22/18)uScraper– SaaS service, free and paid versions available. Functionality primarily for scraping em ail addresses.Web crawling frameworks[edit]These can be used to build web scrapers.ScrapyLegal issues[edit]T he examples and perspective in this articledeal primarily with the United States and do not represent aworldwid e viewof the subject.You mayimprove this article, discuss the issue on thetalk page, orcreate a new article, as ap propriate.(October 2015)(Learn how and when to remove this template message)The legality of web scraping v aries across the world. In general, web scraping may be against theterms of useof some websites, but the enforc eability of these terms is unclear.[6]United States[edit]In the United States, website owners can use three majorl egal claimsto prevent undesired web scraping: (1) copyright infringement (compilation), (2) violation of theCo mputer Fraud and Abuse Act(“CFAA”), and (3)trespass to chattel.[7]However, the effectiveness of these claims relies upon meeting various criteria, and the case law is still evolving. For example, with regard to copyright, w hile outright duplication of original expression will in many cases be illegal, in the United States the courts rule d inFeist Publications v. Rural Telephone Servicethat duplication of facts is allowable.U.S. courts have acknowl edged that users of "scrapers" or "robots" may be held liable for committingtrespass to chattels,[8][9]which invo lves a computer system itself being considered personal property upon which the user of a scraper is trespassing . The best known of these cases,eBay v. Bidder's Edge, resulted in an injunction ordering Bidder's Edge to stop accessing, collecting, and indexing auctions from the eBay web site. This case involved automatic placing of bi ds, known asauction sniping. However, in order to succeed on a claim of trespass tochattels, theplaintiffmust de monstrate that thedefendantintentionally and without authorization interfered with the plaintiff's possessory inte rest in the computer system and that the defendant's unauthorized use caused damage to the plaintiff. Not all cas es of web spidering brought before the courts have been considered trespass to chattels.[10]One of the first majo r tests ofscreen scrapinginvolvedAmerican Airlines(AA), and a firm called FareChase.[11]AA successfully obta ined aninjunctionfrom a Texas trial court, stopping FareChase from selling software that enables users to compa re online fares if the software also searches AA's website. The airline argued that FareChase's websearch softwa re trespassed on AA's servers when it collected the publicly available data. FareChase filed an appeal in March 2003. By June, FareChase and AA agreed to settle and the appeal was dropped.[12]Southwest Airlineshas also c hallenged screen-scraping practices, and has involved both FareChase and another firm, Outtask, in a legal clai m. Southwest Airlines charged that the screen-scraping is Illegal since it is an example of "Computer Fraud and Abuse" and has led to "Damage and Loss" and "Unauthorized Access" of Southwest's site. It also constitutes "In terference with Business Relations", "Trespass", and "Harmful Access by Computer". They also claimed that sc reen-scraping constitutes what is legally known as "Misappropriation and Unjust Enrichment", as well as being a breach of the web site's user agreement. Outtask denied all these claims, claiming that the prevailing law in thi s case should beUS Copyright law, and that under copyright, the pieces of information being scraped would not be subject to copyright protection. Although the cases were never resolved in theSupreme Court of the United S tates, FareChase was eventually shuttered by parent companyYahoo!, and Outtask was purchased by travel expe nse company Concur.[13]In 2012, a startup called 3Taps scraped classified housing ads from Craigslist. Craigsli st sent 3Taps a cease-and-desist letter and blocked their IP addresses and later sued, inCraigslist v. 3Taps. The c ourt held that the cease-and-desist letter and IP blocking was sufficient for Craigslist to properly claim that 3Ta ps had violated theComputer Fraud and Abuse Act.Although these are early scraping decisions, and the theories of liability are not uniform, it is difficult to ignore a pattern emerging that the courts are prepared to protect pro prietary content on commercial sites from uses which are undesirable to the owners of such sites. However, the

degree of protection for such content is not settled, and will depend on the type of access made by the scraper, t he amount of information accessed and copied, the degree to which the access adversely affects the site owner's system and the types and manner of prohibitions on such conduct.[14]While the law in this area becomes more settled, entities contemplating using scraping programs to access a public web site should also consider whether such action is authorized by reviewing the terms of use and other terms or notices posted on or made available t hrough the site. In a 2010 ruling in theCvent, Inc.v.Eventbrite, Inc.In the United States district court for the east ern district of Virginia, the court ruled that the terms of use should be brought to the users' attention In order for abrowse wrapcontract or license to be enforced.[15]In a 2014, filed in theUnited States District Court for the Ea stern District of Pennsylvania,[16]e-commerce siteQVCobjected to the Pinterest-like shopping aggregator Resul tly's `scraping of QVC's site for real-time pricing data. QVC alleges that Resultly "excessively crawled" QVC' s retail site (allegedly sending 200-300 search requests to QVC's website per minute, sometimes to up to 36,000 requests per minute) which caused QVC's site to crash for two days, resulting in lost sales for QVC.[17]QVC's complaint alleges that the defendant disguised its web crawler to mask its source IP address and thus prevented QVC from quickly repairing the problem. This is a particularly interesting scraping case because QVC is seekin g damages for the unavailability of their website, which QVC claims was caused by Resultly.In the plaintiff's w eb site during the period of this trial the terms of use link is displayed among all the links of the site, at the botto m of the page as most sites on the internet. This ruling contradicts the Irish ruling described below. The court al so rejected the plaintiff's argument that the browse wrap restrictions were enforceable in view of Virginia's adop tion of the Uniform Computer Information Transactions Act (UCITA)—a uniform law that many believed was i n favor on common browse wrap contracting practices.[18]InFacebook, Inc. v. Power Ventures, Inc., a district c ourt ruled in 2012 that Power Ventures could not scrape Facebook pages on behalf of a Facebook user. The case is on appeal, and theElectronic Frontier Foundationfiled a brief in 2015 asking that it be overturned.[19][20]In Associated Press v. Meltwater U.S. Holdings, Inc., a court in the US held Meltwater liable for scraping and repu blishing news information from the Associated Press, but a court in the United Kingdom held in favor of Meltw ater.The EU[edit]Outside of the United States, in February 2006, the Danish Maritime and Commercial Court (Copenhagen) ruled that systematic crawling, indexing, and deep linking by portal site ofir.dk of estate site Hom e.dk does not conflict with Danish law or the database directive of the European Union.[21]In a February 2010 case complicated by matters of jurisdiction, Ireland's High Court delivered a verdict that illustrates theinchoatest ate of developing case law. In the case ofRyanair Ltd v Billigfluege.de GmbH, Ireland's High Court ruled Ryana ir's "click-wrap" agreement to be legally binding. In contrast to the findings of the United States District Court Eastern District of Virginia and those of the Danish Maritime and Commercial Court, Mr. Justice Michael Hann a ruled that the hyperlink to Ryanair's terms and conditions was plainly visible, and that placing the onus on the user to agree to terms and conditions in order to gain access to online services is sufficient to comprise a contrac tual relationship.[22]The decision is under appeal in Ireland's Supreme Court.[23]Australia[edit]In Australia, th eSpam Act 2003outlaws some forms of web harvesting, although this only applies to email addresses.[24][25]M ethods to prevent web scraping[edit]The administrator of a website can use various measures to stop or slow a b ot. Some techniques include:Blocking anIP addresseither manually or based on criteria such asgeolocationandD NSRBL. This will also block all browsing from that address.Disabling anyweb serviceAPIthat the website's syst em might expose.Bots sometimes declare who they are (usinguser agentstrings) and can be blocked on that basi s usingrobots.txt; 'googlebot' is an example. Other bots make no distinction between themselves and a human us ing a browser.Bots can be blocked by monitoring excess trafficBots can sometimes be blocked with tools to veri fy that it is a real person accessing the site, like aCAPTCHA. Bots are sometimes coded to explicitly break spec ific CAPTCHA patterns or may employ third-party services that utilize human labor to read and respond in real- time to CAPTCHA challenges.Commercial anti-bot services: Companies offer anti-bot and anti-scraping servic es for websites. A few webapplication firewallshave limited bot detection capabilities as well. However, many s uch solutions are not very effective[26].Locating bots with ahoneypotor other method to identify the IP addresse s of automated crawlers.ObfuscationusingCSS spritesto display such data as phone numbers or email addresses, at the cost ofaccessibilitytoscreen readerusers.Because bots rely on consistency in the front-end code of a target website, adding small variations to the HTML/CSS surrounding important data and navigation elements would r equire more human involvement in the initial set up of a bot and if done effectively may render the target websit e too difficult to scrape due to the diminished ability to automate the scraping process.Websites can declare if cr awling is allowed or not in therobots.txtfile and allow partial access, limit the crawl rate, specify the optimal tim e to crawl and more.See also[edit]Archive.isComparison of feed aggregatorsData scrapingData wranglingImpor terJob wrappingKnowledge extractionOpenSocialScraper siteFake news websiteBlog scrapingSpamdexingDom ain name drop listText corpusWeb archivingBlog networkSearch Engine ScrapingWeb crawlersReferences[edit]^abBoeing, G.; Waddell, P. (2016). "New Insights into Rental Housing Markets across the United States: Web Scraping and Analyzing Craigslist Rental Listings".Journal of Planning Education and Research(0739456X166

