```
 1 import pickle
 2 from PLD2 import PLD
 3 from pprint import pprint as pp
 4 with open('filename.pickle', 'rb') as file_handle:
 5     nodevalues = pickle.load(file_handle)
 6 
 7 class Function(object):
 8     def __init__(self, name="test", args=None, body=None, decorator_list=None,
   returns=None):
 9         pass
10 
11 def dicty(top = 0 ):
12     currentHead = nodevalues[top]
13     node_name = currentHead[4]+'>>'+currentHead[2].split('.')[-1]
14     a_dict = {}
15     a_dict[node_name] = {}
16 
17     # if this node has no children, extract its information
18     if currentHead[1] == []:
19         #if the information exists, use it, don't add empty
20         if currentHead[5]:
21             a_dict[node_name][node_name] = currentHead[5]
22         if a_dict[node_name]:
23             return a_dict
24         else:
25             return None
26 
27     # if this node has children, add their dicts to this dict
28     children = currentHead[1]
29     newdict = {}
30     for child in children:
31         child_dict = dicty(child)
32         child_name = nodevalues[child][4]+'>>'+nodevalues[child][2].split('.')[
   -1]
33         if child_dict:
34             child_d = child_dict[child_name]
35         else:
36             child_d = None
37         if child_dict:
38             a_dict[node_name][child_name] = child_d
39         newdict.update(a_dict)
40     if newdict:
41         return newdict
42     else:
43         return None
44 
45 
46 def stripFunction(values):
47     args = get_function('args', values)
48     body = get_function('body', values)
49     name = get_function('name', values)
50     decorator_list = get_function('decorator_list', values)
51     returns = get_function('returns', values)
52     print('args = ', args)
```

```
53
54 def check_function(key, values):
55     if 'FunctionDef' in key:
56         print('FunctionDefinition found in {0} '.format(key))
57         return True
58     else:
59         print('func not found in {0}'.format(key))
60     return False
61
62 def proc(tree):
63     if isinstance(tree, dict):
64         print('is a dict')
65         for key,values in tree.items():
66             if check_function(key, values):
67                 stripFunction(values)
68             else:
69                 for vkeys, vvalues in values.items():
70                     if isinstance(values, dict):
71                         proc(values)
72
73
74 def main():
75     view_dict = dicty(0)
76  #   dicty2 = proc(view_dict)
77     pp(view_dict)
78   #  print(dicty2)
79     #extract_function_defs()
80 if __name__=='__main__':
81     main()
82
83
```