Manejo de errores

Cuando se produce algun error durante la ejecución Python detiene la ejecución y emite un mensaje de Error.

```
>>> 5./0.
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: float division by zero
```

El mensaje nos da una traza del error dice el archivo (o los archivos) donde se produjo, la línea y el módulo.
Luego viene el tipo de error.

Debugging vs control de errores

Cuando se produce un error existen dos posibilidades:

- El error es imprevisto debido a algun problema del codigo. Debemos hacer un debugging.
- El error es posible en la logica del programa por lo que deberíamos hacer tratiemiento en el codigo a traves de Excepciones.

Debugging el código

Opcion mas artesanal: Se pueden introducir print de las variables antes de la linea donde aparece el error para ver lo que esta sucediendo.
Python trae incorporada una libreria para el debugging:

```
import pdb
# donde se quiere que comience a rastrear el bug agregar:
pdb.set_trace()
```

Comenzar a compilar python problema1.py
(en modo debugging)

Comandos para el debugging

Comandos esenciales:

n next ejecuta la linea

p a,b,c imprime variables

q si queremos salir sin seguir ejecutando.

c si queremos continuar la ejecucion sin seguir debugging

Excepción

Existe una forma de probar si da error y si es asi le decimos que haga algo.

```
try:
    a=1/b
except ZeroDivisionError,a:
    print 'Division por Zero'
```

De esta maneras evitamos el error y se continua con la ejecución. En la programacion se debe tener en cuenta la aparicion del error. En el caso ejemplo a no esta definido

Otros ejemplos

Chequea que exista el modulo a importar:

```
try: #fortran
    import hdmpfmod
except ImportError:
   quit('Requires compilation of fortran modules')
```

Otros ejemplos

Chequea que exista una variable, sino existe la crea con None:

```
def checkpar(par):
    " For input parameters that should be defined with No
    try:
        par
    except NameError:
        par = None
```

Este comando es util para el manejo de diccionarios o atributos de objetos.

Ejemplo

Si una variable no existe me dara un error

```
   print gravedad.g
```

Como hago para definirle un valor a la variable solo en el caso en que no lo tenga?

```
checkpar(gravedad.g) # la defino como None
if gravedad.g is None:
   gravedad.g=9.8
```

Control de los errores

Se puede generar una excepcion voluntaria en nuestro programa mediante la orden 'raise'.
De esta manera podemos controlar cuando aparezcan errores que sabemos de antemano puedan aparecer.
Cuando le pasamos le decimos el tipo de error y una cadena de caracteres que indique el contexto:

```
  >>> raise ZeroDivisionError,"El denominador se hizo 0"
Traceback (most recent call last):
File "<stdio>", line 1, in <module>
ZeroDivisionError: El denominador se hizo 0
```