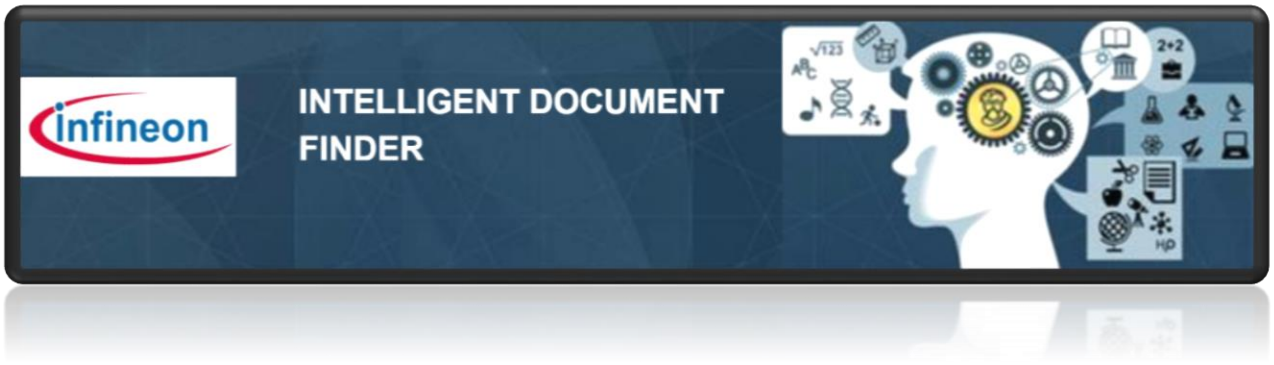
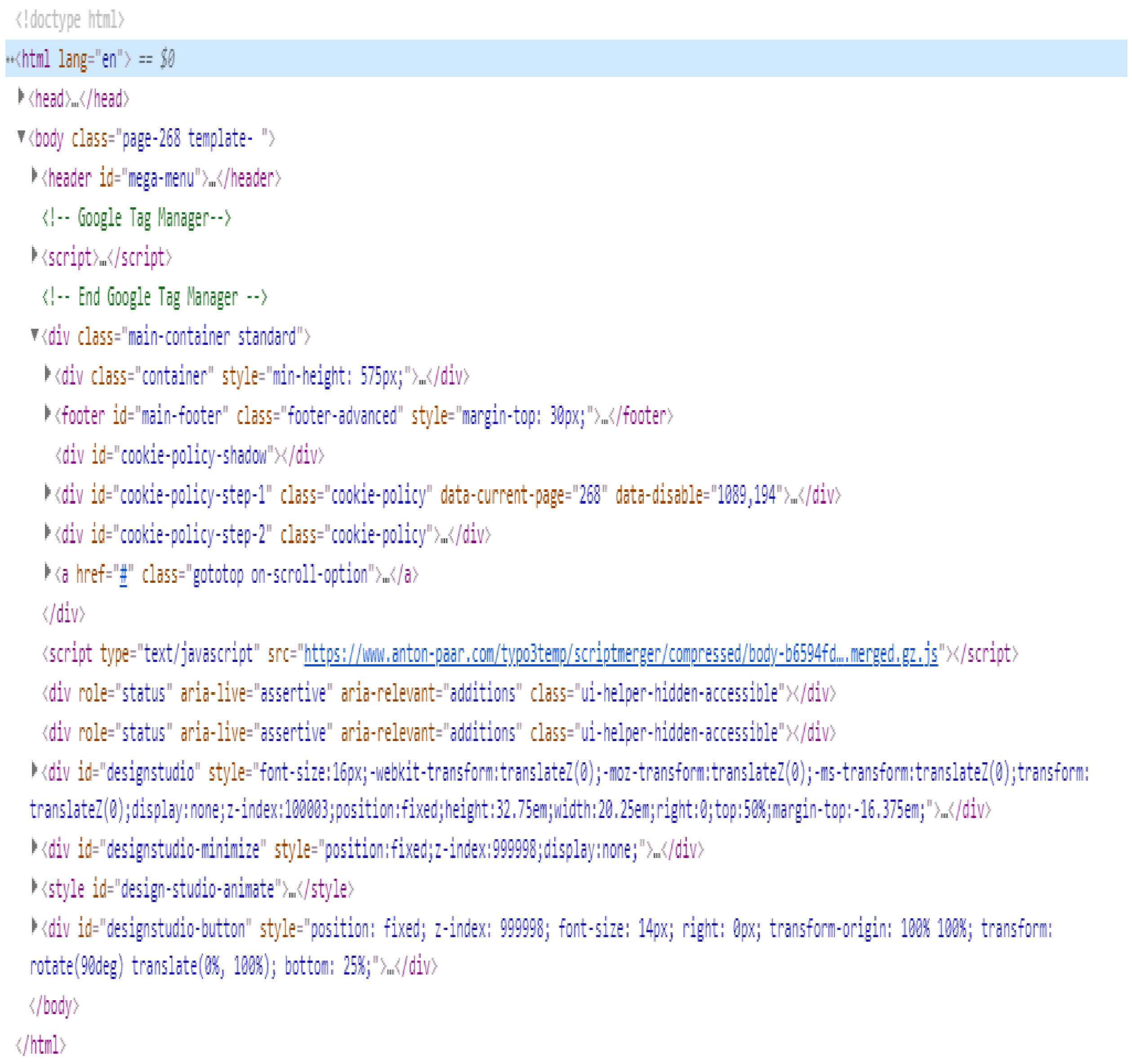


