Topics for Today

- Strings
- Lists
- Arrays (NumPy library)

Problem

- From a list of birds, create a formatted table of all those with “hawk” in their names. Include the order, genus, and species in the table.
- Tools
  - Open file, read lines
  - Split lines into lists of elements
  - Access list elements
  - Convert strings to lower case, search for substrings
  - Use a tuple
  - Format a string
Basic Operations on Strings

- Indexing (or, subscripting) and slicing
- Forms...
  - $s[i]$ – returns $i$th character in $s$ (0-based index)
  - $s[i:j]$ – returns slice of $s$ starting at $i$ until $j$
- Helpful to think of slice positions being between characters
- Other operations...
  - +: concatenation
  - *: repetition ($s * 3$)
  - len($s$)
  - for $c$ in $s$: ...

Slice Examples

- H E L L O
- 0 1 2 3 4 5
- -5 -4 -3 -2 -1

- $S[0:2] = \text{"HE"}$
- $S[-3:-1] = \text{"LL"}$
- Slice indexes out of range equivalent to lowest or highest valid index.
- Fortunately, many operations are built in...
Library Operations on Strings

capitalize("hello there")   -> 'Hello there'
capwords("hello there")   -> 'Hello There'
center("hello there", 20)  -> 'hello there'
count("x y z", " ")       -> '2'
find("hello", "l")        -> '2'
join(["a", "b", "c"] )   -> 'a b c'
ljust("hello", 10)         -> 'hello'
lower("HELLO")            -> 'hello'
lstrip("   hello")         -> 'hello'
replace("hello", "l", "w") -> 'hewwo'
rfind("hello", "l")       -> '3'
rjust("hello", 10)         -> 'hello'
rstrip("hello  ", " ")    -> 'hello'
split("a b c d", " ")     -> ['a', 'b', 'c', 'd']
upper("hello")            -> 'HELLO'

Demo: Generating this slide

from string import *
l = [
    'capitalize("hello there")',
    'capwords("hello there")',
    'center("hello there", 20)',
    'count("x y z", " ")',
    'find("hello", "l")',
    'join(["a", "b", "c"] )',
    'ljust("hello", 10)',
    'lower("HELLO")',
    'lstrip("   hello")',
    'replace("hello", "l", "w")',
    'rfind("hello", "l")',
    'rjust("hello", 10)',
    'rstrip("hello  ", " ")',
    'split("a b c d", " ")',
    'upper("hello")'
]
for s in l:
    print ljust(s, 27), "->" + str(eval(s)) + ""

Generating the Previous Slide

from string import *
l = [
    'capitalize("hello there")',
    'capwords("hello there")',
    'center("hello there", 20)',
    'count("x y z", " ")',
    'find("hello", "l")',
    'join(["a", "b", "c"] )',
    'ljust("hello", 10)',
    'lower("HELLO")',
    'lstrip("   hello")',
    'replace("hello", "l", "w")',
    'rfind("hello", "l")',
    'rjust("hello", 10)',
    'rstrip("hello  ", " ")',
    'split("a b c d", " ")',
    'upper("hello")'
]
for s in l:
    print ljust(s, 27), "->" + str(eval(s)) + ""
Lists

- Sequence of arbitrary values of arbitrary types
- Very similar operations as on strings
  - Indexing and slicing
  - +: concatenation
  - *: repetition (l * 3)
  - len(l)
  - for i in l: …
- Other operations, e.g.,
  - a.append(e); compare: a + e and a.append(l)
- And the big difference compared to strings…

Mutability

- Lists are mutable, strings are not
- Values can be changed by assigning to index or slice
- All references to that value see the change
- Example:
  
  ```python
  x = ["hello", "there", "alice"]
  y = x
  x[2] = "bob"
  x
  y
  ```
- Compare:
  
  ```python
  s = "hello"
  s[2] = "a"
  ```
Arrays

- From the numpy library: array, zeros, ones
- Efficient, math-oriented implementation of lists and more
- All elements are of same data type, declared in advance
- Support multiple dimensions, reshaping dimensions, etc.
- Simple example
  ```python
  from numpy import *
  x = zeros(31)
  p = 1
  for i in range(len(x)):
    x[i] = p/100.0
    p = 2*p
  print x
  ```

Demo: Doubling your money

Array Features

- Can specify data type of elements
  - `x = zeros(100, dtype=int16)`
  - 16 bit integers
  - Used for sound samples in Project 1
- Useful functions
  - `min(x)`
  - `max(x)`
  - `append(x, y)` returns new array that is `x` concatenated with `y` (x + y does element-by-element addition!)
Conditionals

- Another fundamental thing programs can do: test
- Conditional expressions
  - Compare values
  - Perform logical (Boolean: true/false) operations
- Numeric comparisons:
  - $x < y$, $x \leq y$, $x == y$, $x > y$, $x \geq y$, $x != y$
  - In “conditional context”: 0 is False, non-zero is True
- Boolean comparisons: and, or, not
- Conditional context examples:
  - if {condition}:
  - while {condition}:

Example

```python
# Find first Fibonacci number greater than n
# Fibonacci sequence: 1, 1, 2, 3, 5, 8, 13, 21, ...

def fibn(n):
    if 1 > n:
        return 1
    f0 = 1
    f1 = 1
    f = f0 + f1
    while f <= n:
        f0, f1 = f1, f
        f = f0 + f1
        print f
    return f
```

- Warning: Watch for “fencepost errors”!
Clicker Question

Which list does **not** have five elements?

A. range(5)
B. range(1,6)
C. range(10, 100, 20)
D. range(1,5)
E. range(10, 0, -2)

Looping Efficiency

- Standard looping technique…
  - for i in range(100000000):
    - Must create very large list, taking time and memory

- Alternative…
  - for i in xrange(100000000):
    - No list is created, for loop handles “iterator” efficiently