COMP 204

Dictionaries

Mathieu Blanchette, based on material from Carlos Oliver Gonzalez and Christopher Cameron

Note about two-dimensional lists

In your assignment #2, you will need to represent two-dimensional tables, with a fixed number of rows and columns. *Two-dimensional lists* can be used to do this in Python.

A two-dimensional list is a list of lists, where each of the lists is of the same length. Example: A tic-tac-toe grid:

```
1  tictactoe = [ ["X", "", "O"],
2                ["", "X", ""],
3                ["O", "", ""] ]
4
5  print(tictactoe) # [['X','','O'], ['','X',''], ['O','','']]
6
7  # to access an element in a 2D list,
8  # specify the index of the row and column
9  tictactoe[1][2] = "X"
10 print(tictactoe) # [['X','','O'], ['','X','X'], ['O','','']]
```

Note about two-dimensional lists

Example: A position frequency matrix (see assignment #2).

```
# A position frequency matrix of 4 rows and 6 columns
PFM = [ [0, 4, 2, 5, 1, 3],
        [5, 11,4, 10,6, 5],
        [0, 0, 7, 0, 4, 7],
        [10,0, 2, 0, 4, 0] ]

PFM[0][2] = PFM[3][2] + PFM[2][4]
print(PFM[0][2]) # 6
```

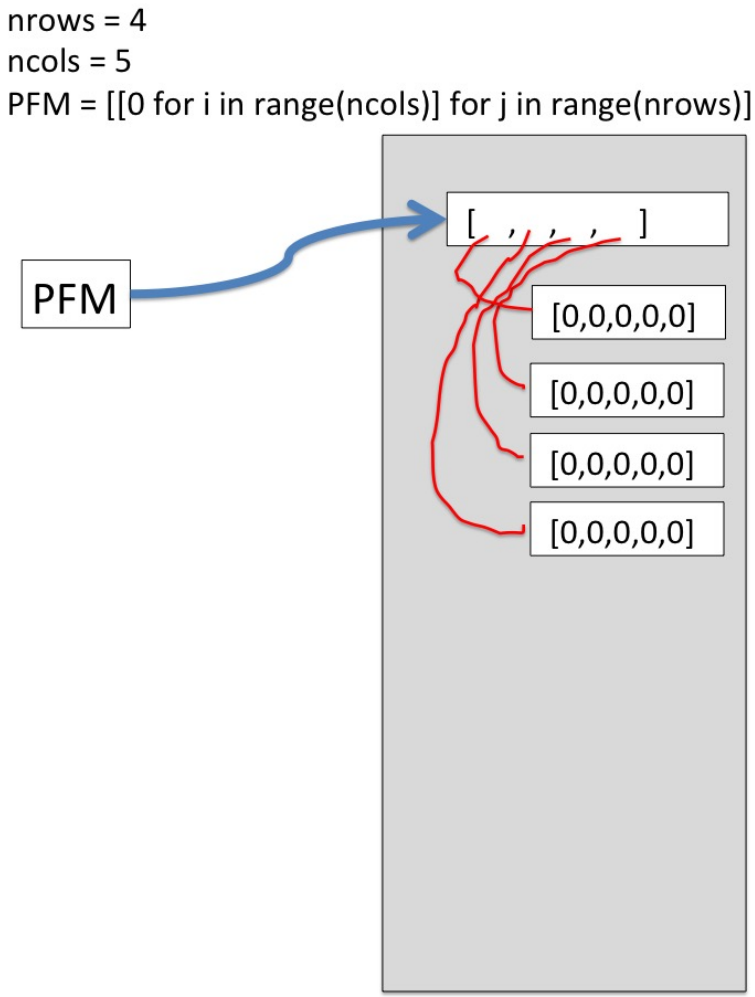
Creating two-dimensional lists

To create a new 2D list filled with zeros:

```
# Creating a two-dimensional list of 4 rows and 5 columns,
# filled with zeros
nrows = 4
ncols = 5
PFM = [[0 for i in range(ncols)] for j in range(nrows)]
print(PFM)
```

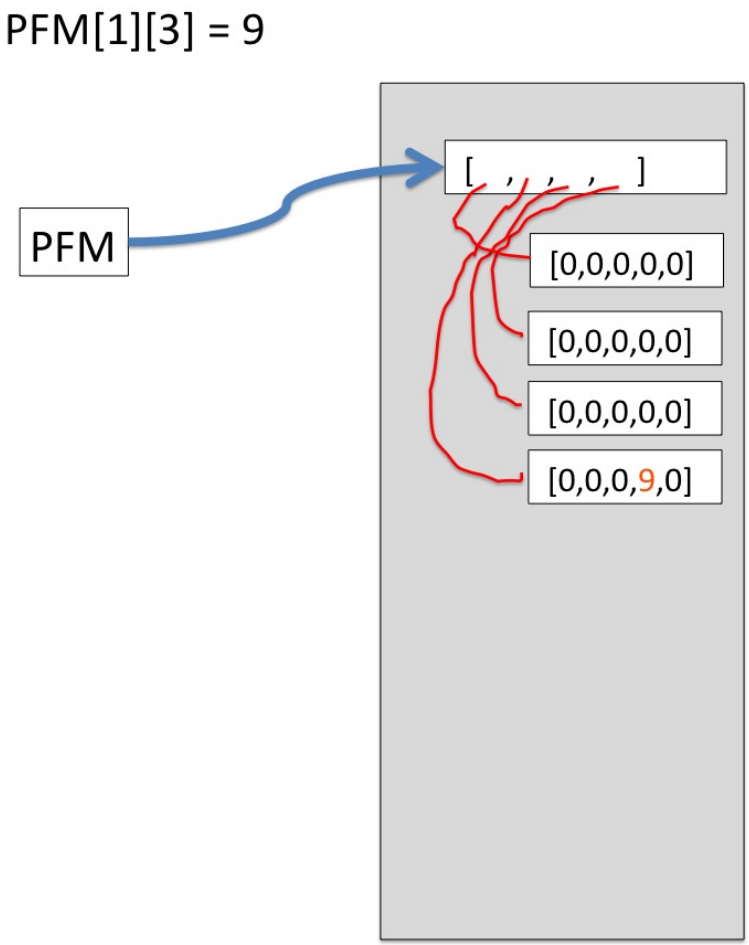
Copying 2D lists

Because lists are compound objects, we need to be careful when copying them.



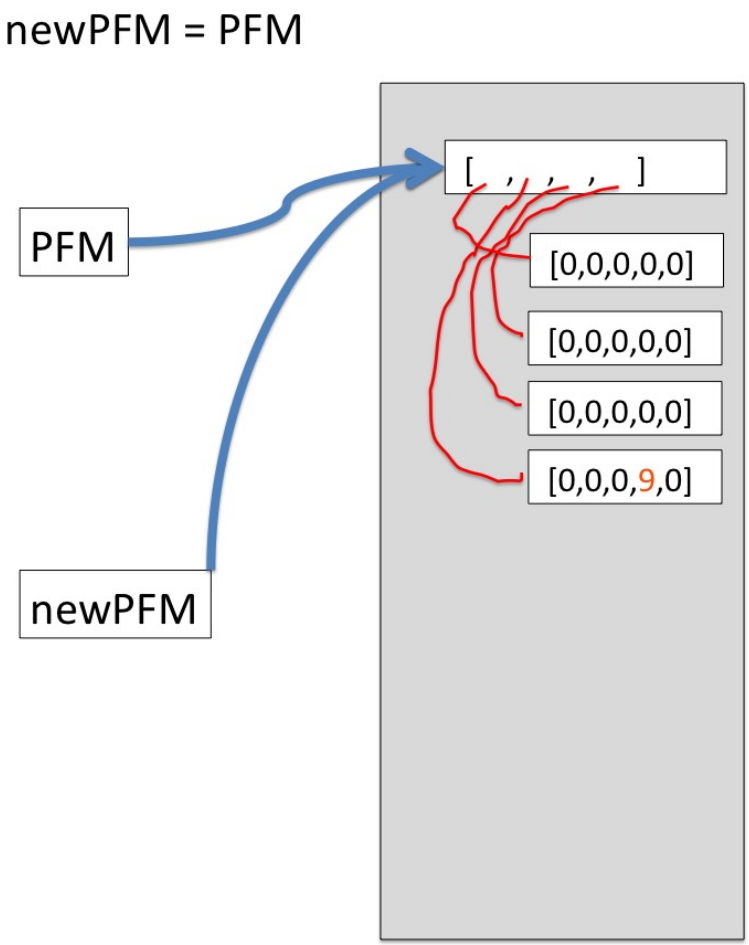
Copying 2D lists

Because lists are compound objects, we need to be careful when copying them.



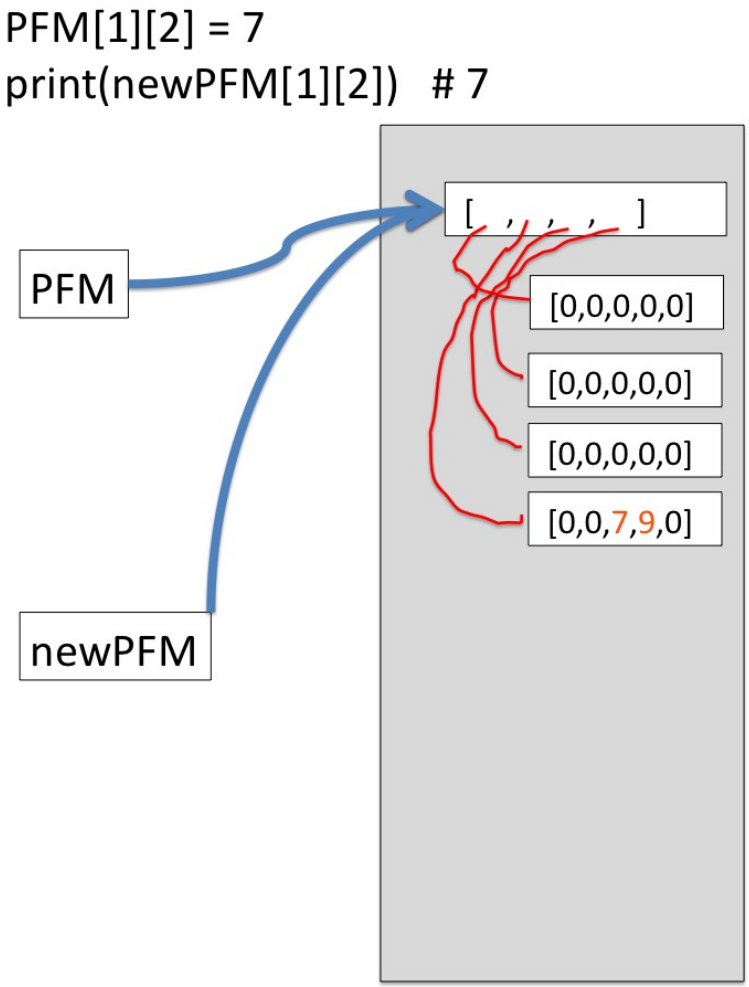
Copying 2D lists

Because lists are compound objects, we need to be careful when copying them.