64789).doi:10.1177/0739456X16664789.^Vargiu & Urru (2013). "Exploiting web scraping in a collaborative fil tering- based approach to web advertising".Artificial Intelligence Research.2(1).doi:10.5430/air.v2n1p44.^Song , Ruihua; Microsoft Research (Sep 14, 2007)."Joint Optimization of Wrapper Generation and Template Detectio n"(PDF).The 13th International Conference on Knowledge Discovery and Data Mining.^Semantic annotation b ased web scraping^Roush, Wade (2012-07-25)."Diffbot Is Using Computer Vision to Reinvent the Semantic W eb". www.xconomy.com. Retrieved2013-03-15.^"FAQ about linking – Are website terms of use binding contra cts?". www.chillingeffects.org. 2007-08-20. Archived fromthe originalon 2002-03-08. Retrieved2007-08-20.^K enneth, Hirschey, Jeffrey (2014-01-01)."Symbiotic Relationships: Pragmatic Acceptance of Data Scraping".Ber keley Technology Law Journal.29(4).doi:10.15779/Z38B39B.ISSN1086-3818.^"Internet Law, Ch. 06: Trespass to Chattels". www.tomwbell.com. 2007-08-20. Retrieved2007-08-20.^"What are the "trespass to chattels" clai ms some companies or website owners have brought?". www.chillingeffects.org. 2007-08-20. Archived fromthe originalon 2002-03-08. Retrieved2007-08-20.^"Ticketmaster Corp. v. Tickets.com, Inc."2007-08-20. Retrieved 2007-08-20.^"American Airlines v. FareChase"(PDF). 2007-08-20. Archived fromthe original(PDF)on 2011-07 -23. Retrieved2007-08-20.^"American Airlines, FareChase Settle Suit". The Free Library. 2003-06-13. Retrieve d2012-02-26.^Imperva (2011).Detecting and Blocking Site Scraping Attacks. Imperva white paper..^Adler, Ken neth A. (2003-07-29)."Controversy Surrounds 'Screen Scrapers': Software Helps Users Access Web Sites But A ctivity by Competitors Comes Under Scrutiny". Retrieved2010-10-27.^"QVC Inc. v. Resultly LLC, No. 14-067 14 (E.D. Pa. filed Nov. 24, 2014)"(PDF). 2014-11-24. Retrieved2015-11-05.^"QVC Inc. v. Resultly LLC, No. 1 4-06714 (E.D. Pa. filed Nov. 24, 2014)".United States District Court for the Eastern District of Pennsylvania. R etrieved5 November2015.^Neuburger, Jeffrey D (5 December 2014)."QVC Sues Shopping App for Web Scrapi ng That Allegedly Triggered Site Outage".The National Law Review. Proskauer Rose LLP. Retrieved5 Novem ber2015.^"Did Iqbal/Twombly Raise the Bar for Browsewrap Claims?"(PDF). 2010-09-17. Retrieved2010-10-2 7.^"Can Scraping Non-Infringing Content Become Copyright Infringement... Because Of How Scrapers Work? | Techdirt".Techdirt. 2009-06-10. Retrieved2016-05-24.^"Facebook v. Power Ventures".Electronic Frontier Fou ndation. Retrieved2016-05-24.^"UDSKRIFT AF SØ- & HANDELSRETTENS DOMBOG"(PDF)(in Danish). b vhd.dk. 2006-02-24. Archived fromthe original(PDF)on 2007-10-12. Retrieved2007-05-30.^"High Court of Irel and Decisions >> Ryanair Ltd -v- Billigfluege.de GMBH 2010 IEHC 47 (26 February 2010)". British and Irish Legal Information Institute. 2010-02-26. Retrieved2012-04-19.^Matthews, Áine (June 2010)."Intellectual Prope rty: Website Terms of Use".Issue 26: June 2010. LK Shields Solicitors Update. p. 03. Retrieved2012-04-19.^Na tional Office for the Information Economy (February 2004)."Spam Act 2003: An overview for business". Austr alian Communications Authority. p. 6. Retrieved2017-12-07.^National Office for the Information Economy (Fe bruary 2004)."Spam Act 2003: A practical guide for business"(PDF). Australian Communications Authority. p. 20. Retrieved2017-12-07.^Mayank DhimanBreaking Fraud & Bot Detection SolutionsOWASP AppSec Cali' 20 18Retrieved February 10, 2018.Retrieved from "https://en.wikipedia.org/w/index.php?title=Web_scraping&oldi d=825057514"Categories:Web scrapingWorld Wide WebSpammingHidden categories:CS1 Danish-language so urces (da)Articles needing additional references from June 2017All articles needing additional referencesArticle s with limited geographic scope from October 2015USA-centricPages using div col with deprecated parameters Navigation menuPersonal toolsNot logged inTalkContributionsCreate accountLog inNamespacesArticleTalkVa riantsViewsReadEditView historyMoreSearchNavigationMain pageContentsFeatured contentCurrent eventsRan dom articleDonate to WikipediaWikipedia storeInteractionHelpAbout WikipediaCommunity portalRecent chan gesContact pageToolsWhat links hereRelated changesUpload fileSpecial pagesPermanent linkPage information Wikidata itemCite this pagePrint/exportCreate a bookDownload as PDFPrintable versionLanguagesالعربيةCatalà DeutschEspañolEuskaraFrançaisÍslenskaItalianoLatviešuNederlands日本語Српски / srpskiTürkçeУкраїнська 中文Edit linksThis page was last edited on 11 February 2018, at 06:38.Text is available under theCreative Com mons Attribution-ShareAlike License;
additional terms may apply. By using this site, you agree to theTerms of UseandPrivacy Policy. Wikipedia® is a registered trademark of theWikimedia Foundation, Inc., a non-profit organization.Privacy policyAbout Wikip ediaDisclaimersContact WikipediaDevelopersCookie statementMobile view(window.RLQ=window.RLQ||[]).p ush(function(){mw.config.set({"wgPageParseReport":{"limitreport":{"cputime":"0.228","walltime":"0.285","p pvisitednodes":{"value":1447,"limit":1000000},"ppgeneratednodes":{"value":0,"limit":1500000},"postexpandi ncludesize":{"value":49571,"limit":2097152},"templateargumentsize":{"value":320,"limit":2097152},"expansi ondepth":{"value":10,"limit":40},"expensivefunctioncount":{"value":3,"limit":500},"entityaccesscount":{"valu e":0,"limit":400},"timingprofile":["100.00% 240.818 1 -total"," 59.86% 144.146 1 Template:Reflist"," 2 8.76% 69.247 18 Template:Cite_web"," 19.28% 46.421 5 Template:Cite_journal"," 17.92% 43.157 1 Template:More_citations_needed"," 13.68% 32.949 2 Template:Ambox"," 4.68% 11.269 1 Templat e:US-centric"," 4.39% 10.578 1 Template:Further_information"," 3.99% 9.620 1 Template:Colbegin" ," 3.20% 7.700 1 Template:Globalize"]},"scribunto":{"limitreport-timeusage":{"value":"0.119","limit":"1

