Tuples + Sorting
Chris Piech and Mehran Sahami
CS106A, Stanford University
Housekeeping

- Contest due June 5th
  - Optional
  - Separate from Assignment #7
Learning Goals

1. Learning about tuples in Python
2. Writing code using tuples
3. Learning about sorting
Tuples
What is a Tuple?

• A tuple is a way to keep track of an ordered collection of items
  – Similar to a list, but immutable (can't be changed in place)
  – Ordered: can refer to elements by their position
  – Collection: list can contain multiple items

• Often used to keep track of data that are conceptually related, such as
  – Coordinates for a point: (x, y)
  – RGB values for a color: (red, green, blue)
  – Elements of an address: (street, city, state, zipcode)

• Can be used to return multiple values from a function
Show Me the Tuples!

• Creating tuples
  – Tuples start/end with parentheses. Elements separated by commas.
    
    ```python
    my_tuple = (1, 2, 3)
    point = (4.7, -6.0)
    strs = ('strings', 'in', 'tuple')
    addr = ('102 Ray Ln', 'Stanford', 'CA', 94305)
    empty_tuple = ()
    ```

• Tuple with one element is the same as the element
  – Could try this out on the console:
    ```python
    >>> tuple_one = (1)
    >>> one = 1
    >>> tuple_one == one
    True
    ```
Accessing Elements of Tuple

• Consider the following tuple:

   \[
   \text{letters} = (\text{'a'}, \text{'b'}, \text{'c'}, \text{'d'}, \text{'e'})
   \]

• Access elements of tuple just like a list:
  – Indexes start from 0

  \[
  \begin{array}{c|c|c|c|c}
  \text{letters} & \text{'a'} & \text{'b'} & \text{'c'} & \text{'d'} & \text{'e'} \\
  \hline
  0 & 1 & 2 & 3 & 4
  \end{array}
  \]

• Access individual elements:

  \[
  \text{letters}[0] \text{ is 'a'}
  \]

  \[
  \text{letters}[4] \text{ is 'e'}
  \]
Accessing Elements of Tuple

• Consider the following tuple:
  
  ```python
  letters = ('a', 'b', 'c', 'd', 'e')
  ```

• Access elements of tuple just like a list:
  
  – Indexes start from 0

  ```
  letters[0] = 'x'
  TypeError: 'tuple' object does not support item assignment
  ```

• **Cannot** assign to individual elements:
  
  – Tuples are **immutable**
Accessing Elements of Tuple

- Consider the following tuple:
  \[ \text{letters} = ('a', 'b', 'c', 'd', 'e') \]

- Access elements of tuple just like a list:
  - Indexes start from 0

- **Cannot** assign to individual elements:
  - Tuples are **immutable**
  - Also, there are no `append/pop` functions for tuples
  - Tuples cannot be changed in place
  - To change, need to create new tuple and overwrite variable
Getting Length of a Tuple

• Consider the following tuple:

\[
\text{letters} = ('a', 'b', 'c', 'd', 'e')
\]

• Can get length of tuple with `len` function:

\[
\text{len(letters) is 5}
\]

– Elements of list are indexed from 0 to length – 1

• Using length to loop through a tuple:

\[
\text{for i in range(len(letters)):
print(str(i) + " -> " + letters[i])}
\]

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>c</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>d</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>e</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Indexes and Slices

• Consider the following tuple:
  ```python
  letters = ('a', 'b', 'c', 'd', 'e')
  ```

• Negative indexes in tuple work just the same as lists
  – Work back from end of tuple
  – Example:
    ```python
    letters[-1] is 'e'
    ```

• Slices work on tuples in the same was as on lists
  ```python
  >>> aslice = letters[2:4]
  >>> aslice
  ('c', 'd')
  ```

  
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

  aslice → 'c' | 'd'
Good Times with Tuples

- More tuple examples:
  
  ```python
  chartreuse_rgb = (127, 255, 0)
  stanford = ('450 Serra Mall', 'Stanford', 'CA', 94305)
  ```

- Printing tuples:
  
