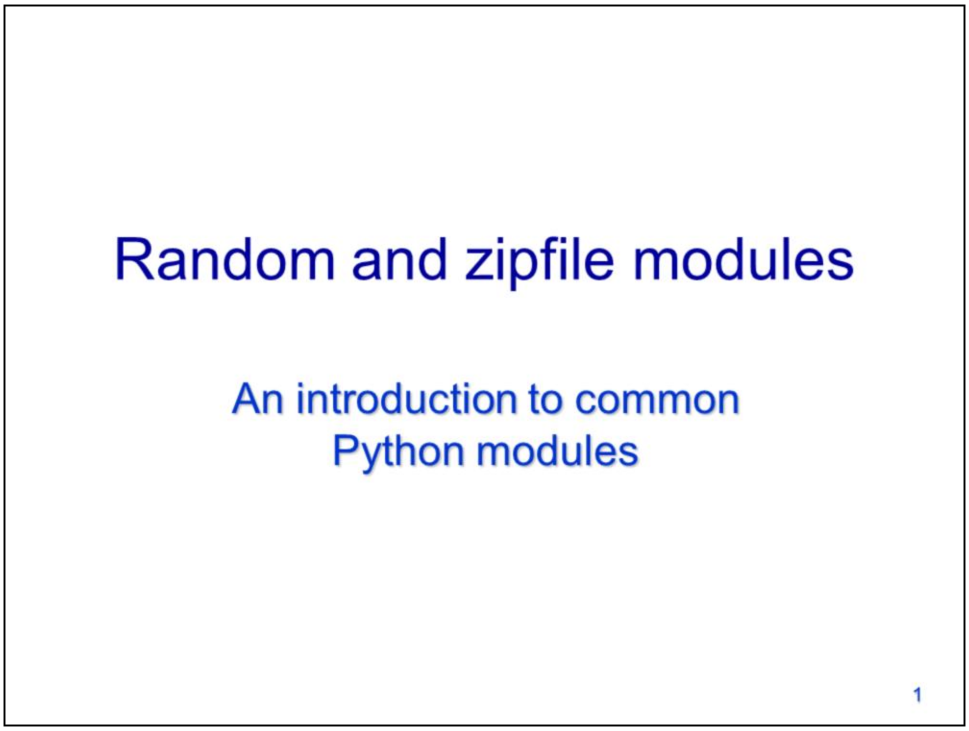
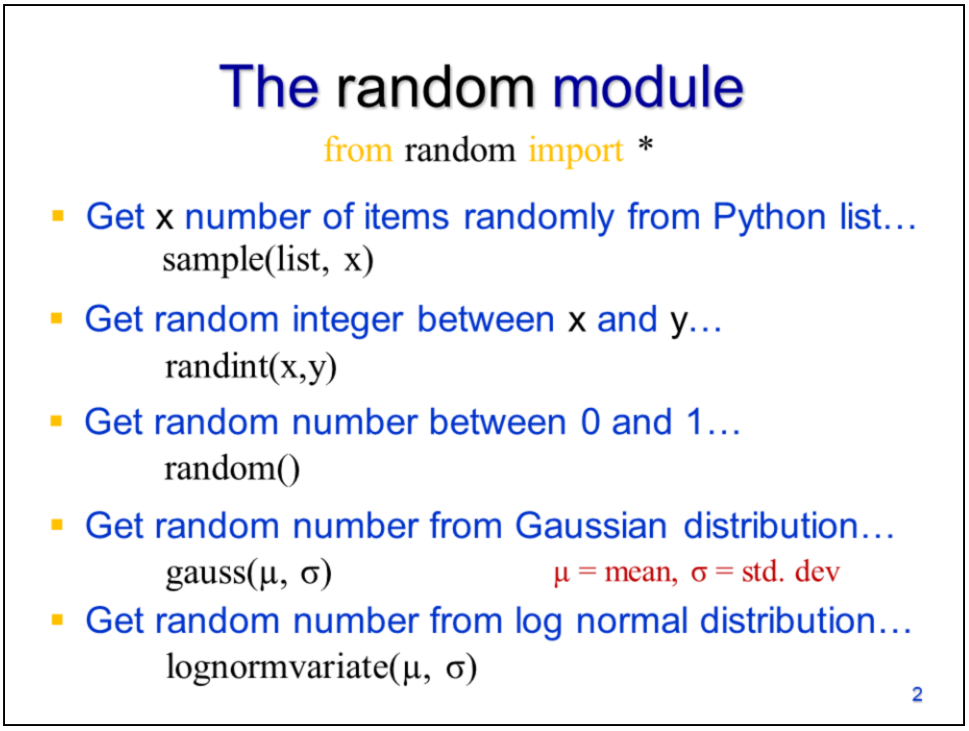


Random and zipfile modules

An introduction to common Python modules

1

This video will present tools from the Random and zipfile modules.



