import json

john = json.dumps({'platform':('Udacity','Cousera')})

kim = json.dumps({'platform':'Udacity'})

ronny = json.dumps({'platform':'freecodecamp'})

alice = json.dumps({'platform':'freecodecamp'})

damian = json.dumps({'platform':('Udacity','Cousera')})

students\_dict = {'John':john,'Kim':kim,'Ronny':ronny,'Alice':alice,'Damian':damian}

def invert\_dict(*d*):

inverse = *dict*()

for key in d:

val = d[key]

if val not in inverse:

inverse[val] = [key]

else:

inverse[val].append(key)

return inverse

print("Original Dictionary \n\n {}".format(students\_dict))

print("")

print("Inverted Dictionary \n\n {}".format(invert\_dict(students\_dict)))

**Original dictionary**

{'John': '{"platform": ["Udacity", "Cousera"]}', 'Kim': '{"platform": "Udacity"}', 'Ronny': '{"platform": "freecodecamp"}', 'Alice': '{"platform": "freecodecamp"}', 'Damian': '{"platform": ["Udacity", "Cousera"]}'}

**Inverted dictionary**

{'{"platform": ["Udacity", "Cousera"]}': ['John', 'Damian'], '{"platform": "Udacity"}': ['Kim'], '{"platform": "freecodecamp"}': ['Ronny', 'Alice']}

**Description**

In my dictionary, I’m handling the various platforms that different students use. A number of these platforms are similar for some students while others are unique. These platform names are my dictionary values. Working with these values that repeat themselves independently would be such a cumbersome and non-programmatic way. Also, my dictionary consists of a nested dictionary for the values of the main dictionary. To enable the inverted\_dict function to hash these values, it would result in an error hashing a list or a dictionary, therefore I’m importing and using JSON to dump the data in a way that is acceptable by the function. The inverted dictionary reduces the ambiguity while printing these records and enhances clarity by grouping keys that share similar values.