

# Introducción a la programación con Python (1)

# Historia

- Implementado a inicio de los 90 en Holanda.
- Version 2.0 en 2000: garbage collector y soporte Unicode.
- Version 3.0 en 2008.
- Versiones 2.6 y 2.7 contienen las características principales de version 3.

# Características

- Gestor de memoria (garbage collector)
- Tipos dinámicos
- Soporte multi-paradigmas
- Diseñado para ser extendible: core pequeño, muchas librerías
- Sintaxis cercana a lenguaje natural
- Bloques definidos por indentación

# Características

- Lenguaje interpretado
- Multiplataforma
- Modos de trabajo:
  - Interactivo
  - Ejecución del script
- Fuertemente tipado

# Características

```
In [1]: a = "10"
```

```
In [2]: b = 5
```

```
In [3]: c = a + b
```

```
-----  
TypeError                                 Traceback (most recent call last)  
/home/jeudy/<ipython-input-3-60f555c9e9aa> in <module>()  
----> 1 c = a + b
```

```
TypeError: cannot concatenate 'str' and 'int' objects
```

```
In [4]: c = b + a
```

```
-----  
TypeError                                 Traceback (most recent call last)  
/home/jeudy/<ipython-input-4-c895e5260a6e> in <module>()  
----> 1 c = b + a
```

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

# Implementaciones

- Cpython (default en Linux)
- Jython: Java
- IronPython: C# (.Net)

# Python interactivo

- ipython (sudo apt-get install ipython si no lo tienen).
- Consola interactiva de python. Funciones de ayuda y autocompletar.

```
Terminal
File Edit View Search Terminal Help
jeudy@machine:~$ ipython
Python 2.7.3 (default, Aug  1 2012, 05:14:39)
Type "copyright", "credits" or "license" for more information.

IPython 0.12.1 -- An enhanced Interactive Python.
?                -> Introduction and overview of IPython's features.
%quickref        -> Quick reference.
help             -> Python's own help system.
object?         -> Details about 'object', use 'object??' for extra details.

In [1]: print "Ola ke ase"
Ola ke ase

In [2]: █
```

## Terminal

File Edit View Search Terminal Help

```
In [10]: a = 10
```

```
In [11]: a?
```

```
Type:      int
```

```
Base Class: <type 'int'>
```

```
String Form:10
```

```
Namespace: Interactive
```

```
Docstring:
```

```
int(x[, base]) -> integer
```

Convert a string or number to an integer, if possible. A floating point argument will be truncated towards zero (this does not include a string representation of a floating point number!) When converting a string, use the optional base. It is an error to supply a base when converting a non-string. If base is zero, the proper base is guessed based on the string content. If the argument is outside the integer range a long object will be returned instead.

```
In [12]: █
```



# Variables - Tipos básicos

- Números:
  - Enteros (Long)
  - Reales (punto flotante)
  - Complejos

```
Terminal
File Edit View Search Terminal Help

In [20]: a = 10

In [21]: b = 3.14

In [22]: c = 7 + 5j

In [23]: print type(a), type(b), type(c)
<type 'int'> <type 'float'> <type 'complex'>

In [24]:
```

# Precisión y notación científica

[illegible]

```
Terminal
File Edit View Search Terminal Help

In [41]: cantidad_cientifica = 1.74E-4

In [42]: print cantidad_cientifica
0.000174

In [43]: cantidad_cientifica = 3.5E+5

In [44]: print cantidad_cientifica
350000.0

In [45]: type(cantidad_cientifica)
Out[45]: float

In [46]:
```

# Operadores matemáticos

```
Terminal
File Edit View Search Terminal Help

In [25]: print 3 + 4
7

In [26]: print 3 - 4
-1

In [27]: print 3 * 4
12

In [28]: print 3 / 4
0

In [29]: print 3.0 / 4.0
0.75

In [30]: print 3 % 2
1

In [31]: print 2 ** 3
8
```