0.000"},"limitreport-memusage":{"value":4752818,"limit":52428800}},"cachereport":{"origin":"mw1249","timestamp":"20180226071249","ttl":1900800,"transientcontent":false}}});});(window.RLQ=window.RLQ||[]).push(function(){mw.config.set({"wgBackendResponseTime":374,"wgHostname":"mw1249"});});

In [60]:
```
type(text)
```

Out[60]: str

In [19]:
```
text.find('scraping')
```

Out[19]: 4

In [20]:
```
soup.find_all('pre')
```

Out[20]: []

How to extract image from Website?

In [21]:
```
# This will return all the images in the page in a list format
soup.find_all('img')
```

Out[21]: [,
,
,
,
]

In [22]:
```
# Get Link of image and concatinate with domain
link_list=[]
for img in soup.find_all('img',alt="Wikimedia Foundation"):
    print('https://en.wikipedia.org/'+img.get('src'))
    link_list.append(('https://en.wikipedia.org/'+img.get('src')))
```

https://en.wikipedia.org//static/images/wikimedia-button.png

In [23]:
```
# Use urllib.request.urlretrieve to retrive image from URL
print(link_list)
for link in link_list:
    image_name = link.split('/')[-1]
    print(image_name)
    if image_name.split('.')[-1] in ['png','jpg','psd']:
        urllib2.urlretrieve(link, image_name)
        print(link,image_name)
```

```
['https://en.wikipedia.org//static/images/wikimedia-button.png']
wikimedia-button.png
https://en.wikipedia.org//static/images/wikimedia-button.png wikimedia-button.png
```

In [24]:
```
pwd
```

Out[24]:
```
'C:\\Users\\murugnav\\Web Scraping'
```

How to extract table data?

In [25]:
```
for table in soup.find_all('table'):

    for rows in table.find_all('tr'):

        for data in rows.find_all('td'):
            print(data.text)
```

This article needs additional citations for verification. Please help improve this article by adding citations to reli able sources. Unsourced material may be challenged and removed. (June 2017) (Learn how and when to remove this template message)

The examples and perspective in this article deal primarily with the United States and do not represent a worldw ide view of the subject. You may improve this article, discuss the issue on the talk page, or create a new article, as appropriate. (October 2015) (Learn how and when to remove this template message)

How to grab meta tag information of a webpage?

In [26]:
```
for meta in soup.find_all('meta'):
    print(meta)
```

```
<meta charset="utf-8"/>
<meta content="" name="ResourceLoaderDynamicStyles"/>
<meta content="MediaWiki 1.31.0-wmf.22" name="generator"/>
<meta content="origin" name="referrer"/>
<meta content="origin-when-crossorigin" name="referrer"/>
<meta content="origin-when-cross-origin" name="referrer"/>
```

In [27]:
```
soup.nav
```

How to grab text from only body tag?

In [28]:
```
body_extract = soup.body
for paragraph in body_extract.find_all('p'):
```

```
    print(paragraph.text)
```

Web scraping, web harvesting, or web data extraction is data scraping used for extracting data from websites.[1] Web scraping software may access the World Wide Web directly using the Hypertext Transfer Protocol, or thro ugh a web browser. While web scraping can be done manually by a software user, the term typically refers to au tomated processes implemented using a bot or web crawler. It is a form of copying, in which specific data is gat hered and copied from the web, typically into a central local database or spreadsheet, for later retrieval or analys is.
Web scraping a web page involves fetching it and extracting from it.[1][2] Fetching is the downloading of a pag e (which a browser does when you view the page). Therefore, web crawling is a main component of web scrapi ng, to fetch pages for later processing. Once fetched, then extraction can take place. The content of a page may be parsed, searched, reformatted, its data copied into a spreadsheet, and so on. Web scrapers typically take some thing out of a page, to make use of it for another purpose somewhere else. An example would be to find and cop y names and phone numbers, or companies and their URLs, to a list (contact scraping).
Web scraping is used for contact scraping, and as a component of applications used for web indexing, web mini ng and data mining, online price change monitoring and price comparison, product review scraping (to watch th e competition), gathering real estate listings, weather data monitoring, website change detection, research, tracki ng online presence and reputation, web mashup and, web data integration.
Web pages are built using text-based mark-up languages (HTML and XHTML), and frequently contain a wealth of useful data in text form. However, most web pages are designed for human end-users and not for ease of aut omated use. Because of this, tool kits that scrape web content were created. A web scraper is an Application Pro gramming Interface (API) to extract data from a web site. Companies like Amazon AWS and Google provide w eb scraping tools, services and public data available free of cost to end users.
Newer forms of web scraping involve listening to data feeds from web servers. For example, JSON is commonl y used as a transport storage mechanism between the client and the web server.
There are methods that some websites use to prevent web scraping, such as detecting and disallowing bots from crawling (viewing) their pages. In response, there are web scraping systems that rely on using techniques in DO M parsing, computer vision and natural language processing to simulate human browsing to enable gathering w eb page content for offline parsing.

