

Slide 1: Title slide for "The I'm in Cali and you're not Lecture" by Alan C Jamieson, PhD. The slide features a dark header with navigation icons (home, back, forward) and a light gray background with a dark footer.

The I'm in Cali and you're not Lecture

Alan C Jamieson, PhD

Slide 2: Lab Solution for Question 1: Factorial. The slide features a dark header with navigation icons and a light gray background with a dark footer.

Lab Solution

- Question 1: Factorial

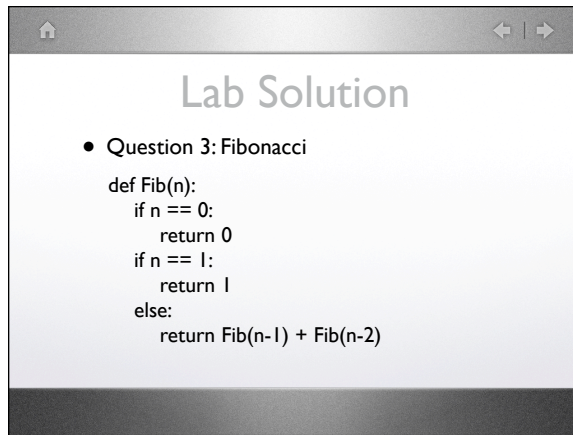
```
def Fact(n):  
    result = 1  
    for i in range(n):  
        result = result * (i+1)  
    return result
```

Slide 3: Lab Solution for Question 2: Summation. The slide features a dark header with navigation icons and a light gray background with a dark footer.

Lab Solution

- Question 2: Summation

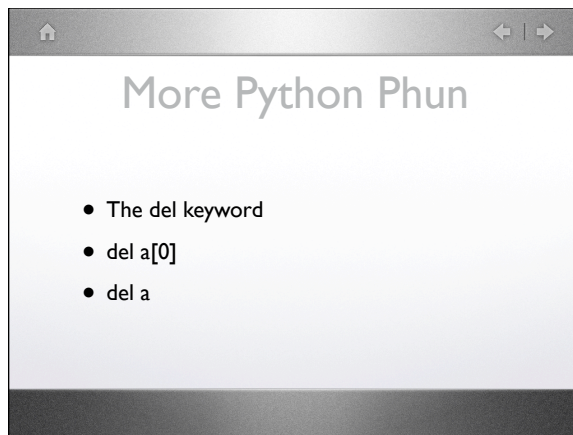
```
def Sum(n):  
    result = 0  
    for i in range(n):  
        result = result + i + 1  
    return result
```



Lab Solution

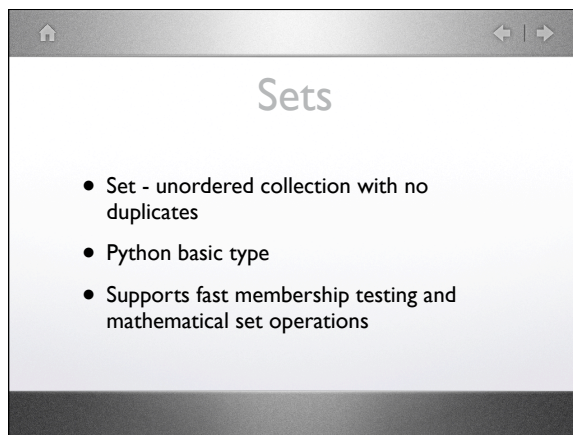
- Question 3: Fibonacci

```
def Fib(n):  
    if n == 0:  
        return 0  
    if n == 1:  
        return 1  
    else:  
        return Fib(n-1) + Fib(n-2)
```



More Python Phun

- The del keyword
- del a[0]
- del a



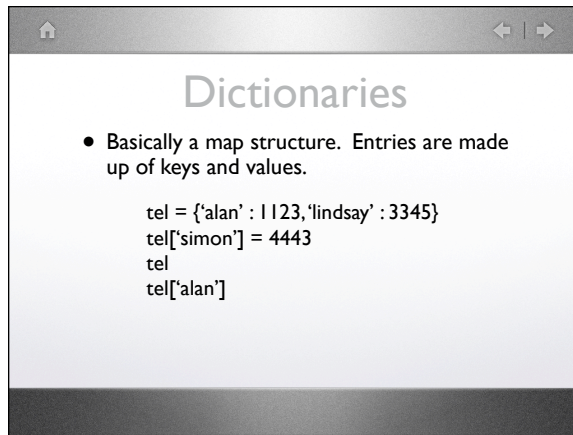
Sets

- Set - unordered collection with no duplicates
- Python basic type
- Supports fast membership testing and mathematical set operations

