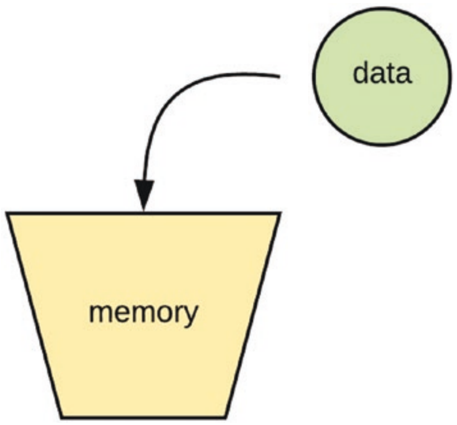

***Figure 9-1.** An illustration of a memory cell holding data*

In programming, a memory location is called a variable. A **variable** is a container for storing the data that is assigned to it. A variable is usually given a unique name by the programmer to represent a particular memory cell. In python, variable names are programmer defined, but it must follow a valid naming condition of only alphanumeric lowercase characters with words separated by an underscore. Also, a variable name should have semantic meaning to the data that is stored in that variable. This helps to improve code readability later in the future.

The act of placing data to a variable is called **assignment**.

```
# assigning data to a variable
x = 1
user_name = 'Emmanuel Okoi'
```

Data Types

Python has the number and string data types in addition to other supported specialized datatypes. The number datatype, for instance, can be an int or a float. Strings are surrounded by quotes in Python.

```
# data types
type(3)
'Output': int
type(3.0)
'Output': float
```

```
type('Jesam Ujong')
'Output': str
```

Other fundamental data types in Python include the lists, tuple, and dictionary. These data types hold a group of items together in sequence. Sequences in Python are indexed from 0.

Tuples are an immutable ordered sequence of items. Immutable means the data cannot be changed after being assigned. Tuple can contain elements of different types. Tuples are surrounded by brackets (…).

```
my_tuple = (5, 4, 3, 2, 1, 'hello')
type(my_tuple)
'Output': tuple
my_tuple[5]           # return the sixth element (indexed from 0)
'Output': 'hello'
my_tuple[5] = 'hi'    # we cannot alter an immutable data type
Traceback (most recent call last):

  File "<ipython-input-49-f0e593f95bc7>", line 1, in <module>
    my_tuple[5] = 'hi'

TypeError: 'tuple' object does not support item assignment
```

Lists are very similar to tuples, only that they are mutable. This means that list elements can be changed after being assigned. Lists are surrounded by square brackets […].

```
my_list = [4, 8, 16, 32, 64]
print(my_list)    # print list items to console
'Output': [4, 8, 16, 32, 64]
my_list[3]        # return the fourth list element (indexed from 0)
'Output': 32
my_list[4] = 256
print(my_list)
'Output': [4, 8, 16, 32, 256]
```

Dictionaries contain a mapping from keys to values. A key/value pair is an item in a dictionary. The items in a dictionary are indexed by their keys. The keys in a dictionary can be any *hashable* datatype (hashing transforms a string of characters into a key to speed up search). Values can be of any datatype. In other languages, a dictionary is analogous to a hash table or a map. Dictionaries are surrounded by a pair of braces {...}. A dictionary is not ordered.

```
my_dict = {'name':'Rijami', 'age':42, 'height':72}
my_dict               # dictionary items are un-ordered
'Output': {'age': 42, 'height': 72, 'name': 'Rijami'}
my_dict['age']        # get dictionary value by indexing on keys
'Output': 42
my_dict['age'] = 35   # change the value of a dictionary item
my_dict['age']
'Output': 35
```

More on Lists

As earlier mentioned, because list items are mutable, they can be changed, deleted, and sliced to produce a new list.

```
my_list = [4, 8, 16, 32, 64]
my_list
'Output': [4, 8, 16, 32, 64]
my_list[1:3]      # slice the 2nd to 4th element (indexed from 0)
'Output': [8, 16]
my_list[2:]       # slice from the 3rd element (indexed from 0)
'Output': [16, 32, 64]
my_list[:4]       # slice till the 5th element (indexed from 0)
'Output': [4, 8, 16, 32]
my_list[-1]       # get the last element in the list
'Output': 64
min(my_list)      # get the minimum element in the list
'Output': 4
max(my_list)      # get the maximum element in the list
'Output': 64
sum(my_list)      # get the sum of elements in the list
'Output': 124
my_list.index(16) # index(k) - return the index of the first occurrence of
item k in the list
'Output': 2
```