Compound data types

- Have seen a sampling of different classes of algorithms
  - Exhaustive enumeration
  - Guess and check
  - Bisection
  - Divide and conquer
- All have been applied so far to simple data types
  - Numbers
  - Strings

Compound data types

- Tuples
- Lists
- Dictionaries

Tuples

- Ordered sequence of elements (similar to strings)
- Elements can be more than just characters

```
t1 = (1, ‘two’, 3)
print(t1)

t2 = (t1, ‘four’)
print(t2)
```

$$(\underbrace{(1, \text{’two’}, 3)}, \underbrace{\text{‘four’}})$$

Operations on tuples

```
t1 = (1, ‘two’, 3)
t2 = (t1, ‘four’)
```

- Concatenation
- Indexing
- Slicing

```
print(t1+t2)
print((t1+t2)[3])
print((t1+t2)[2:5])
```

- Singletons

```
t3 = (‘five’,)
print(t1+t2+t3)
```

Manipulating tuples

- Can iterate over tuples just as we can iterate over strings

```
def findDivisors(n1, n2):
    """assumes n1 and n2 positive ints
       returns tuple containing
       common divisors of n1 and n2"""
    divisors = () # the empty tuple
    for i in range(1, min(n1, n2) + 1):
        if n1%i == 0 and n2%i == 0:
            divisors = divisors + (i,)
    return divisors
```

Manipulating tuples

- Can iterate over tuples just as we can iterate over strings

```
divs = findDivisors(20, 100)
total = 0
for d in divs:
    total += d
print(total)
```