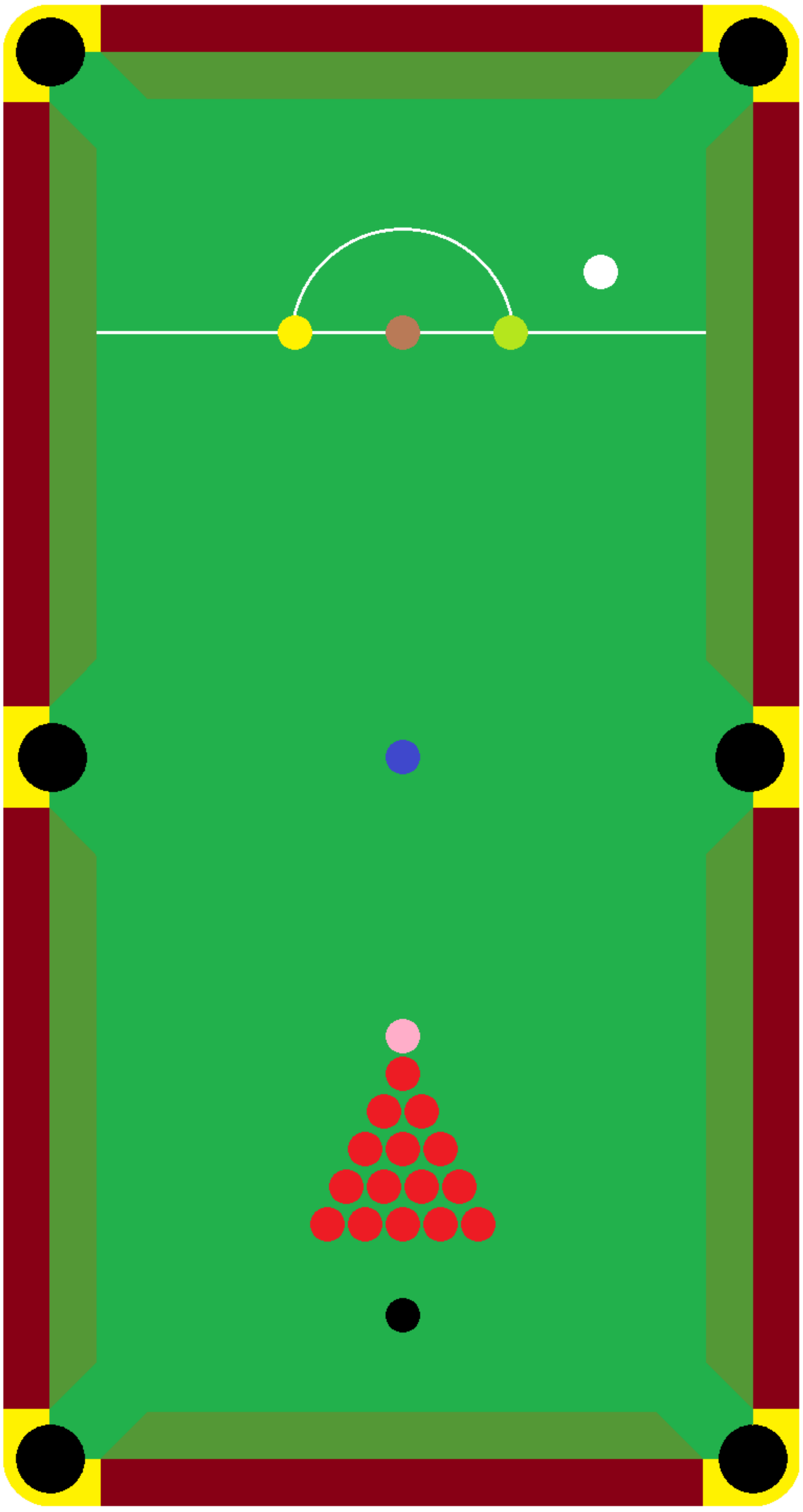
Python Dictionary:



(Preceding Page) **Figure 1:** In this chapter, we are going to use a snooker table, and a dart board, so as to come to grips with dictionaries.

