



COMP2720: Automating Tools for New Media

Dealing with lots of files at once.
Adding more capabilities:
Modules



Processing many files at once

- Sometimes you want to process a bunch of files without knowing the name of every one.
 - Putting a title on every picture in a directory.
 - Creating an index page that references every picture and sound in a directory.
 - Processing every frame in a video, where each frame is stored as a JPEG frame.

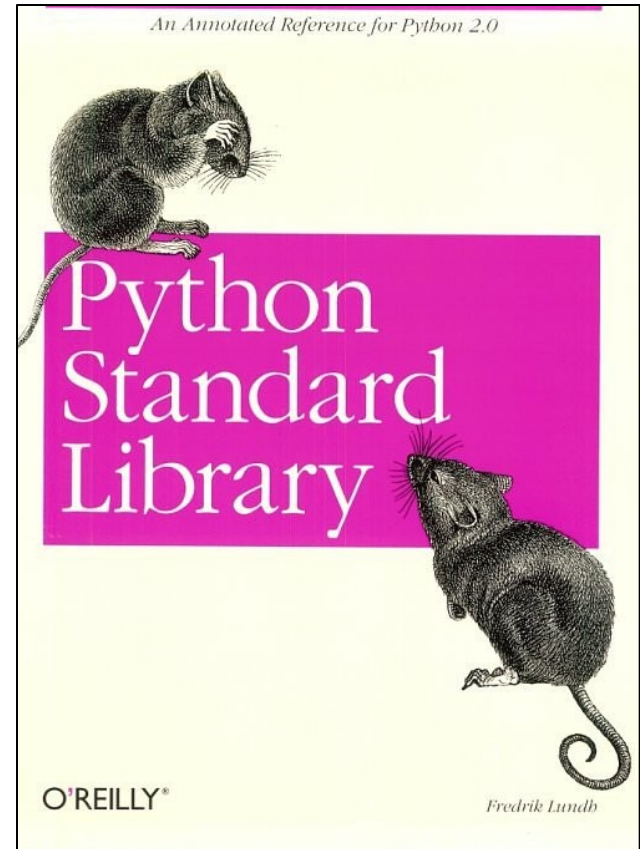


Adding new capabilities: Modules

- What we need to do is to add capabilities to Python that we haven't seen so far.
- We do this by importing *external modules*.
- A module is a file with additional functions and objects defined within it.
 - Some kind of module capability exists in virtually every programming language.
- By *importing* the module, we make the module's capabilities available to our program.
 - Literally, in Python we are evaluating the module, as if we'd typed them into our file.

Python's standard library

- Python has an extensive library of modules that come with it.
- The *Python standard library* includes modules that allow us to access the Internet, deal with time, generate random numbers, and... access files in a directory.
- See module documentation at <http://docs.python.org/modindex>





Accessing pieces of a module

- We access the additional capabilities of a module using *dot notation*, after we import the module.
- How do you know what pieces there are?
 - Check the documentation.
 - Python comes with a Library Guide.
 - There are books like *Python Standard Library* that describe the modules and provide examples.



The `os` (operating system) module

- The `os` module offers a number of powerful capabilities for dealing with files, e.g., renaming files, finding out when a file was last modified, and so on.
- We start accessing the `os` module by typing:
`import os`
- The function that knows about directories is `listdir()`, used as `os.listdir()`
 - `listdir()` takes a path to a directory as input.

Using os.listdir()

```
>>> import os
```

```
>>> print getMediaPath("barbara.jpg")
```

```
H:\comp2720\mediafiles\barbara.jpg
```

```
>>> print getMediaPath("pics")
```

Note: There is no file at H:\comp2720\mediafiles\pics

```
>>> print os.listdir("H:\comp2720\mediafiles")
```

```
['aah.wav', 'barbara.jpg', 'bassoon-c4.wav', 'bassoon-e4.wav', 'bassoon-g4.wav',  
'diving-01.jpg', 'diving-02.jpg', 'diving-03.pg', ...]
```

Note: mediafiles is a folder



Writing a program to title pictures

- We'll input a directory
- We'll use `os.listdir()` to get each filename in the directory.
 - We'll open the file as a picture.
 - We'll title it (add some text to it).
 - We'll save it out as "titled-" and the filename.

Titling pictures

```
import os
```

```
def titleDirectory(dir):
```

```
    for filename in os.listdir(dir):
```

```
        picture = makePicture(dir+filename)
```

```
        addText(picture,10,10,"This is from COMP2720 in 2006")
```

```
        writePictureTo(picture,dir + "titled-"+filename)
```

Okay, that didn't work

```
>>> titleDirectory("H:\comp2720\mediafiles")
```

```
makePicture(filename): There is no file at  
H:\comp2720\mediafilesdiving-01.jpg
```

An error occurred attempting to pass an argument to a function.

Why not?

- Is there a file where we tried to open the picture?
- Actually, no. Look at the output of `os.listdir()` again

```
>>> print os.listdir("H:\comp2720\mediafiles")
```

```
['aah.wav', 'barbara.jpg', 'bassoon-c4.wav', 'bassoon-e4.wav', 'bassoon-g4.wav',  
 ... , 'diving-01.jpg', 'diving-02.jpg', 'diving-03.pg', ...]
```

- The strings in the list are just the *base file names*.
 - No path

Creating the path

- If the directory string is in the placeholder variable `dir`, then `dir+file` is the full pathname, right?
- Close — you still need a *path delimiter*, like “/”
 - But it’s different for each platform (Mac, Windows, Unix)!
 - Python gives us a notation that works: “//” is as a path delimiter for any platform.
 - So: `dir+”//”+file`

A Working titling program

```
import os
```

```
def titleDirectory(dir):
```

```
    for filename in os.listdir(dir):
```

```
        print "Processing:",dir+"//"+filename
```

```
        picture = makePicture(dir+"//"+filename)
```

```
        addText(picture,10,10,"This is from COMP2720 in 2006")
```

```
        writePictureTo(picture,dir+"//"+"titled-"+filename)
```

Showing it work

```
>>> titleDirectory("H:\comp2720\mediafiles")  
Processing: H:\comp2720\mediafiles\diving-01.jpg  
Processing: H:\comp2720\mediafiles\diving-02.jpg  
Processing: H:\comp2720\mediafiles\diving-03.jpg
```

...

```
>>> print os.listdir("H:\comp2720\mediafiles")  
['diving-01.jpg', 'diving-02.jpg', 'diving-03.jpg', ...  
 'titled-diving-01.jpg', 'titled-diving-02.jpg', 'titled-diving-03.jpg', ...]
```

Idea: Inserting a copyright on pictures



What if you want to make sure you've got JPEG files?

```
import os
```

```
def titleDirectory(dir):
```

```
    for file in os.listdir(dir):
```

```
        print "Processing:",dir+"//"+file
```

```
        if file.endswith(".jpg"):
```

```
            picture = makePicture(dir+"//"+file)
```

```
            addText(picture,10,10,"This is from COMP2720 in 2006")
```

```
            writePictureTo(picture,dir+"//"+"titled-"+file)
```

An interesting module: random

```
>>> import random
>>> for i in range(1,10):
...     print random.random()
...
0.8211369314193928
0.6354266779703246
0.9460060163520159
0.904615696559684
0.33500464463254187
0.08124982126940594
0.0711481376807015
0.7255217307346048
0.2920541211845866
```

Randomly choosing words from a list

```
>>> for i in range(1,5):
...     print random.choice(["Here", "is", "a", "list", "of",
...                           "words", "in", "random", "order"])
...
list
a
Here
list
```



Randomly generating language

- Given a list of nouns, verbs that agree in tense and number, and object phrases that all match the verb,
- We can randomly take one from each to make sentences.

Random sentence generator

```
import random
```

```
def sentence():
```

```
    nouns = ["Mark", "Adam", "Angela", "Larry", "Jose", "Matt",  
            "Jim"]
```

```
    verbs = ["runs", "skips", "sings", "leaps", "jumps", "climbs",  
            "argues", "giggles"]
```

```
    phrases = ["in a tree", "over a log", "very loudly", "around the  
              bush", "while reading the Technique"]
```

```
    phrases = phrases + ["very badly", "while skipping", "instead of  
                        grading", "while typing on Phorum."]
```

```
    print random.choice(nouns), random.choice(verbs),  
          random.choice(phrases)
```

Running the sentence generator

```
>>> sentence()
Jose leaps while reading the Technique
>>> sentence()
Jim skips while typing on Phorum.
>>> sentence()
Matt sings very loudly
>>> sentence()
Adam sings in a tree
>>> sentence()
Adam sings around the bush
>>> sentence()
Angela runs while typing on Phorum.
>>> sentence()
Angela sings around the bush
>>> sentence()
Jose runs very badly
```

How much smarter can we make this?

- Can we have different kinds of lists so that, depending on the noun selected, picks the right verb list to get a match in tense and number?
- How about reading input from the user, picking out key words, then generating an “appropriate response”?

```
if input.find(“mother”) <> -1:
```

```
    print “Tell me more about your mother...”
```



Turing's Test & Joseph Weizenbaum's "Eliza"

- Created a program that acted like a therapist.
 - Echoing back to the user whatever they said, as a question.
 - It had rules that triggered on key words in the user's statements.
 - It had a little memory of what it had said before.
- People really believed it was a real therapist!
 - Convinced Weizenbaum of the dangers of computing.



Session with the “Doctor”

>>>My mother bothers me.

Tell me something about your family.

>>>My father was a caterpillar.

You seem to dwell on your family.

>>>My job isn't good either.

Is it because of your plans that you say your job is not good either?

You can import your own files

- Imagine that you have a function named `posterise(picture)` in a file named `hw1.py`
- You can import your own function treating your file as a module

```
import hw1
```

```
hw1.posterise(picture) # That'll work
```

- What if you don't want to use dot notation?

```
from hw1 import posterise
```

Or alternatively

```
from hw1 import *
```

```
posterise(picture) # Will work now
```

The `time` module

- We often need to know what time is it to use inside the program.
- For this we use the Python module called `time`

```
import time
def showTime():
    print time.localtime()
    #this function returns a tuple
>>>showTime()
(2007, 8, 23, 18, 6, 52, 3, 235, 0)
    year  month  dayOM  hour  min  sec  dayOW  dayOY  isDLS
```