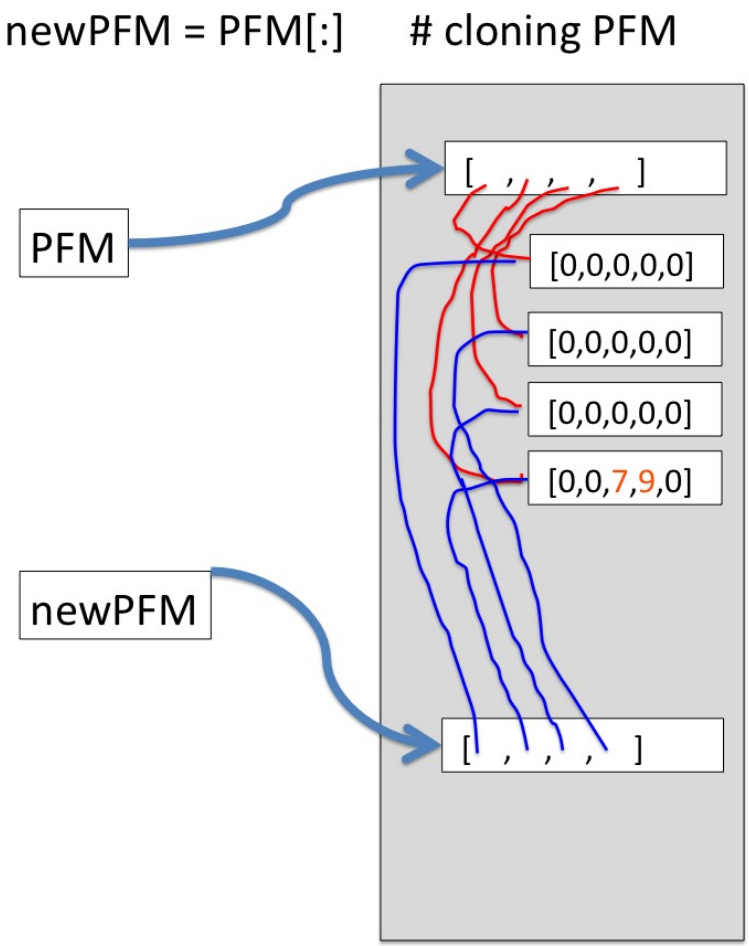
Copying 2D lists

Because lists are compound objects, we need to be careful when copying them.



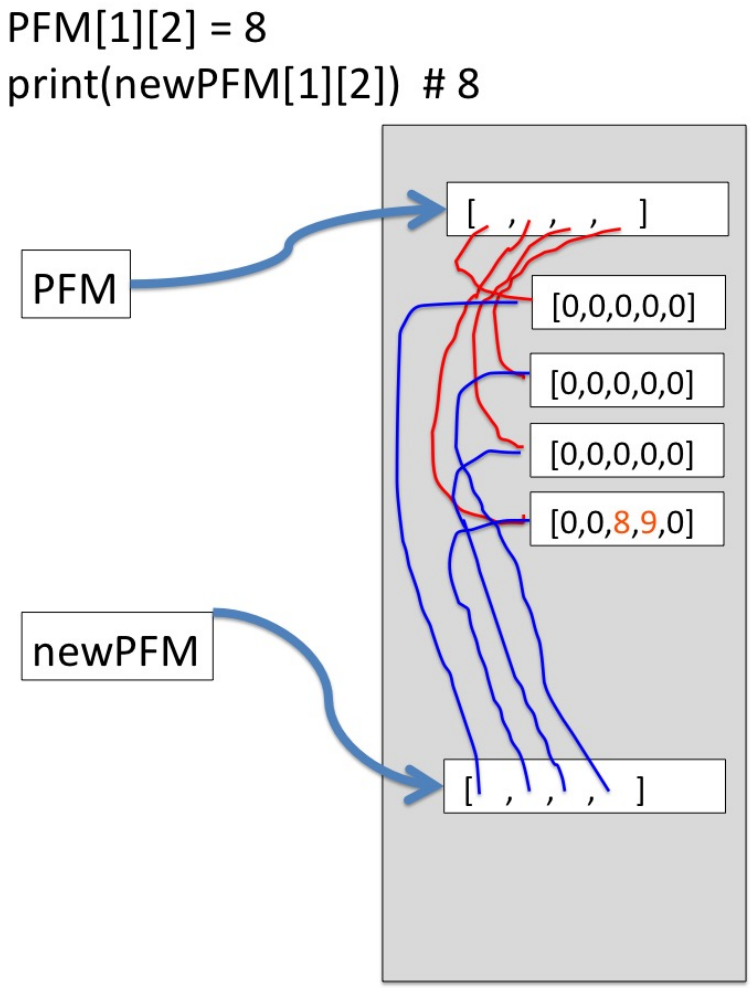
Copying 2D lists

Cloning PFM results in newPFM being a different list object than PFM. *However, the elements of newPFM are the same 1D lists as the elements of PFM.*



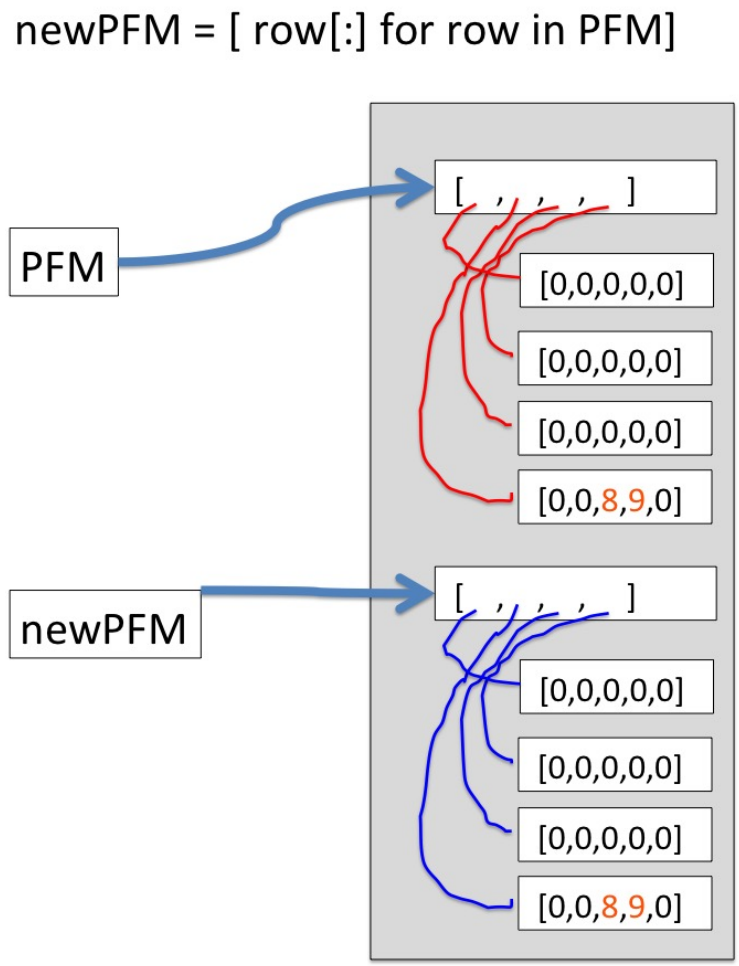
Copying 2D lists

So changing a value in PFM still changes the value in newPFM!



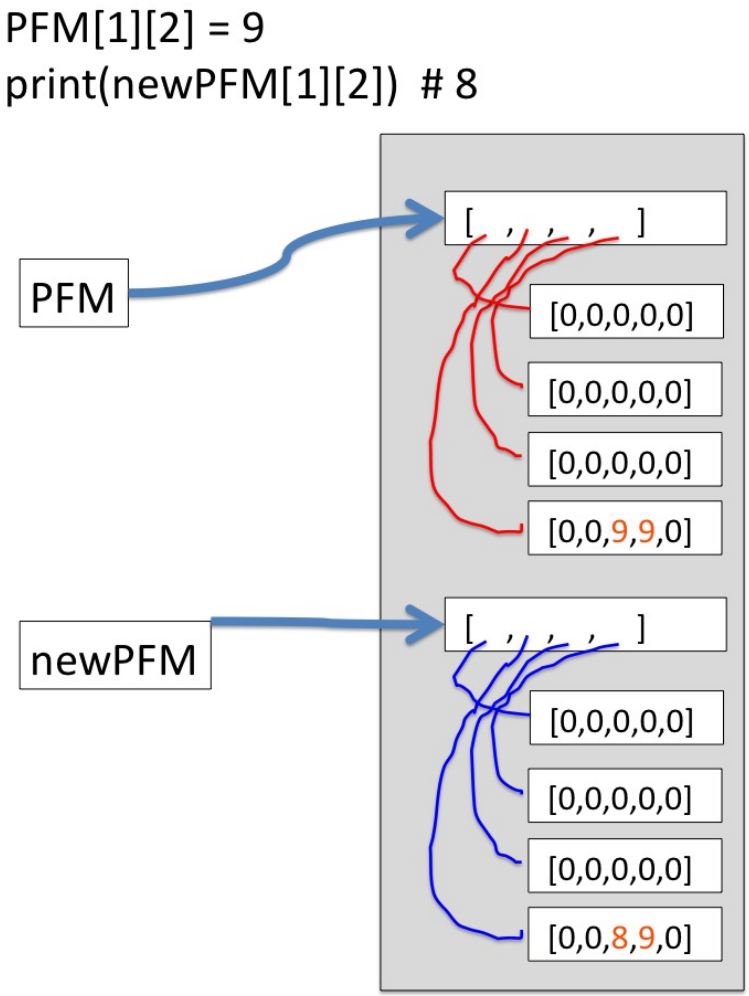
Copying 2D lists

The correct way to clone a 2D list is:



Copying 2D lists

Now the two 2D lists share no elements, and change values in one does not change the values in the other.



