CMSC201
Computer Science I for Majors
Lecture 14 – Tuples
Last Class We Covered

• Mutability
  – Shallow copy
  – Relation to functions

• Modularity
  – Program Design
  – Incremental Development
Any Questions from Last Time?
Announcement – Advising

• **ALL** students must receive advising authorization prior to registering for classes
• Advising Appointments are currently available
• Students can register between Nov 1 – Nov 18
  – Depending on number of earned credits
• CMSC & ENGR - Sign up for an apt via the online scheduler: http://coeadvising.umbc.edu/
• Group Advising is an excellent option!
Announcement – Survey

• Available now on Blackboard
• Due by Tuesday, October 25th at midnight
  – Check completion under “My Grades”
• Some statistics (from Fall 2015):
  – If they had taken the surveys...
    • 9 students would have gotten an A instead of a B
    • 4 students would have gotten a B instead of a C
    • 9 students would have gotten a C instead of a D
Today’s Objectives

• Learn about the *tuple* data structure in Python
• Perform basic operations with tuples including:
  – Creation
  – Conversion
  – Repetition
  – Slicing
  – Traversing
• Use tuples in functions (as return values)
Tuples
The **Tuple** Data Structure

- In Python, a tuple is an *immutable* sequence of values

What does immutable mean?

Tuples are immutable which means you *cannot update or change* the values of tuple elements.
The **Tuple** Data Structure

• Each value in the tuple is an **element** or **item**

• Elements can be **any** Python data type
  – Tuples can mix data types
  – Elements can be nested tuples

```python
```

**tuple name**  
**first element:** a string  
**second element:** an integer
Creating Tuples
Creating Tuples

• The empty tuple is written as two parentheses containing nothing
  \[ \text{tup1} = () \]

• To cast a list as a tuple, you use `tuple()`
  \[
  \text{myList} = [5, 15, 23] \\
  \text{myTuple} = \text{tuple(myList)} \\
  \text{print(type(myTuple))} \\
  \text{<class 'tuple'>}
  \]
Creating Tuples

```python
numbers = (1, 2, 3, 4)
print (numbers)
(1, 2, 3, 4)

cheeses = ('swiss', 'cheddar', 'ricotta', 'gouda')
print (cheeses)
('swiss', 'cheddar', 'ricotta', 'gouda')
```
Creating Tuples

t1 = ('a')
print (t1, type(t1))

Is this a tuple?

t2 = ('a',)
print (t2, type(t2))

Tuples with one element require a comma
Creating Tuples

t3 = tuple('a')
print (t3, type(t3)) ('a',) <class 'tuple'>

empty = tuple() ()
print (empty)
Creating Tuples

```python
aList = [1, 2, 3, 4]
aTuple = tuple(aList)
print(aTuple)  # Output: (1, 2, 3, 4)

aStr = 'parrot'
aTuple2 = tuple(aStr)
print(aTuple2)  # Output: ('p', 'a', 'r', 'r', 'o', 't')
```
Indexing and Slicing Tuples
Tuple Indexing

• Just like other sequences (strings), elements within a tuple are indexed

```python
cheeses = ('swiss', 'cheddar', 'ricotta', 'gouda')
print(cheeses[0])
```

```python
cheeses[0] = 'swiss'
```

What does this do?  
Nothing! (an error)  
Tuples are immutable.
Slicing a Tuple

• Like other sequences, tuples can be sliced
• Slicing a tuple creates a new tuple. It does not change the original tuple.

```python
cheeses = ('swiss', 'cheddar', 'ricotta', 'gouda')
print(cheeses[1:4])
```

What does this output?

`('cheddar', 'ricotta', 'gouda')`
Tuple Operations
Operations on Tuples

• Tuples support all the standard sequence operations, including:
  – Membership tests (using the `in` keyword)
  – Comparison (element-wise)
  – Iteration (e.g., in a `for` loop)
  – Concatenation and repetition
  – The `len()` function
  – The `min()` and `max()` functions
Membership Tests (**in**)  

- In Python, the **in** keyword is used to test if a sequence (list, tuple, string etc.) contains a value.
  
  - Returns **True** or **False**

```python
a = [1, 2, 3, 4, 5]
print(5 in a)  # True
print(10 in a) # False
```

What does this output?
Comparison

• In Python 3.3, we can use the comparison operator, `==`, to do tuple comparison
  – Returns True or False

```python
tuple1, tuple2 = (123, 'xyz'), (456, 'abc')
tuple3 = (456, 'abc')
print (tuple1==tuple2)
print (tuple2==tuple3)
```

What does this output?

False
True
Iteration

teams = ((1, 'Ravens'), (2, 'Panthers'), (5, 'Eagles'), (7, 'Steelers'))

for (index, name) in teams:
    print(index, name)

1 Ravens
2 Panthers
5 Eagles
7 Steelers
Iteration

\[ t = [('a', 0), ('b', 1), ('c', 2)] \]

for letter, number in t:
    print (number, letter)

What does this output?

\[ \begin{align*}
0 & a \\
1 & b \\
2 & c
\end{align*} \]
Concatenation (+)

• The + operator returns a new tuple that is a concatenation of two tuples

\[ a = (1, 2, 3) \]
\[ b = (4, 5, 6) \]
\[ c = a + b \]

print (a, b, c)

(1, 2, 3) (4, 5, 6) (1, 2, 3, 4, 5, 6)
Repetition (*)

• The * operator returns a new tuple that repeats the tuple.

```python
a = (1, 2, 3)
b = (4, 5, 6)
print (a*2, b)
```

(1, 2, 3, 1, 2, 3) (4, 5, 6)
len() Functions

• The method `len()` returns the number of elements in the tuple.

```python
tuple0 = ()
print(len(tuple0))
tupleA = ("UMBC", "is", "the", "best")
print(len(tupleA))
```

What does this output?

0
4
**min() and max() Functions**

- **max(tuple)**
  - Returns item from the tuple with max value

- **min(tuple)**
  - Returns item from the tuple with min value

```python
myTuple = tuple('parrot')
print(myTuple)
print(min(myTuple))
print(max(myTuple))
```

What does this output?
Tuples and Functions

(return)
Tuples and functions

• Python functions (as is true of most languages) can only return one value
  – But... but... we’ve returned multiple values before!
• If multiple objects are packaged together into a tuple, then the function can return the objects as a single tuple
• Many Python functions return tuples
Example: min_max.py

# Returns the smallest and largest
# elements of a sequence as a tuple
def min_max(t):
    return min(t), max(t)

testSeq = [12, 98, 23, 74, 3, 54]
print (min_max(testSeq))

UnitTest = 'She turned me into a newt!'
print (min_max(UnitTest))

myMin, myMax = min_max(UnitTest)
print ((myMin, myMax))
Any Other Questions?
Announcements

• Homework 6 is due Wednesday

• We’ll cover the midterm in class next time