UNIVERSIDAD DE MÁLAGA

FGUMA FUNDACIÓN GENERAL UNIVERSIDAD DE MÁLAGA

CURSOS ONLINE

FUNDACIÓN GENERAL UNIVERSIDAD DE MÁLAGA

Diccionarios en Python

Unidad 9

Python para principiantes

dgm@uma.es

¿Qué es una colección?

- Usar colecciones es interesante, porque podemos tener muchos valores en un solo "paquete".
- Hacemos esto teniendo más de un lugar “en” la variable
- Python proporciona formas de acceder a los diferentes lugares en la variable

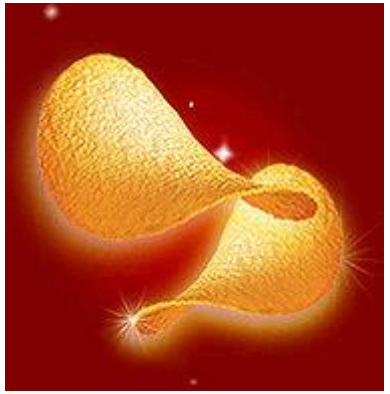
¿Qué no es una "Colección?

La mayor parte de nuestras variables tienen un único valor en ellas – cuando ponemos un nuevo valor en la variable, el antiguo valor es sobreescrito

```
$ python
>>> x = 2
>>> x = 4
>>> print(x)
4
```

Una historia de dos colecciones…

- **Lista**
  - Una colección lineal de valores que permanecen en orden



- **Diccionario**
  - Una “bolsa” de valores, cada uno con su propia etiqueta

Diccionarios

http://en.wikipedia.org/wiki/Associative_array

Diccionarios

- Los diccionarios son las colecciones de datos **más poderosas** de Python
- Los diccionarios nos permiten hacer **operaciones rápidas** similares a las de bases de datos en Python
- Los diccionarios tienen nombres diferentes en otros lenguajes
  - Arrays asociativos - Perl / PHP
  - Properties o Map o HashMap - Java
  - Propiedad Bag - C# / .NET

Diccionarios

- Las listas permiten acceder a sus elementos basándose en la posición en la lista
- Los **diccionarios** son como bolsas – **no hay orden**
- Así que accedemos a lo que ponemos en el diccionario a través de una "**etiqueta de búsqueda**"

```
>>> bolso = dict()
>>> bolso['dinero'] = 12
>>> bolso['caramelos'] = 3
>>> bolso['pañuelos'] = 75
>>> print(bolso)
{'dinero': 12, 'pañuelos': 75, 'caramelos': 3}
>>> print(bolso['caramelos'])
3
>>> bolso['caramelos'] = bolso['caramelos'] + 2
>>> print(bolso)
{'dinero': 12, 'pañuelos': 75, 'caramelos': 5}
```

Comparando listas y diccionarios

Los diccionarios son como listas excepto que usan **claves** (keys) en lugar de números para acceder a valores

```
>>> lst = list()
>>> lst.append(21)
>>> lst.append(183)
>>> print(lst)
[21, 183]
>>> lst[0] = 23
>>> print(lst)
[23, 183]
```

```
>>> ddd = dict()
>>> ddd['age'] = 21
>>> ddd['course'] = 182
>>> print(ddd)
{'course': 182, 'age': 21}
>>> ddd['age'] = 23
>>> print(ddd)
{'course': 182, 'age': 23}
```

```
>>> lst = list()
>>> lst.append(21)
>>> lst.append(183)
>>> print(lst)
[21, 183]
>>> lst[0] = 23
>>> print(lst)
[23, 183]
```

Lista

Clave	Valor
[0]	21
[1]	183

lst

```
>>> ddd = dict()
>>> ddd['age'] = 21
>>> ddd['course'] = 182
>>> print(ddd)
{'course': 182, 'age': 21}
>>> ddd['age'] = 23
>>> print(ddd)
{'course': 182, 'age': 23}
```

Diccionario

Clave	Valor
['course']	182
['age']	21

ddd

Literales Diccionarios (Constantes)

- Los literales de diccionarios utilizan **llaves { }** y tienen una lista de pares clave: valor

- Se puede hacer un **diccionario vacío** simplemente con { }

```
>>> jjj = { 'chuck' : 1 , 'fred' : 42, 'jan': 100}
>>> print(jjj)
{'jan': 100, 'chuck': 1, 'fred': 42}
>>> ooo = { }
>>> print(ooo)
{}
>>>
```

¿El nombre más común?