Web scraping is the process of automatically mining data or collecting information from the World Wide Web. It is a field with active developments sharing a common goal with the semantic web vision, an ambitious initiati ve that still requires breakthroughs in text processing, semantic understanding, artificial intelligence and human-computer interactions. Current web scraping solutions range from the ad-hoc, requiring human effort, to fully au tomated systems that are able to convert entire web sites into structured information, with limitations.
Sometimes even the best web-scraping technology cannot replace a human’s manual examination and copy-and -paste, and sometimes this may be the only workable solution when the websites for scraping explicitly set up b arriers to prevent machine automation.
A simple yet powerful approach to extract information from web pages can be based on the UNIX grep comma nd or regular expression-matching facilities of programming languages (for instance Perl or Python).
Static and dynamic web pages can be retrieved by posting HTTP requests to the remote web server using socket programming.
Many websites have large collections of pages generated dynamically from an underlying structured source like a database. Data of the same category are typically encoded into similar pages by a common script or template. In data mining, a program that detects such templates in a particular information source, extracts its content and translates it into a relational form, is called a wrapper. Wrapper generation algorithms assume that input pages o f a wrapper induction system conform to a common template and that they can be easily identified in terms of a URL common scheme.[3] Moreover, some semi-structured data query languages, such as XQuery and the HTQ L, can be used to parse HTML pages and to retrieve and transform page content.
By embedding a full-fledged web browser, such as the Internet Explorer or the Mozilla browser control, progra ms can retrieve the dynamic content generated by client-side scripts. These browser controls also parse web pag es into a DOM tree, based on which programs can retrieve parts of the pages.
There are several companies that have developed vertical specific harvesting platforms. These platforms create and monitor a multitude of “bots” for specific verticals with no "man in the loop" (no direct human involvement), and no work related to a specific target site. The preparation involves establishing the knowledge base for the entire vertical and then the platform creates the bots automatically. The platform's robustness is measured by the

quality of the information it retrieves (usually number of fields) and its scalability (how quick it can scale up to hundreds or thousands of sites). This scalability is mostly used to target the Long Tail of sites that common aggr egators find complicated or too labor-intensive to harvest content from.

The pages being scraped may embrace metadata or semantic markups and annotations, which can be used to loc ate specific data snippets. If the annotations are embedded in the pages, as Microformat does, this technique can be viewed as a special case of DOM parsing. In another case, the annotations, organized into a semantic layer,[4] are stored and managed separately from the web pages, so the scrapers can retrieve data schema and instructi ons from this layer before scraping the pages.

There are efforts using machine learning and computer vision that attempt to identify and extract information fr om web pages by interpreting pages visually as a human being might.[5]

There are many software tools available that can be used to customize web-scraping solutions. This software ma y attempt to automatically recognize the data structure of a page or provide a recording interface that removes th e necessity to manually write web-scraping code, or some scripting functions that can be used to extract and tra nsform content, and database interfaces that can store the scraped data in local databases. Some web scraping so ftware can also be used to extract data from an API directly.

These can be used to build web scrapers.

The legality of web scraping varies across the world. In general, web scraping may be against the terms of use o f some websites, but the enforceability of these terms is unclear.[6]

In the United States, website owners can use three major legal claims to prevent undesired web scraping: (1) co pyright infringement (compilation), (2) violation of the Computer Fraud and Abuse Act (“CFAA”), and (3) tres pass to chattel.[7] However, the effectiveness of these claims relies upon meeting various criteria, and the case l aw is still evolving. For example, with regard to copyright, while outright duplication of original expression will in many cases be illegal, in the United States the courts ruled in Feist Publications v. Rural Telephone Service t hat duplication of facts is allowable.

U.S. courts have acknowledged that users of "scrapers" or "robots" may be held liable for committing trespass t o chattels,[8][9] which involves a computer system itself being considered personal property upon which the us er of a scraper is trespassing. The best known of these cases, eBay v. Bidder's Edge, resulted in an injunction or dering Bidder's Edge to stop accessing, collecting, and indexing auctions from the eBay web site. This case invo lved automatic placing of bids, known as auction sniping. However, in order to succeed on a claim of trespass to chattels, the plaintiff must demonstrate that the defendant intentionally and without authorization interfered wit h the plaintiff's possessory interest in the computer system and that the defendant's unauthorized use caused da mage to the plaintiff. Not all cases of web spidering brought before the courts have been considered trespass to chattels.[10]

One of the first major tests of screen scraping involved American Airlines (AA), and a firm called FareChase.[1 1] AA successfully obtained an injunction from a Texas trial court, stopping FareChase from selling software th at enables users to compare online fares if the software also searches AA's website. The airline argued that Fare Chase's websearch software trespassed on AA's servers when it collected the publicly available data. FareChase filed an appeal in March 2003. By June, FareChase and AA agreed to settle and the appeal was dropped.[12]

Southwest Airlines has also challenged screen-scraping practices, and has involved both FareChase and another firm, Outtask, in a legal claim. Southwest Airlines charged that the screen-scraping is Illegal since it is an exam ple of "Computer Fraud and Abuse" and has led to "Damage and Loss" and "Unauthorized Access" of Southwes t's site. It also constitutes "Interference with Business Relations", "Trespass", and "Harmful Access by Compute r". They also claimed that screen-scraping constitutes what is legally known as "Misappropriation and Unjust E nrichment", as well as being a breach of the web site's user agreement. Outtask denied all these claims, claiming that the prevailing law in this case should be US Copyright law, and that under copyright, the pieces of informa tion being scraped would not be subject to copyright protection. Although the cases were never resolved in the Supreme Court of the United States, FareChase was eventually shuttered by parent company Yahoo!, and Outta sk was purchased by travel expense company Concur.[13] In 2012, a startup called 3Taps scraped classified hou sing ads from Craigslist. Craigslist sent 3Taps a cease-and-desist letter and blocked their IP addresses and later s ued, in Craigslist v. 3Taps. The court held that the cease-and-desist letter and IP blocking was sufficient for Crai gslist to properly claim that 3Taps had violated the Computer Fraud and Abuse Act.

Although these are early scraping decisions, and the theories of liability are not uniform, it is difficult to ignore a pattern emerging that the courts are prepared to protect proprietary content on commercial sites from uses whi ch are undesirable to the owners of such sites. However, the degree of protection for such content is not settled, and will depend on the type of access made by the scraper, the amount of information accessed and copied, the degree to which the access adversely affects the site owner’s system and the types and manner of prohibitions o n such conduct.[14]

