

# 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 = ('fnorth', '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[4] = 'n' direction[4] = 'northeast'

## 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

Renn Engineering

nerty of Denn Engineering

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)

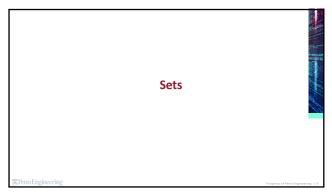
minimum = maxandmin[1] #access the min value from tuple
print(type(minimum), minimum)

#program entry point
if \_\_name\_\_ == '\_\_main\_\_':
 main()

Renn Engineering

operty of Penn Engineering

• 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)	
<pre>max1, min1 = max_and_min(list1)</pre>	



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)

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)	
<ul> <li>Note, an empty set cannot be written as {}</li> <li>Instead, use the set function</li> <li>empty_set = set()</li> </ul>	

