Tuples & Sets

Brandon Krakowsky

Penn Engineering

1

Tuples

2

Tuples

- A *tuple* is an *immutable* sequence of values
  - Once defined, you cannot change the individual elements
  - This is unlike lists, which are *mutable*
- Like lists, values included do not need to be all of the same type
- Creating a tuple is as simple as listing comma-separated values, enclosed in parentheses ()
- Here's a tuple with 4 values:

```
direction = ('north', 'south', 'east', 'west')
type(direction)
```

  - The type is 'tuple'
- If you try to update a tuple, you will get an error – so these won't work:

```
direction[0] = 'N'
direction[4] = 'northeast'
```


3

Tuples

- You can actually create a tuple without the parentheses

```
possible_grades = 'A', 'B', 'C', 'D', 'F'
print(type(possible_grades))
```

  - Again, the type is 'tuple'
- You can also create a tuple from a string (or even a list) with Python's built-in *tuple* function

```
possible_grades = tuple('ABCDF')
print(possible_grades)
```

- Tuples are very useful if/when you want to return multiple things from a function

4

Tuples - Exercise

- Create a *max_and_min* function that returns a *tuple* containing the max and min of a given list
- First, create the *max_and_min* function itself

```
def max_and_min(lst):
    '''Returns the max and min values in the given list.
    '''
    return (max(lst), min(lst))
```


5

Tuples - Exercise

- Create a *max_and_min* function that returns a *tuple* containing the max and min of a given list
- Then, create the *main* function and get the max and min from list: 10, -1, -34, 56

```
def main():
    list1 = [10, -1, -34, 56]
    maxandmin = max_and_min(list1) #get the returned tuple
    print(type(maxandmin)) #type should be tuple

    maximum = maxandmin[0] #access the max value from tuple
    print(type(maximum), maximum)

    minimum = maxandmin[1] #access the min value from tuple
    print(type(minimum), minimum)

#program entry point
if __name__ == '__main__':
    main()
```


6

Tuples - Exercise

- Create a *max_and_min* function that returns a *tuple* containing the max and min of a given list
- You can also get both *tuple* values with two variable assignments

```
#doing 2 assignments from a tuple
max1, min1 = max_and_min(list1)
print(max1, min1)
```


7

Sets

8

Sets

- A *set* is an unordered collection
  - The order doesn't matter and can't be specified
  - It does not support things like indexing
- A *set* does not allow repeated elements
- Values included do not need to be all of the same type
- Sets are *mutable*, so once defined, elements can be changed
- To create a *set*, create a list of comma-separated values within curly braces {}

```
fruit = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
print(type(fruit))
print(fruit)
```


9

Sets

- You can also create a set from a string using Python's built-in *set* function

```
a = 'abracadabra'
a_set = set(a) #unique letters in string a
print(a_set)
```

- Or from a list

```
b = [1, 2, 1, 3, 1, 4, 1, 5, 1, 6, 1, 7, 1, 8, 1, 9, 1, 10]
b_set = set(b) #unique numbers in list b
print(b_set)
```

- Note, an empty set cannot be written as {}
  - Instead, use the *set* function

```
empty_set = set()
```

Penn Engineering

Property of Penn Engineering | 10

10

Sets

- You can iterate over a set to get the values

```
for c in a_set:
    print(c, end = ' ')

for n in b_set:
    print(c, end = ' ')
```

- Add an element to a set

```
a_set.add('c')
print(a_set)
```

- Remove an element from a set

```
b_set.remove(10)
print(b_set)
```

For reference: https://docs.python.org/3/library/stdtypes.html#set

Penn Engineering

Property of Penn Engineering | 11

11