A SIMPLE CRAWLER IN PYTHON:

**HOTEL CRAWLER:**

***Note:*** Always follow the [***robots.txt***](https://www.goibibo.com/robots.txt) file of the target website which is also known as the robot exclusion protocol. This tells web robots which pages not to crawl.

[](https://s3-ap-south-1.amazonaws.com/av-blog-media/wp-content/uploads/2019/09/updated_robots_goibibo.png)

# importing required libraries

import requests

from bs4 import BeautifulSoup

import pandas as pd

# target URL to scrap

url = "https://www.goibibo.com/hotels/hotels-in-shimla-ct/"

# headers

headers = {

'User-Agent': "Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.90 Safari/537.36"

}

# send request to download the data

response = requests.request("GET", url, headers=headers)

# parse the downloaded data

data = BeautifulSoup(response.text, 'html.parser')

print(data)

# find all the sections with specified class name

cards\_data = data.find\_all('div', attrs={'class', 'width100 fl htlListSeo hotel-tile-srp-container hotel-tile-srp-container-template new-htl-design-tile-main-block'})

# total number of cards

print('Total Number of Cards Found : ', len(cards\_data))

# source code of hotel cards

for card in cards\_data:

print(card)

# extract the hotel name and price per room

for card in cards\_data:

# get the hotel name

hotel\_name = card.find('p')

# get the room price

room\_price = card.find('li', attrs={'class': 'htl-tile-discount-prc'})

print(hotel\_name.text, room\_price.text)

# create a list to store the data

scraped\_data = []

for card in cards\_data:

# initialize the dictionary

card\_details = {}

# get the hotel name

hotel\_name = card.find('p')

# get the room price

room\_price = card.find('li', attrs={'class': 'htl-tile-discount-prc'})

# add data to the dictionary

card\_details['hotel\_name'] = hotel\_name.text

card\_details['room\_price'] = room\_price.text

# append the scraped data to the list

scraped\_data.append(card\_details)

# create a data frame from the list of dictionaries

dataFrame = pd.DataFrame.from\_dict(scraped\_data)

# save the scraped data as CSV file

dataFrame.to\_csv('hotels\_data.csv', index=False)

-------------------------------------------------------------------------------------------------------------------------------------

IMAGE SCRAPING:

"""

Web Scraping - Scrap Images

"""

# importing required libraries

import requests

from bs4 import BeautifulSoup

# target URL

url = "https://www.goibibo.com/hotels/hotels-in-shimla-ct/"

headers = {

'User-Agent': "Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.90 Safari/537.36"

}

response = requests.request("GET", url, headers=headers)

data = BeautifulSoup(response.text, 'html.parser')

# find all with the image tag

images = data.find\_all('img', src=True)

print('Number of Images: ', len(images))

for image in images:

print(image)

# select src tag

image\_src = [x['src'] for x in images]

# select only jp format images

image\_src = [x for x in image\_src if x.endswith('.jpg')]

for image in image\_src:

print(image)

image\_count = 1

for image in image\_src:

with open('image\_'+str(image\_count)+'.jpg', 'wb') as f:

res = requests.get(image)

f.write(res.content)

image\_count = image\_count+1

--------------------------------------------------------------------------------------------------------------------------------------

**WIKIPEDIA CRAWLER**

The following is a sample crawler that crawls Wikipedia and prints out the list of search results for a given url.

PGM 1:

from bs4 import BeautifulSoup

import requests

from collections import deque

def parse\_html(html):

soup = BeautifulSoup(html)

title\_element = soup.find("div", attrs={"class": "mw-search-result-heading"}).find("a")

title = title\_element.attrs["title"] # in this crawler, we will collect the title element

links = soup.find\_all("a", attrs={"class": "mw-nextlink"}) # get the next links to crawl

return title, [x.attrs["href"] for x in links if "href" in x.attrs]

def crawl(url\_list):

urls\_to\_crawl = deque(url\_list)

output\_data = []

while len(urls\_to\_crawl) > 0:

url = urls\_to\_crawl.pop()

url\_contents = requests.get(url).text

data, links = parse\_html(url\_contents)

output\_data.append(data)

urls\_to\_crawl.extend(links)

return output\_data

-----------------------------------------------------------------------------------------------------------------------------------

PGM 2:

import re, urllib

textfile = file('depth\_1.txt','wt')

print "Enter the URL you wish to crawl.."

myurl = input()

for i in re.findall('''href=["'](.[^"']+)["']''', urllib.urlopen(myurl).read(), re.I):

print i

for ee in re.findall('''href=["'](.[^"']+)["']''', urllib.urlopen(i).read(), re.I):

print ee

textfile.write(ee+'\n')

textfile.close()

PRINCETON UNIVERSITY CRAWLER:

import urllib2

from urlparse import urlparse

import BeautifulSoup

regex = re.compile(

r'^(?:http|ftp)s?://' # http:// or https://

r'(?:(?:[A-Z0-9](?:[A-Z0-9-]{0,61}[A-Z0-9])?\.)+(?:[A-Z]{2,6}\.?|[A-Z0-9-]{2,}\.?)|' #domain...

r'localhost|' #localhost...

r'\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})' # ...or ip

r'(?::\d+)?' # optional port

r'(?:/?|[/?]\S+)$', re.IGNORECASE)

def isValidUrl(url):

if regex.match(url) is not None:

return True;

return False

def crawler(SeedUrl):

tocrawl=[SeedUrl]

crawled=[]

while tocrawl:

page=tocrawl.pop()

print 'Crawled:'+page

pagesource=urllib2.urlopen(page)

s=pagesource.read()

soup=BeautifulSoup.BeautifulSoup(s)

links=soup.findAll('a',href=True)

if page not in crawled:

for l in links:

if isValidUrl(l['href']):

tocrawl.append(l['href'])

crawled.append(page)

return crawled

crawler('http://www.princeton.edu/main/')

**FOOTBALLERS CRAWLER**