Programming Thinking

Session 6

Pepe García

SCHOOL OF SCIENCE & TECHNOLOGY

Plan for this session

- Learn about dictionaries

SCHOOL OF SCIENCE & TECHNOLOGY

Dictionaries

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

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

Adding elements

We add elements to dictionaries given their specific index:

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

Demo

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

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

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

Demo

SCHOOL OF SCIENCE & TECHNOLOGY

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

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

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

Demo

SCHOOL OF SCIENCE & TECHNOLOGY

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()
```

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()
```

Demo

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

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")
```

Demo

SCHOOL OF SCIENCE & TECHNOLOGY

Exercises

1. Create a function that receives a text and returns the frequency of each word in the text (as a dictionary).

(cont)

SCHOOL OF SCIENCE & TECHNOLOGY

Exercises

2. Create a function that uses the previously generated dictionary and prints a bars diagram of the frequencies. For example, the following:

```
dictionary = {
  "a": 4,
  "hello": 1,
  "world": 3
}
diagram(dictionary)
```

should print:

```
  a       | ****
  hello   | *
  world   | ***
```

SCHOOL OF SCIENCE & TECHNOLOGY