Programming Thinking

Session 10

Pepe García

SCHOOL OF SCIENCE & TECHNOLOGY

Mutating lists

Lists are mutable values, and they provide functionality to add, delete, and update elements

Updating elements in the list

To update an element inside the list, we use a syntax similar to the one for declaring variables, but using the brackets and the index we refer to.

```
numbers = [1,2,4]
numbers[2] = 3
print(numbers) # prints [1,2,3]
```

SCHOOL OF SCIENCE & TECHNOLOGY

Updating elements in the list

To update an element inside the list, we use a syntax similar to the one for declaring variables, but using the brackets and the index we refer to.

```
numbers = [1,2,4]
numbers[2] = 3
print(numbers) # prints [1,2,3]
```

Demo

SCHOOL OF SCIENCE & TECHNOLOGY

Appending elements to the list

To add a new element to the end of the list we use the append() method on it.

```
numbers = [1,2,3]
numbers.append(4)
print(numbers) # prints [1,2,3,4]
```

SCHOOL OF SCIENCE & TECHNOLOGY

Appending elements to the list

To add a new element to the end of the list we use the append() method on it.

```
numbers = [1,2,3]
numbers.append(4)
print(numbers) # prints [1,2,3,4]
```

Demo

SCHOOL OF SCIENCE & TECHNOLOGY

Inserting elements in the list

There's an alternative way of adding new elements to the list, and it's using the insert() method on it:

```
words = ["hello","my","friends"]
words.insert(2, "dear")
print(words) # prints ["hello", "my", "dear", "friends"]
```

The difference between this and append is that with insert we can choose where to put it by using the target index

SCHOOL OF SCIENCE & TECHNOLOGY

Inserting elements in the list

There's an alternative way of adding new elements to the list, and it's using the insert() method on it:

```
words = ["hello","my","friends"]
words.insert(2, "dear")
print(words) # prints ["hello", "my", "dear", "friends"]
```

The difference between this and append is that with insert we can choose where to put it by using the target index

Demo

SCHOOL OF SCIENCE & TECHNOLOGY

Removing elements from the list

In order to remove an element from a list, we should use the .pop() method, and pass the index of the element we want to remove

```
words = ["hello","my","friend"]
words.pop(1)
print(words)
```

SCHOOL OF SCIENCE & TECHNOLOGY

Removing elements from the list

In order to remove an element from a list, we should use the .pop() method, and pass the index of the element we want to remove

```
words = ["hello","my","friend"]
words.pop(1)
print(words)
```

Demo

SCHOOL OF SCIENCE & TECHNOLOGY

For loops

For loops are simpler than while loops. They iterate over elements in a list. They loop once for every element in the list.

For loops

For loops are simpler than while loops. They iterate over elements in a list. They loop once for every element in the list.

```
for <iteration_variable> in <list>:
    <body>
```

In each iteration, we will have a new value for the **`iteration_variable`**.

SCHOOL OF SCIENCE & TECHNOLOGY

For loops

For loops are simpler than while loops. They iterate over elements in a list. They loop once for every element in the list.

```
for <iteration_variable> in <list>:
    <body>
```

In each iteration, we will have a new value for the **`iteration_variable`**.

Demo

Let's see an example for adding all numbers in a list

SCHOOL OF SCIENCE & TECHNOLOGY

Practice

Create a function `to_string` that receives a list of strings, concatenates all of them and returns it as a single string.

SCHOOL OF SCIENCE & TECHNOLOGY

Checkpoint

Checkpoint

Is everybody following so far? Is there any question, comment?

Dictionaries

Dictionaries are another kind of collection in Python. Dictionaries map keys to values.

SCHOOL OF SCIENCE & TECHNOLOGY

Creating dictionaries

We use curly brackets {} to declare dictionaries.

```
translations = {
    "es": "Hola!",
    "it": "Ciao!",
    "en": "Hello!"
}
```

colon for separating key and value

comma for separating entries

SCHOOL OF SCIENCE & TECHNOLOGY

Creating dictionaries

We can also create empty dictionaries

```
translations = {}
```

Creating dictionaries

Adding elements

We add elements to dictionaries given their specific index:

```
translations = {}
translations["en"] = "Hello"
translations["it"] = "Ciao"
translations["es"] = "Hola"
```

SCHOOL OF SCIENCE & TECHNOLOGY

Updating elements

we always can change a value in the dictionary by re-assigning the key

```
translations = {}
translations["en"] = "Hello"
translations["en"] = "WHATUP!"
```

SCHOOL OF SCIENCE & TECHNOLOGY

Updating elements

Deleting elements

We can delete an element of the dictionary using the **pop** method

```
translations = {}
translations["en"] = "Hello"
translations.pop("en")
```

SCHOOL OF SCIENCE & TECHNOLOGY

Deleting elements

Getting all keys or values

We can allways get all **keys** or **values** from the dict as a list using either the **.keys()** or **.values()** method

```
users = {
  1: "Pepe",
  22: "Peter",
  44143: "Johnny",
  2: "Chuck"
}

users.keys()
users.values()
```

SCHOOL OF SCIENCE & TECHNOLOGY

Getting all keys or values

for loops

In the same way we used **for** loops to iterate over elements of a list, we can use them to iterate over elements of a dictionary.

The difference is that, with dictionaries, the **iteration variable** will represent the **current key**, not the **current value**.

SCHOOL OF SCIENCE & TECHNOLOGY

for loops

```
band = {
  "johnny": "plays drums",
  "joey": "plays guitar",
  "markee": "sings",
  "dee-dee": "plays bass-guitar"
}

for member in band:
  print(member + " " + band[member] + " in The Ramones")
```

SCHOOL OF SCIENCE & TECHNOLOGY