Slide titled "Sets" with a home icon and navigation arrows at the top. The text reads: "Load up IDLE (or a Python interpreter of your choice) Try the following:" followed by the code: `a = set('hi how are you')`, `basket = ['apple', 'orange', 'apple', 'pear', 'orange', 'banana']`, and `fruit = set(basket)`.

Slide titled "Sets" with a home icon and navigation arrows at the top. It contains two bullet points: "• Notice that Python takes care of duplicates for you. The 'hi how are you' example shows that Python will even split up a string into characters and eliminate duplicates that way." and "• Fast membership testing - 'orange' in fruit".

Slide titled "Sets" with a home icon and navigation arrows at the top. It contains a bullet point: "• Mathematical Set Functions" followed by the code: `a = set('hi, how are you?')` and `b = set('not bad, you?')`. Below this is a list of operators and their meanings: `a - b` #a but not b, `a | b` #a or b, `a & b` #a and b, and `a ^ b` #a or b but not both.

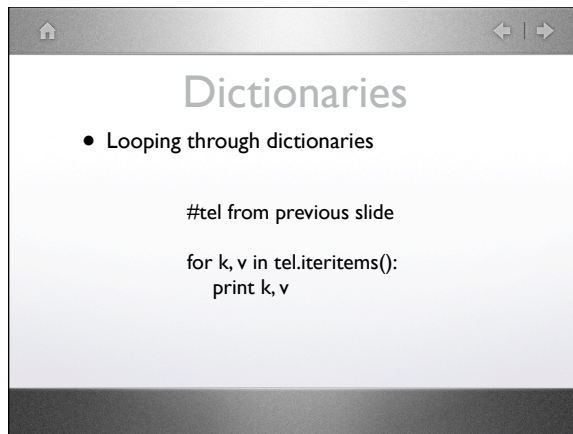


A presentation slide titled "Dictionaries". It features a header bar with a home icon and navigation arrows. The main content includes a bullet point describing dictionaries as map structures and a code snippet for a dictionary named 'tel'.

Dictionaries

- Basically a map structure. Entries are made up of keys and values.

```
tel = {'alan' : 1123, 'lindsay' : 3345}
tel['simon'] = 4443
tel
tel['alan']
```

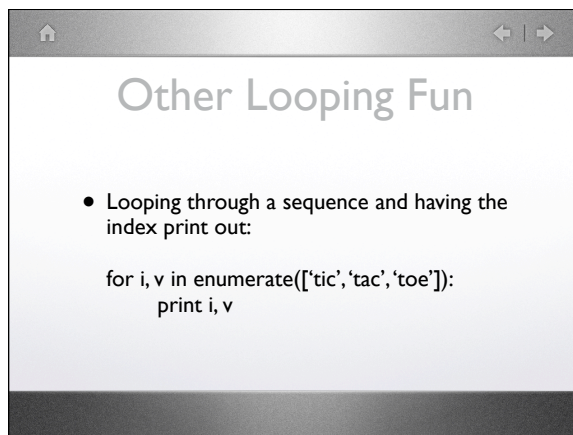


A presentation slide titled "Dictionaries". It features a header bar with a home icon and navigation arrows. The main content includes a bullet point about looping through dictionaries and a code snippet using 'tel.iteritems()'.

Dictionaries

- Looping through dictionaries

```
#tel from previous slide
for k, v in tel.iteritems():
    print k, v
```



A presentation slide titled "Other Looping Fun". It features a header bar with a home icon and navigation arrows. The main content includes a bullet point about looping through a sequence with indices and a code snippet using 'enumerate()'.

