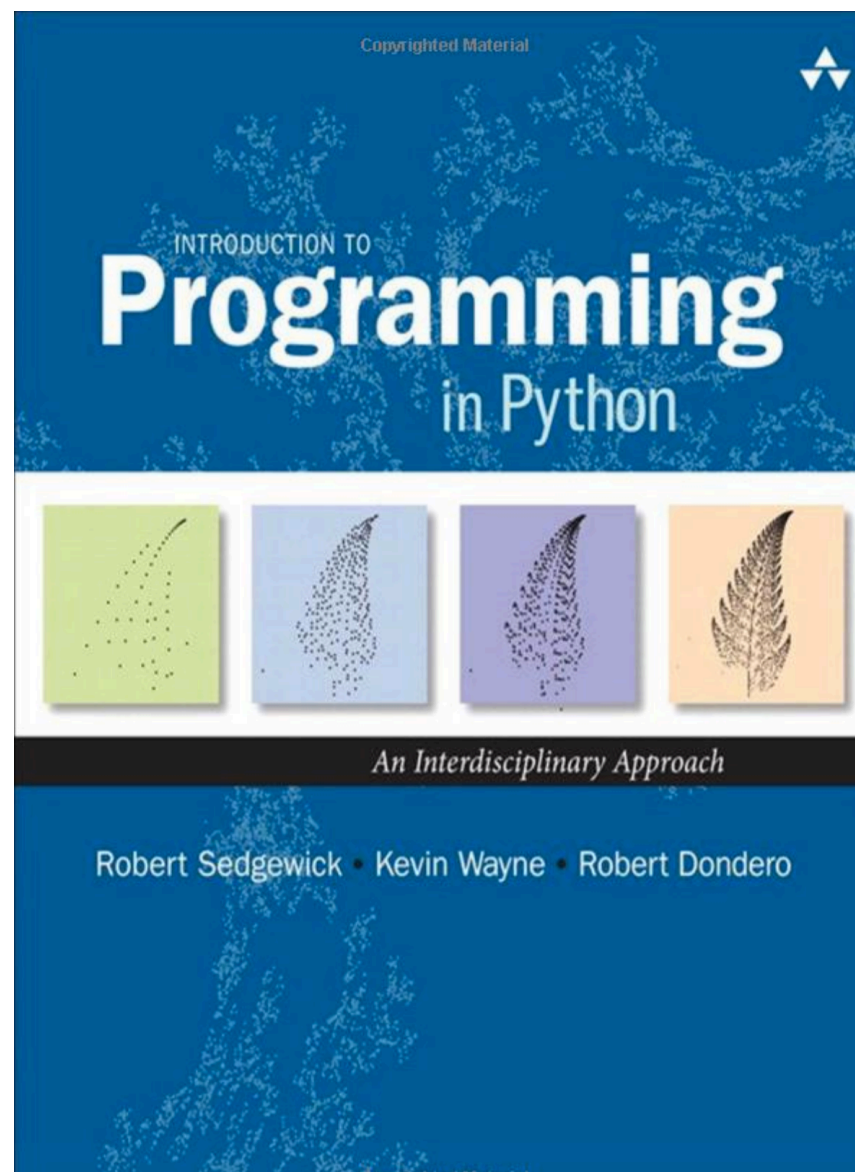


# Taller de Programación

## Clase 08: Tuplas, sets y diccionarios

Daniela Opitz, Diego Caro  
[dopitz@udd.cl](mailto:dopitz@udd.cl)



Basada en presentaciones oficiales de libro Introduction to Programming in Python (Sedgewick, Wayne, Dondero).

