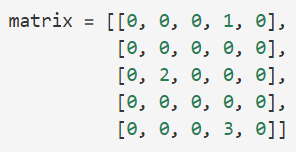
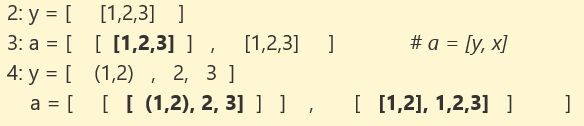
**Dictionaries**

* Format
  + a\_dict = {}  *# creates empty dictionary*
  + a\_dict = {‘key’: ‘value’, ‘key’: ‘value’}
  + *OR* eng2p = {‘three’: ‘tres’, ‘one’: ‘uno’, ‘two’: ‘dos’}
* Mapping type (unordered, associative collection, from a key, maps keys(immutable) to value)
  + key\_list[ value\_list.index(400) ]
* Delete
  + del inventory[‘pears’]
* Modify
  + inventory[‘pears’] = inventory[‘pears’] + 300
  + inventory[‘new key’] = inventory[‘pears’] + inventory[‘apple’]
  + inventory[‘new key’] = 500 *# create new pair*
* Accessing elements
  + a\_dict.values() *# gives a view of values*
  + a\_dict.keys() *# gives a view of keys*
  + a\_dict.get(‘key’, alt) *# gives value associated with key*
  + a\_dict[‘key’] *# gives value associated with key or assign new value*
* Iteration
  + for key in a\_dict:
  + for key in a\_dict:
  + for key in a\_dict.items():
  + for key, value in a\_dict.items():

| **Method** | **Parameters** | **Description** |
| --- | --- | --- |
| keys | none | Returns a view of the keys in the dictionary |
| values | none | Returns a view of the values in the dictionary |
| items | none | Returns a view of the key-value pairs in the dictionary (as tuples) |
| get | key | Returns the value associated with key; None otherwise |
| get | key, alt | Returns the value associated with key; alt otherwise |

* Iteration: **for** iterating over a dictionary implicitly iterates over its keys
* in, not in: test whether a key exists (causes runtime error, returns Boolean)
* get: access the value using a key (returns None)
  + print(inventory.get(“cherries”, 0)) *# 0 if no such key*
* list: creates a list from dictionary
  + keylist = list(a\_dict.keys())
  + valuelist = list(a\_dict.values())
* Aliasing & Copying: if 2 variables refer to same dict object, changes are linked.
  + acopy = a\_dict.copy()
  + list\_a = list\_b *# modifying list\_a also modifies list\_b*

**Sparse Matrix**

* matrices with mostly zeroes (nested list)
  + 
  + matrix = {(0, 3): 1, (2,1): 2, (4,3): 3}
  + matrix[(0,3)] # uses tuples as parameters
  + m.get((row\_idx, col\_idx)),0) # access matrix elements, alt return 0
* Indexing: M[row][column]
  + 
* Row & Columns:
  + a[0][0][0][0] gives 1
  + list[1:1] is empty
  + list[2:3] gives the 3rd element
  + list[2:3] = 4 gives Type Error # Cannot assign integer into list

|  |
| --- |
| **Shallow copy**    not a full copy, sublists are not copied, only references copied. Edits to reference will affect both lists. |
| **Deep copy**    full copy, full replicate as separate objects, fully independent copy |