Data Analysis

Mutability
Coursework
hw03 due Friday 2/17
Gradebook resources: Compass and RELATE
Recap: for loops
def total_length(words):
    total = 0
    for word in words:
        total += len(word)
    return total

total_length(['red', 'green', 'blue'])  # => 12
def word_lengths(words):
    lengths = ____
    for word in words:
        lengths ____
    return lengths

word_lengths(["red", "green", "blue"]) # => [3,
def concatenate_all(words):
    result = ____
    for ____ in ____:
        ____result
    return result

concatenate_all(["red", "green", "blue"]) # => "
Recap: for loops

def acronymize(words):
    acronymize(["red", "green", "blue"]) # => "RGB"
Functions can accept defaults for arguments.

def within_range( val1, val2, abstol=1e-3 ):
    return abs( val1 - val2 ) <= abstol

within_range( 1, 1.001 )
within_range( 1, 1.0001, abstol=1e-2 )

You must include the argument name in this case!
A Quick Quiz
s = 'ABcd'
if not s[0:2].isupper():
    if s[0] == s[2]:
        print( s[0] )
    else:
        print( s[1] )
else:
    if s[1] != s[2]:
        print( s[-1] )
    else:
        print( s[-2] )
s = 'ABcd'
if not s[0:2].isupper():
    if s[0] == s[2]:
        print( s[0] )
    else:
        print( s[1] )
else:
    if s[1] != s[2]:
        print( s[-1] )
    else:
        print( s[-2] )
'd'
s = 'abcd'
if not s.isalpha():
    print( s[0] )
elif s.isupper():
    print( s[-1] )
elif 'ab' in s:
    print( s[-2] )
else:
    print( s[1] )
s = 'abcd'
if not s.isalpha():
    print( s[0] )
elif s.isupper():
    print( s[-1] )
elif 'ab' in s:
    print( s[-2] )
else:
    print( s[1] )

'c'
Example

A Quick Quiz

```plaintext
x = 1
y = x
y = 2
# what is x?

x = [ 1,2,3 ]
y = x
y[0] = 6
# what is x?
```
Mutability & Aliasing
We distinguished mutability and immutability. The distinction arises from the storage in memory.
Immutability occurs when values are copies in memory.

- \( x = 3.14 \)
- \( y = x \)

- \( x = 'good\ advice' \)
- \( y = x \)
Mutability occurs when values share the same location.

The distinction arises from the storage in memory.

\[
x = [1, 2, 3, 4] \\
y = x
\]
- **Aliasing** occurs when one memory location has two names.
- **Aliasing causes mutable types to behave unexpectedly!**
Aliasing

\[ x = [1, 2, 3, 4] \]
\[ y = x \]
\[ x[-1] = 2 \]
Example

```python
x = [ 1, 2, 3 ]
y = x
y[0] = 6
# what is x?
```
```
a = [ 'a', 'b', 'c', 'd' ]
b = a
b[3] = '*'
```

What is the final value of `a`?  
A [ 'a', 'b', '*', 'd' ]  
B [ 'a', 'b', 'c', '*' ]  
C [ 'a', 'b', 'c', 'd' ]  
D None of the above.
a = [ 'a', 'b', 'c', 'd' ]
b = a
b[3] = '*'

What is the final value of a?
A [ 'a', 'b', '*', 'd' ]
B [ 'a', 'b', 'c', '*' ] *
C [ 'a', 'b', 'c', 'd' ]
D None of the above.
The immutable analogue of a list is a tuple.

We form a tuple by using parentheses () instead of square brackets [].
 tuples can be used to format multiple values for print.

'\%i \%i \%i' \% (1,2,3)
Example

s = ???
x = 10
y = 'Hello'
z = 3.14
print(s % x, y, z)

What should replace the ????

A  '%i %f %s'
B  '%f %s %i'
C  '%i %s %f'
D  None of the above.
```python
s = ???
x = 10
y = 'Hello'
z = 3.14
print(s % x, y, z)
```

What should replace the `????`?

A `’%i %f %s’`
B `’%f %s %i’`
C `’%i %s %f’` ★
D None of the above.
tuples can also be used on the left-hand side of an assignment operator.