The random module

from random import *

- Get x number of items randomly from Python list…
  sample(list, x)
- Get random integer between x and y…
  randint(x,y)
- Get random number between 0 and 1…
  random()
- Get random number from Gaussian distribution…
  gauss(μ, σ)   μ = mean, σ = std. dev
- Get random number from log normal distribution…
  lognormvariate(μ, σ)

2

The **random** module provides tools for generating random numbers or taking random samples from a set of data.

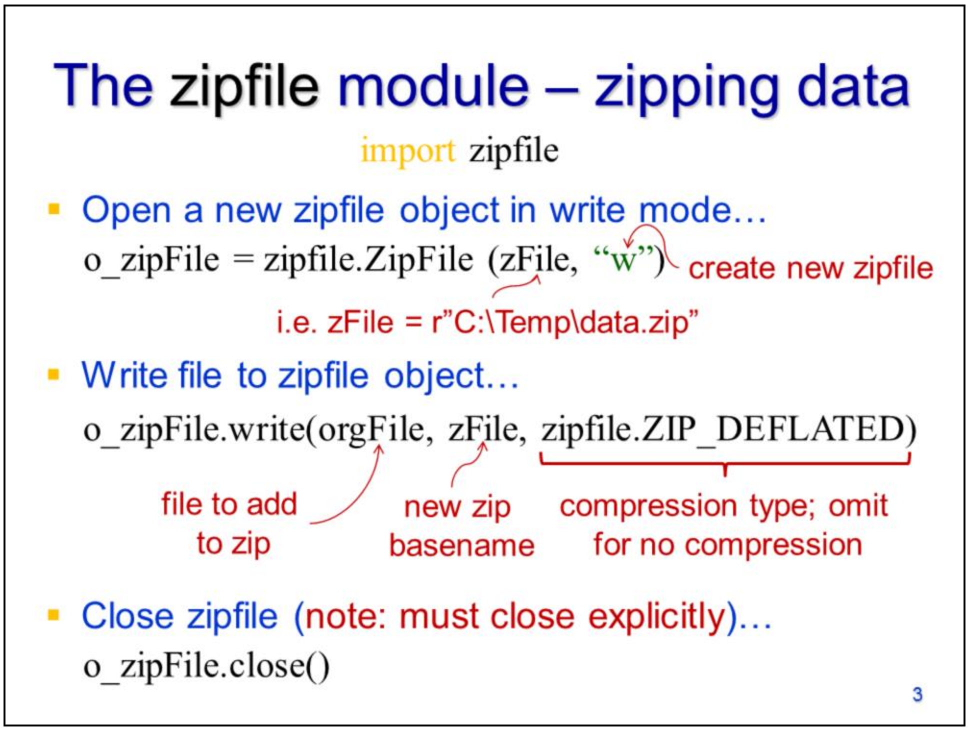
The **sample** tool will take a random sample of items from a list.

The **randint** tool will generate a random integer between the lower and upper limit that is specified.

The **random** tool will generate a random decimal number between 0 and 1. The random number can be scaled by multiplying the result by a whole number. For example, if you wanted to generate random decimal numbers between 0 and 100, you would multiply the output of the random tool by 100.

The **gauss** tool generates a random number from a Gaussian distribution

And the **lognormvariate** tool generates a random number from a log normal distribution. The random tool can generate numbers from several other distributions also – see module documentation for further information.



