1 **import** ast

2 *##common= { 'col\_offset', 'lineno'}*

3 # *Python Language Dictionary* = *PLD refers to Python version* - *3.5.2*

4 **from** PLD2 **import** PLD, PLDLIST

5 **from** textfile **import** text **as** text

6 tree = ast.parse{text)

7

8 **def** add\_to\_list{base, sep, bits):

9 **return** [sep.join{[base, bit]) **for** bit in bits]

10

11

12 **def** tie\_in{endlist):

13 **for** row **in** sorted{endlist):

14 rowl = len{str{row[1]))

15 print{' '\*{60-rowl),row[l],' ',eval(row[l]))

16

17

18 **def** get\_children\_of{item):

19 {count, parent\_full\_name) = item

20 child\_list = [{count,parent\_full\_name)]

21 print{'starting **analysis of** ',parent\_full name)

22 parent\_is\_list = eval{'isinstance({O}, **list)**'.format(

parent full\_name))

23 **if** parent\_is\_list:

24 child\_list\_len = len{eval(parent full name))

25 list\_of\_children = list(map(lambda x: '[{0}]'.

format{x),range{child\_list len)))

26 child\_list.extend{add\_to\_list(parent\_full\_name, ''

, list\_of\_children))

27 print('children **from list** ', child list)

28 **else:**

29 parent= eval('type({O})'.format(parent\_full name)

30 parent value = str(parent)

31 parent\_value2 = (parent\_value)[13:-2] **if**

parent\_value.startswith("<class **'\_ast.") else** parent\_value

[8:-2]

32 **if** parent\_value2 **in** PLD:

33 sub\_list = PLD[parent\_value2]

34 **for** item **in** sub list:

35 child\_list.append('.'.join([

parent full\_name, item]))

36 **else:**

37 print{'not **found in PLD** ',parent\_value2)

38 print{'returning **child\_list for {0}** = {l}'.format(

parent full\_name, child\_list))

39 **return** child list

40

41

42 **def** main{):

43 count = 0

44 stacker [{count, **'tree')]**

45 endlist=[]

46 parent\_array ={ }

47 child\_array ={ }

48 **while** len{stacker) > 0:

49 parent\_sent = stacker[O]

50 current nodes children= get\_children\_of(

parent\_sent)

51 **for** child **in** current nodes children:

52 current count = count

53 **if** child == parent sent:

54 print{child, ' **has returned')**

55 endlist.append(parent\_sent)

56 current\_parent = parent\_sent[O]

57 stacker.remove(child)

58 **else:**

59 print{child,' is **new for stacker')**

60 count += 1

61 stacker.append((count, child))

62

63

.append{count)

parent\_array[count] = current\_parent child\_array.setdefault(current\_parent, [])

64 print{'stack **currently holds** : ',stacker)

65 print{'endlist **currently holds** : ', endlist)

66 print{'count **is at** ', count)

67 print{'end **result=** ', endlist)

68 print{parent\_array)

69 print{child\_array)

70 goget = tie\_in{endlist)

71

72 **if** name ==' **main** '·

73 main{)

74