Working with ZIP archives

All Versions

Examples

Examining Zipfile Contents

There are a few ways to inspect the contents of a zipfile. You can use the printdir to just get a variety of information sent to stdout

```
with zipfile.ZipFile(filename) as zip:
    zip.printdir()

    # Out:
    # File Name                                             Modified             Size
    # pyexpat.pyd                                    2016-06-25 22:13:34       157336
    # python.exe                                     2016-06-25 22:13:34        39576
    # python3.dll                                    2016-06-25 22:13:34        51864
    # python35.dll                                   2016-06-25 22:13:34      3127960
    # etc.
```

We can also get a list of filenames with the namelist method. Here, we simply print the list:

```
with zipfile.ZipFile(filename) as zip:
    print(zip.namelist())

# Out: ['pyexpat.pyd', 'python.exe', 'python3.dll', 'python35.dll', ... etc. ...]
```

Instead of namelist , we can call the infolist method, which returns a list of ZipInfo objects, which contain additional information about each file, for instance a timestamp and file size:

```
with zipfile.ZipFile(filename) as zip:
    info = zip.infolist()
    print(zip[0].filename)
    print(zip[0].date_time)
    print(info[0].file_size)

# Out: pyexpat.pyd
# Out: (2016, 6, 25, 22, 13, 34)
# Out: 157336
```

Opening Zip Files

To start, import the zipfile module, and set the filename.

```
import zipfile
filename = 'zipfile.zip'
```

Working with zip archives is very similar to working with files , you create the object by opening the zipfile, which lets you work on it before closing the file up again.

```
zip = zipfile.ZipFile(filename)
print(zip)
# <zipfile.ZipFile object at 0x0000000002E51A90>
zip.close()
```

In Python 2.7 and in Python 3 versions higher than 3.2, we can use the with context manager. We open the file in "read" mode, and then print a list of filenames:

```
with zipfile.ZipFile(filename, 'r') as z:
    print(zip)
    # <zipfile.ZipFile object at 0x0000000002E51A90>
```

Creating new archives

To create new archive open zipfile with write mode.

```
import zipfile
new_arch=zipfile.ZipFile("filename.zip",mode="w")
```

To add files to this archive use write() method.

```
new_arch.write('filename.txt','filename_in_archive.txt') #first parameter is filename and seconc
new_arch.close()
```

If you want to write string of bytes into the archive you can use writestr() method.

```
str_bytes="string buffer"
new_arch.writestr('filename_string_in_archive.txt',str_bytes)
new_arch.close()
```

Extracting zip file contents to a directory

Extract all file contents of a zip file

```
import zipfile
with zipfile.ZipFile('zipfile.zip','r') as zfile:
    zfile.extractall('path')
```

If you want extract single files use extract method, it takes name list and path as input parameter

```
import zipfile
f=open('zipfile.zip','rb')
zfile=zipfile.ZipFile(f)
for cont in zfile.namelist():
    zfile.extract(cont,path)
```

Syntax

```
import zipfile
class zipfile. ZipFile ( file, mode='r', compression=ZIP_STORED, allowZip64=True )
```

Parameters

Remarks

If you try to open a file that is not a ZIP file, the exception zipfile.BadZipFile is raised.

In Python 2.7, this was spelled zipfile.BadZipfile , and this old name is retained alongside the new one in Python 3.2+