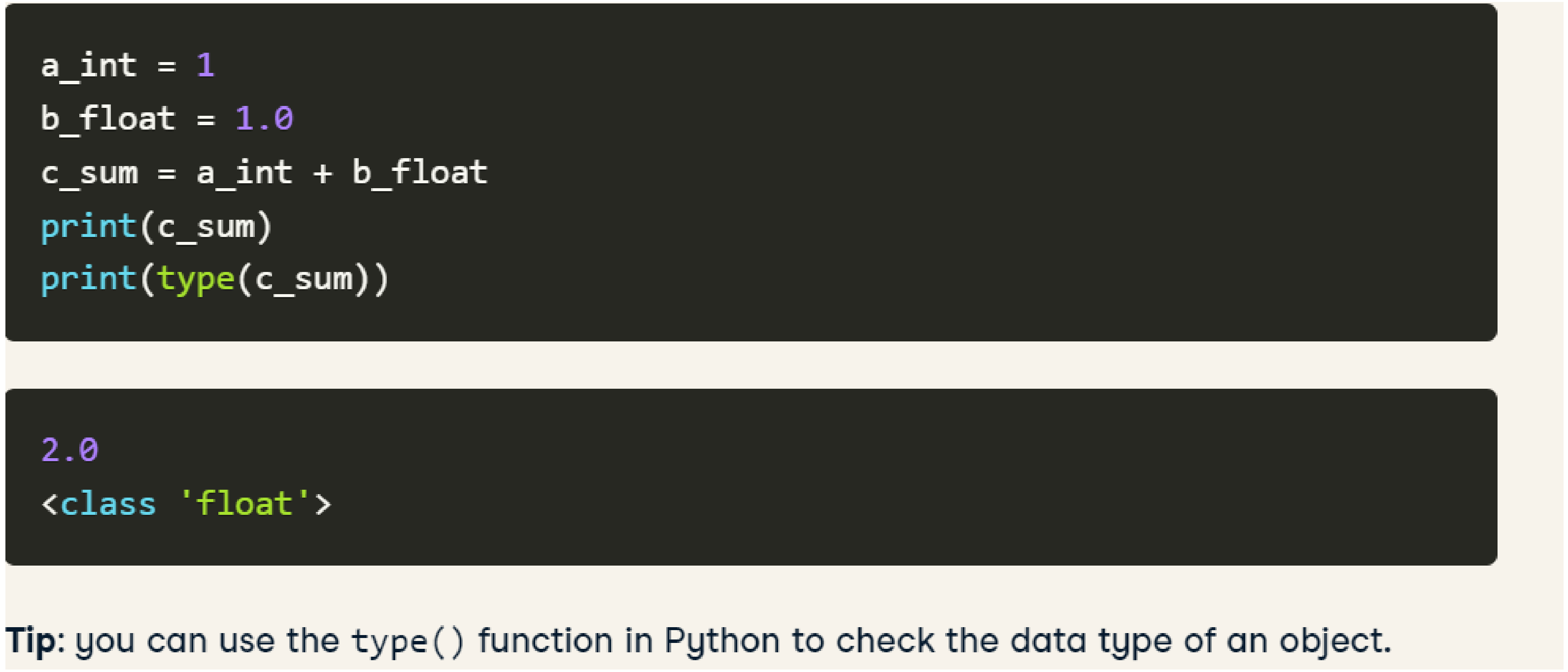
Data Type Conversion

Python has many data types. You must have already seen and worked with some of them.
You have integers and float to deal with numerical values, boolean (bool) to deal with true/false values
and strings to work with alphanumeric characters.
You can make use of lists, tuples, dictionary, and sets that are data structures where you can store a collection of values.

Python Implicit Data Type Conversion



```python
a_int = 1
b_float = 1.0
c_sum = a_int + b_float
print(c_sum)
print(type(c_sum))
```

```
2.0
<class 'float'>
```

Tip: you can use the `type()` function in Python to check the data type of an object.