Problem Statement:

We are looking for an Intelligent Document Finder tool that can provide easy and intelligent searches among the document files. The required document type includes presentations, pdf, doc and txt files. The main idea behind this problem statement is combining human tagging with an automated semantic search for efficient document finding. The tool is supposed to have a manual as well as auto-tagging capabilities. Once the document are tagged, the user will enter a few queries in the search page of the tool to look for the most relevant document.

DOCUMENT FINDER TOOL



```
<!doctype html>
…<html lang="en"> == $0
 ▶<head>…</head>
 ▼<body class="page-268 template- ">
   ▶<header id="mega-menu">…</header>
    <!-- Google Tag Manager-->
   ▶<script>…</script>
    <!-- End Google Tag Manager -->
   ▼<div class="main-container standard">
     ▶<div class="container" style="min-height: 575px;">…</div>
     ▶<footer id="main-footer" class="footer-advanced" style="margin-top: 30px;">…</footer>
      <div id="cookie-policy-shadow"></div>
     ▶<div id="cookie-policy-step-1" class="cookie-policy" data-current-page="268" data-disable="1089,194">…</div>
     ▶<div id="cookie-policy-step-2" class="cookie-policy">…</div>
     ▶<a href="#" class="gototop on-scroll-option">…</a>
    </div>
    <script type="text/javascript" src="https://www.anton-paar.com/typo3temp/scriptmerger/compressed/body-b6594fd….merged.gz.js"></script>
    <div role="status" aria-live="assertive" aria-relevant="additions" class="ui-helper-hidden-accessible"></div>
    <div role="status" aria-live="assertive" aria-relevant="additions" class="ui-helper-hidden-accessible"></div>
   ▶<div id="designstudio" style="font-size:16px;-webkit-transform:translateZ(0);-moz-transform:translateZ(0);-ms-transform:translateZ(0);transform:
   translateZ(0);display:none;z-index:100003;position:fixed;height:32.75em;width:20.25em;right:0;top:50%;margin-top:-16.375em;">…</div>
   ▶<div id="designstudio-minimize" style="position:fixed;z-index:999998;display:none;">…</div>
   ▶<style id="design-studio-animate">…</style>
   ▶<div id="designstudio-button" style="position: fixed; z-index: 999998; font-size: 14px; right: 0px; transform-origin: 100% 100%; transform:
   rotate(90deg) translate(0%, 100%); bottom: 25%;">…</div>
  </body>
</html>
```

Manual tagging:

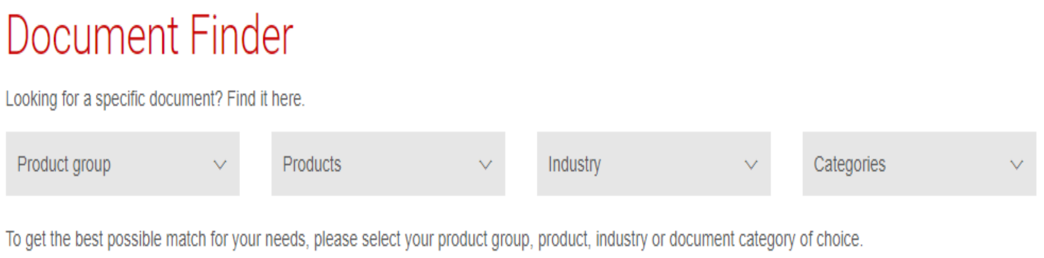
Manual tagging can provide data for only the following dimensions: Campaign, Source, Medium, Content, Keyword. When you use auto-tagging, however, you can see data for several additional dimensions, including: Query Match Type (How your keyword was actually matched to the search query).