The zipfile module – zipping data

```
import zipfile
```

- Open a new zipfile object in write mode…

```
o_zipFile = zipfile.ZipFile (zFile, "w")
```

create new zipfile

i.e. zFile = r"C:\Temp\data.zip"

- Write file to zipfile object…

```
o_zipFile.write(orgFile, zFile, zipfile.ZIP_DEFLATED)
```

file to add to zip

new zip basename

compression type; omit for no compression

- Close zipfile (note: must close explicitly)…

```
o_zipFile.close()
```

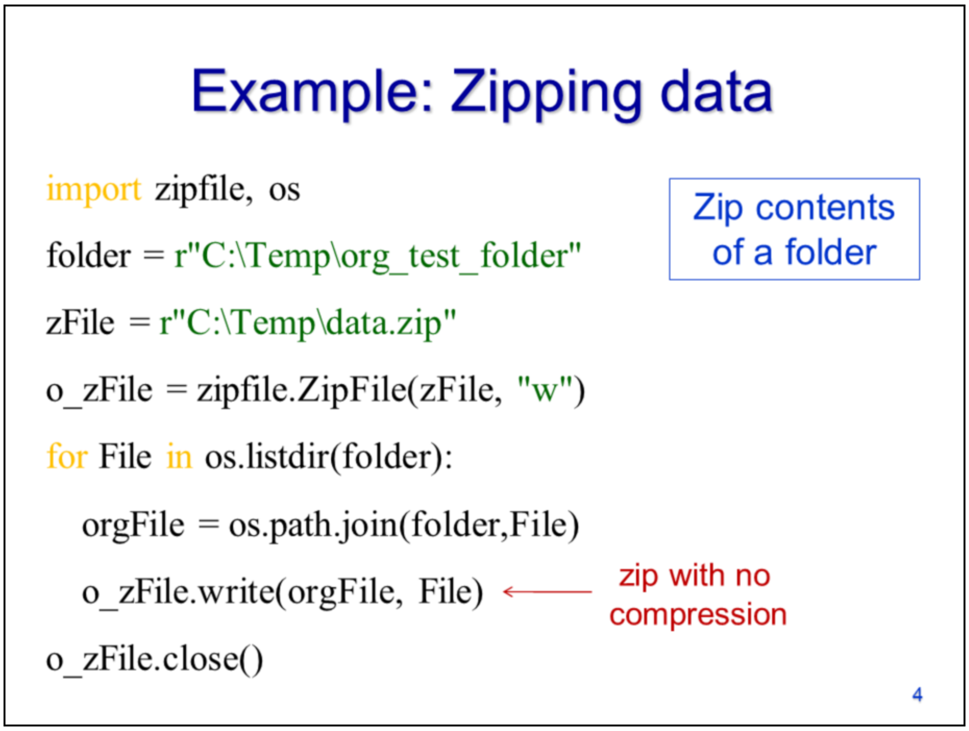
3

The **zipfile** module allows files to be zipped or unzipped. This module can only be used with files that have a .zip extension. Python's built-in **tarfile** module is very similar to the zipfile module and will work with .tar or .tar.gz files – these compression formats are common when working with certain remote sensing datasets (i..e Landsat). Other modules exist for use with different types of file compression and these modules can often be easily found online. In this video, we will focus on zipfiles.

To create a zipfile, use the **ZipFile** tool to create a new zipfile object in **write** mode. Note that the zip file name should have a .zip extension. Once the zipfile object is created, it can then be populated with files.

Use the **zipfile object's write** tool to add files to the zipfile. This tool requires you to specify the pathname of the file to be added to the zipfile as well as the basename that will be given the file after it is added to the zipfile. The type of compression may also be specified if you would like to reduce the zipfile size.

The zipfile must be closed after it is populated by using the zipfile objects **close** method.



Example: Zipping data

```
import zipfile, os
folder = r"C:\Temp\org_test_folder"
zFile  = r"C:\Temp\data.zip"
o_zFile = zipfile.ZipFile(zFile, "w")
for File in os.listdir(folder):
    orgFile = os.path.join(folder,File)
    o_zFile.write(orgFile, File)
o_zFile.close()
```