  ```python
  >>> print(chartreuse_rgb)
  (127, 255, 0)
  >>> print(stanford)
  ('450 Serra Mall', 'Stanford', 'CA', 94305)
  ```

- Check if tuple is empty (empty tuple is like "False")
  
  ```python
  if stanford:
      print('stanford is not empty')
  else:
      print('stanford is empty')
  ```
More Good Times with Tuples

• More tuple examples:

  chartreuse_rgb = (127, 255, 0)
  stanford = ('450 Serra Mall', 'Stanford', 'CA', 94305)

• Check to see if a tuple contains an element:

  state = 'CA'
  if state in stanford:
    # do something

• General form of test (evaluates to a Boolean):

  \textit{element in tuple}

  – Returns \texttt{True} if \textit{element} is a value in \textit{tuple}, \texttt{False} otherwise
  – Can also test if element is not in tuple using \texttt{not in}
chartreuse_rgb = (127, 255, 0)

• Function: `max(chartreuse_rgb)`
  – Returns maximal value in the tuple
  ```python
  >>> max(chartreuse_rgb)
  255
  ```

• Function: `min(chartreuse_rgb)`
  – Returns minimal value in the tuple
  ```python
  >>> min(chartreuse_rgb)
  0
  ```

• Function: `sum(chartreuse_rgb)`
  – Returns sum of the values in the tuple
  ```python
  >>> sum(chartreuse_rgb)
  382
  ```
Looping Through Tuple Elements

stanford = ('450 Serra Mall', 'Stanford', 'CA', 94305)

• For loop using `range`:

```python
for i in range(len(stanford)):
    elem = stanford[i]
    print(elem)
```

• For-each loop:

```python
for elem in stanford:
    print(elem)
```

Output:

450 Serra Mall
Stanford
CA
94305

• These loops both iterate over all elements of the tuple
  – Variable `elem` is set to each value in list (in order)
  – Works just the same as iterating through a list
Tuples as Parameters

• When you pass a tuple as a parameter, think of it like passing an integer
  – In function, changing tuple parameter is changing a copy

```python
def remove_red(rgb_tuple):
    rgb_tuple = (0, rgb_tuple[1], rgb_tuple[2])
    print("In remove_red: " + str(rgb_tuple))

def main():
    chartreuse_rgb = (127, 255, 0)
    remove_red(chartreuse_rgb)
    print("In main: " + str(chartreuse_rgb))
```

Output:

```
In remove_red: (0, 255, 0)
In main: (127, 255, 0)
```
Assignment with Tuples

• Can use tuples to assign multiple variables at once:
  – Number of variables on left-hand side of assignment needs to be the same as the size of the tuple on the right-hand side

```python
>>> (x, y) = (3, 4)
>>> x
3
>>> y
4
```
Returning Tuples from Functions

• Can use tuples to return multiple values from function
  
  – Stylistic point: values returned should make sense as something that is grouped together (e.g., (x, y) coordinate)

```python
def get_date():
    day = int(input("Day (DD): "))
    month = int(input("Month (MM): "))
    year = int(input("Year (YYYY): "))
    return day, month, year

def main():
    (dd, mm, yyyy) = get_date()
    print(str(mm) + "/" + str(dd) + "/" + str(yyyy))
```

Terminal:

Day (DD): 10
Month (MM): 05
Year (YYYY): 1970
5/10/1970
Returning Tuples from Functions

• Can use tuples to return multiple values from function
  – Stylistic point: values returned should make sense as something that is grouped together (e.g., (x, y) coordinate)

```python
def get_date():
    day = int(input("Day (DD): "))
    month = int(input("Month (MM): "))
    year = int(input("Year (YYYY): "))
    return day, month, year

def main():
    (dd, mm, yyyy) = get_date()
    print(str(mm) + "/" + str(dd) + "/" + str(yyyy))
```

– Note: all paths through a function should return a tuple of the same length, otherwise program might crash

– For functions that return tuples, comment should specify the number of return values (and their types)
• Can create lists from tuples using `list` function:

```python
>>> my_tuple = (10, 20, 30, 40, 50)
>>> my_list = list(my_tuple)
>>> my_list
[10, 20, 30, 40, 50]
```