Example : google ads...

Code:

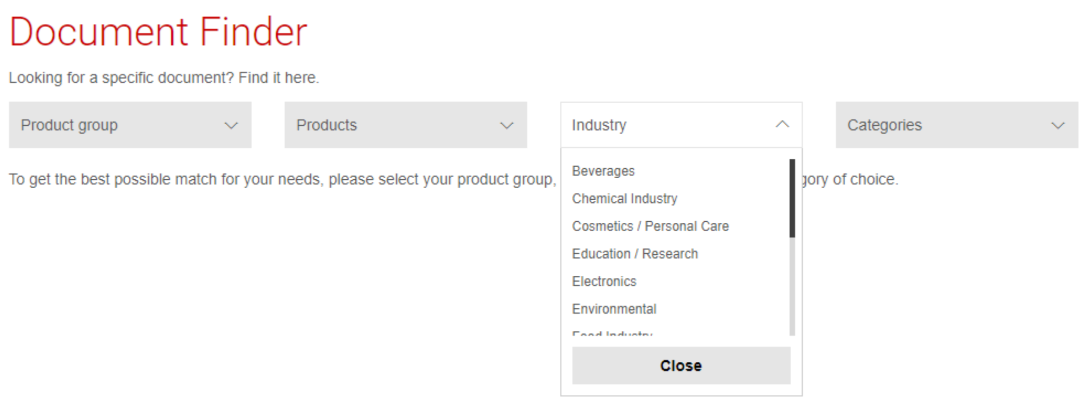
```
import os
import re
import sys
from threading import  Thread
from datetime import datetime
import subprocess
import cPickle
dict1 = {}
def get_drives():
            response = os.popen("wmic logicaldisk get caption")
            list1 = []
            total_file = []
            t1= datetime.now()
            for line in response.readlines():
                        line = line.strip("\n")
                        line = line.strip("\r")
                        line = line.strip(" ")
                        if (line == "Caption" or line == ""):
                                    continue
                        list1.append(line)
            return list1
def search1(drive):
       for root, dir, files in os.walk(drive, topdown = True):
                  for file in files:
                            file= file.lower()
                          if file in dict1:
                                          file = file+”_1”
                                          dict1[file]= root
                                    else :
                                          dict1[file]= rootdef
create():
        t1= datetime.now()
        list2 = []   # empty list is created
        list1 = get_drives()
        print list1
        for each in list1:
            process1 = Thread(target=search1, args=(each,))
            process1.start()
            list2.append(process1)

         for t in list2:
                  t.join() # Terminate the threads
pickle_file = open(“finder_data”,”w”)
            cPickle.dump(dict1,pickle_file)
            pickle_file.close()
            t2= datetime.now()
            total =t2-t1
            print “Time taken to create “ , total
            print “Thanks for using L4wisdom.com”
if len(sys.argv) < 2 or len(sys.argv) > 2:
            print “Please use proper format”
            print “Use <finder -c >  to create database file”
            print “Use <finder file-name> to search file”
            print “Thanks for using L4wisdom.com”

elif sys.argv[1] == ‘-c’:
            create()

else:
            t1= datetime.now()
            try:
                     pickle_file  = open(“finder_data”, “r”)
                     file_dict = cPickle.load(pickle_file)
                     pickle_file.close()
            except IOError:
                        create()
            except Exception as e : print e
                        sys.exit()
            file_to_be_searched = sys.argv[1].lower()
            list1= []
            print “Path \t\t: File-name”
for key in file_dict:
                   if re.search(file_to_be_searched, key):
                           str1 =  file_dict[key]+” : “+key
                                    list1.append(str1)
            list1.sort()
            for each in list1:
                              print each
                            print “-----------------------”
            t2= datetime.now()
            total =t2-t1
            print “Total files are”, len(list1)
            print “Time taken to search “ , total
            print “Thanks for using L4wisdom.com”
```

Sample output:


Document Finder

Looking for a specific document? Find it here.