¿El nombre más común?

marquard

zhen

csev

zhen

cwen

marquard

zhen

csev

cwen

zhen

csev

marquard

zhen

¿El nombre más común?

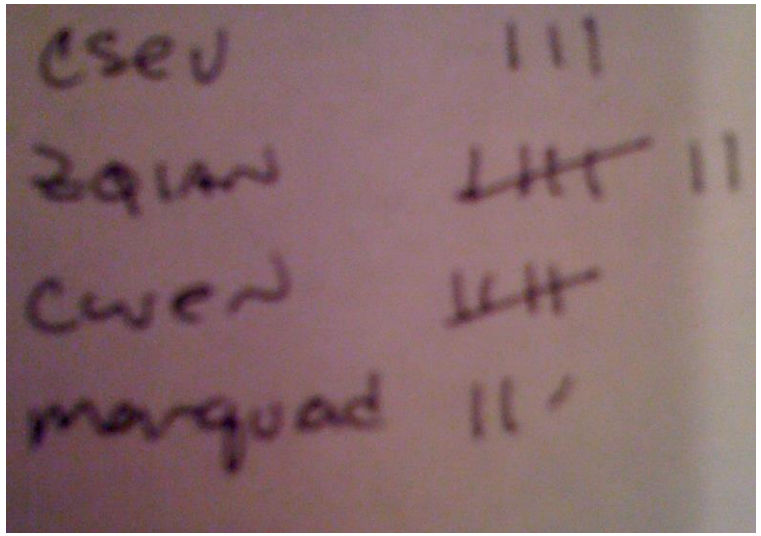
marquard

zhen

csev

zhen

cwen



csev

cwen

zhen

csev

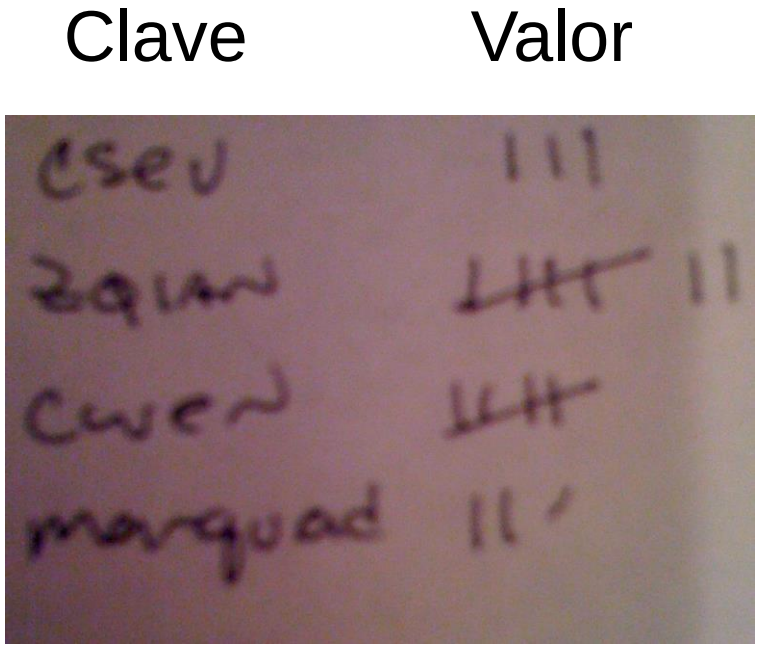
marquard

zhen

Muchos contadores con un diccionario

Un uso común de los diccionarios es **contar** cuántas veces “vemos” algo

```
>>> ccc = dict()
>>> ccc['csev'] = 1
>>> ccc['cwen'] = 1
>>> print(ccc)
{'csev': 1, 'cwen': 1}
>>> ccc['cwen'] = ccc['cwen'] + 1
>>> print(ccc)
{'csev': 1, 'cwen': 2}
```



