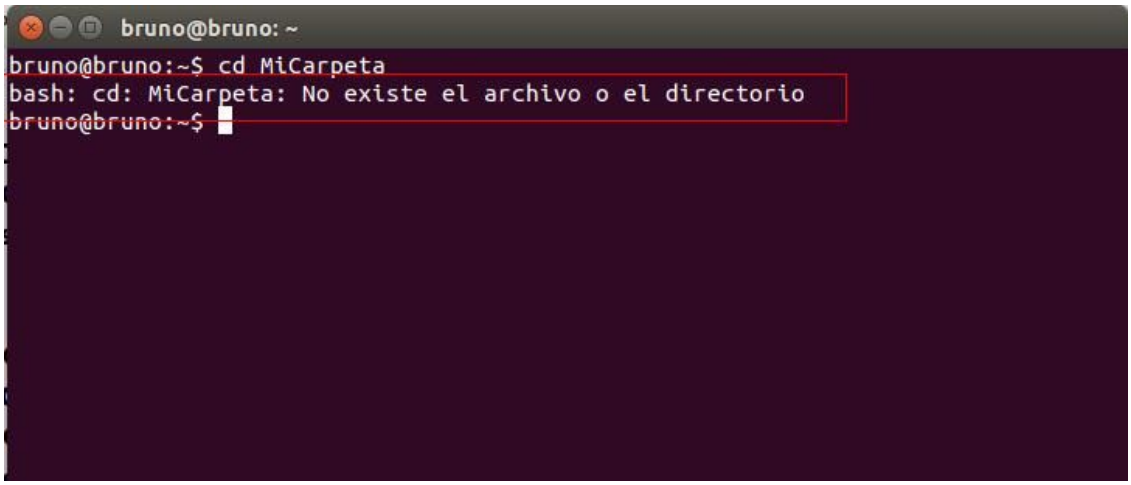
PREGUNTAS FRECUENTES

TÉCNICAS:

a. ¿Qué significa cuando nos aparece en la terminal el mensaje: *No existe el archivo o directorio*?

Al trabajar en la terminal nos puede aparecer el siguiente error:



Esto significa que estamos intentando acceder a través de la terminal, a una carpeta o archivo que:

- No existe, ó
- Especificamos mal su ruta

Si no existe, podemos crearla: Hacemos donde se desee click derecho y *nueva carpeta*.
En el caso de que se haya especificado mal su ruta, podemos ver la misma haciendo click derecho sobre la carpeta, elegimos *Propiedades*, y en la pestaña llamada *Básico*, vemos en *Lugar* su ruta:

b. ¿Qué es la ruta de una carpeta?

La ruta de una carpeta, es lo que identifica a la misma dentro de un sistema de archivos. Existe una carpeta "raíz" que es la que contiene a todas las demás, por lo general en Ubuntu se llama *home.*

La ruta de una carpeta se construye a partir de los nombres de las carpetas que la contienen. Por ejemplo, supongamos que dentro de *home* tenemos la carpeta *carp1*, y dentro de *carp1* tenemos *carp2*. Por consiguiente la ruta de *carp2* es ***/home/carp1***

c. ¿Cómo me muevo de carpeta en carpeta en la terminal?

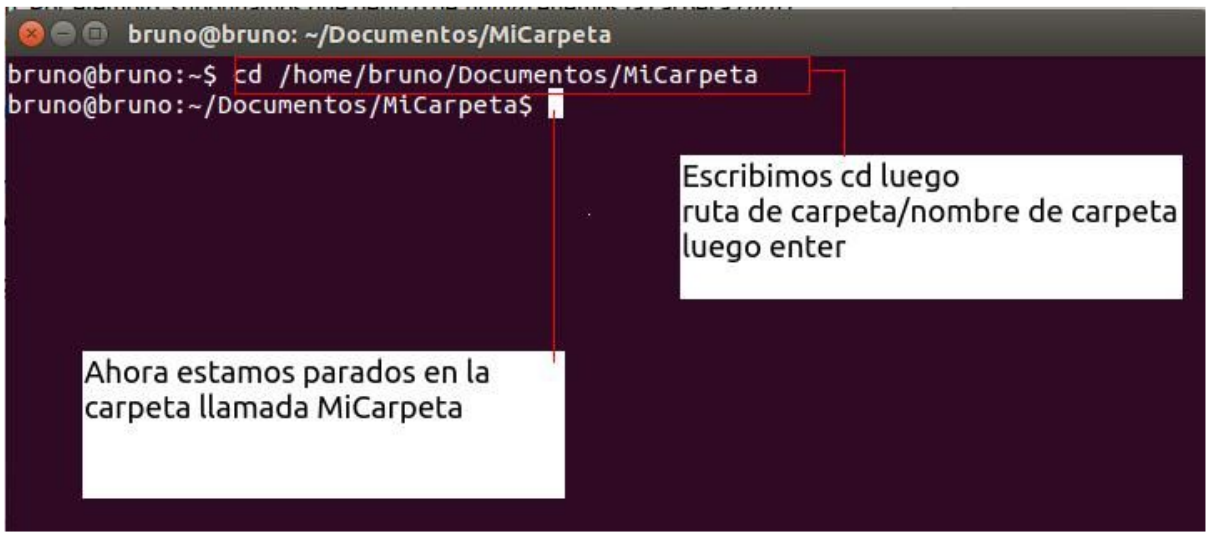
Existen muchos comandos útiles para trabajar usando la terminal. Para acceder a una carpeta. Algunos son **cd**(por **c**hange **d**irectory que significa cambiar de directorio) y **ls** (por **l**i**s**t que significa listar). Veamos cómo se ejecutan:

cd *ruta_de_carpeta/nombre_de_carpeta :* Para acceder desde la terminal a la carpeta deseada

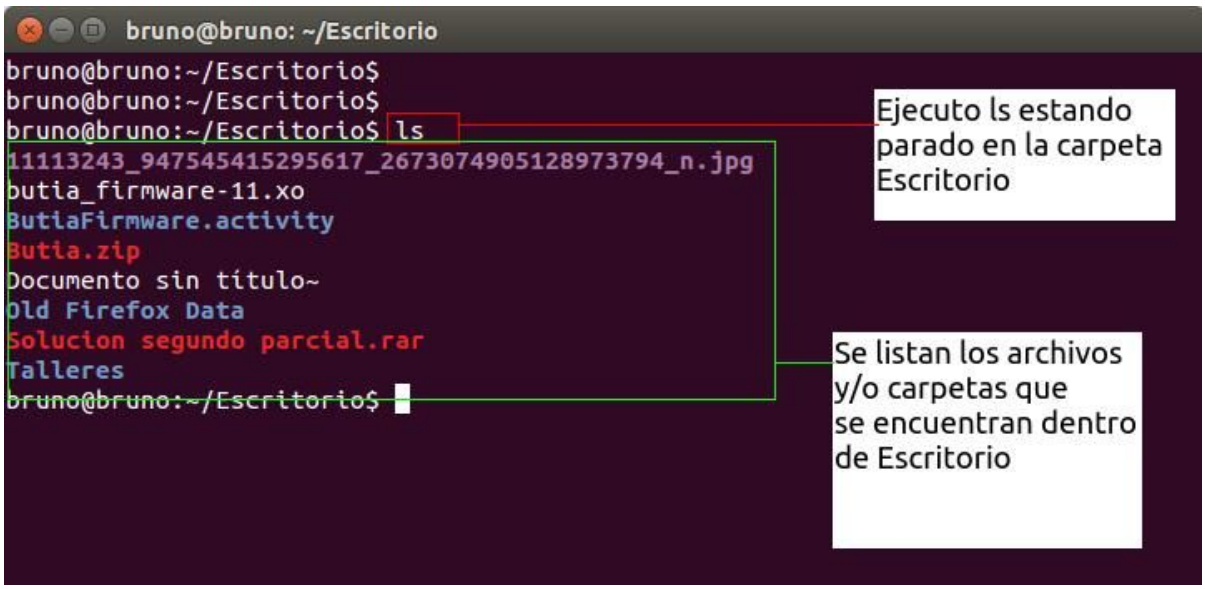
cd .. : para acceder a la carpeta anterior

ls : para ver el contenido de la carpeta en la que estamos parados

Veamos algún ejemplo de uso de cd:



Ahora un ejemplo de usar ls:

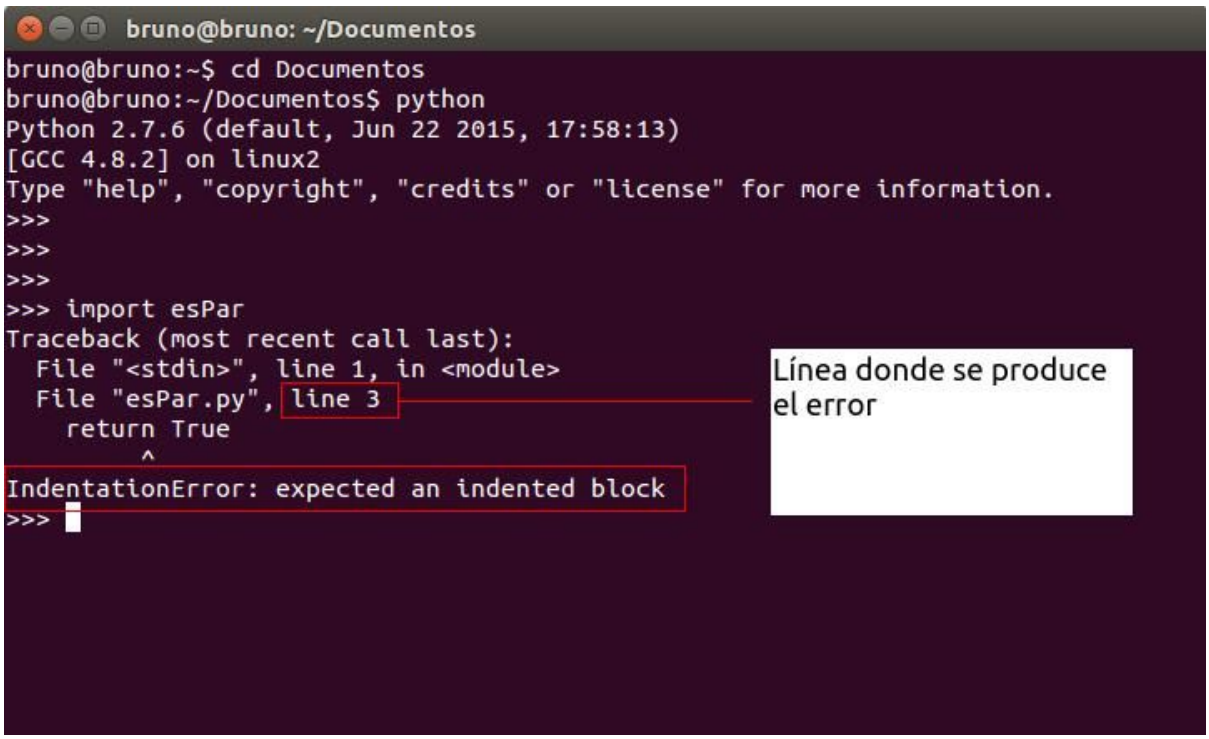


d. ¿Qué significa el error: *Identationerror : expected an indented block*?

Esto significa que estamos teniendo un error de indentación en el programa que queremos ejecutar. Por ejemplo, el siguiente programa:

```
1  a = 26
2  if (a%2 == 0):
3  print 'a es par'
4  else:
5      print 'a es impar'
```

Tiene un error de indentación en la línea 3. Si hacemos import de este programa (que le llamamos esPar):



DE PROGRAMACIÓN:

a. ¿Para qué sirven los comentarios en python?

Los comentarios en un programa son textos que **no** se ejecutan. Por lo que sólamente son útiles para las personas que trabajan con dichos programas. Los comentarios nos pueden servir para indicar qué estamos haciendo, o qué pretendemos hacer; de esta manera quien lo lea lo entenderá más fácilmente. Veamos un ejemplo:

```
1  #este es un pequeño programa para indicar si a es par o impar
2  a = 26
3  if (a%2 == 0): #si a módulo 2 me da 0 entonces es par, sino es impar
4      print 'a es par'
5  else:
6      print 'a es impar'
```

Podemos ver que en esta imagen en la línea 1 hemos comentado de qué trata el programa, y en la 3 comentamos qué se evalúa en el **if.** Para comentar una línea se pone primero el símbolo # seguido del texto. Veamos cómo comentar varias líneas:

```
1  #este es un pequeño programa para indicar si a es par o impar
2  a = 26
3  if (a%2 == 0): #si a módulo 2 me da 0 entonces es par, sino es impar
4      print 'a es par'
5  else:
6      print 'a es impar'
7
8  ''' Esto es un comentario de varias líneas,
9  las líneas de la 8 hasta la 10 no serán tomadas en cuenta porque
10 son parte del comentario. '''
```

Para comentar más de una línea, se pone todo entre **tres comillas simples**, es decir que se transforma automáticamente en comentario todo lo que empiece y termine con **'''.**

b. ¿Cómo se puede habilitar el uso de tildes en los comentarios de un programa?