To get the best possible match for your needs, please select your product group, product, industry or document category of choice.



Document Finder

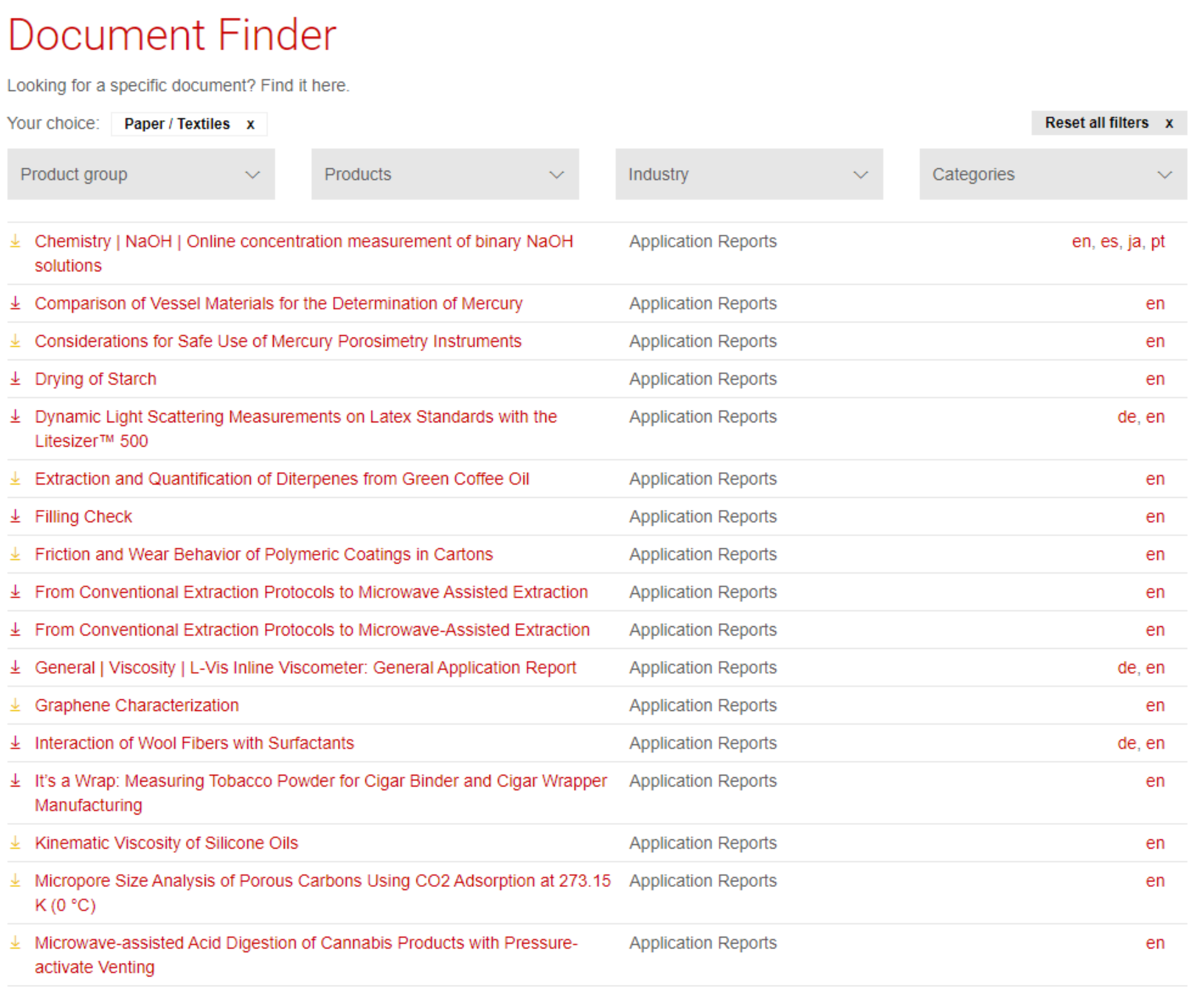
Looking for a specific document? Find it here.

Product group | Products | Industry | Categories

- Beverages
- Chemical Industry
- Cosmetics / Personal Care
- Education / Research
- Electronics
- Environmental

Close

To get the best possible match for your needs, please select your product group, gory of choice.



Document Finder

Looking for a specific document? Find it here.

