Tuples

Tuples group multiple values into a single compound value. The values within a tuple can be of any type and don't have to be of the same type as each other.

In this example, `(404, "Not Found")` is a tuple that describes an *HTTP status code*. An HTTP status code is a special value returned by a web server whenever you request a web page. A status code of `404 Not Found` is returned if you request a webpage that doesn't exist.

```
let http404Error = (404, "Not Found")
// http404Error is of type (Int, String), and equals (404, "Not Found")
```

The `(404, "Not Found")` tuple groups together an `Int` and a `String` to give the HTTP status code two separate values: a number and a human-readable description. It can be described as "a tuple of type `(Int, String)`".

You can create tuples from any permutation of types, and they can contain as many different types as you like. There's nothing stopping you from having a tuple of type `(Int, Int, Int)`, or `(String, Bool)`, or indeed any other permutation you require.

You can *decompose* a tuple's contents into separate constants or variables, which you then access as usual:

```
let (statusCode, statusMessage) = http404Error
print("The status code is \(statusCode)")
// Prints "The status code is 404"
print("The status message is \(statusMessage)")
// Prints "The status message is Not Found"
```

If you only need some of the tuple's values, ignore parts of the tuple with an underscore (_) when you decompose the tuple:

```
let (justTheStatusCode, _) = http404Error
print("The status code is \(justTheStatusCode)")
// Prints "The status code is 404"
```

Alternatively, access the individual element values in a tuple using index numbers starting at zero:

```
print("The status code is \(http404Error.0)")
// Prints "The status code is 404"
print("The status message is \(http404Error.1)")
// Prints "The status message is Not Found"
```

You can name the individual elements in a tuple when the tuple is defined:

```
let http200Status = (statusCode: 200, description: "OK")
```

If you name the elements in a tuple, you can use the element names to access the values of those elements:

```
print("The status code is \(http200Status.statusCode)")
// Prints "The status code is 200"
print("The status message is \(http200Status.description)")
// Prints "The status message is OK"
```

Tuples are particularly useful as the return values of functions. A function that tries to retrieve a web page might return the `(Int, String)` tuple type to describe the success or failure of the page retrieval. By returning a tuple with two distinct values, each of a different type, the function provides more useful information about its outcome than if it could only return a single value of a single type. For more information, see Functions with Multiple Return Values.

> NOTE
>
> Tuples are useful for simple groups of related values. They're not suited to the creation of complex data structures. If your data structure is likely to be more complex, model it as a class or structure, rather than as a tuple. For more information, see Structures and Classes.