While the law in this area becomes more settled, entities contemplating using scraping programs to access a pub lic web site should also consider whether such action is authorized by reviewing the terms of use and other term s or notices posted on or made available through the site. In a 2010 ruling in the Cvent, Inc. v. Eventbrite, Inc. I n the United States district court for the eastern district of Virginia, the court ruled that the terms of use should b e brought to the users' attention In order for a browse wrap contract or license to be enforced.[15] In a 2014, file d in the United States District Court for the Eastern District of Pennsylvania,[16] e-commerce site QVC objecte d to the Pinterest-like shopping aggregator Resultly's `scraping of QVC's site for real-time pricing data. QVC al leges that Resultly "excessively crawled" QVC's retail site (allegedly sending 200-300 search requests to QVC' s website per minute, sometimes to up to 36,000 requests per minute) which caused QVC's site to crash for two days, resulting in lost sales for QVC.[17] QVC's complaint alleges that the defendant disguised its web crawler t o mask its source IP address and thus prevented QVC from quickly repairing the problem. This is a particularly interesting scraping case because QVC is seeking damages for the unavailability of their website, which QVC cl aims was caused by Resultly.
In the plaintiff's web site during the period of this trial the terms of use link is displayed among all the links of t he site, at the bottom of the page as most sites on the internet. This ruling contradicts the Irish ruling described b elow. The court also rejected the plaintiff's argument that the browse wrap restrictions were enforceable in view of Virginia's adoption of the Uniform Computer Information Transactions Act (UCITA)—a uniform law that m any believed was in favor on common browse wrap contracting practices.[18]
In Facebook, Inc. v. Power Ventures, Inc., a district court ruled in 2012 that Power Ventures could not scrape F acebook pages on behalf of a Facebook user. The case is on appeal, and the Electronic Frontier Foundation filed a brief in 2015 asking that it be overturned.[19][20] In Associated Press v. Meltwater U.S. Holdings, Inc., a cou rt in the US held Meltwater liable for scraping and republishing news information from the Associated Press, bu t a court in the United Kingdom held in favor of Meltwater.
Outside of the United States, in February 2006, the Danish Maritime and Commercial Court (Copenhagen) rule d that systematic crawling, indexing, and deep linking by portal site ofir.dk of estate site Home.dk does not conf lict with Danish law or the database directive of the European Union.[21]
In a February 2010 case complicated by matters of jurisdiction, Ireland's High Court delivered a verdict that illu strates the inchoate state of developing case law. In the case of Ryanair Ltd v Billigfluege.de GmbH, Ireland's H igh Court ruled Ryanair's "click-wrap" agreement to be legally binding. In contrast to the findings of the United States District Court Eastern District of Virginia and those of the Danish Maritime and Commercial Court, Mr. Justice Michael Hanna ruled that the hyperlink to Ryanair's terms and conditions was plainly visible, and that pl acing the onus on the user to agree to terms and conditions in order to gain access to online services is sufficient to comprise a contractual relationship. [22] The decision is under appeal in Ireland's Supreme Court.[23]
In Australia, the Spam Act 2003 outlaws some forms of web harvesting, although this only applies to email addr esses.[24][25]
The administrator of a website can use various measures to stop or slow a bot. Some techniques include:

How to get list items?

In [29]:
```
for list_item in soup.find_all('li'):
    print(list_item.text)
```

1 Techniques

1.1 Human copy-and-paste
1.2 Text pattern matching
1.3 HTTP programming
1.4 HTML parsing
1.5 DOM parsing
1.6 Vertical aggregation
1.7 Semantic annotation recognizing
1.8 Computer vision web-page analysis

1.1 Human copy-and-paste

2.1 Example tools

2.1.1 Javascript tools
2.1.2 SaaS version
2.1.3 Web crawling frameworks

3.1 United States
3.2 The EU
3.3 Australia

cURL – command line tool and library for transferring (including getting) data with URLs supporting a wide ra nge of HTTP methods (GET, POST, cookies, etc.)
Data Toolbar – web scraping add-on for Internet Explorer, Mozilla Firefox, and Google Chrome Web browsers that collects and converts structured data from web pages into a tabular format that can be loaded into a spreads heet or database management program.
Diffbot – uses computer vision and machine learning to automatically extract data from web pages by interpreti ng pages visually as a human being might.
Heritrix – gets pages (lots of them). It is a web crawler designed for web archiving, written by the Internet Archi ve (see Wayback Machine).
HtmlUnit – headless browser that can be used for retrieving web pages, web scraping, and more.
HTTrack – free and open source Web crawler and offline browser, designed to download websites.
iMacros –a browser extension to record, code, share and replay browser automation (javascript)
Kantu – uses screenshots and OCR for scraping
Selenium (software) – a portable software-testing framework for web applications
Jaxer
nokogiri

OutWit Hub – Web scraping application including built-in data, image, document extractors and editors for cust om scrapers and automatic exploration and extraction jobs (free and paid versions)
watir –
Wget – computer program that retrieves content from web servers. It is part of the GNU Project. It supports do wnloading via the HTTP, HTTPS, and FTP protocols.
WSO2 Mashup Server –
Yahoo! Query Language (YQL) –
Data Scraping Studio – Stand alone windows desktop software to scrape data using CSS selectors and REGEX.
Greasemonkey
Node.js
PhantomJS – scripted, headless browser used for automating web page interaction.
jQuery
Agenty – SaaS solution, paid versions available from $29 (06/09/17)
Apify – Web scraping and automation platform, free and paid versions available (10/10/17)
dexi.io – SaaS solution, free and paid versions available from $119 USD (31/10/17)
diggernaut.com – Turn websites into datasets, free and paid (from $9.99 USD) subscriptions available (02/05/18)
fScraper – Facebook friendly scraper, SaaS solution, free and paid versions available
Import.io – SaaS solution, paid versions available from $299 USD (06/09/17)
Listly.io – HTML to Excel in seconds, free SaaS service (04/10/17)
Mozenda – SaaS solution, is a web-based platform for web data extraction (01/22/18)
uScraper – SaaS service, free and paid versions available. Functionality primarily for scraping email addresses.
Scrapy
Blocking an IP address either manually or based on criteria such as geolocation and DNSRBL. This will also bl ock all browsing from that address.
Disabling any web service API that the website's system might expose.
Bots sometimes declare who they are (using user agent strings) and can be blocked on that basis using robots.txt ; 'googlebot' is an example. Other bots make no distinction between themselves and a human using a browser.
Bots can be blocked by monitoring excess traffic
Bots can sometimes be blocked with tools to verify that it is a real person accessing the site, like a CAPTCHA. Bots are sometimes coded to explicitly break specific CAPTCHA patterns or may employ third-party services t hat utilize human labor to read and respond in real-time to CAPTCHA challenges.
Commercial anti-bot services: Companies offer anti-bot and anti-scraping services for websites. A few web appl ication firewalls have limited bot detection capabilities as well. However, many such solutions are not very effe ctive[26].
Locating bots with a honeypot or other method to identify the IP addresses of automated crawlers.
Obfuscation using CSS sprites to display such data as phone numbers or email addresses, at the cost of accessibi lity to screen reader users.
Because bots rely on consistency in the front-end code of a target website, adding small variations to the HTML /CSS surrounding important data and navigation elements would require more human involvement in the initial set up of a bot and if done effectively may render the target website too difficult to scrape due to the diminished ability to automate the scraping process.
Websites can declare if crawling is allowed or not in the robots.txt file and allow partial access, limit the crawl r ate, specify the optimal time to crawl and more.
Archive.is
Comparison of feed aggregators
Data scraping
Data wrangling
Importer
Job wrapping
Knowledge extraction
OpenSocial
Scraper site
Fake news website
Blog scraping
Spamdexing
Domain name drop list