Disponible en <https://introcs.cs.princeton.edu/python>

# Outline

- Listas anidadas
- Tuplas
- Sets
- Diccionarios

# Listas Anidadas

- Una lista anidada es una lista que aparece como un elemento en otra lista
- Son útiles para representar matrices
- Una matriz  $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$  por ejemplo en python quedaría así:

```
[ [a11, a12, a13], [a21, a22, a23], [a31, a32, a33] ]
```

# Listas Anidadas

```
1 from random import random
2
3 rows = 3
4 cols = 4
5 matrix = []
6 for i in range(rows):
7     matrix.append([0]*cols)
8
9 print(matrix)
```

Número de filas

Número de columnas

```
1 def fillrandom(matrix, rows, cols):
2     for i in range(rows):
3         for j in range(cols):
4             matrix[i][j] = random()
5
6 fillrandom(matrix, rows, cols)
7 print(matrix)
```

## Matriz de ceros

Número de filas

matrix[0] → [0, 0, 0, 0]  
matrix[1] → [0, 0, 0, 0]  
matrix[2] → [0, 0, 0, 0]

Número de columnas

# Listas Anidadas

- Cómo llamo a cada elemento de la lista anidada en python?

<code>matrix[0][0]</code>	<code>matrix[0][1]</code>	<code>matrix[0][2]</code>	<code>matrix[0][3]</code>
<code>matrix[1][0]</code>	<code>matrix[1][1]</code>	<code>matrix[1][2]</code>	<code>matrix[1][3]</code>
<code>matrix[2][0]</code>	<code>matrix[2][1]</code>	<code>matrix[2][2]</code>	<code>matrix[2][3]</code>

# tuple()

- Una tupla es una secuencia de valores agrupados. Sirve para agrupar en un único valor, valores que deben ir juntos.
- El tipo de datos que representa una tupla se llama **tuple()**.
- Las tuplas se crean con valores separados por comas y entre paréntesis (elem1, elem2)
- La estructura de datos **tuple()** es parecida a estructura de lista pero a diferencia de esta, **tuple()** es **INMUTABLE**.

# set()

- Un set es una colección **no ordenada ni indexada** de elementos.
- Los elementos son **únicos** y no se pueden repetir.
- El tipo de datos que representa un set es **set()**
- Los sets pueden crear con el operador corchete {elem1, elem2}
- Los sets son **mutables** pero **no pueden contener elementos mutables**
- Sirven para:
  - chequear si existe un elemento
  - realizar operaciones de unión, intersección, diferencia entre conjuntos
  - eliminar duplicados de una secuencia

# Tuplas y sets



```
1 A = {[1, 2, 3, 4], list(['a', 'b', 'c'])}
2 print('A:', A)
3 print('(1, 2, 3, 4) in A?:', (1, 2, 3, 4) in A)
```

```
$ python3 badset.py
Traceback (most recent call last):
  File "badset.py", line 1, in <module>
    A = {[1, 2, 3, 4], ['a', 'b', 'c']}
TypeError: unhashable type: 'list'
```



```
1 A = {(1, 2, 3, 4), tuple(['a', 'b', 'c'])}
2 print('A:', A)
3 print('(1, 2, 3, 4) in A?:', (1, 2, 3, 4) in A)
```