When commencing the science known as computer programming, it is soon discovered that a lot of the skills, that must be learned, so as to become competent in this profession, has to do with ordering data.

We have already seen two other ways that we may order data, i.e. lists and tuples[1].

What concerns us now is *dictionaries*.

Dictionaries are Python-specific, which means that they do not really exist in other programming languages.

Python Dictionaries have been compared with Associative Arrays[2] in other programming languages, though.

In Python, a Dictionary is an unordered Database of sorts. In a dictionary, each datum has a unique numeric key associated with it.

It is not possible, in dictionaries, to assign two pieces of data to the same numeric key. This is a rule consistent with most databases. Indeed, the definition of a 'database' is a table of data, where each row of the table is assigned a unique numeric key. If one were to assign to two rows of a table of data the same numeric key, then that table of data would then cease, by that very deed, to be a database.

In a Python dictionary, we assign each datum a unique numeric value, or *key*. Hence, in Python, a dictionary is a type of *database*.

In a dictionary, the numeric values that we assign to objects must be *unique*. The numeric values need not be in numeric order. One could have a dictionary containing merely two pieces of data, one having a numeric key of 1, and the other having a numeric key of 60:

[1] Please see the relevant chapters on LISTS and TUPLES.
[2] Please see the relevant chapter on ARRAYS.

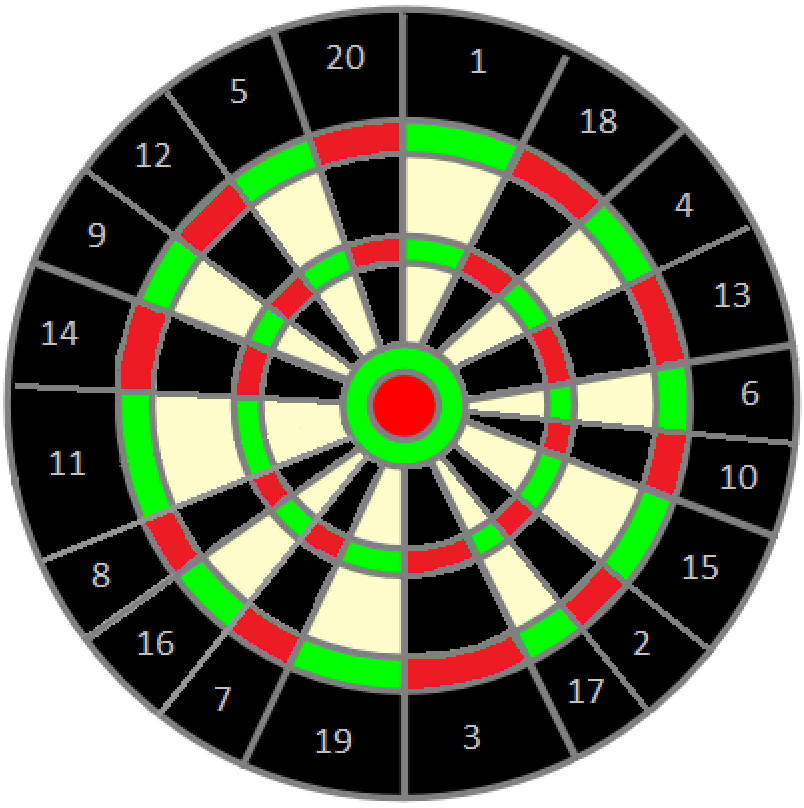
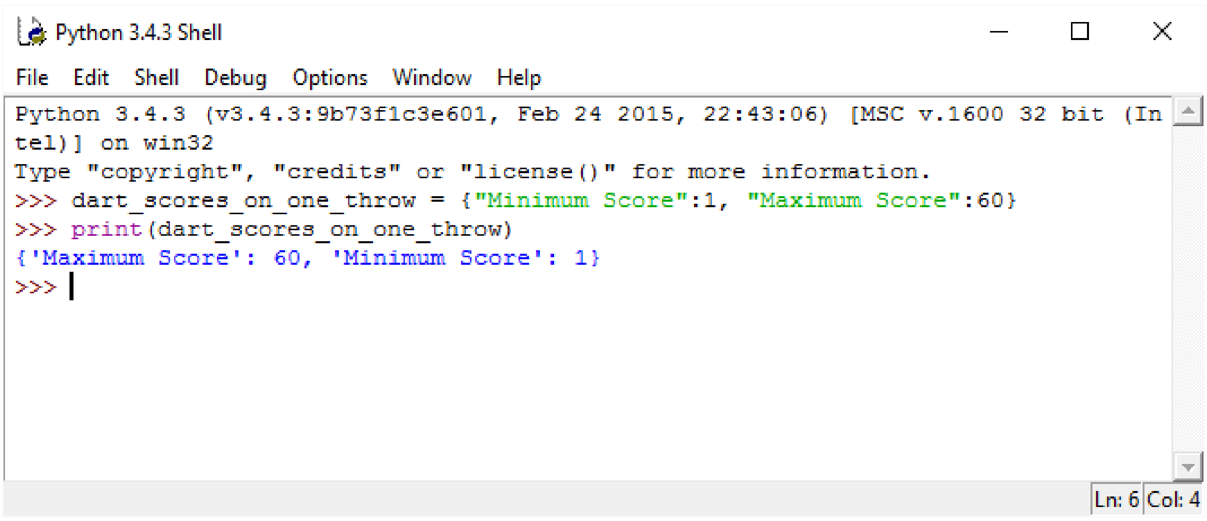