Text corpus
Web archiving
Blog network
Search Engine Scraping
Web crawlers
^ a b Boeing, G.; Waddell, P. (2016). "New Insights into Rental Housing Markets across the United States: Web Scraping and Analyzing Craigslist Rental Listings". Journal of Planning Education and Research (0739456X16664789). doi:10.1177/0739456X16664789.
^ Vargiu & Urru (2013). "Exploiting web scraping in a collaborative filtering- based approach to web advertising". Artificial Intelligence Research. 2 (1). doi:10.5430/air.v2n1p44.
^ Song, Ruihua; Microsoft Research (Sep 14, 2007). "Joint Optimization of Wrapper Generation and Template Detection" (PDF). The 13th International Conference on Knowledge Discovery and Data Mining.
^ Semantic annotation based web scraping
^ Roush, Wade (2012-07-25). "Diffbot Is Using Computer Vision to Reinvent the Semantic Web". www.xconomy.com. Retrieved 2013-03-15.
^ "FAQ about linking – Are website terms of use binding contracts?". www.chillingeffects.org. 2007-08-20. Archived from the original on 2002-03-08. Retrieved 2007-08-20.
^ Kenneth, Hirschey, Jeffrey (2014-01-01). "Symbiotic Relationships: Pragmatic Acceptance of Data Scraping". Berkeley Technology Law Journal. 29 (4). doi:10.15779/Z38B39B. ISSN 1086-3818.
^ "Internet Law, Ch. 06: Trespass to Chattels". www.tomwbell.com. 2007-08-20. Retrieved 2007-08-20.
^ "What are the "trespass to chattels" claims some companies or website owners have brought?". www.chillingeffects.org. 2007-08-20. Archived from the original on 2002-03-08. Retrieved 2007-08-20.
^ "Ticketmaster Corp. v. Tickets.com, Inc." 2007-08-20. Retrieved 2007-08-20.
^ "American Airlines v. FareChase" (PDF). 2007-08-20. Archived from the original (PDF) on 2011-07-23. Retrieved 2007-08-20.
^ "American Airlines, FareChase Settle Suit". The Free Library. 2003-06-13. Retrieved 2012-02-26.
^ Imperva (2011). Detecting and Blocking Site Scraping Attacks. Imperva white paper..
^ Adler, Kenneth A. (2003-07-29). "Controversy Surrounds 'Screen Scrapers': Software Helps Users Access Web Sites But Activity by Competitors Comes Under Scrutiny". Retrieved 2010-10-27.
^ "QVC Inc. v. Resultly LLC, No. 14-06714 (E.D. Pa. filed Nov. 24, 2014)" (PDF). 2014-11-24. Retrieved 2015-11-05.
^ "QVC Inc. v. Resultly LLC, No. 14-06714 (E.D. Pa. filed Nov. 24, 2014)". United States District Court for the Eastern District of Pennsylvania. Retrieved 5 November 2015.
^ Neuburger, Jeffrey D (5 December 2014). "QVC Sues Shopping App for Web Scraping That Allegedly Triggered Site Outage". The National Law Review. Proskauer Rose LLP. Retrieved 5 November 2015.
^ "Did Iqbal/Twombly Raise the Bar for Browsewrap Claims?" (PDF). 2010-09-17. Retrieved 2010-10-27.
^ "Can Scraping Non-Infringing Content Become Copyright Infringement... Because Of How Scrapers Work? | Techdirt". Techdirt. 2009-06-10. Retrieved 2016-05-24.
^ "Facebook v. Power Ventures". Electronic Frontier Foundation. Retrieved 2016-05-24.
^ "UDSKRIFT AF SØ- & HANDELSRETTENS DOMBOG" (PDF) (in Danish). bvhd.dk. 2006-02-24. Archived from the original (PDF) on 2007-10-12. Retrieved 2007-05-30.
^ "High Court of Ireland Decisions >> Ryanair Ltd -v- Billigfluege.de GMBH 2010 IEHC 47 (26 February 2010)". British and Irish Legal Information Institute. 2010-02-26. Retrieved 2012-04-19.
^ Matthews, Áine (June 2010). "Intellectual Property: Website Terms of Use". Issue 26: June 2010. LK Shields Solicitors Update. p. 03. Retrieved 2012-04-19.
^ National Office for the Information Economy (February 2004). "Spam Act 2003: An overview for business". Australian Communications Authority. p. 6. Retrieved 2017-12-07.
^ National Office for the Information Economy (February 2004). "Spam Act 2003: A practical guide for business" (PDF). Australian Communications Authority. p. 20. Retrieved 2017-12-07.
^ Mayank Dhiman Breaking Fraud & Bot Detection Solutions OWASP AppSec Cali' 2018 Retrieved February 10, 2018.
Web scraping
World Wide Web
Spamming
CS1 Danish-language sources (da)
Articles needing additional references from June 2017
All articles needing additional references

Articles with limited geographic scope from October 2015
USA-centric
Pages using div col with deprecated parameters
Not logged in
Talk
Contributions
Create account
Log in
Article
Talk
Read
Edit
View history
Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store
Help
About Wikipedia
Community portal
Recent changes
Contact page
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page
Create a book
Download as PDF
Printable version
العربية
Català
Deutsch
Español
Euskara
Français
Íslenska
Italiano
Latviešu
Nederlands
日本語
Српски / srpski
Türkçe
Українська
中文
This page was last edited on 11 February 2018, at 06:38.

Disclaimers
Contact Wikipedia
Developers
Cookie statement
Mobile view

Whaterver we have seen so far is for scraping single webpage? if suppose you need to pass in some values to the webpage and should scrape the information of dynamically landed website what will you do?

The Answer is selinium! Next let's see how to scrape dynamically

Dynamic Web Scraping Using Python

Selenium is actually as automation testing tool to simulate the user actions on a webpage through webdriver. But, we can use it for web scraping as well.

It currently supports all the browsers such as Firefox,Chrome,IE,safari.

How to to install selenium?

pip install selenium

Let's scrape Realtime train info website

In [1]:
```
# Import selinium webdriver
from selenium import webdriver
```

In [2]:
```
# Execute or start the Chrome driver simulation. It will open Chrome brows
er.
driver = webdriver.Chrome(executable_path=r'C:\Users\murugnav\Downloads\ch
romedriver_win32\chromedriver.exe')
```

In [3]:
```
# Load a webpage through get method
driver.get('https://www.bahn.com/en/view/index.shtml')
```

In [4]:
```
# Inspect the element you require on a webpage and check what the element
it has such as class,id,Xpath and more
# once, The element you want have any attributes mentioned above then sele
ct that element by find_element_by_**(method)

from_loc = driver.find_element_by_id('js-auskunft-autocomplete-from')
```

In [5]:
```
to_loc = driver.find_element_by_id('js-auskunft-autocomplete-to')
```

In [6]:
```
# Let's clear that field before we pass in any parameter
from_loc.clear()
to_loc.clear()
```

In [7]:
```
# Send the values for that key to Tag element through send_keys() method
from_loc.send_keys('Aachen Hbf')
```

In [8]:
```
to_loc.send_keys('Aalen')
```

In [9]:
```
# find the search button
```

```
submit=driver.find_element_by_xpath('//*[@id="js-tab-auskunft"]/div/form/fieldset[5]/div/input[1]')
```

In [10]:
```
# Now we have found the button that we need click and let's click it through click() method
submit.click()
```

In [11]:
```
URL=driver.current_url
```

In [12]:
```
print(URL)
```

```
https://reiseauskunft.bahn.de/bin/query.exe/en?revia=yes&existOptimizePrice=1&country=GBR&dbkanal_007=L04_S02_D002_KIN0059_qf-bahn-svb-kl2_lz03&start=1&protocol=https%3A&S=Aachen+Hbf&REQ0JourneyStopsSID=&Z=Aalen&REQ0JourneyStopsZID=&date=Tue%2C+27.02.18&time=15%3A40&timesel=depart&returnDate=&returnTime=&returnTimesel=depart&optimize=0&auskunft_travelers_number=1&tariffTravellerType.1=E&tariffTravellerReductionClass.1=0&tariffClass=2&rtMode=DB-HYBRID&externRequest=yes&HWAI=JS%21js%3Dyes%21ajax%3Dyes%21
```

In [13]:
```
from bs4 import BeautifulSoup
```

In [14]:
```
import urllib.request as urls
```

In [15]:
```
source = urls.urlopen(URL).read()
```

In [16]:
```
soup = BeautifulSoup(source,'lxml')
```

In [17]:
```
# Find the result table
table = soup.find('table',class_='result')
```

