AL NAFI,
A company with a focus on education,
wellbeing and renewable energy.

Python Alpha 200

Lists, Tuples and Dictionaries

Dua of the day to recite after Takbeer in Salah

اللَّهُمَّ بَاعِدْ بَيْنِي وَ بَيْنَ خَطَايَايَ كَمَا
بَاعَدْتَ بَيْنَ الْمَشْرِقِ وَ الْمَغْرِبِ،
اللَّهُمَّ نَقِّنِي مِنْ خَطَايَايَ كَمَا يُنَقَّى الثَّوْبُ
الأَبْيَضُ مِنَ الدَّنَسِ، اللَّهُمَّ اغْسِلْنِي مِنْ
خَطَايَايَ بِالثَّلْجِ وَ الْمَاءِ وَ الْبَرَدِ

O Allah , separate me from my sins as You have separated the East from the West. O Allah, cleanse me of my transgressions as the white garment is cleansed of stains . O Allah , wash away my sins with ice and water and frost.

Al-Bukhari 1/181, Muslim 1/419

Study, Rinse and Repeat

- Please subscribe to our YouTube Channel to be on top of your studies.
- Please logon to our website https://alnafi.com/login/
- Use your username and password to logon
- Please keep an eye on zone@alnafi.com emails
- Please review the videos of 30 minutes daily
- Please review the notes daily.
- Please take time for clearing up your mind and reflect on how things are proceeding in your studies.

List, Tuples and Dictionaries

Python has several other important data types like Lists Tuples and Dictionaries, that we'll probably use every day down the line in using ML, DL, AI, Cyber Security and even software development

The 200 lecture aim is to get you all acquainted with each of these data types.

Once you have mastered these three data types plus the string data type from the previous lecture, you will be quite a ways along in your education of Python. You'll be using these four building blocks in 99% of all the applications you will write.

Lists

A Python list is similar to displaying results or a range.

In Python, an empty list can be created in the following ways.

```
my_list = []
or
my_list = list()
```

As you can see, you can create the list using square brackets or by using the Python built-in, list. A list contains a list of elements, such as strings, integers, objects or a mixture of types.

List examples

Let's take a look at some examples:

```
my_list = [1, 2, 3]
```

Or

```
my_list2 = ["a", "b", "c"]
```

Or

```
my_list3 = ["a", 1, "Python", 5]
```

Nested list

You can also create lists of lists like this:

```
my_list = [1, 2, 3]

my_list2 = ["a", "b", "c"]

my_list3 = ["a", 1, "Python", 5]

my_nested_list = [my_list, my_list2, my_list3]

print(my_nested_list)
```

Occasionally we will combine two or three lists together.

List extend method

```
combo_list = []
one_list = [4, 5]
combo_list.extend(one_list)
combo_list
```

A slightly easier method to add two lists

```
my_list = [1, 2, 3]
my_list2 = ["a", "b", "c"]
combo_list = my_list + my_list2
combo_list
```

Sorting out a list

```
alpha_list = [34, 23, 67, 100, 88, 2]
alpha_list.sort()
alpha_list
```

A none list

```
alpha_list = [34, 23, 67, 100, 88, 2]
sorted_list = alpha_list.sort()
sorted_list
print(sorted_list)
```

In this example, if we try to assign the sorted list to a variable.

However, when you call the sort() method on a list, it sorts the list in-place. So if we try to assign the result to another variable, then we'll find out that we'll get a None object, which is like a Null in other languages.

Thus when we want to sort something, just remember that we sort them in-place and we cannot assign it to a different variable.

Slicing a list

```
alpha_list = [34, 23, 67, 100, 88, 2]
alpha_list[0:3]
```

Tuples

A tuple is similar to a list, but you create them with parentheses instead of square brackets.

You can also use the tuple built-in. The main difference is that a tuple is immutable (Cannot be changed) while the list is mutable(can be changed).

How I remember Tuples ☺

Tuples are like couples which means they cannot be changed ☺

Tuple example

```
my_tuple=(1,2,3,4,5)
my_tuple[0:3]
```

Or an empty

```
another_tuple=tuple()
```

Tuple casting which means a list inside a tuple

```
abc=tuple([1,2,3])
```

This example has three elements inside it. Notice that it has a list inside it. This is an example of casting. We can change or cast an item from one data type to another. In this case we cast a list into a tuple.