Figure 2: Apart from darts that miss the dartboard completely – humorously termed 'masonry darts' – or darts that ricochet off the metal stripes, the minimum score from one throw is 1. The maximum score from one throw is 60. Below, we contrive a *dictionary* so as to represent this data.

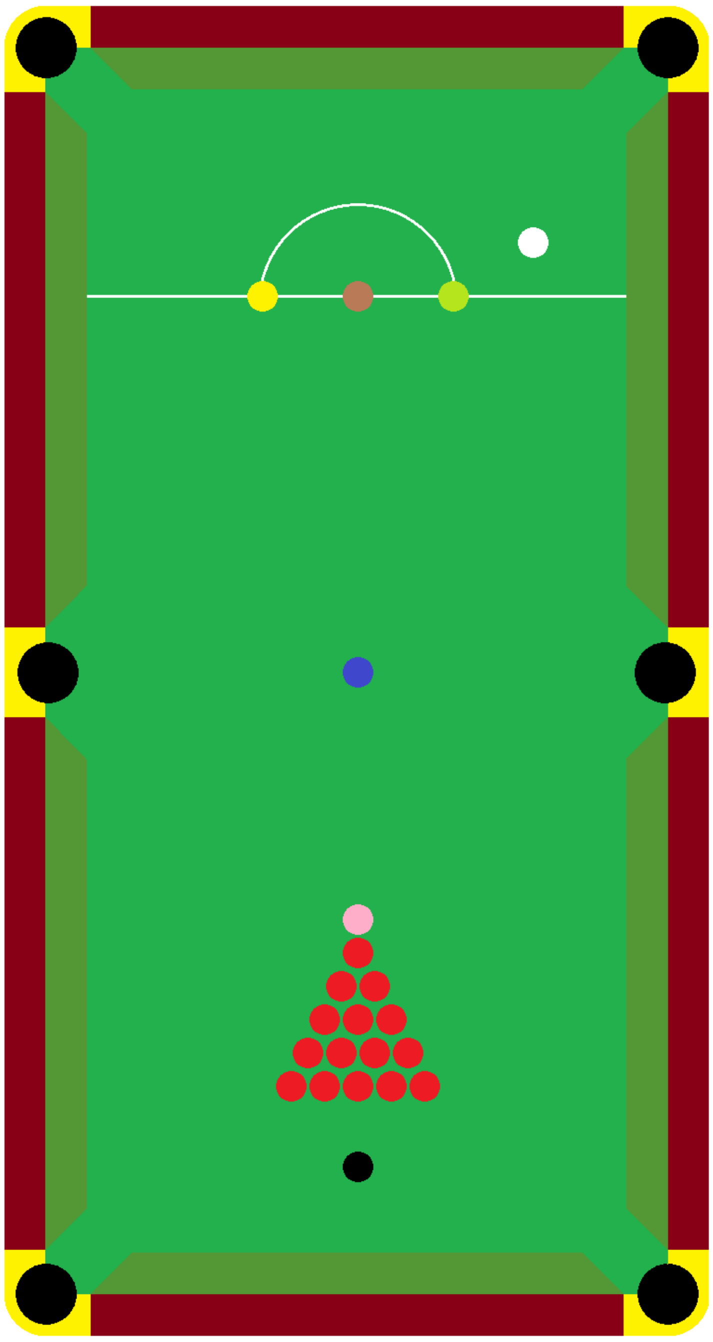
```
>>>dart_scores_on_one_throw = {"Minimum Score":1, "Maximum Score":60}↵

>>> print(dart_scores_on_one_throw)↵

{'Maximum Score': 60, 'Minimum Score': 1,}
```



```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> dart_scores_on_one_throw = {"Minimum Score":1, "Maximum Score":60}
>>> print(dart_scores_on_one_throw)
{'Maximum Score': 60, 'Minimum Score': 1}
>>> 
```