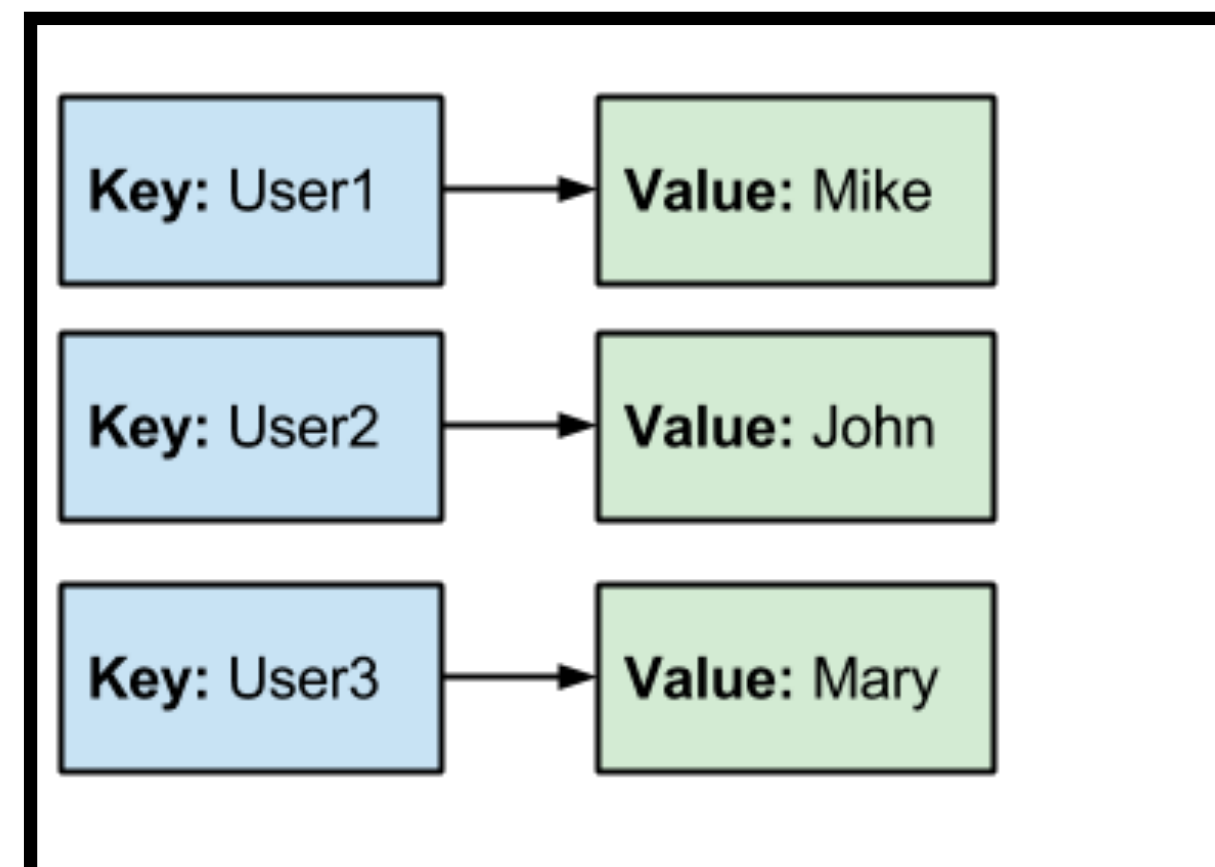
```
$ python3 tupleset.py
A: {(1, 2, 3, 4), ('a', 'b', 'c')}
(1, 2, 3, 4) in A?: True
```