# Asignación

```
Terminal
File Edit View Search Terminal Help

In [47]: a,b,c = 10, "hola mundo", 3.14E-5

In [48]: print a, b, c
10 hola mundo 3.14e-05

In [49]: print type(a), type(b), type(c)
<type 'int'> <type 'str'> <type 'float'>

In [50]:
```

```
Terminal
File Edit View Search Terminal Help

In [33]: x = 10 #Operador de asignación

In [34]: x += 1 #Equivalente a x = x + 1

In [35]: print x
11

In [36]: x *= 10

In [37]: print x
110

In [38]: x -= 10

In [39]: print x
100

In [40]: x /= 4

In [41]: print x
25
```

# Variables - Tipos básicos

- Cadenas de texto

```
Terminal
File Edit View Search Terminal Help

In [57]: cadena = "hola"

In [58]: cadena2 = cadena + " mundo"

In [59]: print cadena
hola

In [60]: print cadena2
hola mundo

In [61]: cadena += " ke ase"

In [62]: print cadena
hola ke ase

In [63]: █
```

```
Terminal
File Edit View Search Terminal Help

In [70]: otra_cadena = "uno"

In [71]: cadena_magica = otra_cadena * 3

In [72]: print cadena_magica
unounouno

In [73]: █
```

# Insertar valores en cadenas

## Terminal

File Edit View Search Terminal Help

```
In [91]: cadena = "HOLA"
```

```
In [92]: valor = 1000
```

```
In [93]: cadena_compleja = "%s contengo al valor %d"%(cadena, valor)
```



```
In [94]: print cadena_compleja  
HOLA contengo al valor 1000
```

# Reemplazando

```
In [7]: str_paises = "CR US MX AR ES PE IT GR FR"
```

```
In [8]: str_paises.replace(" ", ",")
```

```
Out[8]: 'CR,US,MX,AR,ES,PE,IT,GR,FR'
```

```
In [9]: str_paises
```

```
Out[9]: 'CR US MX AR ES PE IT GR FR'
```

```
In [10]: str_paises_csv = str_paises.replace(" ", ",")
```

```
In [11]: str_paises_csv
```

```
Out[11]: 'CR,US,MX,AR,ES,PE,IT,GR,FR'
```