Your choice: **Paper / Textiles x** **Reset all filters x**

Product group | Products | Industry | Categories

Chemistry | NaOH | Online concentration measurement of binary NaOH solutions	Application Reports	en, es, ja, pt
Comparison of Vessel Materials for the Determination of Mercury	Application Reports	en
Considerations for Safe Use of Mercury Porosimetry Instruments	Application Reports	en
Drying of Starch	Application Reports	en
Dynamic Light Scattering Measurements on Latex Standards with the Litesizer™ 500	Application Reports	de, en
Extraction and Quantification of Diterpenes from Green Coffee Oil	Application Reports	en
Filling Check	Application Reports	en
Friction and Wear Behavior of Polymeric Coatings in Cartons	Application Reports	en
From Conventional Extraction Protocols to Microwave Assisted Extraction	Application Reports	en
From Conventional Extraction Protocols to Microwave-Assisted Extraction	Application Reports	en
General | Viscosity | L-Vis Inline Viscometer: General Application Report	Application Reports	de, en
Graphene Characterization	Application Reports	en
Interaction of Wool Fibers with Surfactants	Application Reports	de, en
It's a Wrap: Measuring Tobacco Powder for Cigar Binder and Cigar Wrapper Manufacturing	Application Reports	en
Kinematic Viscosity of Silicone Oils	Application Reports	en
Micropore Size Analysis of Porous Carbons Using CO2 Adsorption at 273.15 K (0 °C)	Application Reports	en
Microwave-assisted Acid Digestion of Cannabis Products with Pressure-activate Venting	Application Reports	en

Auto-tagging:

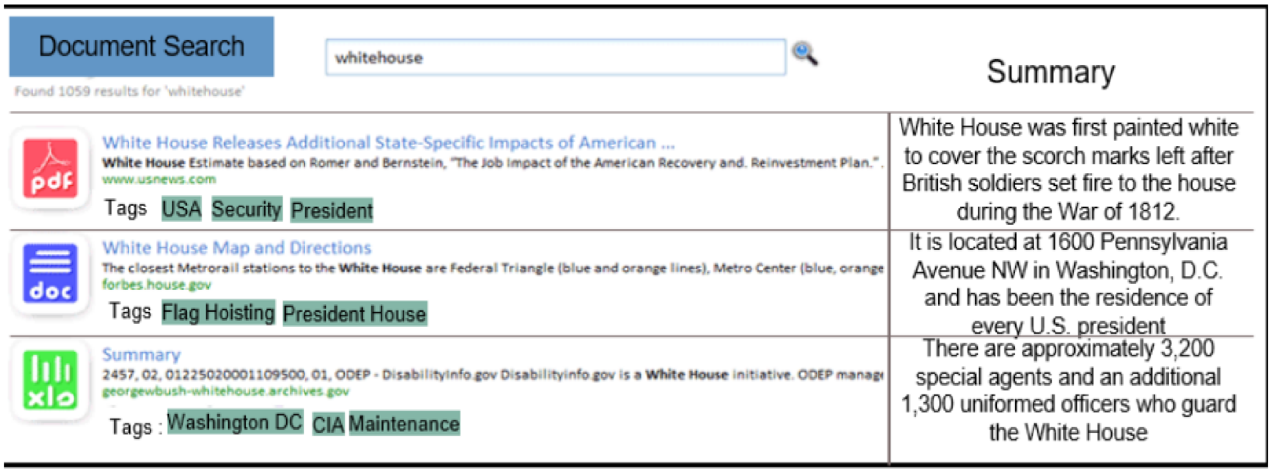
Auto-tagging: A feature that automatically adds a parameter to your URLs to help you track offline conversions and report on your ad performance using website tracking programs like Google Analytics. ...You can also use this information to import complex conversions into Google Ads, whether online or offline.

Example : google ads...