• Can create tuples from lists using `tuple` function:

```python
>>> a_list = ['congratulations', 'class', 'of', 2020]
>>> a_tuple = tuple(a_list)
>>> a_tuple
('congratulations', 'class', 'of', 2020)
```
Tuples and Dictionaries

• Can get key/value pairs from dictionaries as tuples using the `items` function:

```python
>>> dict = {'a':1, 'b':2, 'c':3, 'd':4}
>>> list(dict.items())
[('a', 1), ('b', 2), ('c', 3), ('d', 4)]
```

• Can loop through key/value pairs as tuples:

```python
for key, value in dict.items():
    print(str(key) + " -> " + str(value))
```

Output:

```
a  ->  1
b  ->  2
c  ->  3
d  ->  4
```
Tuples in Dictionaries

• Can use tuples as keys in dictionaries:

```python
>>> dict = {('a',1): 10, ('b',1): 20, ('a',2): 30}
>>> list(dict.keys())
[('a', 1), ('b', 1), ('a', 2)]
>>> list(dict.values())
[10, 20, 30]
```

• Can use tuples as values in dictionaries:

```python
>>> colors = { 'orange': (255, 165, 0),
              'yellow': (255, 255, 0),
              'aqua':  (0, 128, 128)  }
>>> list(colors.values())
[(255, 165, 0), (255, 255, 0), (0, 128, 128)]
>>> list(colors.keys())
['orange', 'yellow', 'aqua']
```
Putting it all together: colors.py
Sorting
Basic Sorting

• The `sorted` function orders elements in a collection in increasing (non-decreasing) order
  – Can sort any type that support `<` and `==` operations
  – For example: int, float, string
  – `sorted` returns new collection (original collection unchanged)

```python
>>> nums = [8, 42, 4, 8, 15, 16]
>>> sorted(nums)
[4, 8, 8, 15, 16, 42]
>>> nums
[8, 42, 4, 8, 15, 16]    # original list not changed

>>> strs = ['banana', 'zebra', 'apple', 'donut']
>>> sorted(strs)
['apple', 'banana', 'donut', 'zebra']
```
Intermediate Sorting

• Can sort elements in decreasing (non-increasing) order
  – Use the optional parameter `reverse=True`

```python
>>> nums = [8, 42, 4, 8, 15, 16]
>>> sorted(nums, reverse=True)
[42, 16, 15, 8, 8, 4]

>>> strs = ['banana', 'APPLE', 'apple', 'donut']
>>> sorted(strs, reverse=True)
['donut', 'banana', 'apple', 'APPLE']
```

• Note case sensitivity of sorting strings!
  – Any uppercase letter is less than any lowercase letter
  – For example: 'z' < 'a'
Advanced Sorting

• Sorting using a custom function
  – Use the optional parameter `key=<function name>`

```python
def get_len(s):
    return len(s)

def main():
    strs = ['a', 'bbbbb', 'cc', 'zzzz']
    sorted_strs = sorted(strs, key=get_len)
    print(sorted_strs)
```

Output:

```
['a', 'cc', 'bbbb', 'zzzz']
```
Super Deluxe Advanced Sorting

• Sorting a list of tuples with a custom function
  – Use the optional parameter `key=<function name>`

```python
def get_count(food):
    return food[1]

def main():
    foods = [('apple', 5), ('banana', 2), ('chocolate', 137)]
    sort_names = sorted(foods)
    print(sort_names)
    sort_count = sorted(foods, key=get_count)
    print(sort_count)
    rev_sort_count = sorted(foods, key=get_count, reverse=True)
    print(rev_sort_count)
```

Output:

```python
[('apple', 5), ('banana', 2), ('chocolate', 137)]
[('banana', 2), ('apple', 5), ('chocolate', 137)]
[('chocolate', 137), ('apple', 5), ('banana', 2)]
```
Learning Goals

1. Learning about tuples in Python
2. Writing code using tuples
3. Learning about sorting
Yes, that's in sorted order!