In [18]:
```
table_rows = table.find_all('tr')

res = []

for tr in table_rows:
    # filter table rows based on tag attribute class
    td = tr.find_all('td',class_=['station','time','duration','changes','products','farePep','fareStd'])
    row = [i.text for i in td]
    res.append(row)

res = res[:-3]

print(res)
```

```
[[], ['\xa0', '\xa0', '\xa0'], ['Aachen Hbf  ', '15:39', '\n4:16\n', '2', '\nICE, RE\n', 'No special fares available\xa0', '128,00\xa0EUR\xa0'], ['\nAalen Hbf \n', '\n19:55'], ['\n', '\nTo\xa0offer\xa0selection\n'], [], [], ['\nAachen Hbf  \n', '\n15:51\xa0', '\n4:34\n', '3', '\nRE, ICE\n', '173,90\xa0EUR\xa0\u2002\xa0in 1st class', '128,00\xa0EUR\xa0'], ['\nAalen Hbf \n', '\n20:25'], ['\n', '\nTo\xa0offer\xa0selection\n'], [], [], ['\nAachen Hbf  \n', '\n15:51\xa0', '\n5:04\n', '2', '\nRE, ICE, IC\n', '123,90\xa0EUR\xa0\u2002', '128,00\xa0EUR\xa0'], ['\nAalen Hbf \n', '\n20:55'], ['\n', '\nTo\xa0offer\xa0selection\n'], [], [], ['\nAachen Hbf  \n', '\n15:51\
```

```
xa0', '\n5:34\n', '3', '\nRE, IC, ICE\n', '79,90\xa0EUR\xa0\u2002', '102,0
0\xa0EUR\xa0'], ['\nAalen Hbf \n', '\n21:25'], ['\n', '\nTo\xa0offer\xa0se
lection\n']]
```

In [19]:
```
# remove unwanted table rows
l=[]

for sel in res:
    if not sel or sel[0]=='\n' or sel[0]=='\xa0':
        pass
    else:
        l.append(sel)


print(l)
```

```
[['Aachen Hbf  ', '15:39', '\n4:16\n', '2', '\nICE, RE\n', 'No special far
es available\xa0', '128,00\xa0EUR\xa0'], ['\nAalen Hbf \n', '\n19:55'], ['
\nAachen Hbf  \n', '\n15:51\xa0', '\n4:34\n', '3', '\nRE, ICE\n', '173,90\
xa0EUR\xa0\u2002\xa0in 1st class', '128,00\xa0EUR\xa0'], ['\nAalen Hbf \n'
, '\n20:25'], ['\nAachen Hbf  \n', '\n15:51\xa0', '\n5:04\n', '2', '\nRE,
ICE, IC\n', '123,90\xa0EUR\xa0\u2002', '128,00\xa0EUR\xa0'], ['\nAalen Hbf
 \n', '\n20:55'], ['\nAachen Hbf  \n', '\n15:51\xa0', '\n5:34\n', '3', '\n
RE, IC, ICE\n', '79,90\xa0EUR\xa0\u2002', '102,00\xa0EUR\xa0'], ['\nAalen
Hbf \n', '\n21:25']]
```

In [20]:
```
# strip escape characters and unpack list for formatting
new = []
for item in l:
    for it in item:
        it=it.strip('\n')
        it=it.strip('\xa0')

        new.append(it)
```

In [21]:
```
c = len(new)/9

start=0
end = 9
i=0
final = []
while True:
    set1= new[start:end]
    final.append(tuple(set1))
    start =end
    end = end + 9
    i = i + 1
    if i >= c:
        break

print(final)
```

```
[('Aachen Hbf  ', '15:39', '4:16', '2', 'ICE, RE', 'No special fares avail
able', '128,00\xa0EUR', 'Aalen Hbf ', '19:55'), ('Aachen Hbf  ', '15:51',
'4:34', '3', 'RE, ICE', '173,90\xa0EUR\xa0\u2002\xa0in 1st class', '128,00
\xa0EUR', 'Aalen Hbf ', '20:25'), ('Aachen Hbf  ', '15:51', '5:04', '2', '
RE, ICE, IC', '123,90\xa0EUR\xa0\u2002', '128,00\xa0EUR', 'Aalen Hbf ', '2
0:55'), ('Aachen Hbf  ', '15:51', '5:34', '3', 'RE, IC, ICE', '79,90\xa0EU
R\xa0\u2002', '102,00\xa0EUR', 'Aalen Hbf ', '21:25')]
```

In [22]:
```
# import data analysis library pandas to deal with dataframes and CSV
import pandas as pd
```

In [23]:
```
# convert List of tuples to dataframe
df = pd.DataFrame.from_records(data=final,columns=['source_station','start
_time','duration','changes','products','farePep','fareStd','end_station','
end_time'])
```

In [24]:
```
# Lets see what we have in dataframe
df.head()
```

Out[24]:

	source_station	start_time	duration	changes	products	farePep	fareStd	end_
0	Aachen Hbf	15:39	4:16	2	ICE, RE	No special fares available	128,00 EUR	Aalen
1	Aachen Hbf	15:51	4:34	3	RE, ICE	173,90 EUR in 1st class	128,00 EUR	Aalen
2	Aachen Hbf	15:51	5:04	2	RE, ICE, IC	123,90 EUR	128,00 EUR	Aalen
3	Aachen Hbf	15:51	5:34	3	RE, IC, ICE	79,90 EUR	102,00 EUR	Aalen

In [25]:
```
# writing data frame to CSV file
df.to_csv('Train_Detail_new.csv')
```

Where my file got saved?

In [26]:
```
pwd
```

Out[26]:
```
'C:\\Users\\murugnav\\Web Scraping'
```

How to go Backward in Browser?

In [27]:
```
driver.back()
```

How to fo forward in Browser?

In [28]:
```
driver.forward()
```

How to refresh the page in Browser?

In [47]:
```
driver.refresh()
```

How to close the driver?

In [29]:
```
driver.close()
```

In [1]:
```
# importing required modules

import PyPDF2
```

In [4]:
```
# creating a pdf file object
f = open('Statistics.pdf', 'rb')
```

In [5]:
```
# creating a pdf reader object
pdfReader = PyPDF2.PdfFileReader(f)
```

In [7]:
```
# printing number of pages in pdf file
print(pdfReader.numPages)
```

```
236
```