# Cadenas: funciones varias

```
In [30]: str_paises
Out[30]: 'CR US MX AR ES PE IT GR FR'

In [31]: str_paises.title() #Capitaliza palabras dentro de cadena
Out[31]: 'Cr Us Mx Ar Es Pe It Gr Fr'

In [32]: str_paises.lower() #Convierte a minuscula
Out[32]: 'cr us mx ar es pe it gr fr'

In [33]: str_paises_cap = str_paises.title() #Captura resultado (no actua directo)

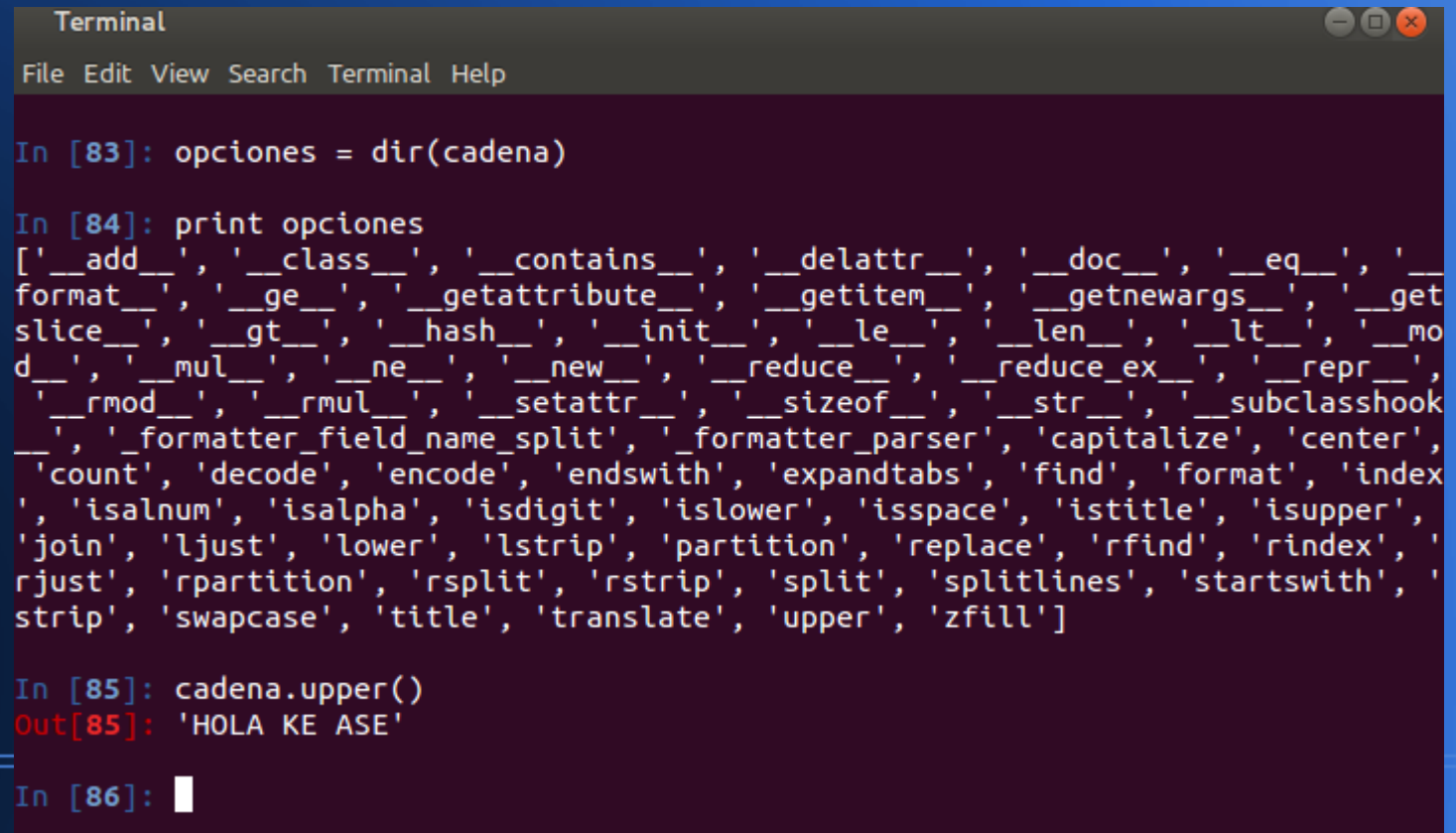
In [34]: str_paises_cap
Out[34]: 'Cr Us Mx Ar Es Pe It Gr Fr'

In [35]: str_paises_cap.swapcase()
Out[35]: 'cR uS mX aR eS pE iT gR fR'
```



# Ejercicio: funciones de cadenas

- En el shell de python, podemos usar la función `dir(<expresión>)` para ver su estructura y funciones.



```
Terminal
File Edit View Search Terminal Help

In [83]: opciones = dir(cadena)

In [84]: print opciones
['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__getslice__', '__gt__', '__hash__', '__init__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '_formatter_field_name_split', '_formatter_parser', 'capitalize', 'center', 'count', 'decode', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'index', 'isalnum', 'isalpha', 'isdigit', 'islower', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']

In [85]: cadena.upper()
Out[85]: 'HOLA KE ASE'

In [86]:
```

# Variables - Tipos básicos

- Lógicos

```
Terminal
File Edit View Search Terminal Help

In [96]: verdadero = True

In [97]: falso = False

In [98]: print type(verdadero), type(falso)
<type 'bool'> <type 'bool'>

In [99]: print verdadero, falso
True False

In [100]: █
```

# Operadores de comparación

```
Terminal
File Edit View Search Terminal Help

In [113]: valor1 = 1000

In [114]: print valor1 == 100 #El operador == compara valores
False

In [115]: valor2 = 1000

In [116]: print valor1 == valor2
True

In [117]: print valor1 != 34567 #Operador "No igual"
True

In [118]: resultado = 10 < valor1 #Se puede asignar a variable

In [119]: print resultado
True

In [120]: █
```