Primitive Versus Non-Primitive Data Structures

Primitive data structures are the building blocks for data manipulation and contain pure, simple values of data. Python has four primitive variable types:

- Integers
- Float
- Strings
- Boolean

Non-primitive data structures don't just store a value, but rather a collection of values in various formats. In Python, you have the following non-primitive data structures:

- Lists
- Tuples
- Dictionary
- Sets

Integer and Float Conversions

Integers and floats are data types that deal with numbers.

To convert the integer to float, use the `float()` function in Python. Similarly, if you want to convert a float to an integer, you can use the `int()` function.

```python
a_int = 3
b_int = 2

# Explicit type conversion from int to float
c_float_sum = float(a_int + b_int)
print(c_float_sum)
```

```
5.0
```

```python
a_float = 3.3
b_float = 2.0

# Explicit type conversion from float to int
c_int_sum = int(a_float + b_float)
print(c_int_sum)

c_float_sum = a_float + b_float
print(c_float_sum)
```

```
5
5.3
```

Data Type Conversion with Strings

A string is a collection of one or more characters (letters, numbers, symbols). You may need to convert strings to numbers or numbers to strings fairly often. Check out how you can do this using the `str()` function:

```
price_cake = 15
price_cookie = 6
total = price_cake + price_cookie
print("The total is: " + total  + "$")
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-12-54bd76b9b4bd> in <module>()
      2 price_cookie = 6
      3 total = price_cake + price_cookie
----> 4 print("The total is: " + total  + "$")


TypeError: Can't convert 'int' object to str implicitly
```

```
price_cake = 15
price_cookie = 6
total = price_cake + price_cookie
print("The total is: " + str(total)  + "$")
```

```
The total is: 21$
```

In Python, you can also convert strings to integer and float values whenever possible.
Let's see what this means:

```
price_cake = '15'
price_cookie = '6'

# String concatenation
total = price_cake + price_cookie
print("The total is: " + total + "$")

# Explicit type conversion to integer
total = int(price_cake) + int(price_cookie)
print("The total is: " + str(total)  + "$")
```

```
The total is: 156$
The total is: 21$
```

Type Conversion to Tuples and Lists

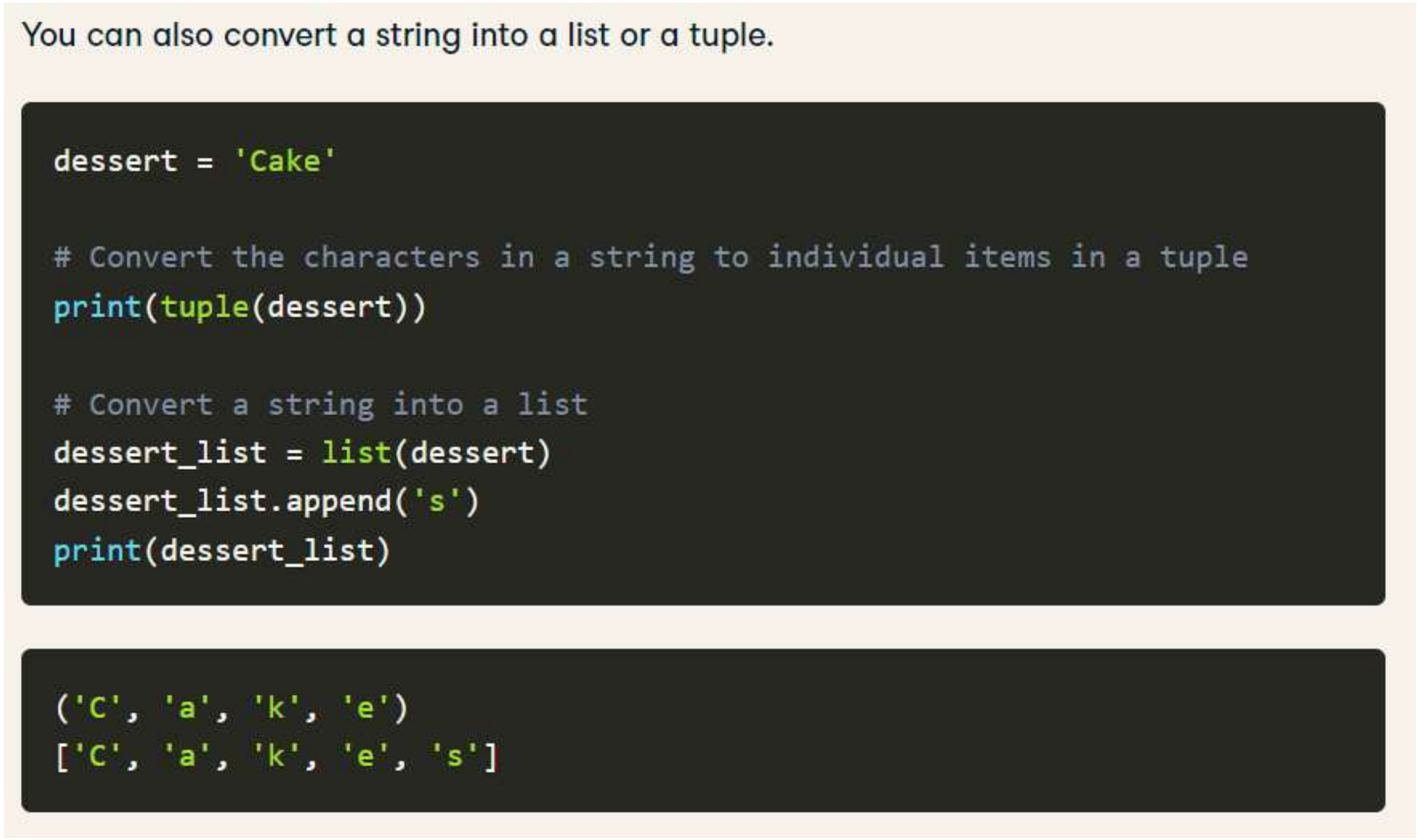
Just like with integers and floats, you can also convert lists to tuples and tuples to lists.