# dict()

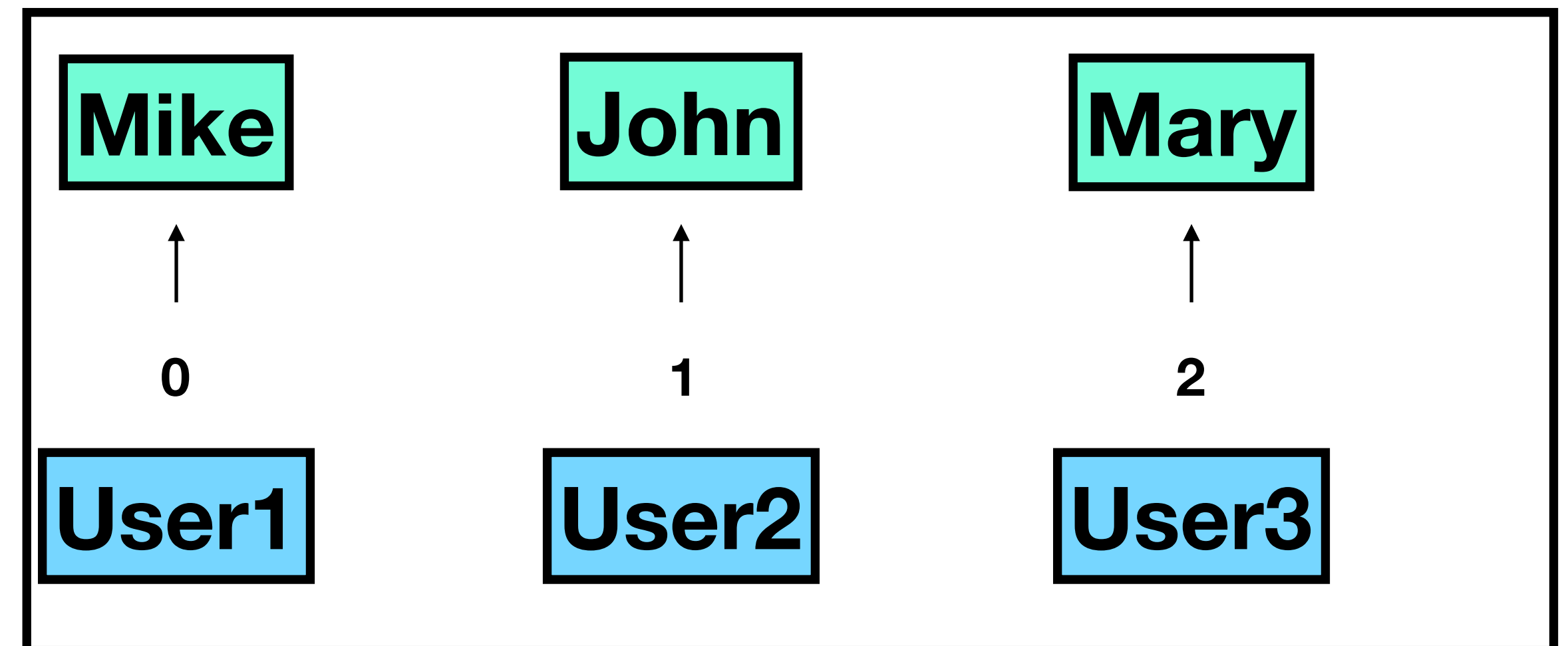
- Un diccionario es una colección de datos que no esta ordenada pero está indexada
- El tipo de datos de los diccionarios es **dict()**
- Un diccionario asocia un valor a una clave (key)
- La clave debe ser inmutable (de tipo int(), str(), tuple())

Diccionario



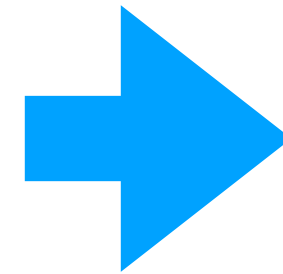
Lista o Tupla 2

Lista o Tupla 1



# dict()

Dominio	Dirección IP
udd.cl	201.221.123.142
ingenieria.udd.cl	201.221.123.142
google.com	64.233.190.101
pokemongo.com	13.33.131.6
<b>llave (key)</b>	<b>valor</b>



```
1 ipaddress = dict()
2 ipaddress['udd.cl'] = '201.221.123.142'
3 ipaddress['ingenieria.udd.cl'] = '201.221.123.142'
4 ipaddress['google.com'] = '64.233.190.101'
5 ipaddress['pokemongo.com'] = '13.33.131.6'
6
7 print('Dirección IP de udd.cl:', ipaddress['udd.cl'])
```

```
1 ipaddress = {'udd.cl': '201.221.123.142',
2              'google.com': '64.233.190.101'}
3 ipaddress['ingenieria.udd.cl'] = '201.221.123.142'
4 ipaddress['pokemongo.com'] = '13.33.131.6'
5
6 print('Dirección IP de udd.cl:', ipaddress['udd.cl'])
```

# Ejemplo: ¿Cómo almacenar las notas de un curso? .list()

- Podríamos usar una lista para nombre de alumno, notas y el curso:

```
nombres = ['Diego', 'Francisca', 'Loreto', 'Leo']  
notas = [4.1, 5.5, 6.8, 3.9]
```

