Python Basics!

mutability, container methods

CS101 Lecture #9

2016-09-21

Warmup Quiz

Question #1

```
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] )
```

Question #2

```
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] )
```

for loops

Example

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

Example

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

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

Example

```
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

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

Example

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

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

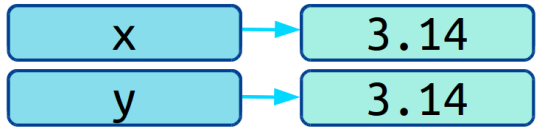
Mutability

- We distinguished mutability and immutability.
- The distinction arises from the storage in memory.

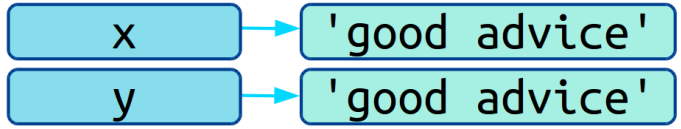
Mutability

- Immutability occurs when values are copies in memory.

```
x = 3.14
y = x
```



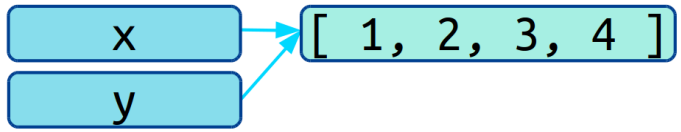

```
x = 'good advice'
y = x
```



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
```

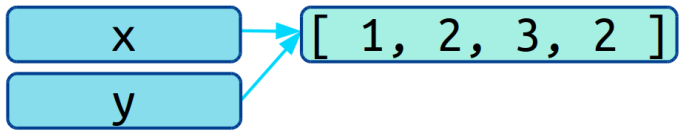


Aliasing

- 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

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

Example

```
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.

Tuples

- The immutable analogue of a `list` is a `tuple`.
- We form a tuple by using parentheses `()` instead of square brackets `[]`.

Where can I use tuples?

- `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.

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.

```
one,pi,hello = ( 1,3.14,'Hi' ) %
```

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.

```
one,pi,hello = ( 1,3.14,'Hi' ) %
x,y = y,x
```

Where can I use tuples?

- `tuples` can return multiple values from a function.

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

a,b,c = fun()
```

Container Methods

Container Methods

- Because `lists` are mutable, we can change their contents.

```
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`!

```
x = [ 4,1,2,3 ]
x = x.sort()     # MANY of you will do this.  Thi
print(x)
```

Example

```
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

string.split method

- `split` returns a `list`.
- Takes a single string argument, the delimiter.

```
name = 'Oliver Wendell Holmes'
names = name.split(' ')
print(m[-1])
```

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

```
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

```
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

```
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`

string.join method

- `join` returns a `str`.
- Takes a single `list` argument.
- Returns the `list` elements joined as a string.

```
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.

```
names = [ ”Geoffrey”, ”Richard”,
          ”Aloysius”, ”Johnston” ]
#  GOAL:  ””””Geoffrey Richard
#             Aloysius Johnston”””” %
’ ’.join(names)      # note the odd syntax!
                     # join is a STRING method
```

Example

```
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'` ⋆

One more thing...

```
range( 0, 6, 2 )
list( range( 0, 6, 2 ) ) %\pause
[ 0, 2, 4 ]
```

Reminders

Reminders

- Homework #4 is due Friday Sep. 23.
- Midterm #1 will be Monday Oct. 3. (evening)