Other Looping Fun

- Looping through a sequence and having the index print out:

```
for i, v in enumerate(['tic', 'tac', 'toe']):
    print i, v
```

Other Looping Fun

- Looping through two or more sequences at the same time:

```
questions = ['name', 'quest', 'favorite color']
answers = ['lancelot', 'the holy grail', 'blue']
for q, a in zip(questions, answers):
    print 'What is your %s? It is %s.' % (q, a)
```

Other Looping Fun

- Reversed and/or non-1 increments

```
for i in reversed(xrange(1, 10, 2)):
    print i
```

Modules

- We've already had a look at modules.
- Any file that we have defined functions, etc. in is considered a module.
- `import modulename`
- `modulename.functionname(params)`
- `dir(modulename)`

Modules

- Multiple modules packaged together are considered a package.
- Provides structure
- `import moduleA.moduleB`
- importing utilizing `*` (does not work in Windows!)

Standard Library

- `os` module (`import os`)
- large module, specifically designed for interaction with OS via command line.

```
os.getcwd()
os.chdir()
```

Standard Library

- `glob` (`import glob`)
- designed to handle pathname pattern expansions
- Wildcard searches to the OS

```
glob.glob("*.py")
```

Standard Library

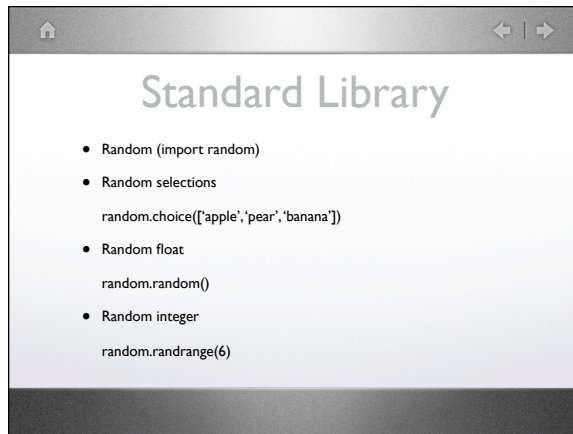
- Command Line Arguments (`import sys`)
`print sys.argv`
- `argv` is a list of strings

Standard Library

- Regular Expression pattern matching (`import re`)
`re.findall(r'\b[a-z]*', 'which foot or hand fell fastest')`
- When only simple capabilities are needed, you should use the String functionalities

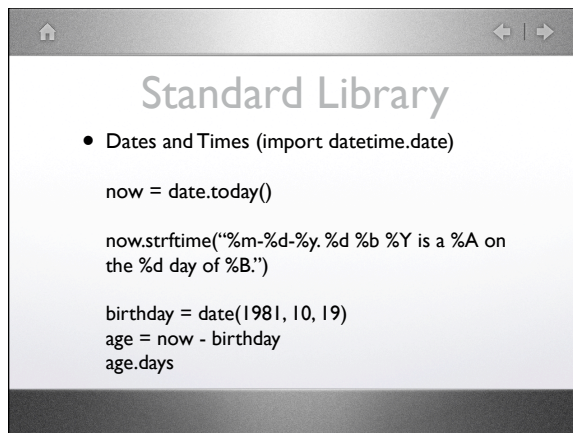
Standard Library

- Math (`import math`)
- Gives access to the C mathematics library
`math.cos(math.pi / 4.0)`
`math.log(1024, 2)`



Standard Library

- Random (import random)
- Random selections
`random.choice(['apple', 'pear', 'banana'])`
- Random float
`random.random()`
- Random integer
`random.randrange(6)`