In [8]:
```
# creating a page object
pageObj = pdfReader.getPage(10)
```

In [11]:
```
# extracting text from page
text=pageObj.extractText()

print(text)
```

```
Introduction
1.2.4ModernRegression
Regressionmodelsrelatevariablestoeachotherinalinearfashion.Forexample,ifyou
recordedtheheightsandweightsofseveralpeopleandplottedthemagainsteachother,
youwould˝ndthatasheightincreases,weighttendstoincreasetoo.Youwouldprobably
alsoseethatastraightlinethroughthedataisaboutasgoodawayofapproximatingthe
relationshipasyouwillbeableto˝nd,thoughtherewillbesomevariabilityabouttheline.
Suchlinearmodelsarepossiblythemostimportanttoolavailabletostatisticians.They
havealonghistoryandmanyofthemoredetailedtheoreticalaspectswerediscoveredinthe
1970s.Theusualmethodfor˝ttingsuchmodelsisby"leastsquares"estimation,though
othermethodsareavailableandareoftenmoreappropriate,especiallywhenthedataare
notnormallydistributed.
Whathappens,though,iftherelationshipisnotastraightline?Howcanacurvebe˝tto
thedata?Therearemanyanswerstothisquestion.Onesimplesolutionisto˝taquadratic
relationship,butinpracticesuchacurveisoftennot¸exibleenough.Also,whatifyou
havemanyvariablesandrelationshipsbetweenthemaredissimilarandcomplicated?
Modernregressionmethodsaimataddressingtheseproblems.Methodssuchasgeneralized
additivemodels,projectionpursuitregression,neuralnetworksandboostingallowforvery
generalrelationshipsbetweenexplanatoryvariablesandresponsevariables,andmodern
computingpowermakesthesemethodsapracticaloptionformanyapplications
1.2.5Classi˝cation
Somethingsaredi˙erentfromothers.How?Thatis,howareobjectsclassi˝edintotheir
respectivegroups?Considerabankthatishopingtolendmoneytocustomers.Some
```

customerswhoborrowmoneywillbeunableorunwillingtopayitback,thoughmostwill payitbackasregularrepayments.Howisthebanktoclassifycustomersintothesetwo groupswhendecidingwhichonestolendmoneyto?
Theanswertothisquestionnodoubtisin ̧uencedbymanythings,includingacustomer's income,credithistory,assets,alreadyexistingdebt,ageandprofession.Theremaybeother in ̧uential,measurablecharacteristicsthatcanbeusedtopredictwhatkindofcustomera particularindividualis.Howshouldthebankdecidewhichcharacteristicsareimportant, andhowshoulditcombinethisinformationintoarulethattellsitwhetherornottolend themoney?
Thisisanexampleofaclassi ́cationproblem,andstatisticalclassi ́cationisalarge ́eld containingmethodssuchaslineardiscriminantanalysis,classi ́cationtrees,neuralnetworks andothermethods.
1.2.6TimeSeries
Manytypesofresearchlookatdatathataregatheredovertime,whereanobservationtaken todaymayhavesomecorrelationwiththeobservationtakentomorrow.Twoprominent examplesofthisarethe ́eldsof ́nance(thestockmarket)andatmosphericscience.
6

In [9]:
```
# closing the pdf file object
pdfFileObj.close()
```

In [2]:
```
# pip install SpeechRecognition
#pip install pyaudio
```

In [1]:
```
import speech_recognition as sr
```

In [4]:
```
# Record Audio
r = sr.Recognizer()
with sr.Microphone() as source:
    print("Say something!")
    audio = r.listen(source)

# Speech recognition using Google Speech Recognition
try:
    # for testing purposes, we're just using the default API key
    # to use another API key, use `r.recognize_google(audio, key="GOOGLE_SPEECH_RECOGNITION_API_KEY")`
    # instead of `r.recognize_google(audio)`
    print("You said: " + r.recognize_google(audio))
except sr.UnknownValueError:
    print("Google Speech Recognition could not understand audio")
except sr.RequestError as e:
    print("Could not request results from Google Speech Recognition service; {0}".format(e))
```

```
Say something!
You said: hi Chandru
```

In [3]:
```
class newstr(str):

    def find_all(string,substring):
        """
        Function: Returning all the index of substring in a string
        Arguments: String and the search string
        Return:Returning a list
        """
        # finding length of a subtring
        length = len(substring)
        #initializing counter or poniter
        c=0
        # declaring empty list. Remember you cannot append a list without
declaring list.
        indexes = []
        #loop through each letter of string until EOF string
        while c < len(string):
            if string[c:c+length] == substring:
                indexes.append(c)
            #Incrementing the counter
            c=c+1
        return indexes
```

In [2]:
```
newstr.find_all('Am i repeating ? yes am repeating so what','repeating')
```

Out[2]:
```
[5, 24]
```

In [4]:
```
newstr.find_all('Am i repeating ? yes am repeating so what','s')
```

Out[4]:
```
[19, 34]
```

In [11]:
```
### %timeit function used to test how much time it took to execute the fun
cion.
### Actually, it will execute 1000000 times and will give you fastest and
slowest run details
### Remember, These timings will change time to time based on your RAM cap
acity or resource available
```

In [8]:
```
%timeit newstr.find('Am i repeating ? yes am repeating so what','repeating
')
```

```
The slowest run took 13.23 times longer than the fastest. This could mean
that an intermediate result is being cached.
1000000 loops, best of 3: 480 ns per loop
```

In [39]:
```
basket = ['pappaya','Orange','apple','mango','strawberry']
```

In [40]:
```
sorted(basket,key=lambda x: x[-1])
```

Out[40]:
```
['pappaya', 'Orange', 'apple', 'mango', 'strawberry']
```

In [36]:
```
x='pappaya'
```

In [37]:
```
x[-1]
```

Out[37]:
```
'a'
```

In [43]:
```
list(filter(lambda n:n%2 ==1,[1,2,3,34,4,499]))
```

Out[43]:
```
[1, 3, 499]
```

In [44]:
```
def square(n):
    return n**2
```

In [45]:
```
square(4)
```

Out[45]:
```
16
```

In [46]:
```
lambda n: n**2
```

Out[46]:
```
<function __main__.<lambda>>
```

In [48]:
```
list(map(lambda n: n**2,[1,2,3,4,4]))
```

Out[48]:
```
[1, 4, 9, 16, 16]
```

In [49]:
```
def test(a,b):
    if a>b:
        return a
    else:
        return b
```

In [50]:
```
test(10,1)
```

Out[50]:
```
10
```

Managing Python Packages with Conda

To know details about conda

conda info

Help on conda commands

conda help

Help on particular conda command

conda help install

Listing all the packages installed

conda list

searching for conda packages

conda search numpy

Creating virtual environment through conda

creating conda py27env

conda create --name py27env python==2.7

cloning from other environment

conda create --name myenv --clone copyenv

Activating conda environment

activate py27env

Deactivating conda environment

deactivate py27env

Removing packages or environment

To remove environment

conda remove --name py27env --all

To remove packages

conda remove --name py27env numpy

or

To remove from current environment

conda remove numpy

Installing packages

conda install numpy

Uninstalling packages

conda uninstall numpy

Listing virtual environments

conda info --envs

or

conda info -e

In [ ]:

Managing Python Packages with PIP

Pypi(Python Package Index) is the repository of python packages which were created by open source community. As per November 2017, there are 1,22,887 packages available in Pypi.

Repository Link: https://pypi.python.org/

Ok, How will i install or uninstall or maintain the packges?

You can manage python packages through pip(Pip installs packages) or conda tools to managa python packages.

Help Command

pip help

Help on particular command

pip help install

Listing the packages available on your system

pip list

Searching packages in Pypi

pip search numpy

How will i install packages?

pip install scipy

How will i install specific version of the packages?

pip install numpy==1.13.1

How will i setup my collegue's system environment who is going to work with me for the same project?

pip freeze > requirement.txt

This freeze command will write all the package names and versions into the requirement.txt file(in this case)

and you can install all the packages using the below install command

pip install -r requirement.txt

How can i uninstall the package?

pip uninstall numpy

How can i find outdated packages?

pip list --outdated or pip list -o

How can i know the details about package?

pip show numpy

In [ ]: