Lecture 4 - Lists, Files, Modules, Input Output.

First An Intro for Veronica

Impact 307 - Funding

A little note on Starting Businesses

Code Re usability / Functions

It would be really long and error prone to have a program where you put in all the values into the code and every calculation was inline. That is what we did in the Jupyter Notebook. In the previous class we created a “function” that allowed us to convert from miles to kilometers. It took some steps to build this. We started out with the inline code and then slowly evolved it into a function and added tests to verify that it worked.

To create a function you use the Python `def` followed by a space and a name for the function. The name should start with a letter, a..z, then you can have letters or digits and underscore characters, `_` . Then you have an open parenthesis, `(` and a list of name of parameters, then a close parenthesis, `)` and a colon `:` .

The list of parameters is used in an order dependent way.

Let’s build a simple function that calculates the length of the hypotenuse of a right triangle.

```
import math

def hypotenuse ( a, b ) :
    h = math.sqrt ( ( a * a ) + ( b * b ) )
    return h

print ( hypotenuse ( 3, 4 ) )
```

Let’s try it with some variables:

```
height = 6
width = 8
print ( hypotenuse ( height, width ) )

hh = 10
```

```
ww = 22
print ( hypotenuse ( hh, ww ) )

height = 3
width = 4
print ( hypotenuse ( height, width ) )
```

Lists. Chapter 3 in the book.

Life would be very boring if all we could do was to work on one thin at once. That would be like what you are stuck in in most GUI applications - one document, one image. There are very good tools for each of these. For example Adobe Photoshop - part of the creative suite - is fantastic and working on 1 image at once. I use it.

A couple of years ago a business hear in town was adding water marks to images. They had an intern, being payed $15 an hour, open an image, drag the watermark onto the image in the bottom right, save the image. They had 2,500,000 images to process. At my best estimate this person would finish this task in 3.4 years working full time. Long before that time his break would turn to goo and he would get mouse shoulder and fall off of his chair in total abject boredom.

I used a 20 line Python program to open the image file, open the water mark file, position it, paint the watermark onto the image, save the file. It started out with a list of files, 2.5 million of them, and in about 35 minutes it has a perfectly positioned watermark on every one.

One of the super-powers of computers is to work at scale. Amazon's authentication service performs 400,000,000 requests a second. That is scale.

A list in python.

```
cars = ['Tesla','Bmw','Mercedes-Benz','Aion']
cars
print(cars)
cars.sort()
print(cars)
```

An important note about upper/lower case characters.

```
cars = ['Tesla','bmw','Mercedes-Benz','Aion']
cars
print(cars)
```

```
cars.sort()
print(cars)
```

What are lists - syntax for them.

Use square for lists, round for tuples.

```
list1 = [ "abc", 3 ]
```

Accessing the elements in a list.

```
a = list1[0]
list1[0] = "def"
list1.remove ( 3 )
list1.append ( 4 )
print ( list1 )
del list1[1:2]
print ( list1 )
```

The value inside the `list1[0]` - in the square brackets is often referred to as the subscript.

List Comprehensions

```
list1 = [ 4, 2, 20, 1,0,10,3 ]

l2 = [ i for i in list1 if i < 10 ]
print ( l2 )

for i in range(10):
        print i

sqr = [ i*i for i in range(10) ]
sqr2 = [ i**2 for i in range(10) ]

obj = ["Even" if i%2==0 else "Odd" for i in range(10)]
print(obj)
```

Let's build a list program.

1. Let's create a directory for the program to be in.
2. An input file to read from.
3. The code -- open the file, read the file, return the list.
4. Let's test this chunk.
5. Let's print out all the even numbers in the list.

```
$ pwd
$ mkdir ex1
$ cd ex1
```

```
$ vi data1.txt
```

```
$ vi ex1.py
```

Let's build a string - list processor.

Strings are list of characters.

```
$ pwd
$ cd ..
$ mkdir ex2
$ cd ex2
```

1. Switch directories.
2. Let's create a directory for the program to be in.
3. An input file to read from.
4. The code -- open the file, read the file, return the list of strings.
5. Let's test this chunk.
6. Let's count the characters on each line.

Let's build a matrix add.

You can have lists inside lists.

```
$ pwd
$ cd ..
$ mkdir ex3
$ cd ex3
```

1. Switch directories.
2. Let's create a directory for the program to be in.
3. Read in a matrix.
4. Read in a 2nd matrix (same code)
5. See if an appropriate size.
6. A function to add matrices.
7. Output a matrix.

Copyright