Figure 3: We have created our first dictionary. See how, when we ask it to print out our dictionary, it does not output the data in the same way that we inputted it, i.e. in numeric order. Remember: a dictionary is an *unordered assortment of data*.

We are going to create another dictionary. This dictionary will be a little more complex, in that it will concern the values of balls on a snooker table:

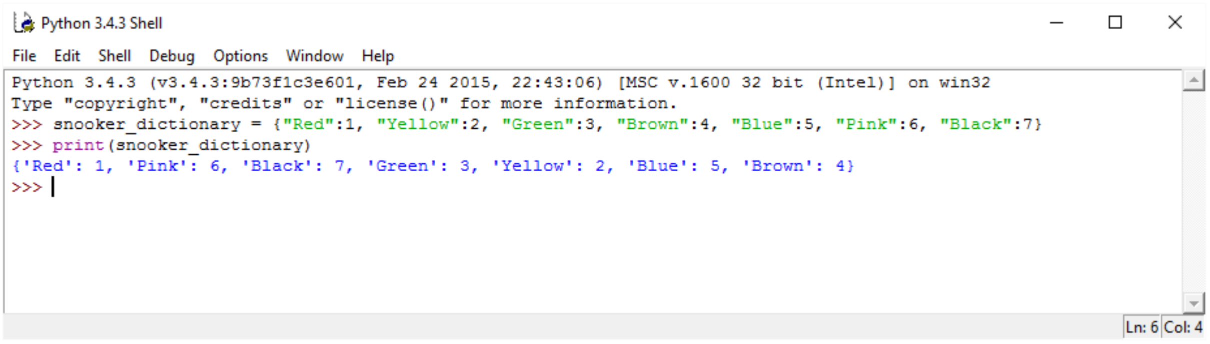


<u>Value of Snooker Table Balls:</u>

Red:	1;
Yellow:	2;
Green:	3;
Brown:	4;
Blue:	5;
Pink:	6;
Black:	7;

(Preceding Page) **Figure 4:** The values of balls on a snooker table. We are going to create a *dictionary* in Python so as to express the above information.

```
>>>snooker_dictionary = {"Red":1, "Yellow":2, "Green":3, "Brown":4, "Blue":5, "Pink":6, "Black":7}↵

>>>print(snooker_dictionary)↵

{'Red': 1, 'Yellow': 2, 'Green': 3, 'Brown': 4, 'Blue': 5, 'Pink': 6, 'Black': 7}
```



```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> snooker_dictionary = {"Red":1, "Yellow":2, "Green":3, "Brown":4, "Blue":5, "Pink":6, "Black":7}
>>> print(snooker_dictionary)
{'Red': 1, 'Pink': 6, 'Black': 7, 'Green': 3, 'Yellow': 2, 'Blue': 5, 'Brown': 4}
>>> 
```

Figure 5: Above is the Python code that we use to create our Python Dictionary.

We employ:

snooker_dictionary

as our dictionary's variable name. We then assign to that variable, the dictionary value:

{"Red":1, "Yellow":2, "Green":3, "Brown":4, "Blue":5, "Pink":6, "Black":7}

by using the assignment/assignation operator:

=

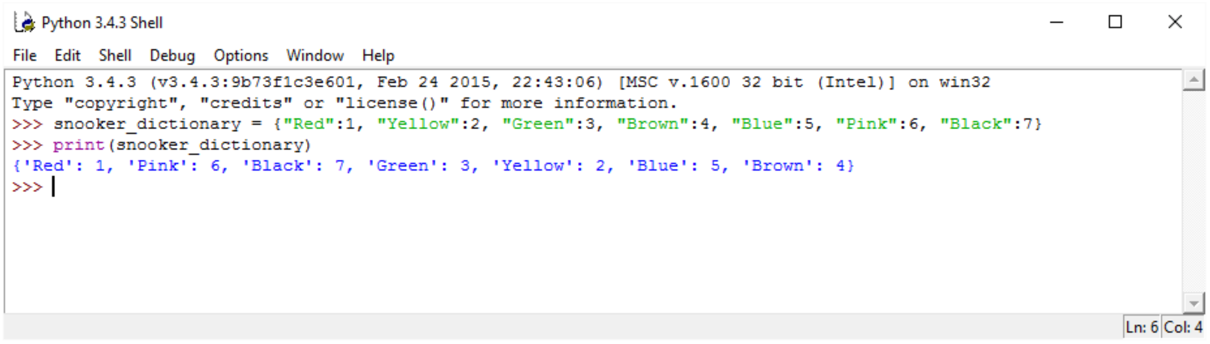
.

We ask Python to print the snooker dictionary:

print(snooker_dictionary)

and Python outputs:

{'Red': 1, 'Pink': 6, 'Black': 7, 'Green': 3, 'Yellow': 2, 'Blue': 5, 'Brown': 4}

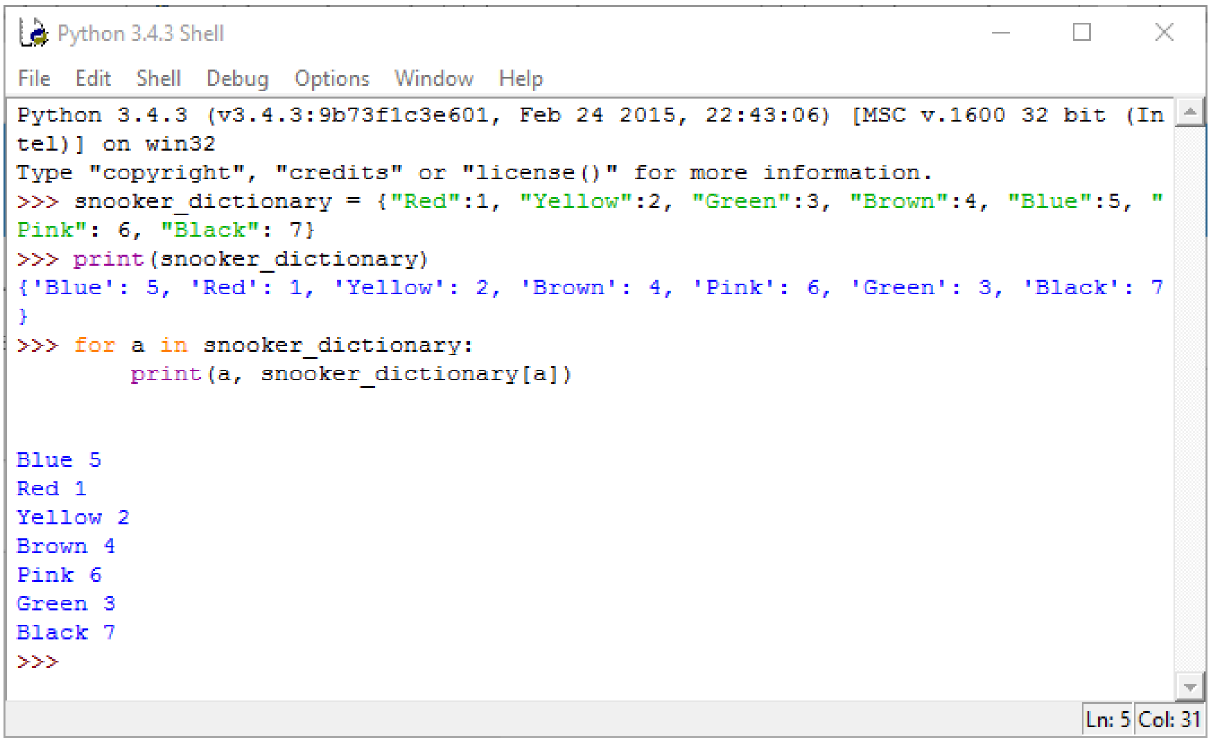


```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> snooker_dictionary = {"Red":1, "Yellow":2, "Green":3, "Brown":4, "Blue":5, "Pink":6, "Black":7}
>>> print(snooker_dictionary)
{'Red': 1, 'Pink': 6, 'Black': 7, 'Green': 3, 'Yellow': 2, 'Blue': 5, 'Brown': 4}
>>> 
```