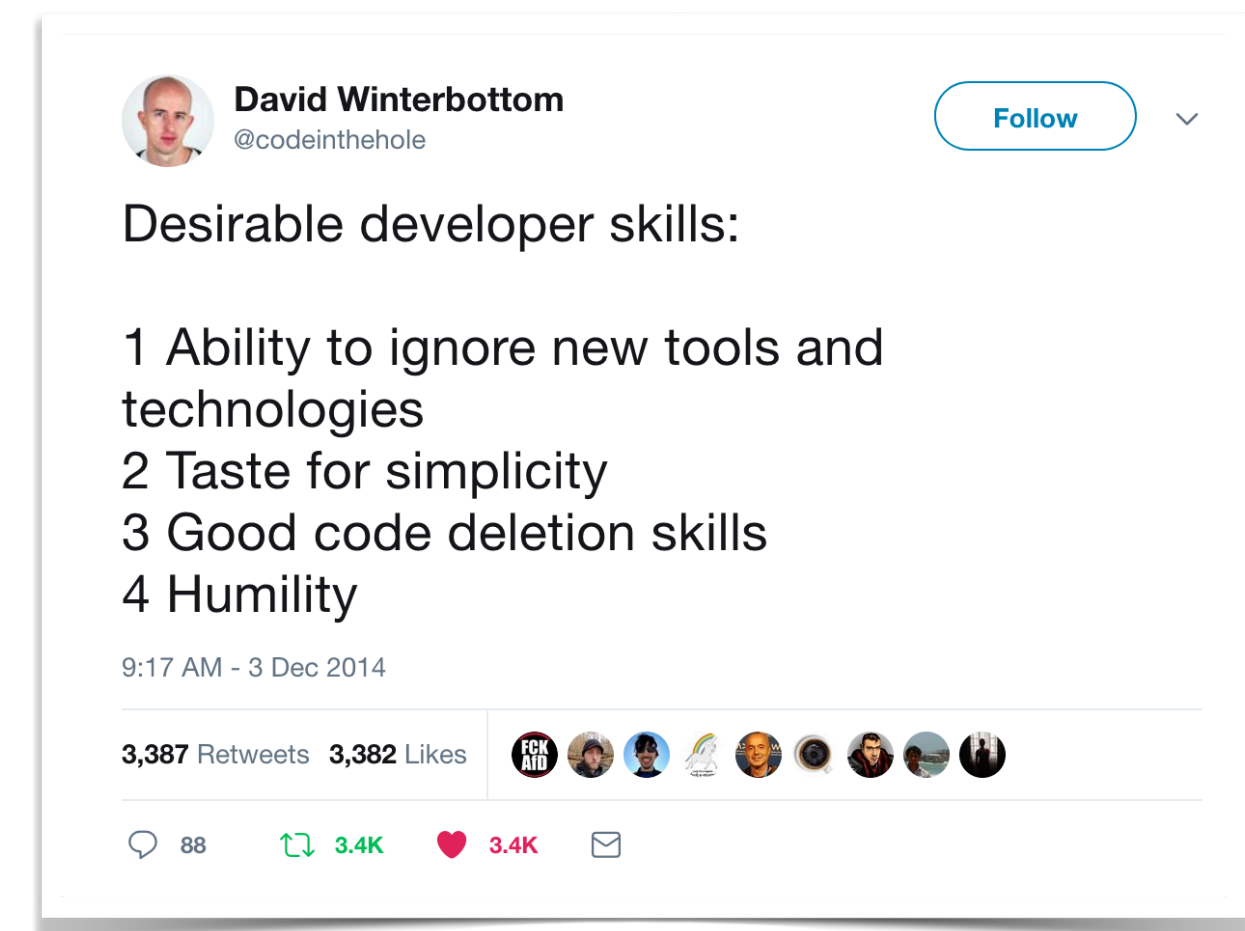
- Cada lista contiene información distinta.
- Las listas deben ser del **mismo tamaño**.
- Información entre las listas deben estar en la misma **posición**.