Zip contents of a folder

zip with no compression

4

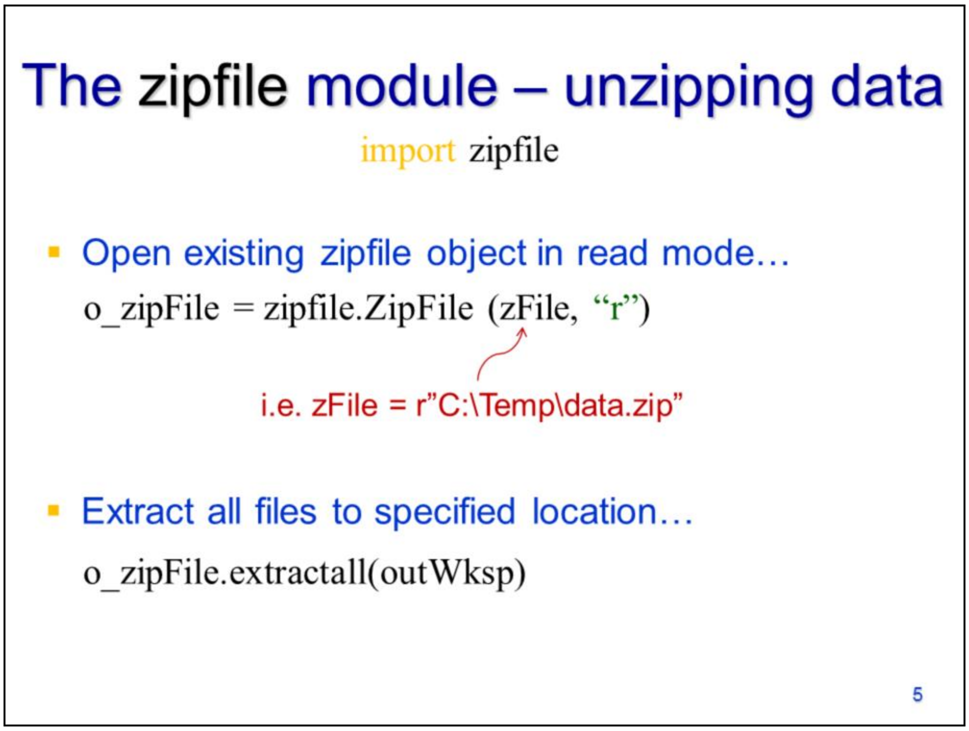
Let’s look at an example of a script that uses the zipfile module to create a zipfile. In this case, I will be zipping an entire folder. The output zipfile pathname is defined and includes a .zip extension. The **ZipFile** tool is used to create a zipfile object in write mode.

A **for loop** is then used to iterate through the contents of the folder containing the items that will be zipped. Recall that the os module’s listdir tool creates a list and so can be imbedded directly in the loop header line.

The os.path module’s join tool is used to join the workspace and the file basename.

The zipfile object’s write method is used to add the file to the zipfile object. In this case, I left the zipped version of the file with its original name. I did not specify a compression type for the zipfile so it will remain uncompressed.

After finished adding all files to the zipfile, the zipfile object is closed.



The zipfile module – unzipping data

```
import zipfile
```

- Open existing zipfile object in read mode…

```
o_zipFile = zipfile.ZipFile (zFile, "r")
```

i.e. zFile = r"C:\Temp\data.zip"

