\_\_author\_\_ = "7093700, Schenk"

*# epr6 Aufgabe -- ANALYSE*Programmierung des Testens von Graphen auf die Baumeigenschaft in python==3.0 oder neuer  
  
*Ein- und Ausgabeformat:  
------------------------------*Ein: Die Kanten des gewünschten Graphen in Koordinatenform wie im Beispiel angegeben  
Aus: Baum - Ja oder Nein, Wenn Ja dann auch Wurzel und Blätter des Baums.  
  
*Annahmen:  
------------------------------*Es muss sich um einen gerichteten Graphen handeln. Daas eingeben der Kanten (von Knoten v1 zu Knoten v2) in Koordinatenform ist völlig ausreichend,   
da durch das Festlegen der Kanten auch direkt die Position der Knoten gegeben ist. Ein Graph sieht nach Eingabe z.B. wie folgt aus:  
[((1, 1), (3, 3)), ((1, 1), (4, 1)), ((7, 7), (4, 2)), ((7, 7), (1, 1))]  
  
  
*Entwurfsmuster:  
------------------------------*Ich werde auch in Zukunft nach einem ähnlichen Muster vorgehen, nämlich dass die Überfunktion "def main" regelmäßig weitere Funktionen aufruft und somit als "oberste Funktion" gilt.   
Aus Gründen der Übersichtlichkeit füge ich mehrere ähnliche Aufgaben dann zu kleinen Unterfunktionen zusammen, sodass diese von "def main" aus gesteuert werden.  
Dieses Mal wurden auch die Tests als eigene Funktion von def main aus ausgerufen.  
  
*# epr6 Aufgabe -- TESTS (stehen auch als extra Funktion bereit)  
------------------------------  
  
# TEST 1:  
# IN: [((1, 1), (3, 3)), ((1, 1), (4, 1)), ((7, 7), (4, 2)), ((7, 7), (1, 1))]  
# SHOULD: be no tree  
# OUT: Reason for being no tree: (1, 1) (3, 3) (7, 7) (1, 1)  
# Reason for being no tree: (1, 1) (4, 1) (7, 7) (1, 1)  
# The provided graph is no tree!  
  
# Test 2:  
# IN: [((1, 1), (3, 3)), (("a", "k"), (4, 1.9)), ((7, 7), (4, 2))]  
# SHOULD: be a tree  
# OUT: The provided graph is not just a graph but also a tree!  
# The leaves are: [(3, 3), (4, 1.9), (4, 2)]  
# The root of the tree is: [(1, 1), (7, 7), ('a', 'k')]  
  
# Test 3:  
# [((1, 1, 3, 3), (1, 1), (4, 1)), ((7, 7), (4, 2)), ((7, 7), (1, 1))]  
# SHOULD  
# The provided graph is not just a graph but also a tree!  
# The leaves are: [(1, 1), (4, 2)]  
# The root of the tree is: [(1, 1, 3, 3), (7, 7)]  
  
  
# epr6 Aufgabe -- DOKUMENTATION  
Beschreibung des Programms:  
------------------------------*Das vorliegende Python-Skript beschreibt das Prüfen von Graphen auf Baumeigenschaft, das aus mehreren Funktionen besteht, welche jeweils einen spezifischen Teil des Testens von Graphen abdecken.   
Zunächst müssen die Kanten in Koordinatenform (Tupel) eingegeben werden, die dann in einer Liste gespeichert werden. Die Koordinaten der Knoten müssen durch Kommas getrennt eingegeben werden.  
Danach wird zuerst geprüft, ob es sich bei dem eingegebenen Graphen gleichzeitig auch um einen Baum handelt.  
Wenn ein Baum vorhanden ist, wird auch in der Funktion get\_leaves(graphing) nach dessen Blättern gesucht und diese ausgegeben.  
Zuletzt prüft die Funktion get\_root(graphing) ob der Baum eine Wurzel besitzt und gibt diese dementsprechend aus.   
Schließlich wird der Nutzer auf alle Details hinsichtlich der korrekten Eingabe und der Grapheigenschaften informiert.