Python no permite que los comentarios tengan tildes, pero podemos colocar en la primer línea:

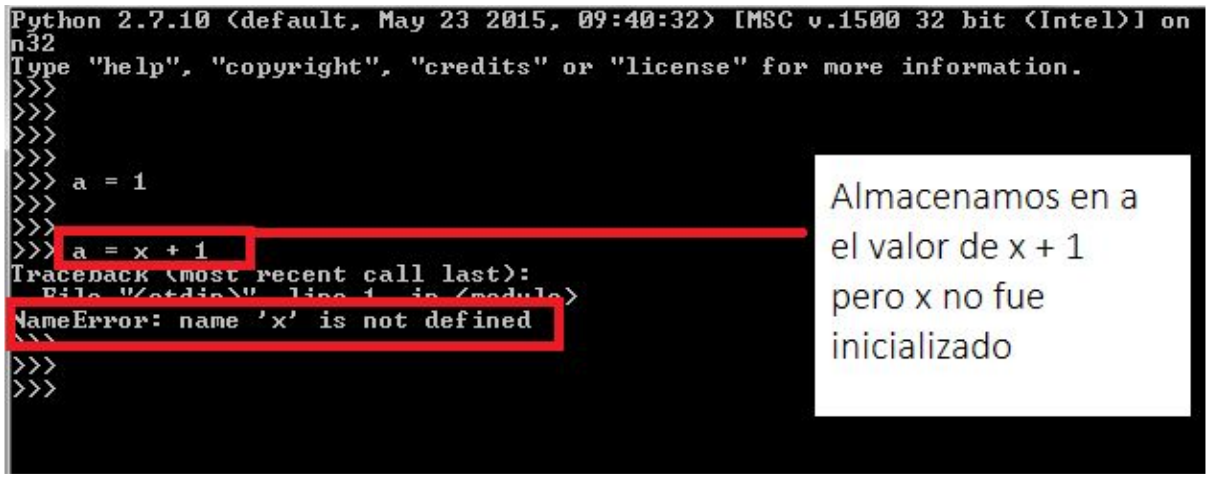
-*- coding: utf-8 -*-

```
1  # -*- coding: utf-8 -*-
2  #este es un pequeño programa para indicar si a es par o impar
3  a = 26
4  if (a%2 == 0): #si a módulo 2 me da 0 entonces es par, sino es impar
5      print 'a es par'
6  else:
7      print 'a es impar'
8  ''' Esto es un comentario de varias líneas,
9  las líneas de la 8 hasta la 10 no serán tomadas en cuenta porque
10 son parte del comentario. '''
```

Ahora este programa no va a mostrar error al tener comentarios que contengan tildes, como en la línea 8.

c. ¿Por qué es importante inicializar las variables?

Al inicializar variables, nos aseguramos que nuestros programas no accedan a lugares de la memoria donde no sabemos qué hay. Veamos el siguiente ejemplo ejecutando en la línea de comandos de Python:



Nos dio un error porque estamos haciendo la suma de x + 1, pero ¿Cuánto vale x? Python no sabe porque no le dimos valor a **x**, por lo tanto el programa no va a funcionar. En conclusión: **si queremos usar una variable es fundamental inicializarla.**

d. ¿Cómo puedo asegurarme en python que estoy trabajando con una división real?

En python 2.X:

- // es el cociente de la división entera entre los operandos. Por ejemplo:
    - **2//3** evalúa a **0**.
    - **-8//4** evalúa a **-2**.
    - **6.1//3** evalúa a **2.0**. (no está claro qué resultado es)
    - **2.9//2** evalúa a **1.0**. (no está claro qué resultado es)

- / es la división real. Por ejemplo:
    - **12.0/3.0** evalúa a **4.0**.
    - **-6.2/2.1** evalúa a **-2.9523809523809526**.
    - **12.0/5.0** evalúa a **2.4**.

¿Pero qué pasa si evaluamos **3/2**? La respuesta es que **el resultado es 1, cuando debería ser 1.5**. Es decir que **si los operandos son números enteros, la división que debería ser real, es división entera**.

Este problema se soluciona poniendo al comienzo del archivo en el que estemos trabajando:

```
from __future__ import division
```

De esta forma nos aseguramos que se utilice la implementación de python 3.X donde la división / siempre es real aunque se utilicen números enteros como operandos.

De todos modos hay un problema con la división entera:

1.9//2 da como resultado 0, cuando en realidad debería indicarse que no es correcto usar operandos reales.