# Operadores lógicos

X	Y	And	Or
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

```
Terminal
File Edit View Search Terminal Help

In [121]: True and True
Out[121]: True

In [122]: True and False
Out[122]: False

In [123]: False and False
Out[123]: False

In [124]: True or True
Out[124]: True

In [125]: True or False
Out[125]: True

In [126]: False or False
Out[126]: False
```

# Operadores lógicos en expresiones complejas

```
Terminal
File Edit View Search Terminal Help

In [128]: valor = 1000

In [129]: resultado = (valor >= 100) and (valor != 1000)

In [130]: print resultado
False

In [131]: not resultado
Out[131]: True
```

# Estructuras de datos

- Listas
- Diccionarios
- Conjuntos
- Tuplas

# Listas

Terminal	Terminal	Terminal
<pre>File Edit View Search Terminal Help</pre>	<pre>File Edit View Search Terminal Help</pre>	<pre>File Edit View Search Terminal Help</pre>
<pre>In [2]: lista</pre>	<pre>In [15]: lista_inception = [1, "dos", [10,11,12]]</pre>	<pre>In [14]: lista_inception</pre>
<pre>In [3]: type(lista)</pre>	<pre>In [16]: len(lista_inception)</pre>	<pre>In [15]: lista_inception</pre>
<pre>Out[3]: list</pre>	<pre>Out[16]: 3</pre>	<pre>Out[14]: [1, 'dos', [10, 11, 12]]</pre>
<pre>In [4]: len(lista)</pre>		<pre>In [16]: len(lista_inception)</pre>
<pre>Out[4]: 0</pre>		<pre>Out[16]: 3</pre>
<pre>In [5]:</pre>	<pre>In [17]:</pre>	

# Listas – insertando elementos

## Terminal

File Edit View Search Terminal Help

```
In [41]: lista = []
```

```
In [42]: lista.append(10)
```

```
In [43]: lista.append(20)
```

```
In [44]: lista.append(30)
```

```
In [45]: len(lista)
```

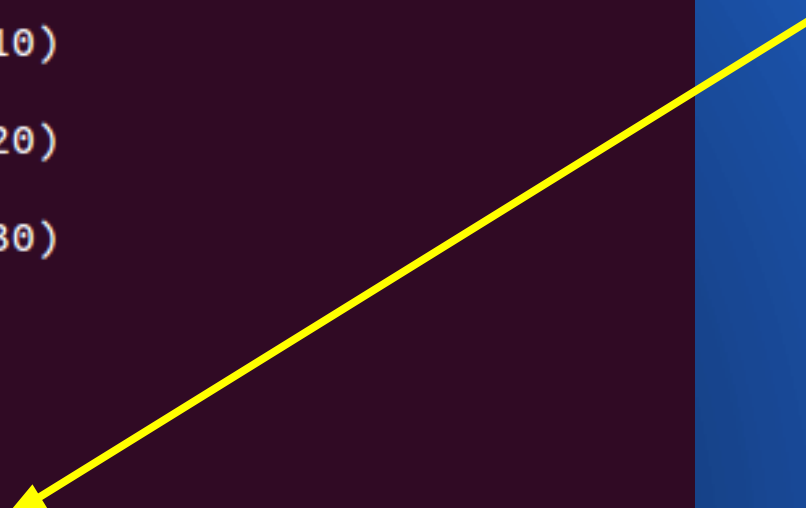
```
Out[45]: 3
```

```
In [46]: lista
```

```
Out[46]: [10, 20, 30]
```

```
In [47]: █
```

El append hace que la lista se comporte como una cola.