Figure 5: The code necessary to create a dictionary and its output. Note how Python outputs the dictionary neither how we entered it; neither in alphabetical order; neither in numerical order. Again, a *dictionary* is an unordered database of sorts.

It is possible for us, once a dictionary has been created, to split a dictionary up into separate values. We can do this through the following code:

```
>>>snooker_dictionary = {"Red":1, "Yellow":2, "Green":3, "Brown": 4,
"Pink":6, "Black":7}↵

>>>print(snooker_dictionary)↵

{'Blue': 5, 'Red': 1, 'Yellow': 2, 'Brown': 4, 'Pink': 6, 'Green': 3, 'Black': 7}

>>>for a in snooker_dictionary:↵
        print(a, snooker_dictionary[a])↵

Blue 5

Red 1

Yellow 2

Brown 4

Pink 6

Green 3

Black 7
```



```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> snooker_dictionary = {"Red":1, "Yellow":2, "Green":3, "Brown":4, "Blue":5, "
Pink": 6, "Black": 7}
>>> print(snooker_dictionary)
{'Blue': 5, 'Red': 1, 'Yellow': 2, 'Brown': 4, 'Pink': 6, 'Green': 3, 'Black': 7
}
>>> for a in snooker_dictionary:
        print(a, snooker_dictionary[a])


Blue 5
Red 1
Yellow 2
Brown 4
Pink 6
Green 3
Black 7
>>>
```

Figure 6: What the above code looks like in a Python Interactive window.

A dictionary is a *value* in and of itself. This value, we assigned to the variable:

snooker_dictionary

.

By executing the above code, we have *extracted* the *component values* of the *dictionary* value.

There is another useful thing that we can do to dictionaries: we can *append* new terms to it.

In anatomy, *appendages*, such as arms and legs, are those organs that *hang on* to the trunk, or core.

The term ‘to append,’ etymologically, means ‘to hang [something] on to [something.]’

present active: ‘**pendō**,’ *present infinitive*: ‘**pendere**,’ *perfect active*: ‘**pependī**,’ *supine*: ‘**pensum**’; *third conjugation*.

is the Latin verb, ‘to hang.’

‘ad’

is the Latin preposition that means:

‘to, toward.’

Affix the preposition, ‘ad,’ to the verb, ‘pendō,’ and we get ‘appendō,’ which is the Latin verb, ‘to hang [something] towards [something else.]’

In Python, we can *append* or *hang another* [component value] *on to* our dictionary.

We are able to *change* dictionaries. This means that dictionaries, in programing jargon, are *mutable*[3].

Let us return to our:

snooker_dictionary

and let us imagine a rule change, the addition of a purple ball of value: 8.