If we want to change the abc tuple back into a list, we can do the following:

```
abc_list=list(abc)
```

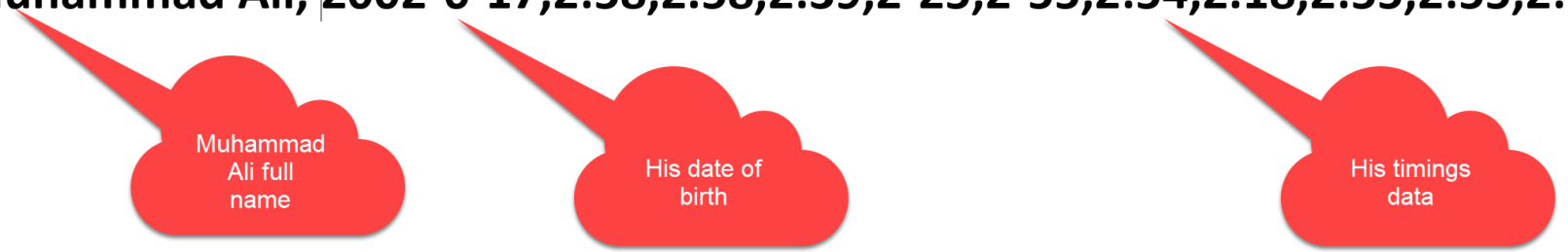
Dictionaries

Using a dictionary to associate value

Lists are *great*, but they are not always the best data structure for every situation.

Dictionary example

Muhammad Ali, 2002-6-17,2:58,2.58,2:39,2-25,2-55,2:54,2.18,2:55,2:55,2:22,2-21,2.22



Let's take a look at Muhammad Ali's data:

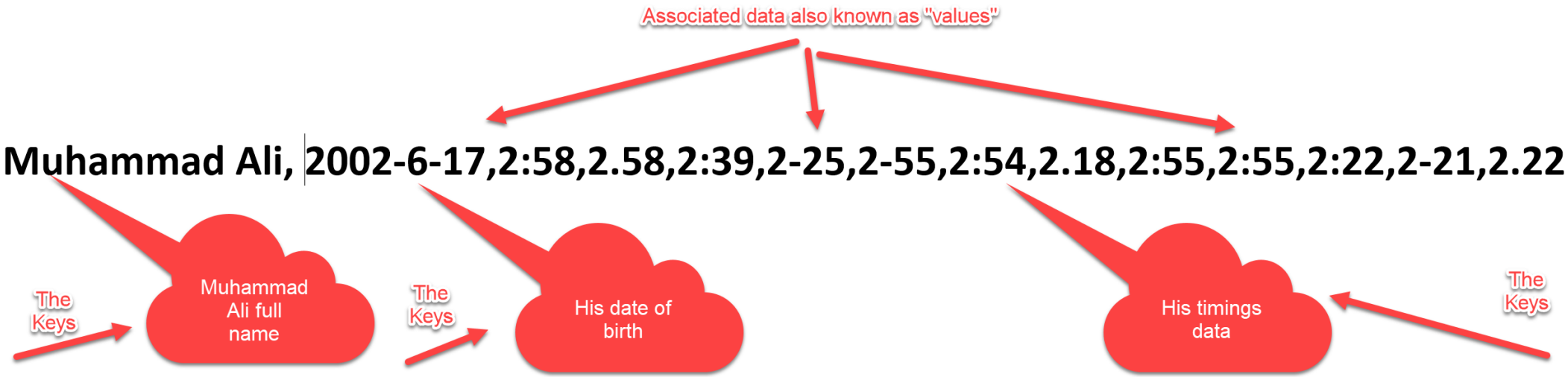
There's a definite **structure** here: the athlete's name, the date of birth, and then the list of times.

Let's continue to use a list for the timing data, because that still makes sense.

But let's make the timing data part of *another* data structure, which associates all the data for an athlete with a single variable.

For that we use a Python **dictionary**, which *associates data values with keys*:

Another closer look at the data



Python Dictionary revisited

Python dictionary is a hash table with indexed keys which can be immutable type. For example a string or number can be a key.

We need to be aware that dictionary is an unordered set of key:value pairs and the keys must be unique.

We can also call a dictionary instance's **key** to check if dictionary has a key using **in**

Empty Dictionary code examples

Empty dictionary

```
my_dict = {}
```

Or

```
another_dict = dict()
```

Dictionary with values

```
my_other_dict = {"one":1, "two":2, "three":3}
my_other_dict
```

Note:

All dictionaries are enclosed with curly braces. The last line is printed out so you can see how unordered a dictionary is. Now lets find out how to access a value in a dictionary.

Accessing a value within a dictionary

```
my_dict = {"name":"Sharfoo", "address":"123 Gulshan"}
my_dict["name"]
```

Adding true and false statements

"name" in my_dict

True

"state" in my_dict

False

Boolean True and false is something we will cover later ☺ this is just a place holder for the upcoming lecture.

Checking all the keys in a dictionary

my_dict = {"name":"Sharfoo", "address":"123 Gulshan"}

my_dict.keys()

To check values we use

my_dict.keys()

The importance of **in**

```
"name" in my_dict        # this is good
"name" in my_dict.keys()  # this works too, but is slower
```

While this probably won't matter much to us right now, in a real job situation, seconds matter. When we have thousands of files to process, these little tricks can save us a lot of time in the long run!

جزاك اللهُ

To ask questions, please logon to the portal https://alnafi.com/login/ and use your username and password. From within the portal you can ask questions. We will only answer questions if they are coming through the portal and not on email.

For any other queries please reach out on info@alnafi.com