# Listas: accedendo elementos

```
Terminal
File Edit View Search Terminal Help

In [76]: lista[0] = 3.14 #Podemos editar cualquier elemento de la lista

In [77]: lista
Out[77]: [3.14, 20, 30, 100, 200, 300]

In [78]: lista[-1] #Podemos indexar a la lista de atrás para delante
Out[78]: 300

In [79]: lista[-2] #Penultimo elemento
Out[79]: 200

In [80]:
```

# Listas: accediendo elementos

```
Terminal
File Edit View Search Terminal Help

In [85]: lista
Out[85]: [3.14, 20, 30, 100, 200, 300]

In [86]: lista[0:2] #Podemos acceder a la lista por rango
Out[86]: [3.14, 20]
```

```
Terminal
File Edit View Search Terminal Help

In [96]: lista
Out[96]: [3.14, 20, 30, 100, 200, 300]

In [97]: lista[2:3]
Out[97]: [30]

In [98]: lista[2:2]
Out[98]: []

In [99]: lista[2:5]
Out[99]: [30, 100, 200]

In [100]: █
```

# Listas: accedendo elementos

```
Terminal
File Edit View Search Terminal Help

In [101]: lista
Out[101]: [3.14, 20, 30, 100, 200, 300]

In [102]: lista[:3] #Si omito el primer indice, toma el inicio
Out[102]: [3.14, 20, 30]

In [103]: lista[2:] #Si omito el ultimo indice, tomar el final
Out[103]: [30, 100, 200, 300]
```

```
Terminal
File Edit View Search Terminal Help

In [110]: lista
Out[110]: [3.14, 20, 30, 100, 200, 300]

In [111]: lista[2:4] = [-100, -300] #Se pueden usar rangos para reemplazar

In [112]: lista
Out[112]: [3.14, 20, -100, -300, 200, 300]

In [113]:
```

# Listas: funciones disponibles

```
Terminal
File Edit View Search Terminal Help

In [127]: lista.
lista.append    lista.extend    lista.insert    lista.remove    lista.sort
lista.count     lista.index     lista.pop       lista.reverse

In [127]: lista.pop?
Type:          builtin_function_or_method
Base Class:    <type 'builtin_function_or_method'>
String Form:   <built-in method pop of list object at 0x17d2830>
Namespace:     Interactive
Docstring:
L.pop([index]) -> item -- remove and return item at index (default last).
Raises IndexError if list is empty or index is out of range.

In [128]: lista.append?
Type:          builtin_function_or_method
Base Class:    <type 'builtin_function_or_method'>
String Form:   <built-in method append of list object at 0x17d2830>
Namespace:     Interactive
Docstring:     L.append(object) -- append object to end
```

# Funciones de listas

## Terminal

File Edit View Search Terminal Help

```
In [130]: lista_mixta
```

```
Out[130]: [False, 13, 13.74, 50, '50', 'altair', 'cincuenta']
```

```
In [131]: lista_mixta.insert(4, 13)
```

```
In [132]: lista_mixta
```

```
Out[132]: [False, 13, 13.74, 50, 13, '50', 'altair', 'cincuenta']
```

```
In [133]: lista_mixta.insert(0, 50)
```

```
In [134]: lista_mixta
```

```
Out[134]: [50, False, 13, 13.74, 50, 13, '50', 'altair', 'cincuenta']
```

# Funciones de listas – extender lista

## Terminal

File Edit View Search Terminal Help

```
In [136]: lista
```

```
Out[136]: [-300, -100, 3.14, 20, 200, 300]
```

```
In [137]: lista_mixta
```

```
Out[137]: [50, False, 13, 13.74, 50, 13, '50', 'altair', 'cincuenta']
```

```
In [138]: lista.extend(lista_mixta)
```

```
In [139]: lista
```

```
Out[139]:
```

```
[-300,  
 -100,  
 3.14,  
 20,  
 200,  
 300,  
 50,  
 False,  
 13,  
 13.74,  
 50,  
 13,  
 '50',  
 'altair',  
 'cincuenta']
```

# Funciones de listas: buscar

```
Terminal
File Edit View Search Terminal Help

In [153]: lista_mixta.index?
Type:      builtin_function_or_method
Base Class: <type 'builtin_function_or_method'>
String Form: <built-in method index of list object at 0x17bfc20>
Namespace: Interactive
Docstring:
L.index(value, [start, [stop]]) -> integer -- return first index of value.
Raises ValueError if the value is not present.

In [154]: lista_mixta
Out[154]: [50, False, 13, 13.74, 50, 13, '50', 'altair', 'cincuenta']

In [155]: lista_mixta.index(50)
Out[155]: 0

In [156]: lista_mixta.index('50')
Out[156]: 6
```