A very useful type: Dictionary

- A dictionary is said to be a *mapping* type because it maps *key* objects to *value* objects.
- Dictionaries are immensely useful and are the magic behind a lot of Python functionality
- Syntax:

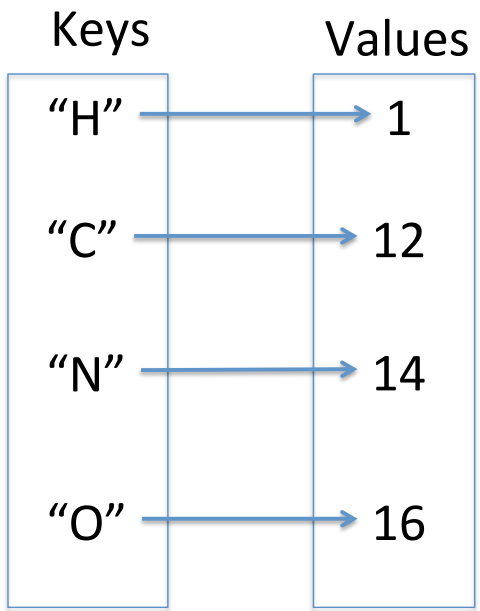
```
1 my_dict = { [key1]: [value1], [key2]: [value2],...}
```

- The analogy to a real dictionary works. The word you look up is the **key** and the definition is the **value**

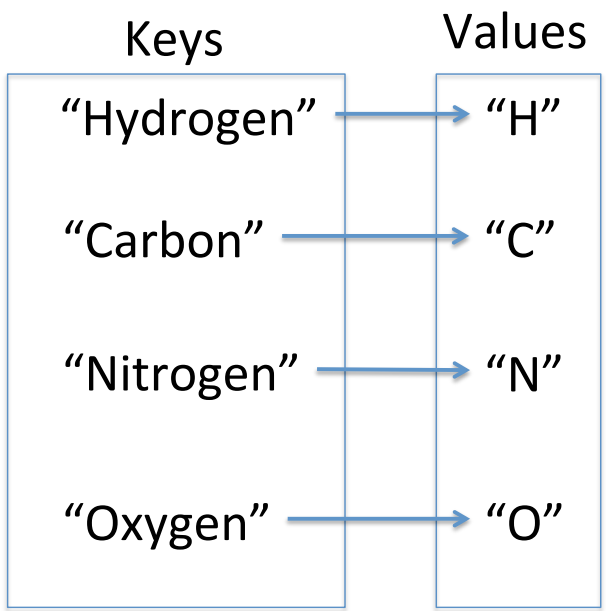
```
# this dictionary maps strings to integers
periodicTable = {"H":1, "C":12, "N":14, "O": 16}

elementsCodes = {"Hydrogen":"H", "Carbon":"C",
                 "Nitrogen":"N", "Oxygen":"O"}
```

periodicTable dictionary:



elementCodes dictionary:



Accessing elements in a dictionary

```
# this dictionary maps strings to integers
periodicTable = {"H":1, "C":12, "N":14, "O": 16}

elementsCodes = {"Hydrogen":"H", "Carbon":"C",
                 "Nitrogen":"N", "Oxygen":"O"}

mass = periodicTable["K"]


periodicTable["He"] = 4  # adds key "He" with value 4
periodicTable["Na"]= 23  # adds key "Na" with value 23

#periodicTable now contains 6 keys,value pairs

periodicTable["C"] = 12.01  # overwrites value for key "C"

del periodicTable["N"]  # deletes key "N" and its value 14
```

Adding and deleting key/value pairs to a dictionary

Adding new key/value pairs:

- Syntax: myDict[key] = value
- If key does not already exist in the dictionary, the new key/value pair is added
- If the key already exists, its previous value is overwritten

Deleting key/values: del myDict[key]

```
# this dictionary maps strings to integers
periodicTable = {"H":1, "C":12, "N":14, "O": 16}

elementsCodes = {"Hydrogen":"H", "Carbon":"C",
                 "Nitrogen":"N", "Oxygen":"O"}

mass = periodicTable["K"]


periodicTable["He"] = 4  # adds key "He" with value 4
periodicTable["Na"]= 23  # adds key "Na" with value 23

#periodicTable now contains 6 keys,value pairs

periodicTable["C"] = 12.01  # overwrites value for key "C"

del periodicTable["N"]  # deletes key "N" and its value 14
```

About keys and values

Keys:

- Have to be immutable objects: int, float, str, tuple.
- Have to be unique in the dictionary: A dictionary cannot contain two elements with the same key.

