Statistical Programming with Python

Pepe García

2020-04-20

Dictionaries & Sets

Plan for this session

- Learn about dictionaries
- Learn about sets

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

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"
```

Adding elements

Updating elements

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

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

Updating elements

Deleting elements

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

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

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()
```

Getting all keys or values

Sets

Sets are collections of elements with no duplicated entries, and no specific order.

The sets below are equal:

```
a = { 1, 2, 3 }
b = { 3, 2, 1 }
c = { 1, 3, 2, 3, 3, 2}
```

Creating sets

We can use curly brackets ({}) to create sets, separating their items with commas.

The **set** function lets us convert a list into a set.

```
a = { 1, 2, 3 }
b = set([1, 3, 2, 3, 3, 2])
```

Adding & removing

```
a = { 1, 2, 3 }

a.add(4)     # Add item 4
a.discard(2) # Remove item 2
a.remove(1)  # Remove item 1

# Discarding a non-present item does nothing
a.discard(888)

# Removing a non-present item causes an error
a.remove(888)
```

Operations on Sets

Operation	Description
a | b, a.union(b)	Union of sets a and b
a & b, a.intersection(b)	Intersection of sets a and b
a - b, a.difference(b)	Difference of sets a and b
a ^ b, a.symmetric_difference(b)	Symmetric difference of sets a and b
a <= b, a.issubset(b)	Check if a is a subset of b
a >= b, a.issuperset(b)	Check if a is a superset of b

Working with sets

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 or a set.

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

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")
```

for loops

```
primes = { 2, 3, 5, 7, 11 }

for number in primes:
  print(number)
```

for loops

Exercises

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

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, "another": 2}
   diagram(dictionary)
   ```

   should print:

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