

Open Source

- Merge 1
- Merge 2
- Merge 3
- Merge 4
- Merge 5
- Merge 6
- Merge 7
- Merge 8
- Merge 9
- Merge 10
- Merge 11
- Merge 12
- Merge 13
- Merge 14
- Merge 15
- Merge 16
- Merge 17
- Merge 18

Possible Merges

[Mon 17 Mar 14:16:07 2014] - EasyMerge started
[Mon 17 Mar 14:16:07 2014] - UI initialization completed
[Mon 17 Mar 14:16:07 2014] - Please pick source tree to analyze
[Mon 17 Mar 14:16:18 2014] - Start running CloneDigger...
[Mon 17 Mar 14:16:48 2014] - 18 mergeable clone sets found
[Mon 17 Mar 14:17:00 2014] - Mergeable 1 evaluation:
File 1:
/Users/shengyingan/Projects/easymerge/EasyMerge/tests/beets/ui/clone.py
File 2:
/Users/shengyingan/Projects/easymerge/EasyMerge/tests/beets/ui/migrate.py

Mergeable 1 evaluation:
File 1:
/Users/shengyingan/Projects/easymerge/EasyMerge/tests/beets/ui/clone.py
File 2:
/Users/shengyingan/Projects/easymerge/EasyMerge/tests/beets/ui/migrate.py
File 1 package: beets > util > clone
File 2 package: beets > ui > migrate
External class reference: 2
External function reference: 2
Extra parameters: 2
Clone length: 30 lines
Merge is somewhat recommended

Merge Unmerge

/Users/shengyingan/Projects/easymerge/EasyMerge/tests/beets

Source Tree Path

confit.py migrate.py

/Users/shengyingan/Projects/easymerge/EasyMerge/tests/beets/ui/migrate.py

```
645:         if self.alias_key is not None:
646:             self.represented_objects[self.alias_key] = node
647:         best_style = True
648:         testFunc = outsideFunc()
649:         testFunc = insideFunc()
650:         testFunc = outsideClass()
651:         testFunc = Dumper.insideClass()
652:         if hasattr(mapping, 'items'):
653:             mapping = list(mapping.items())
654:         for item_key, item_value in mapping:
655:             node_key = self.represent_data(item_key)
656:             node_value = self.represent_data(item_value)
657:             if not ((isinstance(node_key, yaml.ScalarNode)
658:                     and not node_key.style)
659:                     or (isinstance(node_value, yaml.ScalarNode)
660:                         and not node_value.style)):
661:                 best_style = False
662:             if not ((isinstance(node_value, yaml.ScalarNode)
663:                     and not node_value.style)
664:                     or (isinstance(node_value, yaml.ScalarNode)
665:                         and not node_value.style)):
666:                 node_flow_style = self.default_flow_style
667:             else:
668:                 node_flow_style = best_style
669:             return node
670:         ##REPLACE CODE ABOVE##
671:         ##REPLACE WITH##
672:         return represent_mapping_helper(self, Dumper.insideClass, outsideFunc, yaml.MappingNode, outsideClass, tag, mapping, flow_style)
673:         ##REPLACE END##
674:
675:     def represent_list(self, data):
676:         """If a list has less than 4 items, represent it in inline style
677:         (i.e. comma separated, within square brackets)"""
```

Source Code Browser

```
def represent_mapping_helper(self, Dumper.insideClass, outsideFunc, yaml.MappingNode, outsideClass, tag, mapping, flow_style=None):
    def insideFunc(inside):
        return 'inside'
        value = []
        node = yaml.MappingNode(tag, value, flow_style=flow_style)
        if (self.alias_key is not None):
            self.represented_objects[self.alias_key] = node
        best_style = True
        testFunc = outsideFunc()
        testFunc = insideFunc()
        testFunc = outsideClass()
        testFunc = Dumper.insideClass()
        if hasattr(mapping, 'items'):
            mapping = list(mapping.items())
        for (item_key, item_value) in mapping:
            node_key = self.represent_data(item_key)
            node_value = self.represent_data(item_value)
            if not ((isinstance(node_key, yaml.ScalarNode) and (not node_key.style)):
                    or (isinstance(node_value, yaml.ScalarNode) and (not node_value.style))):
                best_style = False
            if not ((isinstance(node_value, yaml.ScalarNode) and (not node_value.style)):
                    or (isinstance(node_value, yaml.ScalarNode) and (not node_value.style))):
                best_style = False
            value.append((node_key, node_value))
        if (flow_style is None):
            if (self.default_flow_style is not None):
                node_flow_style = self.default_flow_style
            else:
                node_flow_style = best_style
        return node
```

Extracted Method

Evaluation