Values:

- Values can be any type of object: int, float, str, tuple, list, dictionary, etc.
- Many keys can map to the same value

A dictionary can contain keys of many different types, and values of many different types:

```
# a dictionary with keys and values of different types
mixedDict = {"H":"Hydrogen", 17:"prime",
             30:[1,2, 3, 5], (4,5):20}


product = mixedDict[(4,5)]  # 20
primeFactors = mixedDict[30]     # [1,2, 3, 5]

fac = mixedDict[20]   # KeyError: 20 not in mixedDict
```

Dictionaries of dictionaries

The values stored in a dictionary can themselves be dictionaries!

```
# a dictionary where each value is itself a dictionary
periodicTable = {"H": {"name":"Hydrogen", "mass":1},
                 "C": {"name":"Carbon", "mass":12},
                 "N": {"name":"Nitrogen", "mass":14},
                 "O": {"name":"Oxygen", "mass":16} }

carbonDic = periodicTable["C"]  # {"name":"Carbon", "mass":12}
m = carbonDic["mass"]  # 12

#or more directly
m = periodicTable["C"]["mass"]  #12
```

Iterating through dictionaries

The function keys() returns all the keys present in the dictionary.

```
per = {"H":1, "C":12, "N":14, "O": 16}

keyList = list( per.keys() ) # ["H", "C", "N", "O"]
# Note: the keys() function returns an object of
#       type dict_keys. This object is converted to a
#       list using the list() function

for k in keyList:
    print("Key",k,"has value",per[k])
```

The function items() returns the key/value tuples in the dictionary

```
per = {"H":1, "C":12, "N":14, "O": 16}

itemList = list( per.items() )
# Note: the items() function returns an object of
#       type dict_items. This object is converted to a
#       list using the list() function

# itemList is now a list of tuples:
# [('H', 1), ('C', 12), ('N', 14), ('O', 16)]

for k,v in itemList:
    print("Key",k,"has value",v)
```

More functions on dictionaries

To test if a key is present in a dictionary, use the in operator: key in myDict , which evaluates to True if key is in myDict.

```
periodic = {"H":1, "C":12, "N":14, "O": 16}

newElement = "Na"
if newElement in periodic:
    print("Na is already in the dictionary")
else:
    print("Na is not in the dictionary")
```

To add the content of one dictionary, use the update() function.

```
per = {"H":1, "C":12, "N":14, "O": 16}
newTable = {"Na":23, "K":39}

# Add the content of newTable to per
per.update(newTable) # per now has 6 elements
                     # newTable still has 2
```

For more functions on dictionaries:

`https://docs.python.org/3/library/stdtypes.html#mapping-types-dict`

Example

Goal: Count the number of occurrences of all characters in a string.

```
1  sequence = "Hello my name is MAThieu"
2  nucleotides = "acgt"
3  counts = {}  # an empty dictionary
4  for nuc in sequence:
5      if nuc in nucleotides:
6          if (nuc in counts)==False:
7              counts[nuc] = 1
8          else:
9              counts[nuc] += 1
10
11 print(counts)
```

Example

Goal: Compute the mass of a molecule based on its chemical composition. Assume that you have access to a dictionary of atomic masses.

```
periodicTable = {"H":1, "C":12, "N":14, "O": 16}

aceticAcid = "CHHHCOOH"

mass = 0
for element in aceticAcid:
    mass += periodicTable[element]

print("Mass of acetic acid is",mass)
```

Example

Goal: Create a dictionary using as keys the english name of molecules and as values their molar mass. Assume that you have access to a dictionary of atomic masses and a dictionary of chemical compositions:

```
periodicTable = {"H":1, "C":12, "N":14, "O": 16}

molecules = {"Carbon dioxyde":"COO",
             "Nitric oxyde":"NO",
             "Acetic acid":"CHHHCOOH"}

moleculeMass = {}  # the new dictionary we are about
                   # to populate with name/mass pairs
for name, composition in molecules.items():
    mass = 0
    for atom in composition:
        mass += periodicTable[atom]
    moleculeMass[name] = mass

print(moleculeMass)
```