Errores con diccionarios

Es un **error** hacer referencia a una clave **que no está** en el diccionario

Podemos usar el operador **in** para ver si una clave está en el diccionario

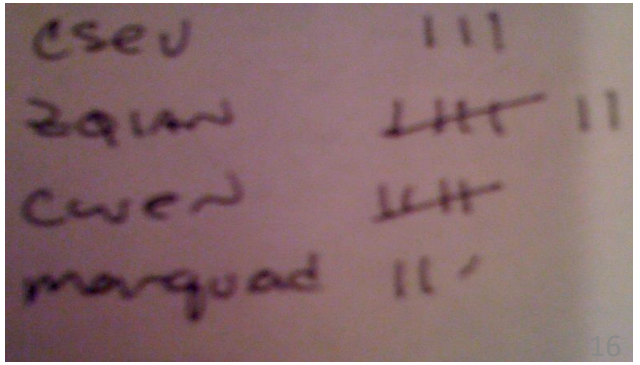
```
>>> ccc = dict()
>>> print(ccc['csev'])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'csev'
>>> 'csev' in ccc
False
```

Cuando vemos un nuevo nombre

Cuando encontramos un nuevo nombre, tenemos que sumar una nueva entrada al diccionario, y si es la segunda o posteriores veces que vemos el nombre, simplemente sumanos uno a la cuenta en el diccionario con ese nombre

```
counts = dict()
names = ['csev', 'cwen', 'csev', 'zqian', 'cwen']
for name in names :
    if name not in counts:
       counts[name] = 1
    else :
        counts[name] = counts[name] + 1
print(counts)
```

{'csev': 2, 'zqian': 1, 'cwen': 2}



El método get para diccionarios

El patrón para comprobar si un valor **está actualmente** en un diccionario y asumir un **valor por defecto** si no lo está, es tan común, que el método **get()** lo hace por nosotros

```
if name in counts:
    x = counts[name]
else :
    x = 0
```

```
x = counts.get(name, 0)
```

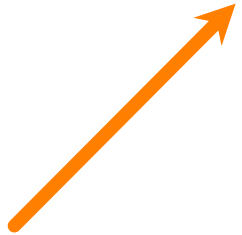
Valor por defecto si la clave no existe (sin error)

{'csev': 2, 'zqian': 1, 'cwen': 2}

Cuenta simplificada con get()

Podemos usar get() y proporcionar un valor por defecto de 0 cuando la clave no está todavía en el diccionario – y entonces sumar solo 1

```
counts = dict()
names = ['csev', 'cwen', 'csev', 'zqian', 'cwen']
for name in names :
    counts[name] = counts.get(name, 0) + 1
print(counts)
```



Valor por defecto

{'csev': 2, 'zqian': 1, 'cwen': 2}

Cuenta simplificada con get()

```
counts = dict()
names = ['csev', 'cwen', 'csev', 'zqian', 'cwen']
for name in names :
    counts[name] = counts.get(name, 0) + 1
print(counts)
```

https://www.youtube.com/watch?v=2AoxCkySv34

Contando palabras en texto

Writing programs (or programming) is a very creative and rewarding activity. You can write programs for many reasons ranging from making your living to solving a difficult data analysis problem to having fun to helping someone else solve a problem. This book assumes that everyone needs to know how to program and that once you know how to program, you will figure out what you want to do with your newfound skills.

We are surrounded in our daily lives with computers ranging from laptops to cell phones. We can think of these computers as our “personal assistants” who can take care of many things on our behalf. The hardware in our current-day computers is essentially built to continuously ask us the question, “What would you like me to do next?”

Our computers are fast and have vast amounts of memory and could be very helpful to us if we only knew the language to speak to explain to the computer what we would like it to do next. If we knew this language we could tell the computer to do tasks on our behalf that were repetitive. Interestingly, the kinds of things computers can do best are often the kinds of things that we humans find boring and mind-numbing.

