COMP7022
Programming for Artificial Intelligence

Class 3

The aim of this class is to continue practising data structures and loops in Python. To start, create a new Jupyter Notebook – you could name it ‘Class 3’. Each task provides a sample input values, but your solution must work with any input values.

Note: The exercise marked with a * is part of Assessment 1 – you should show your solution to your class supervisor by the end of Class 5.

1. Write a Python code that, given a list of names, creates a dictionary where each name is grouped by the first letter. Your code should work for any list of names – e.g.:

```
names = ["Jim", "Hetty", "Kirsten", "Theo", "Henry", "Paul"]
grouped = {}
"""
dictionary grouping names based on the first letter

"H" → "Henry", "Hetty"
"J" → "Jim"
"K" → "Kirsten"
"P" → "Paul"
"T" → "Theo"

"""
print(grouped)
```

2. Given a dictionary, write a Python code that create a new dictionary where the keys are used as values and the values are used as keys – e.g.: "computer" → "apple" becomes "apple" → "computer" in the new dictionary. Your solution must work with any dictionary.

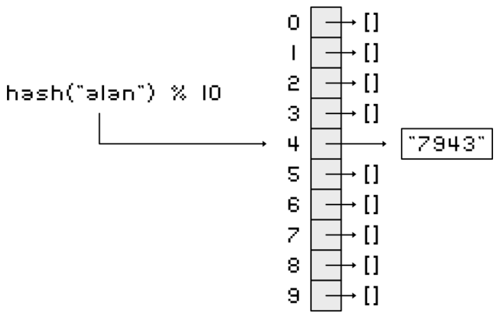
```
my_dict = {"computer" : "apple", "model" : "macbook pro", "processor" : "M1
Pro", "year" : 2021}
inverted_dict = {}

# write logic here

print(inverted_dict)
```

3. * A simple way to implement a **dictionary-like structure** is to use a list of lists. The outer list represents the available *positions* (or *buckets*) where data can be stored. Each element of the outer list is another list that will hold one or more *(key, value)* pairs that fall into the same position. To decide which position a key belongs to, we can use Python’s built-in hash() function

and apply the modulo operator (%) to ensure the result fits within the range of the list.

```python
key = "alan"
# assuming that our inner list has 10 positions
index = hash(key) % 10
```

Since there are only a limited number of positions, different keys may produce the same index (a situation known as a *collision*).
That's why each element of the outer list is itself another list — to store all pairs that share the same index.
The diagram below illustrates the process of adding the pair `{"alan": "7943"}.`



Using this approach, write a Python code that creates a list of lists that mimics a Dictionary.