Code:

```
import os
import boto3
logging.basicConfig(level=os.environ.get('LOG_LEVEL', 'INFO'))
ec2 = boto3.client('ec2')
logger = logging.getLogger(__name__)
def tag_snapshots():
    snapshots = {}
    for response in
ec2.get_paginator('describe_snapshots').paginate(OwnerIds=['self']):
        snapshots.update([(snapshot['SnapshotId'], snapshot) for snapshot in
response['Snapshots']])
    for image in ec2.describe_images(Owners=['self'])['Images']:
        tags = boto3_tag_list_to_ansible_dict(image.get('Tags', []))
        for device in image['BlockDeviceMappings']:
            if 'SnapshotId' in device['Ebs']:
                snapshot = snapshots[device['Ebs']['SnapshotId']]
                snapshot['Used'] = True
                cur_tags = boto3_tag_list_to_ansible_dict(snapshot.get('Tags', []))
                new_tags = copy.deepcopy(cur_tags)
                new_tags.update(tags)
                new_tags['ImageId'] = image['ImageId']
                new_tags['Name'] += ' ' + device['DeviceName']
                if new_tags != cur_tags:
                    logger.info('{0}: Tags changed to
{1}'.format(snapshot['SnapshotId'], new_tags))
                    ec2.create_tags(Resources=[snapshot['SnapshotId']],
Tags=ansible_dict_to_boto3_tag_list(new_tags))
    for snapshot in snapshots.values():
        if 'Used' not in snapshot:
            cur_tags = boto3_tag_list_to_ansible_dict(snapshot.get('Tags', []))
            name = cur_tags.get('Name', snapshot['SnapshotId'])
            if not name.startswith('UNUSED'):
                logger.warning('{0} Unused!'.format(snapshot['SnapshotId']))
                cur_tags['Name'] = 'UNUSED ' + name
                ec2.create_tags(Resources=[snapshot['SnapshotId']],
Tags=ansible_dict_to_boto3_tag_list(cur_tags))
def tag_volumes():
    volumes = {}
    for response in ec2.get_paginator('describe_volumes').paginate():
        volumes.update([(volume['VolumeId'], volume) for volume in
response['Volumes']])
    for response in ec2.get_paginator('describe_instances').paginate():
        for reservation in response['Reservations']:
            for instance in reservation['Instances']:
                tags = boto3_tag_list_to_ansible_dict(instance.get('Tags', []))
                for device in instance['BlockDeviceMappings']:
                    volume = volumes[device['Ebs']['VolumeId']]
                    volume['Used'] = True
                    cur_tags = boto3_tag_list_to_ansible_dict(volume.get('Tags', []))
                    new_tags = copy.deepcopy(cur_tags)
                    new_tags.update(tags)
                    new_tags['Name'] += ' ' + device['DeviceName']
                    if new_tags != cur_tags:
                        logger.info('{0} Tags changed to
{1}'.format(volume['VolumeId'], new_tags))
                        ec2.create_tags(Resources=[volume['VolumeId']],
Tags=ansible_dict_to_boto3_tag_list(new_tags))
    for volume in volumes.values():
        if 'Used' not in volume:
            cur_tags = boto3_tag_list_to_ansible_dict(volume.get('Tags', []))
            name = cur_tags.get('Name', volume['VolumeId'])
            if not name.startswith('UNUSED'):
                logger.warning('{0} Unused!'.format(volume['VolumeId']))
                cur_tags['Name'] = 'UNUSED ' + name
                ec2.create_tags(Resources=[volume['VolumeId']],
Tags=ansible_dict_to_boto3_tag_list(cur_tags))
def tag_everything():
    tag_snapshots()
    tag_volumes()
def boto3_tag_list_to_ansible_dict(tags_list):
    tags_dict = {}
    for tag in tags_list:
        if 'key' in tag and not tag['key'].startswith('aws:'):
            tags_dict[tag['key']] = tag['value']
        elif 'Key' in tag and not tag['Key'].startswith('aws:'):
            tags_dict[tag['Key']] = tag['Value']
    return tags_dict
def ansible_dict_to_boto3_tag_list(tags_dict):
    tags_list = []
    for k, v in tags_dict.items():
        tags_list.append({'Key': k, 'Value': v})
    return tags_list
def handler(event, context):
    tag_everything()
if __name__ == '__main__':
    tag_everything()
```

Sample output:



Document Search

whitehouse

Found 1059 results for 'whitehouse'

White House Releases Additional State-Specific Impacts of American ...
White House Estimate based on Romer and Bernstein, "The Job Impact of the American Recovery and. Reinvestment Plan." .
www.usnews.com
Tags USA Security President

White House Map and Directions
The closest Metrorail stations to the **White House** are Federal Triangle (blue and orange lines), Metro Center (blue, orange
forbes.house.gov
Tags Flag Hoisting President House