This lets us make multiple assignments at once.

```python
one, pi, hello = (1, 3.14, 'Hi')
x, y = y, x
```
Where can I use tuples?

- Tuples can return multiple values from a function.

```python
def fun():
    return 'hi', 3, 'lo'

a, b, c = fun()
```
Container Methods
Because lists are mutable, we can change their contents.

```python
x = [4, 1, 2, 3]
x[3] = -2  # item assignment
x.append(5)  # appending items
del x[1]  # removing items
x.sort()  # changing item order
```
Container Methods

- sort and append modify the list itself.

Warning!
This explains why sort and append return None!

```
x = [ 4,1,2,3 ]
x.sort()  # This is the right way to sort
print(x)
```
Container Methods

- sort, reverse, and append modify the list itself.

Warning!
This explains why sort and append return None!

```python
x = [ 4,1,2,3 ]
x = x.sort()  # MANY of you will do this. This is wrong!
print(x)
```
y = [ 3, 2, 1 ]
x = y.append( 5 )
y[-1] = 3

What is the final value of x?
A [ 3, 2, 1, 3 ]
B [ 3, 2, 1, 5 ]
C [ 3, 2, 1 ]
D None
y = [ 3,2,1 ]
x = y.append( 5 )
y[-1] = 3

What is the final value of x?
A [ 3, 2, 1, 3 ]
B [ 3, 2, 1, 5 ]
C [ 3, 2, 1 ]
D None ★
- **index** returns the index of the first occurrence of a value in a list.
- **count** returns how many times a value occurs.
- **in** returns membership in the list.
- *** repeats a list.**
- **+ extends a list (also extend).**
- **max, min, len, etc.**
String/List Methods
split returns a list.
Takes a single string argument, the delimiter.

```python
name = 'Oliver Wendell Holmes'
names = name.split(' ')
print(names[-1])
```
\textbf{Example}

\begin{verbatim}
x = 'A+B+C'
y = x.split()
\end{verbatim}

What is the final value of \texttt{y}?

\begin{itemize}
  \item A \textquotesingle ABC\textquotesingle
  \item B [ \textquotesingle A\textquotesingle, \textquotesingle B\textquotesingle, \textquotesingle C\textquotesingle ]
  \item C [ \textquotesingle A+B+C\textquotesingle ]
  \item D \textquotesingle A\textquotesingle, \textquotesingle B\textquotesingle, \textquotesingle C\textquotesingle
  \item E None
\end{itemize}
Example

x = 'A+B+C'
y = x.split()

What is the final value of y?
A  'ABC'
B  ['A','B','C']
C  ['A+B+C']  ⋆
D  'A','B','C'
E  None
Example

```python
x = 'A+B+C'
y = x.split('+')
```

What is the final value of `y`?

A  'ABC'
B  ['A','B','C']
C  ['A+B+C']
D  'A','B','C'
E  None
```python
x = 'A+B+C'
y = x.split('+')
```

What is the final value of \( y \)?

A  'ABC'
B  ['A', 'B', 'C']  ✗
C  ['A+B+C']
D  'A', 'B', 'C'
E  None
x = 'A+B+C'
y = x.split('-')

What is the final value of y?
A 'A+B+C'
B [ 'A+B+C' ]
C ( 'A+B+C' )
D None
x = 'A+B+C+'
y = x.split('+')

What is the final value of y?
A  'ABC'
B  ['A', 'B', 'C']
C  ['A', 'B', 'C']  ★
D  ['A+B+C']
E  None
join returns a str.

- Takes a single list argument.
- Returns the list elements joined as a string.

```python
names = [ "Geoffrey", "Richard", "Aloysius", "Johnston" ]
# GOAL: "", "Geoffrey Richard Aloysius Johnston"
# ".join(names) # note the odd syntax!
# join is a STRING method
```
a = [ 'X', 'A', 'G' ]
b = a[:]
a.sort()
x = ','.join(b)

What is the final value of x?
A 'XAG'
B [ 'X,A,G' ]
C 'A,G,X'
D ',A,G,X,'
E 'X,A,G' ✓
range( 0, 6, 2 )
list( range( 0, 6, 2 ) )
[ 0, 2, 4 ]
Next steps
Next steps

- Complete quiz07 (due 2/14)
- Complete hw03 (due 2/17)
- Read for the next class