# Funciones de listas: contar

## Terminal

File Edit View Search Terminal Help

```
In [177]: lista_mixta
```

```
Out[177]: [50, False, 13, 13.74, 50, 13, '50', 'altair', 'cincuenta']
```

```
In [178]: lista_mixta.count(50)
```

```
Out[178]: 2
```

```
In [179]: lista_mixta.count('50')
```

```
Out[179]: 1
```

```
In [180]: █
```



# Funciones de listas: remove

```
Terminal
File Edit View Search Terminal Help

In [181]: lista_mixta.pop?
Type:      builtin_function_or_method
Base Class: <type 'builtin_function_or_method'>
String Form: <built-in method pop of list object at 0x19d5998>
Namespace: Interactive
Docstring:
L.pop([index]) -> item -- remove and return item at index (default last).
Raises IndexError if list is empty or index is out of range.

In [182]: lista_mixta
Out[182]: [50, False, 13, 13.74, 50, 13, '50', 'altair', 'cincuenta']

In [183]: expulsado = lista_mixta.pop() #LIFO, como en colas!

In [184]: expulsado
Out[184]: 'cincuenta'

In [185]: lista_mixta
Out[185]: [50, False, 13, 13.74, 50, 13, '50', 'altair']
```

# Recorrer una lista

```
In [227]: for elemento in lista_mixta:
.....:     print elemento
.....:
50
False
13
13.74
50
13
50
altair

In [228]:
```

# Cadenas a listas

```
In [37]: str_paises_csv
```

```
Out[37]: 'CR,US,MX,AR,ES,PE,IT,GR,FR'
```

```
In [38]: lista_paises = str_paises_csv.split(",")
```

```
In [39]: lista_paises
```

```
Out[39]: ['CR', 'US', 'MX', 'AR', 'ES', 'PE', 'IT', 'GR', 'FR']
```

```
In [40]: type(lista_paises)
```

```
Out[40]: list
```

```
In [41]: len(lista_paises)
```

```
Out[41]: 9
```

# Diccionarios

- Pares llave, valor
- Mapea valores con llaves
- Muchas de las funciones y utilidades que servían para listas, sirven para diccionarios.
- Llaves pueden ser tipos “inmutables”: números, cadenas de texto, etc, pero NO listas u otros diccionarios (que pueden cambiar).

# Diccionarios - inicializando

```
In [199]: paises = {"cr" : "Costa Rica", "us" : "Estados Unidos", "mx" : "Mexico"}  
#Inicializando diccionario con valores
```

```
In [200]: len(paises)
```

```
Out[200]: 3
```

```
In [201]: paises
```

```
Out[201]: {'cr': 'Costa Rica', 'mx': 'Mexico', 'us': 'Estados Unidos'}
```

```
In [202]: paises["cr"]
```

```
Out[202]: 'Costa Rica'
```

```
In [203]: saludo = "Hola, soy Pepe y vivo en %s"%(paises["cr"])
```

```
In [204]: print saludo
```

```
Hola, soy Pepe y vivo en Costa Rica
```

```
In [205]: █
```