- Extract all files to specified location…

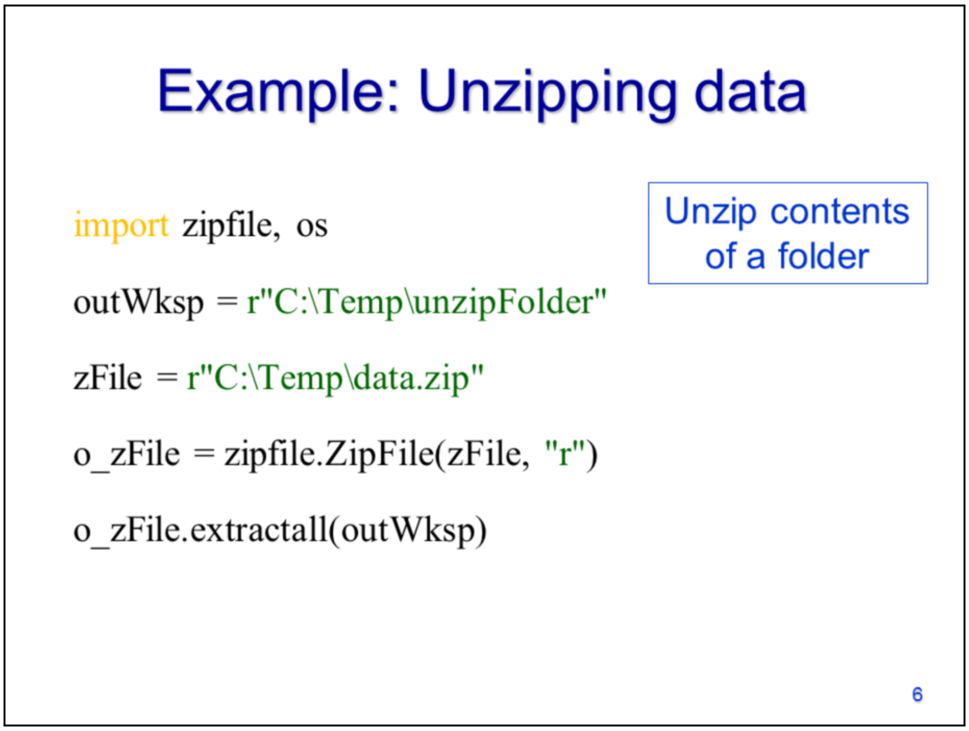
```
o_zipFile.extractall(outWksp)
```

5

The **zipfile** module can also unzip data from an existing zipfile.

When unzipping a file, the **ZipFile** tool should be used to create the zipfile object by opening the zipfile in **read** mode.

The zipfile object's **extractall** method can be used to extract the files to the specified output location.



Example: Unzipping data

Unzip contents of a folder

```
import zipfile, os
outWksp = r"C:\Temp\unzipFolder"
zFile = r"C:\Temp\data.zip"
o_zFile = zipfile.ZipFile(zFile, "r")
o_zFile.extractall(outWksp)
```

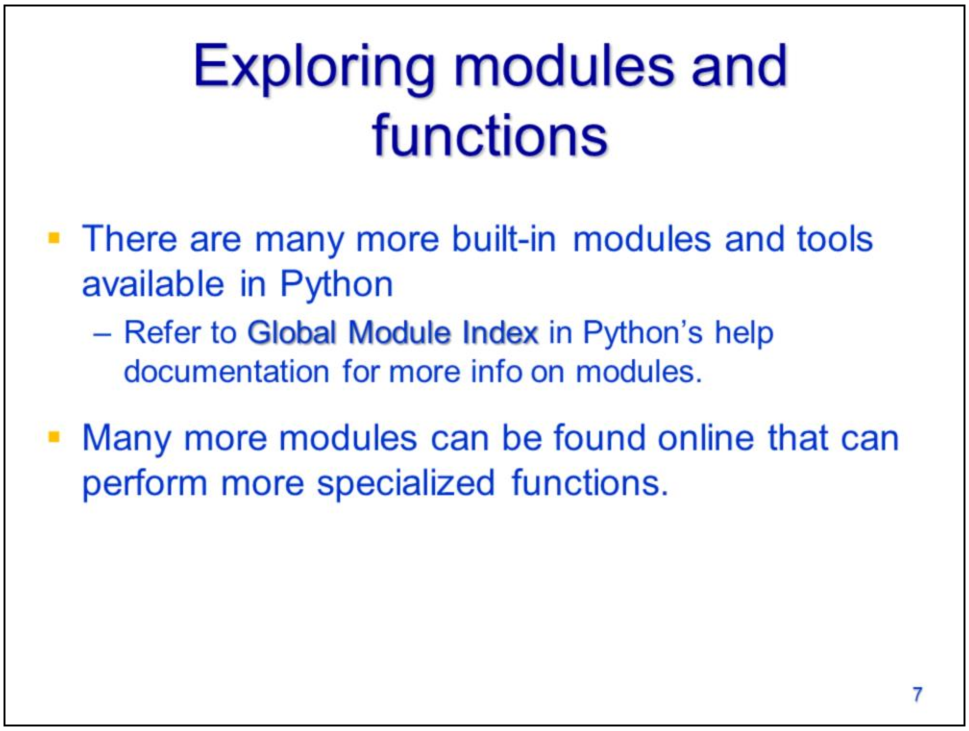
6

This example script demonstrates the process of unzipping a file.

Here I’ve define a zipfile that already exists.

The **ZipFile** tool is used to open the existing zipfile in **read** mode.

The zipfile object’s **extractall** method is used to extract all files to the output folder.



Exploring modules and functions

- There are many more built-in modules and tools available in Python
  - Refer to **Global Module Index** in Python's help documentation for more info on modules.
- Many more modules can be found online that can perform more specialized functions.

7

In the past 4 videos, we've looked at some commonly used tools from a handful of built-in modules. The modules that we explored have additional tools that we did not discuss. In addition, there are many more modules that we did not have time to explore. You can refer to the Python documentation to learn more about the capabilities of all built-in Python modules.

The Python community is very active and a quick online search will often find a solution to a task you are trying to perform.