[3] From the Latin 1st-conjugation verb, ‘mūtō, mūtāre, mūtāvī, mūtātus,’ which means ‘to change,’ and the Latin 3rd-declension adjective, ‘habilis, habile,’ which means ‘having,’ whence we derive the Latin adjective-making suffix, ‘-abilis, -abile.’ Combine the Latin verb, ‘mūtō,’ with the suffix, ‘-abilis, -abile’ and we get the 3rd-declension Latin adjective, ‘mūtābilis, mūtābile,’ which denotes something that *has* [the *ability*] to *change*. In Python Programming, dictionaries *have* the *ability* to *change*.

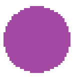
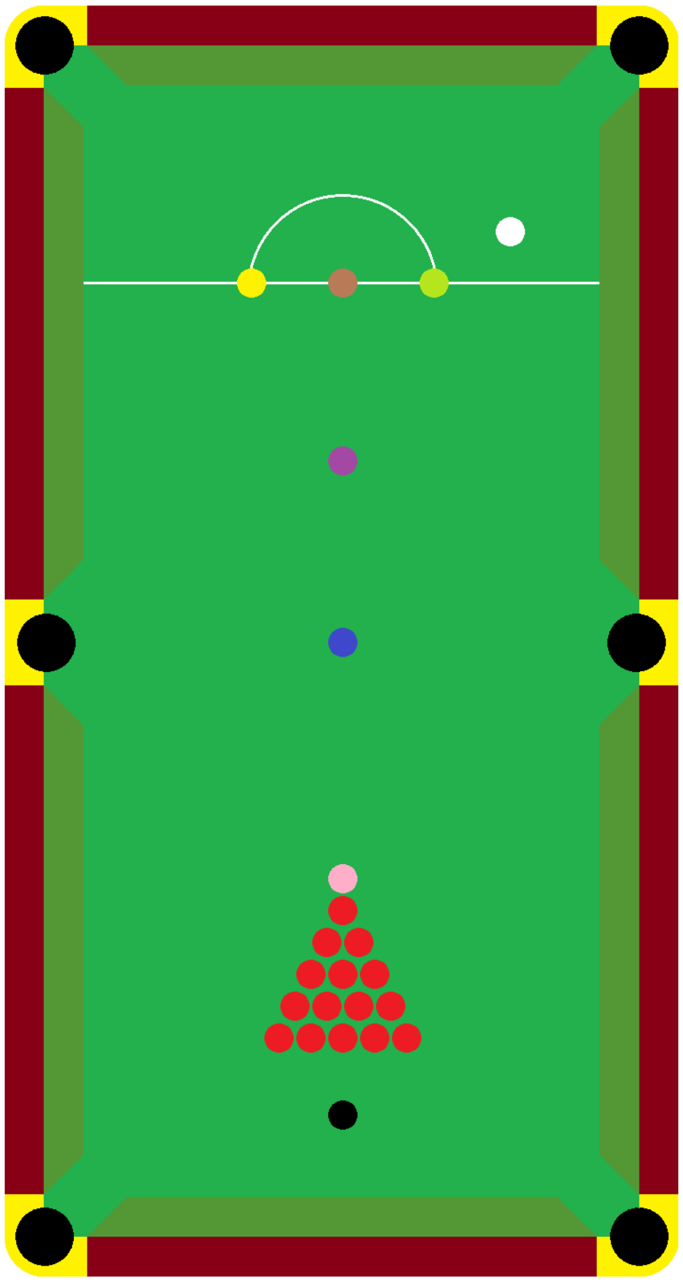


Figure 7: The purple ball. Just imagine the possibilities!



<u>Value of Snooker Table Balls:</u>

Red:	1;
Yellow:	2;
Green:	3;
Brown:	4;
Blue:	5;
Pink:	6;
Black:	7;
Purple:	8.

(From Preceding Page) **Figure 8:** We shall position the purple ball just above the blue, I think.