Remember what tuple and lists are? Lists and Tuples in Python are used to store a collection of homogeneous items. The difference between tuples and list is that tuples are immutable, which means once defined you cannot delete, add or edit any values inside it.

You can use the `tuple()` function to return a tuple version of the value passed to it and similarly the `list()` function to convert to a list:

```
a_tuple = (1,2,3,4,5,6,7,8,9,10)
b_list = [1,2,3,4,5]

print(tuple(b_list))
print(list(a_tuple))
```

```
(1, 2, 3, 4, 5)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```



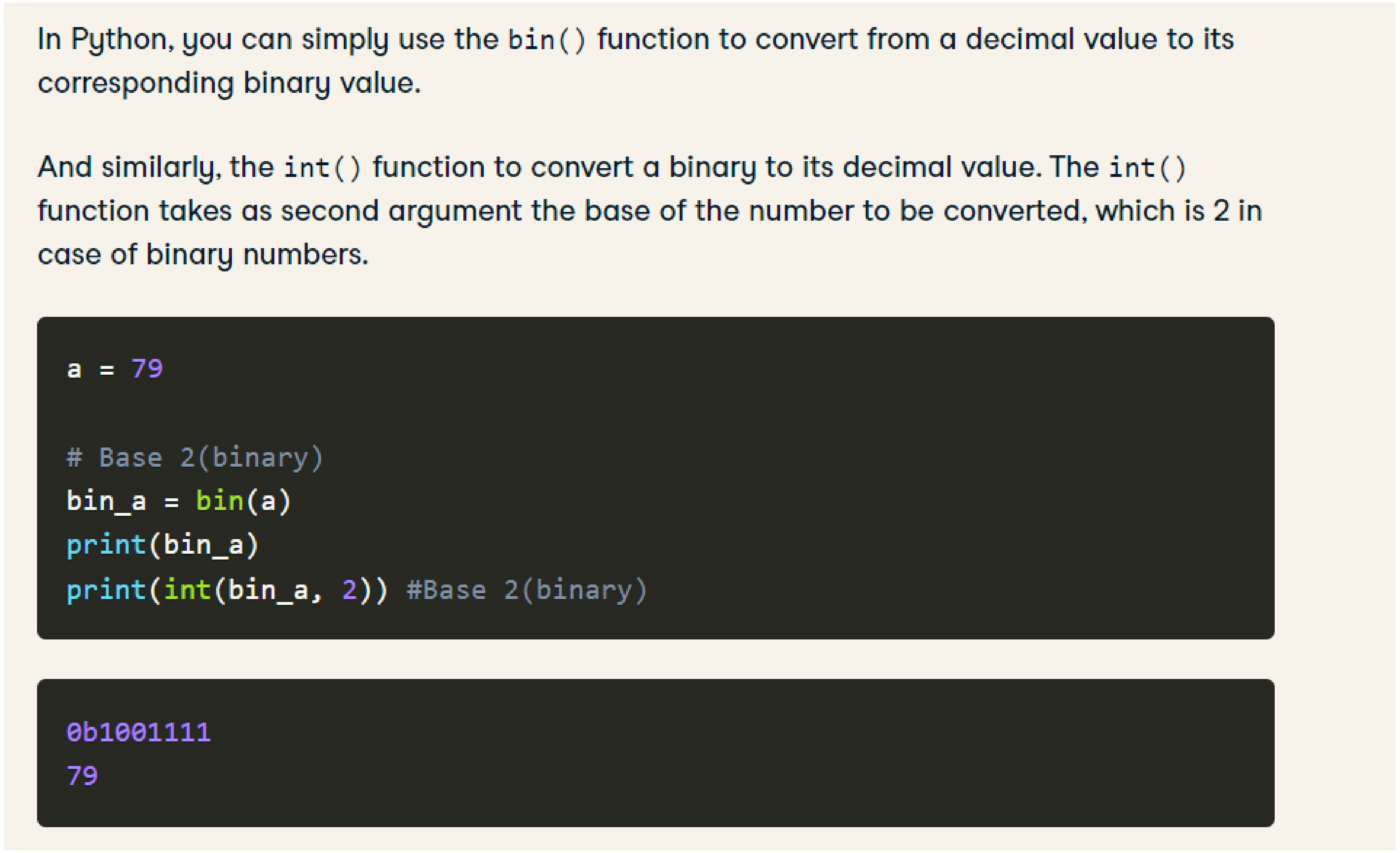
You can also convert a string into a list or a tuple.

```
dessert = 'Cake'

# Convert the characters in a string to individual items in a tuple
print(tuple(dessert))

# Convert a string into a list
dessert_list = list(dessert)
dessert_list.append('s')
print(dessert_list)
```

```
('C', 'a', 'k', 'e')
['C', 'a', 'k', 'e', 's']
```

Convert to Binary Number

Binary integers are the number represented with base two. Which means in the binary number system, there are only two symbols used to represent numbers: 0 and 1.



In Python, you can simply use the `bin()` function to convert from a decimal value to its corresponding binary value.

And similarly, the `int()` function to convert a binary to its decimal value. The `int()` function takes as second argument the base of the number to be converted, which is 2 in case of binary numbers.

```python
a = 79

# Base 2(binary)
bin_a = bin(a)
print(bin_a)
print(int(bin_a, 2)) #Base 2(binary)
```

```
0b1001111
79
```

Conversion to Octal

In Python, you can use the `oct()` function to convert from a decimal value to its corresponding octal value. Alternatively, you can also use the `int()` function along with the correct base which is 8 for the octal number system.

```python
a = 79

# Base 8(octal)
oct_a = oct(a)
print(oct_a)
print(int(oct_a, 8))
```

```
0o117
79
```

Type conversion to Hexadecimal

Hexadecimal is a base 16 number system.

In Python, you can use the `hex()` function to convert from a decimal value to its corresponding hexadecimal value, or the `int()` function with base 16 for the hexadecimal number system.

```
a = 79

# Base 16(hexadecimal)
hex_a = hex(a)
print(hex_a)
print(int(hex_a, 16))
```

```
0x4f
79
```