External Modules

15-110 – Bonus Slides

Useful Modules

This set of bonus slides describes an array of modules that other Python programmers (and CMU students!) find useful.

You are not responsible for understanding any of these modules. Think of this more as a resource to consult if you decide you *want* to use any of them later on.

Module Index

Math: NumPy

Science: SciPy

Data Analysis: pandas, Matplotlib

Webscraping: Beautiful Soup

Websites: Django

Machine Learning: scikit-learn

Natural Language Processing: nltk

Images: PIL

3D Graphics: Panda3D

Audio: PyAudio

Game Design: Pygame

SciPy Collection

SciPy is a group of modules that support advanced mathematical and scientific operations. It can handle large calculations that might take the default Python operations too long to compute. We'll use this a little bit in the Data Analysis lectures.

The group includes NumPy (which focuses on math), SciPy (science), pandas (data analysis), and Matplotlib (plotting of charts and graphs). These can be used separately or as a group. Each need to be installed separately, but can be installed directly with `pip install name`

Website: https://www.scipy.org/

SciPy Collection Example

We'll show how to use Matplotlib in the Data Analysis II lecture. To learn about pandas, watch the following video: http://www.cs.cmu.edu/~15110-s20/hw/hw6_pandas.mp4

Here's a brief demo of running a t-test with Numpy and Scipy:

```
# Run a T-test on two random sets of data
import numpy, scipy
from scipy import stats
vals1 = numpy.random.random(1000) # generates 100 random numbers
vals2 = numpy.random.random(1000)
result = stats.ttest_ind(vals1, vals2)
print(result.pvalue)
```

Beautiful Soup

Beautiful Soup is a module that supports webscraping and HTML parsing. This is useful if you want to gather data from online for use in an application.

Website: https://www.crummy.com/software/BeautifulSoup/

Install:

```
pip install beautifulsoup4
```

Parse HTML as Tags

HTML organizes content on a page using **tags**, like this:

```
<tag attribute="value">
     <subtag> Some content for the subtag </subtag>
</tag>
```

To parse a website, you need to look for a certain type of tag in the file.

Beautiful Soup Example

```
import requests
from bs4 import BeautifulSoup

page = requests.get("https://www.cs.cmu.edu/~110/schedule.html")
soup = BeautifulSoup(page.content, 'html.parser')
for link in soup.find_all('a'):
    url = link["href"]
    if "slides/" in url and ".pdf" in url:
        print(link["href"])
```

Django

Django is a module that lets you build interactive websites using Python. This involves setting up a **frontend** (the part of a website that the user sees while browsing) and a **backend** (the part of a website that processes requests and does the actual work).

Website: https://www.djangoproject.com/

Install:

```
pip install django
```

scikit-learn

scikit-learn is a module that supports a large set of machine learning algorithms in Python. If you want to dabble in machine learning or artificial intelligence, this is a good place to start. Note that you'll still need to provide a starting dataset to get any algorithm to work.

Website: https://scikit-learn.org/stable/

Install:

```
pip install scikit-learn
```

scikit-learn Example

```
# Learn a decision tree from a random set of two-number data points
# that predict a third number
import numpy
import sklearn
from sklearn import tree
import matplotlib.pyplot as plt

trainingX = [ ]
for i in range(1000):
    trainingX.append(numpy.random.random(2)) # generates 1000 two-element random pairs
trainingY = numpy.random.random(1000)

regr = tree.DecisionTreeRegressor(max_depth=2)
regr.fit(trainingX, trainingY)

plt.figure()
tree.plot_tree(regr)
plt.show()
```

nltk

nltk, the Natural Language Toolkit, assists with natural language processing for machine learning purposes. This is useful whenever you're working with a corpus of written texts.

Website: https://www.nltk.org/

Install:

```
pip install nltk
```

nltk Example

```
# Identify the nouns in a document
import nltk
document = "insert example text here"
result = []
words = nltk.word_tokenize(document)
tags = nltk.pos_tag(words)
for tup in tags:
    [word, type] = tup
    if (word.lower() not in result) and (type == 'NN' or type == 'NNS'):
        result.append(word.lower())
result.sort()
print(result)
```

PIL: Python Imaging Library

PIL is a lightweight and easy-to-install module that lets `tkinter` interact with images other than .gif and .ppm files. It also includes functions that support basic image manipulation.

Website: http://www.pythonware.com/products/pil/

Since the main PIL installation is not maintained, most programmers use an offshoot called Pillow instead.

Website: https://pypi.org/project/Pillow/2.2.1/

Install:

```
pip install pillow
```

Panda3D

Panda3D is a module that supports 3D rendering and animation. Like `pyaudio`, it can be very complicated to install and use, but it is still much easier than trying to create 3D animation in a 2D system.

Website: https://www.panda3d.org/manual/

Install:

```
pip install Panda3D
```

PyAudio

PyAudio makes it possible to analyze, create, and play audio files. This module requires some complex pre-existing software, including the language C++; if you get an error message while installing, read it carefully to see how to make the installation work.

Website: https://people.csail.mit.edu/hubert/pyaudio/

Install:

```
pip install pyaudio
```

Note that there are many other audio modules available as well; you can find a list: https://wiki.python.org/moin/Audio

PyAudio Example

```
import pyaudio, math
# Make the sound data
bitrate, freq = 64000, 130.815 # C3 frequency
dataFrames = [""] * 5
for octave in range(5):
    mult = 2*math.pi*(freq*(octave+1))
    for frame in range(bitrate):
        dataFrames[octave] += chr(int(math.sin(mult * frame / bitrate)*127+128))
# Play the sounds!
p = pyaudio.PyAudio()
stream = p.open(format=32, channels = 1, rate = bitrate, output = True)
for frame in dataFrames:
    stream.write(frame)
# Close the stream
stream.stop_stream()
stream.close()
p.terminate()
```

Pygame

Pygame is, like tkinter, a library that lets you make graphical applications. However, Pygame is specifically designed to create games. It has better support for sprites and collision detection than `tkinter`.

Website: https://www.pygame.org/news

Install:

```
pip install pygame
```