Python Basics!

mutability, container methods

CS101 Lecture #9
for loops
Example

for i in range(10):
    print(i ** 2)
for i in range(10):
    print(i ** 2)

for i in range(2, 10):
    print(i ** 2)
for i in range(10):
    print(i ** 2)

for i in range(2,10):
    print(i ** 2)

for i in range(2,10,3):
    print(i ** 2)
Mutability & Aliasing
Example

```python
x = 1
y = x
y = 2
# what is x? %
```

```python
x = [1, 2, 3]
y = x
y[0] = 6
# what is x?
```
Example

\begin{verbatim}
x = 1
y = x
y = 2
# what is x? %

x = [ 1,2,3 ]
y = x
y[0] = 6
# what is x?
\end{verbatim}
We distinguished *mutability* and *immutability*. The distinction arises from the storage in memory.
Immutability occurs when values are copies in memory.

\[\begin{align*}
x &= 3.14 \\
y &= x
\end{align*}\]

\[\begin{align*}
x &= \text{'good advice'} \\
y &= x
\end{align*}\]
Mutability & immutability

- **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

\[
x = [1, 2, 3, 4]
y = x
x[-1] = 2
\]
x = [1, 2, 3]
y = x
y[0] = 6
# what is x?
```python
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)
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.
Where can I use tuples?

- 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')
```
- 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
```
- 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!

```python
x = [ 4,1,2,3 ]
x.sort()  # This is the right way to sort a list.
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 wrong way!
print(x)
```
Example

```python
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
Container Methods

- `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])
```
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 '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
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
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"""" %
```
**string.join method**

- `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 ) )
out: [ 0, 2, 4 ]
Reminders
Reminders

- Homework #4 is due Wed Nov. 4.
- Midterm #1 will be on the day of the 12th lecture (Nov. 7 Monday), covering through Lecture #11. (evening)