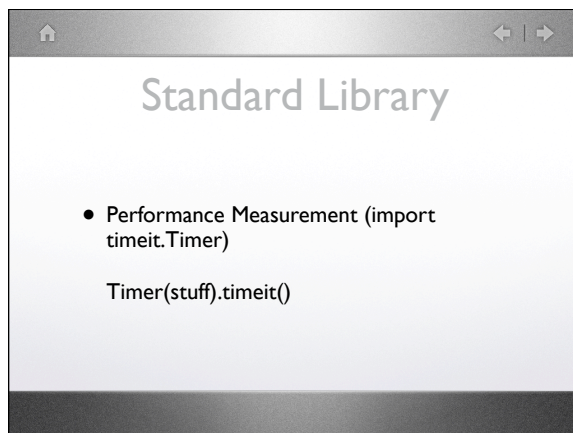
Standard Library

- Dates and Times (import datetime.date)

```
now = date.today()

now.strftime("%m-%d-%y. %d %b %Y is a %A on
the %d day of %B.")

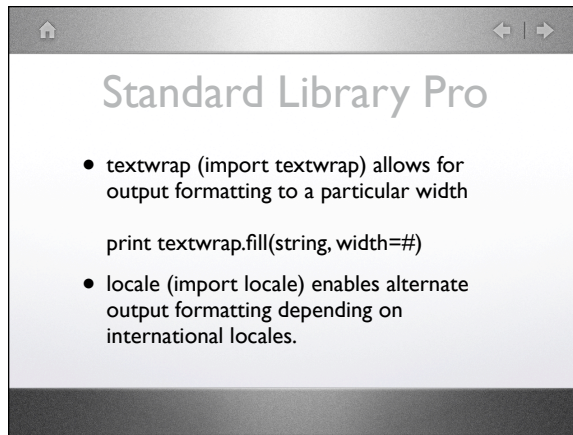
birthday = date(1981, 10, 19)
age = now - birthday
age.days
```



Standard Library

- Performance Measurement (import timeit.Timer)

```
Timer(stuff).timeit()
```

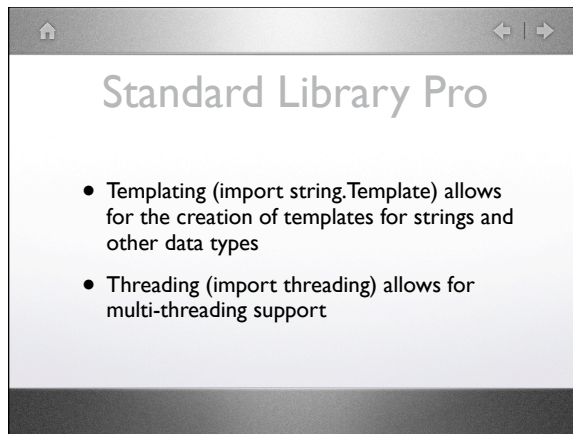
Standard Library Pro

- `textwrap` (`import textwrap`) allows for output formatting to a particular width

```
print textwrap.fill(string, width=##)
```

- `locale` (`import locale`) enables alternate output formatting depending on international locales.

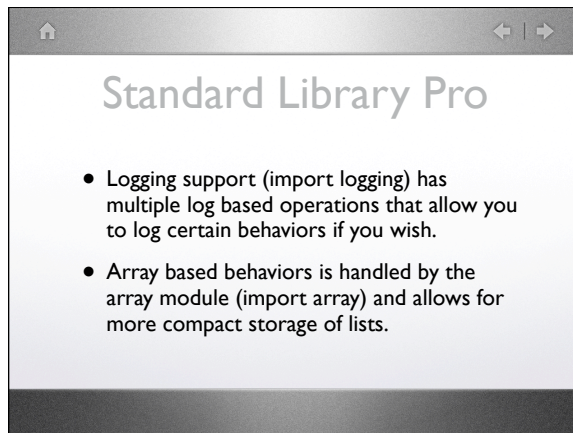
This slide features a title bar with a home icon and navigation arrows. The content is presented on a light gray background with a dark gray footer.



Standard Library Pro

- `Templating` (`import string.Template`) allows for the creation of templates for strings and other data types
- `Threading` (`import threading`) allows for multi-threading support

This slide features a title bar with a home icon and navigation arrows. The content is presented on a light gray background with a dark gray footer.



Standard Library Pro

- `Logging support` (`import logging`) has multiple log based operations that allow you to log certain behaviors if you wish.
- `Array based behaviors` is handled by the `array` module (`import array`) and allows for more compact storage of lists.

This slide features a title bar with a home icon and navigation arrows. The content is presented on a light gray background with a dark gray footer.

Standard Library

- Other uses:
 - Data Compression (using gzip, tarfile, etc)
 - Network functionality (TCP/IP, sockets)
 - Quality Control and Documentation

Monday

- Classes and other Object Oriented Constructs
- Lab solution
- Maybe the project solution

See YA!