The rules of snooker have been changed, which necessitates a *change* in the value of our:

snooker_dictionary

variable. Luckily, *dictionaries* are mutable, and it will not be difficult for us to amend our dictionary so as to reflect the addition of a purple ball into the game of snooker:

```
>>> snooker_dictionary = {"Red":1, "Yellow":2, "Green":3, "Brown":4, "Blue":5, "Pink": 6, "Black": 7}↵

>>> print(snooker_dictionary) ↵

{'Yellow': 2, 'Green': 3, 'Brown': 4, 'Black': 7, 'Red': 1, 'Pink': 6, 'Blue': 5}

>>> snooker_dictionary['Purple'] = 8↵

>>> print(snooker_dictionary)↵

{'Yellow': 2, 'Green': 3, 'Purple': 8, 'Brown': 4, 'Black': 7, 'Red': 1, 'Pink': 6, 'Blue': 5}

>>>
```

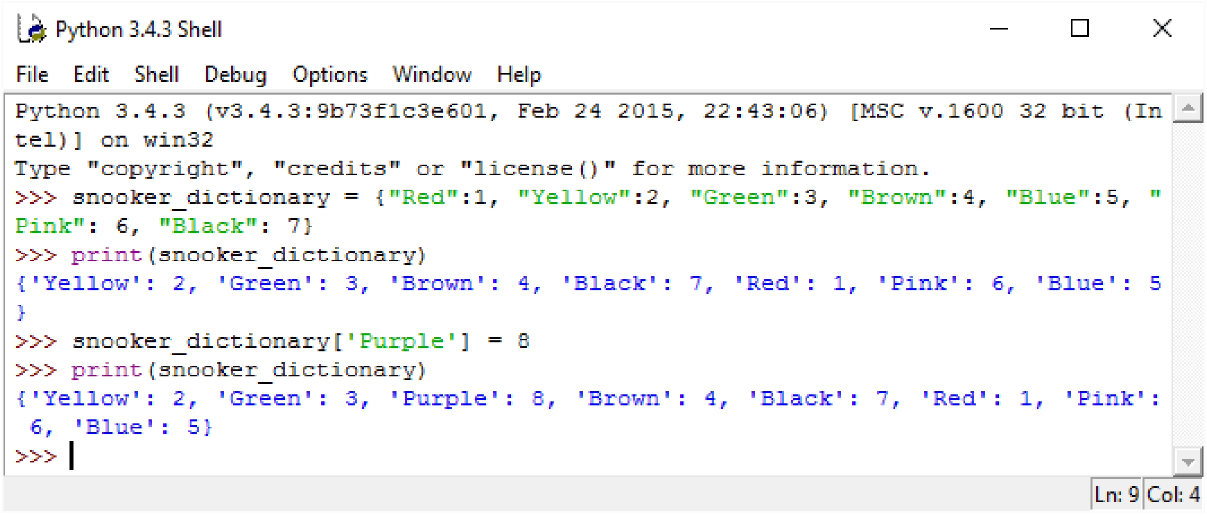
From the above code, we can garner that it is the:

snooker_dictionary['Purple'] = 8

piece of code that appends the value:

'Purple': 8

to our pre-existing dictionary.



```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> snooker_dictionary = {"Red":1, "Yellow":2, "Green":3, "Brown":4, "Blue":5, "
Pink": 6, "Black": 7}
>>> print(snooker_dictionary)
{'Yellow': 2, 'Green': 3, 'Brown': 4, 'Black': 7, 'Red': 1, 'Pink': 6, 'Blue': 5
}
>>> snooker_dictionary['Purple'] = 8
>>> print(snooker_dictionary)
{'Yellow': 2, 'Green': 3, 'Purple': 8, 'Brown': 4, 'Black': 7, 'Red': 1, 'Pink':
 6, 'Blue': 5}
>>> 
```

Figure 8: What the code looks like executed in a Python Interactive Window.

In Conclusion:

In this chapter we have examined what a dictionary is – a type of unordered database; created a dictionary; extracted component values from a dictionary; and we have also appended new information to a dictionary. For a beginner programmer, such as yourself, the reader, this is all you will be required to know of this topic at the moment.

Dictionaries are a kind of an array, which means that a Dictionary can be a sub-element in another dictionary; a super-dictionary. We do not need to examine multidimensional dictionaries yet, though.