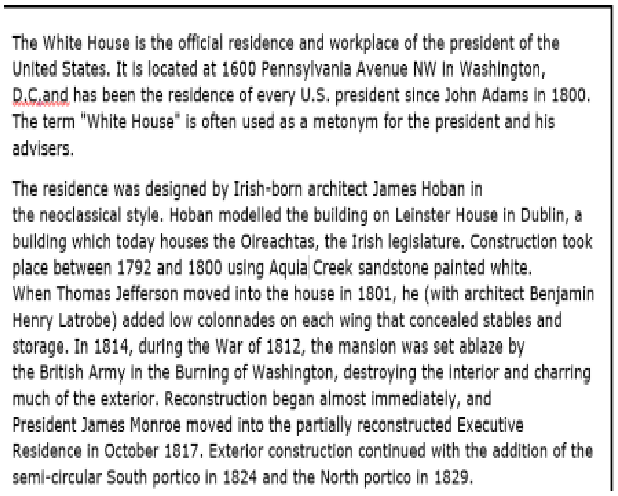
Summary
2457, 02, 01225020001109500, 01, ODEP - Disabilityinfo.gov Disabilityinfo.gov is a **White House** initiative. ODEP manag
georgewbush-whitehouse.archives.gov
Tags : Washington DC CIA Maintenance

Summary

White House was first painted white to cover the scorch marks left after British soldiers set fire to the house during the War of 1812.

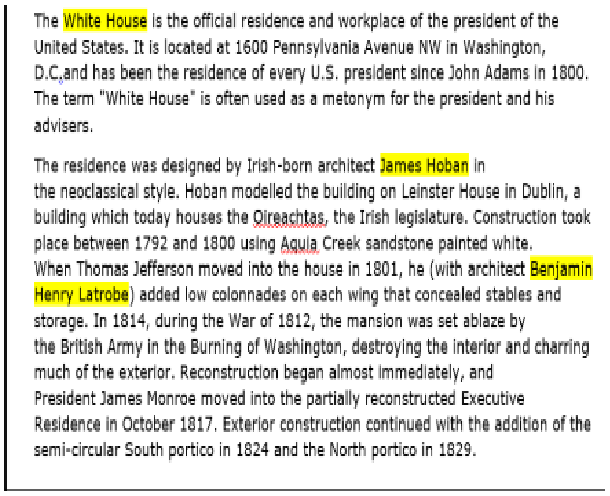
It is located at 1600 Pennsylvania Avenue NW in Washington, D.C. and has been the residence of every U.S. president

There are approximately 3,200 special agents and an additional 1,300 uniformed officers who guard the White House



The White House is the official residence and workplace of the president of the United States. It is located at 1600 Pennsylvania Avenue NW in Washington, D.C.and has been the residence of every U.S. president since John Adams in 1800. The term "White House" is often used as a metonym for the president and his advisers.

The residence was designed by Irish-born architect James Hoban in the neoclassical style. Hoban modelled the building on Leinster House in Dublin, a building which today houses the Oireachtas, the Irish legislature. Construction took place between 1792 and 1800 using Aquia Creek sandstone painted white. When Thomas Jefferson moved into the house in 1801, he (with architect Benjamin Henry Latrobe) added low colonnades on each wing that concealed stables and storage. In 1814, during the War of 1812, the mansion was set ablaze by the British Army in the Burning of Washington, destroying the interior and charring much of the exterior. Reconstruction began almost immediately, and President James Monroe moved into the partially reconstructed Executive Residence in October 1817. Exterior construction continued with the addition of the semi-circular South portico in 1824 and the North portico in 1829.



The White House is the official residence and workplace of the president of the United States. It is located at 1600 Pennsylvania Avenue NW in Washington, D.C.and has been the residence of every U.S. president since John Adams in 1800. The term "White House" is often used as a metonym for the president and his advisers.

The residence was designed by Irish-born architect James Hoban in the neoclassical style. Hoban modelled the building on Leinster House in Dublin, a building which today houses the Oireachtas, the Irish legislature. Construction took place between 1792 and 1800 using Aquia Creek sandstone painted white. When Thomas Jefferson moved into the house in 1801, he (with architect Benjamin Henry Latrobe) added low colonnades on each wing that concealed stables and storage. In 1814, during the War of 1812, the mansion was set ablaze by the British Army in the Burning of Washington, destroying the interior and charring much of the exterior. Reconstruction began almost immediately, and President James Monroe moved into the partially reconstructed Executive Residence in October 1817. Exterior construction continued with the addition of the semi-circular South portico in 1824 and the North portico in 1829.