Patrón de cuenta

```
counts = dict()
print('Escribe una línea:')
line = input('')

words = line.split()

print('Palabras:', words)

print('Contando...')
for word in words:
    counts[word] = counts.get(word,0) + 1
print('Cuentas', counts)
```

El patrón general para contar las palabras en una línea de texto es dividir la línea en palabras (**split**), y entonces iterar a través de las palabras usando un **diccionario** para llevar la cuenta de cada palabra de manera independiente.

```
python wordcount.py
Escribe una línea:
the clown ran after the car and the car ran into the tent
and the tent fell down on the clown and the car

Palabras: ['the', 'clown', 'ran', 'after', 'the', 'car',
'and', 'the', 'car', 'ran', 'into', 'the', 'tent', 'and',
'the', 'tent', 'fell', 'down', 'on', 'the', 'clown',
'and', 'the', 'car']
Contando…

Cuentas {'and': 3, 'on': 1, 'ran': 2, 'car': 3, 'into': 1,
'after': 1, 'clown': 2, 'down': 1, 'fell': 1, 'the': 7,
'tent': 2}
```

http://www.flickr.com/photos/71502646@N00/2526007974/

```
counts = dict()
line = input('Escribe una línea:')
words = line.split()

print('Palabras:', words)
print('Contando...’)

for word in words:
    counts[word] = counts.get(word,0) + 1
print('Cuentas', counts)
```

```
python wordcount.py
```

Escribe una línea:
the clown ran after the car and the car ran into the tent and the tent fell down on the clown and the car

Palabras: ['the', 'clown', 'ran', 'after', 'the', 'car', 'and', 'the', 'car', 'ran', 'into', 'the', 'tent', 'and', 'the', 'tent', 'fell', 'down', 'on', 'the', 'clown', 'and', 'the', 'car']
Counting...

Cuentas {'and': 3, 'on': 1, 'ran': 2, 'car': 3, 'into': 1, 'after': 1, 'clown': 2, 'down': 1, 'fell': 1, 'the': 7, 'tent': 2}

Bucles for y diccionarios

Aunque los diccionarios no están almacenados en orden, podemos escribir un **bucle for** que itera por todas las entradas del diccionario – realmente **las claves**, y entonces mirar los **valores**

```
>>> counts = { 'chuck' : 1 , 'fred' : 42, 'jan': 100}
>>> for key in counts:
...     print(key, counts[key])
...
jan 100
chuck 1
fred 42
>>>
```

Recuperando listas de claves y valores

Se puede recuperar una lista de claves, valores o ítems (clave+valor) de un diccionario

```
>>> jjj = { 'chuck' : 1 , 'fred' : 42, 'jan': 100}
>>> print(list(jjj))
['jan', 'chuck', 'fred']
>>> print(jjj.keys())
['jan', 'chuck', 'fred']
>>> print(jjj.values())
[100, 1, 42]
>>> print(jjj.items())
[('jan', 100), ('chuck', 1), ('fred', 42)]
>>>
```

¿Qué es una “tupla”? - Próximamente...

Extra: ¡Dos variables de iteración!

- Podemos iterar a través de las **parejas clave-valor** de un diccionario usando **dos variables de iteración**

- En cada iteración, la primera variable es la clave y la segunda, el valor correspondiente a esa clave

```
jjj = { 'chuck' : 1 , 'fred' : 42, 'jan': 100}
for aaa,bbb in jjj.items() :
    print(aaa, bbb)
```

```
jan 100
chuck 1
fred 42
```

aaa	bbb
[jan]	100
[chuck]	1
[fred]	42

```
name = input('Nombre de fichero:')
handle = open(name)

counts = dict()
for line in handle:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word,0) + 1

bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

```
python words.py
Nombre de fichero: words.txt
to 16
```

```
python words.py
Nombre de fichero: clown.txt
the 7
```

Usando dos bucles anidados

Resumen

- ¿Qué es una colección?
- Listas versus diccionarios
- Constantes de diccionarios
- La palabra más común
- Usando el método get()
- Búsqueda y falta de orden
- Escribiendo bucles con diccionarios
- Anticipo: tuplas
- Ordenando diccionarios

Ejercicio

Escribe un programa que lea el archivo mbox-short.txt e indique quién ha enviado el mayor número de e-mails junto con el número de e-mails. El programa busca líneas con ‘From ‘ y toma la segunda palabra de estas líneas como la persona que envió el e-mail.

Debes crear un diccionario Python que mapea el nombre del emisor con el número de veces que aparece en el archivo. Después de generar el diccionario, el programa itera a través del diccionario para ver quién ha sido el emisor con más mensajes.

Acknowledgements / Contributions

Initial Development: Charles Severance, University of Michigan School of Information

… Insert new Contributors and Translators here
Spanish Version: Daniel Garrido (dgm@uma.es)

Continue…