# Diccionarios: funciones

```
Terminal
File Edit View Search Terminal Tabs Help

Terminal x Terminal

In [213]: llaves = diccionario.keys() #Lista con las llaves

In [214]: print llaves
['sol', 'tierra']

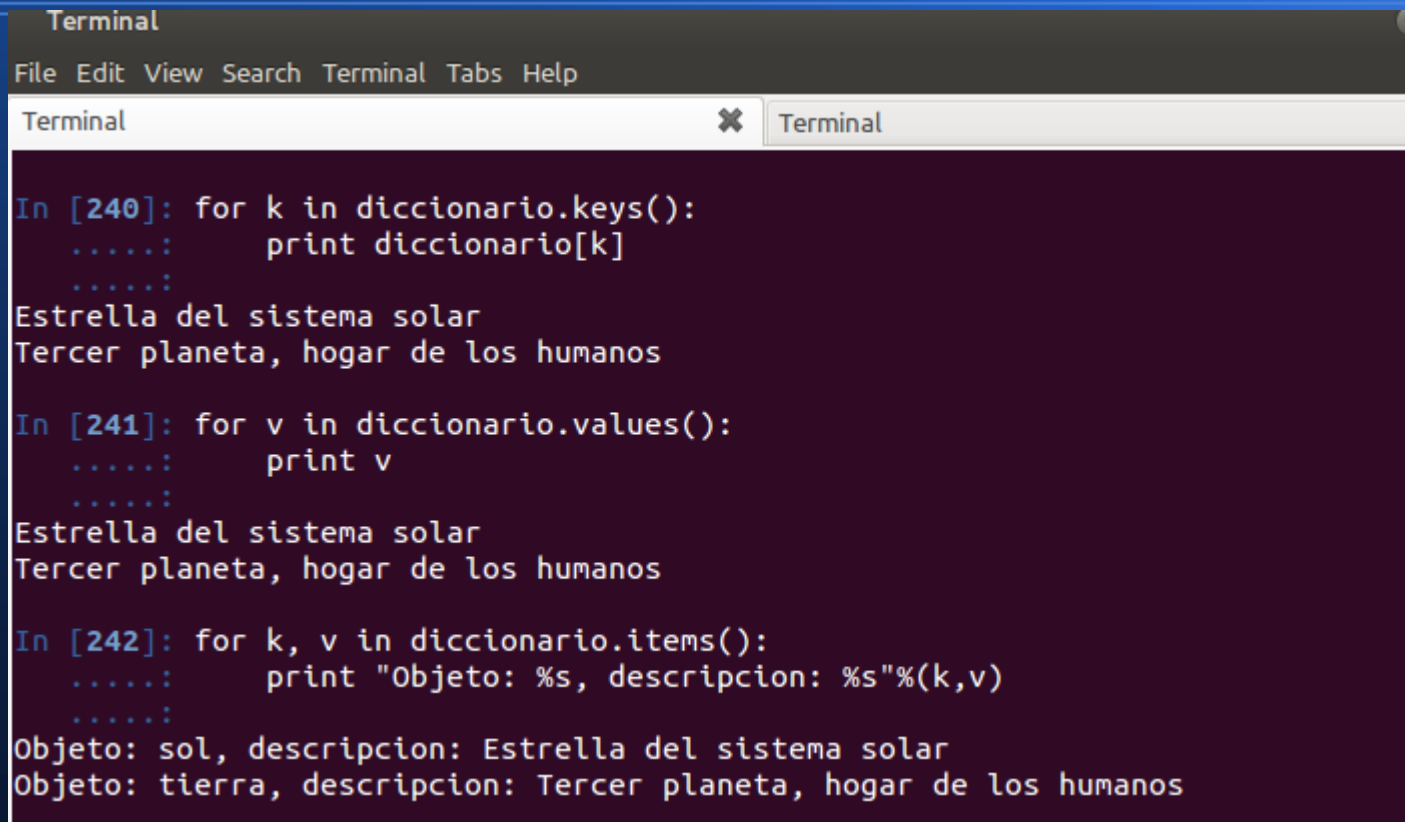
In [215]: valores = diccionario.values() #Lista con los valores

In [216]: print valores
['Estrella del sistema solar', 'Tercer planeta, hogar de los humanos']

In [217]: diccionario.has_key("marte") #Pregunto si el dicc tiene una llave
Out[217]: False

In [218]: diccionario.has_key("sol")
Out[218]: True
```

# Recorrer pares de diccionario



```
Terminal
File Edit View Search Terminal Tabs Help

Terminal

In [240]: for k in diccionario.keys():
.....:     print diccionario[k]
.....:
Estrella del sistema solar
Tercer planeta, hogar de los humanos

In [241]: for v in diccionario.values():
.....:     print v
.....:
Estrella del sistema solar
Tercer planeta, hogar de los humanos

In [242]: for k, v in diccionario.items():
.....:     print "Objeto: %s, descripcion: %s"%(k,v)
.....:
Objeto: sol, descripcion: Estrella del sistema solar
Objeto: tierra, descripcion: Tercer planeta, hogar de los humanos
```