# Ejemplo: ¿Cómo almacenar las notas de un curso? .list()

## Nota de Diego?

```
def obtener_notas(estudiante, lista_nombres, lista_notas):  
    i = lista_nombres.index(estudiante)  
    nota = lista_notas[i]  
  
    return nota  
  
nota = obtener_notas('Diego', nombres, notas)  
  
print('Diego tiene un', nota)
```

- Complicado si tienes varios tipos de información que almacenar
- Debes mantener **varias listas**, y pasarlas como argumento
- Necesita de un **índice** (un **entero** con la posición)
- **DIFICIL DE MANTENER!**



# Ejemplo: ¿Cómo almacenar las notas de un curso?.dict()

**Diccionario** →

```
1 notas = {'Diego'      : 4.1
2          , 'Francisca': 5.5
3          , 'Daniela'  : 6.8
4          , 'Leo'      : 3.9}
5
6 def obtener_notas(estudiante, dict_notas):
7     nota = dict_notas[estudiante]
8     return nota
9
10 nota = obtener_notas('Diego', notas)
11 print('Diego tiene un', nota)
```

Clave ↓

Valor ↓

Int →

Int →

# Patrón típico de uso diccionario

- Recorrer todas las llaves del diccionario:

```
for llave in notas.keys():  
    print(llave)
```

- Recorrer los valores del diccionario:

```
for valor in notas.values():  
    print(valor)
```

```
d = {"Python": 1991,  
     "C": 1972,  
     "Java": 1996}
```

- Recorrer todas las llaves y valores:

```
for llave, valor in notas.items():  
    print(llave, valor)
```

# Ejemplo: Contando Palabras

```
1 def word_count(message):
2     counts = dict()
3     words = message.split()
4
5     for word in words:
6         if word in counts:
7             counts[word] += 1
8         else:
9             counts[word] = 1
10
11     return counts
12
13
14 mambo = '''Desde lima vengo a mi machaguay
15 Desde Lima vengo a mi machaguay
16 A bailar el mambo de mi machaguay
17 A bailar el mambo de mi machaguay'''
18
19 print(word_count(mambo))
```



```
{'A': 2,
 'Desde': 2,
 'Lima': 1,
 'a': 2,
 'bailar': 2,
 'de': 2,
 'el': 2,
 'lima': 1,
 'machaguay': 4,
 'mambo': 2,
 'mi': 4,
 'vengo': 2}
```

# Actividad

El siguiente diccionario contiene información relacionada a la pobreza regional de nuestro país. Las llaves (keys) corresponden a los nombres las regiones de Chile, y los valores (values) corresponden a la información del porcentaje de pobreza de ingreso en la respectiva región (Fuente: Casen 2015). Imprima el nombre la región con mayor población en situación de pobreza de ingresos y su respectivo porcentaje.

```
pobreza_reg={'Tarapaca' : 7.1, 'Antofagasta' : 5.4, 'Atacama' : 6.9, 'Coquimbo' :  
13.8, 'Valparaíso' : 12.0, 'Libertador Bernardo OHiggins' : 13.7, 'Maule' : 18.7,  
'Biobío' : 17.6, 'La Araucanía' : 23.6, 'Los Lagos' : 16.1, 'Aysen' : 6.5,  
'Magallanes y La Antártica Chilena' : 4.4, 'Región Metropolitana de Santiago' : 7.1,  
'Los Ríos' : 16.8, 'Arica y Parinacota' : 9.7}
```



# Resumen

## Estructuras de datos

- `tuple()`: colección inmutable y ordenada
- `set()`: conjunto de elementos, no-ordenada
- `dict()`: tabla de símbolos clave:valor

## Conceptos

- **Inmutable**: que no se puede modificar una vez creada.

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

[https://docs.python.org/3/reference/lexical\\_analysis.html](https://docs.python.org/3/reference/lexical_analysis.html)

		Built-in Functions		
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

<https://docs.python.org/3/library/functions.html>