# Conjuntos

- Una lista de elementos únicos
- Mismo concepto que conjunto en matemáticas

```
In [46]: conjunto = set()

In [47]: type(conjunto)
Out[47]: set

In [48]: len(conjunto)
Out[48]: 0

In [49]: conjunto = set([1,2,3,1,4,2,5]) #Construir un conjunto a partir de lista

In [50]: conjunto
Out[50]: set([1, 2, 3, 4, 5])
```



# Conjuntos - operaciones

```
In [90]: conjunto1
Out[90]: set([1, 2, 3, 4])

In [91]: conjunto1.add(5)

In [92]: conjunto1.add(5) #Solo lo agrega 1 vez, ¡Elementos unicos!

In [93]: conjunto1
Out[93]: set([1, 2, 3, 4, 5])

In [94]: conjunto2
Out[94]: set([3, 4, 5, 6])

In [95]: conjunto1.update(conjunto2)

In [96]: conjunto1
Out[96]: set([1, 2, 3, 4, 5, 6])
```

# Tuplas

- Similares a listas, pero, son inmutables (sirven como llave en diccionarios).

```
In [113]: mitupla_mixta = (1,2,"hola",True,2,1)

In [114]: mitupla_mixta[2]
Out[114]: 'hola'

In [115]: mitupla_mixta[2] = 3
-----
TypeError                                 Traceback (most recent call last)
/home/jeudy/<ipython-input-115-9452fb94d69f> in <module>()
----> 1 mitupla_mixta[2] = 3

TypeError: 'tuple' object does not support item assignment

In [106]: mitupla.index(4)
Out[106]: 3
```

# Ejercicio

- ¿Copia o referencia?

```
> list1 = [1, 5, 9, 13]
> list2 = list1
> list2[0] = -1
> print list1, list2
```

- ¿Como resolvemos el problema?

# Ejecución de script

- Llamando al interprete explicitamente
- Tener un .py con `#!/usr/bin/python` y usar `chmod`
- Clonar repositorio:
  - `git clone git://github.com/jeudy/intropython0415`
- Abrir script con editor de texto
- Ejecutarlo de las 2 formas

# Ejercicio

- Cree un nuevo repositorio en su cuenta de Github.com, inicialícelo localmente, asocie el remote a su cuenta de Github.
- Cree un nuevo script ejercicio0415.py y agréguelo al repositorio, push a master.
- Del repositorio intropython0415, baje el archivo variable\_para\_ejercicio.py que contiene la inicialización de una variable con un texto corto.

# Ejercicio

- Elimine todas las comas del texto.
- Convierta el texto a minúscula
- Cree un diccionario que contenga las palabras del texto como llaves, y el número de veces que aparece cada una (pista: funcion ***fromkeys*** de los diccionarios)
- Imprima una linea por cada elemento del diccionario, con las llaves ordenadas alfabéticamente. La cadena debe ser: “La palabra ? aparece ? veces” donde las ? se reemplazan por la palabra y el conteo.

# Resultado

```
La palabra <<reim>> aparece 1 veces
La palabra <<se>> aparece 3 veces
La palabra <<sentido>> aparece 1 veces
La palabra <<sentir>> aparece 1 veces
La palabra <<ser>> aparece 2 veces
La palabra <<setenta>> aparece 1 veces
La palabra <<significado>> aparece 1 veces
La palabra <<solo>> aparece 1 veces
La palabra <<soy>> aparece 3 veces
La palabra <<tambien>> aparece 1 veces
La palabra <<tengo>> aparece 1 veces
La palabra <<todas>> aparece 1 veces
La palabra <<un>> aparece 2 veces
La palabra <<una>> aparece 3 veces
La palabra <<uno>> aparece 1 veces
La palabra <<veinte>> aparece 1 veces
La palabra <<vida>> aparece 2 veces
La palabra <<voy>> aparece 1 veces
La palabra <<y>> aparece 3 veces
La palabra <<yo>> aparece 1 veces
jeudy@machine:~/Proyectos/UCR/intropython0415 (master)$
```