

Manejo de errores

Prof.Miguel García Silvente

146

Manejo de errores

Solución básica para que no ocurran errores:
evitar que se realice la operación

```
c = [1,"2",2,3,4,5.2]
sumOfc = 0
for numero in c:
    if isinstance(numero,int):
        sumOfc += numero

if "dato" in B:
    A["dato"] = B["dato"]
```

Prof.Miguel García Silvente

147

Manejo de excepciones

O bien se puede manejar el error una vez producido

```
sumOfc = 0
```

```
for numero in c:
```

```
    try:
```

```
        sumOfc += numero
```

```
    except TypeError: # indicando cada opción
```

```
        pass
```

```
try:
```

```
    A["dato"] = B["dato"]
```

```
except KeyError:
```

Prof.Miguel García Silvente

148

```
    pass
```

Excepciones

- Las excepciones son eventos que pueden modificar el flujo de un programa.
- Se lanzan automáticamente con cada error.
- **try/except**: capturan y recuperan una excepción generada por el programador o por Python.
- **try/finally**: realiza acciones si la excepción se produce o no.
- **raise**: genera una excepción manualmente.
- **assert**: genera una excepción de forma condicional.

Tipos de excepciones

- Manejo de errores
 - Cada vez que Python detecta un error, lanza una excepción.
 - Por defecto se detiene la ejecución.
 - En otro caso, se trata de capturar y recuperar la excepción
- Notificación de eventos
 - Puede enviar una señal indicando una situación válida (por ejemplo en una búsqueda)
- Manejo de casos especiales
 - Maneja situaciones inusuales.
- Finalización de acciones
 - Garantiza que se termina una acción (try/finally)
- Control de flujo inusual
 - Un tipo de “goto” de alto nivel

Prof.Miguel García Silvente

150

Ejemplo de manejo de excepciones (I)

```
>>> print (3/0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
```

```
def division(x, y) :
    try :
        return x/y
    except ZeroDivisionError :
        print "división por cero"
```

Prof.Miguel García Silvente

151

Ejemplo de manejo de excepciones (II)

```
def division(x, y) :  
    try :  
        if y == 0 :  
            raise 'cero'  
        return x/y  
    except 'cero' :  
        print "división por cero"
```

Prof.Miguel García Silvente

152

Ejemplo de manejo de excepciones (III)

```
n = int(input("Introduce un entero: "))  
Introduce un entero: 23.5  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
ValueError: invalid literal for int() with base 10: '23.5'"
```

```
while not Terminar ::  
    try:  
        n = int(input('Introduce un entero: '))  
        Terminar = True  
    except ValueError:  
        print('no es un entero! Inténtalo de nuevo')
```

Prof.Miguel García Silvente

153

try / except / else

try:

```
<bloque de sentencias>      #código principal  
except <nombre1>:  
    <bloque de sentencias>  
except <nombre2>,<datos>:  
    <bloque de sentencias>  
except (<nombre3>,<nombre4>):  
    <bloque de sentencias>  
except:  
    <bloque de sentencias>  
else:                  # opcional, si no hay excepción  
    <bloque de sentencias>
```

Prof.Miguel García Silvente

154

Ejemplo de try / except

try:

```
<bloque>  
except NameError(): ...  
except IndexError(): ...  
except KeyError(): ...  
except (AttributeError,TypeError,SyntaxError):...
```

else:

- Capturar todas las excepciones: **except** vacío.
- No es recomendable porque evita que podamos encontrar errores.
- **else** se realiza si no hay ninguna excepción

Prof.Miguel García Silvente

155

try / finally

Con **finally** se realiza el bloque, se produzca o no una excepción

```
try:  
    <bloque>  
finally:  
    <bloque>
```

- Garantiza que algo se haga pase lo que pase.

Prof.Miguel García Silvente

156

Ejemplo (I)

```
>>> try:  
>>> print (3/0)  
>>> finally: print "Terminado"  
Terminado  
Traceback...  
....  
ZeroDivisionError: integer division...
```

```
>>> try:  
>>>     try:  
>>>         print (3/0)  
>>>     except ZeroDivisionError : print ("Excepción")  
>>> finally: print ("Terminado")
```

Excepción
Terminado

Prof.Miguel García Silvente

157

Ejemplo (II)

```
>>> f = open("datos.txt", 'r')
>>> try:
>>> ...
>>> finally:
>>> f.close()
```

Traceback (most recent call last):

File "<stdin>", line 2, in <module>

FileNotFoundException: [Errno 2] No such file or directory: 'datos.txt'

raise

Permite lanzar de forma explícita una excepción

raise <nombre>

raise <nombre>,<datos># proporciona datos al manejador

raise #reenvía la última excepción

```
>>>try:
```

 raise 'cero', (3,0)

except 'cero': print "argumento cero"

except 'cero', datos: print(datos)

Ejemplo de raise

La última opción es útil si se quiere propagar la excepción capturada a otro manejador

try:

```
    raise InterrupcionTeclado
```

except:

```
    print ("propagar")
```

```
    raise
```

```
propagar
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 2, in <module>
```

```
NameError: name 'InterrupcionTeclado' is not defined
```

Prof.Miguel García Silvente

160

Ejemplo con múltiples excepciones

```
import sys
```

try:

```
    f = open('integers.txt')
```

```
    s = f.readline() # uno por línea
```

```
    i = int(s.strip())
```

```
except IOError as (errno, strerror):
```

```
    print "I/O error({0}): {1}".format(errno, strerror)
```

```
except ValueError:
```

```
    print "No valid integer in line."
```

```
except:
```

```
    print "Unexpected error:", sys.exc_info()[0]
```

```
    raise
```

Prof.Miguel García Silvente

161

assert

Es como **raise** pero de forma condicional

assert <condición>, <datos>

assert <condición>

Si la condición es falsa se lanza una excepción **AssertionError**

```
def f(x,y)
    assert x>0, 'x debe ser positivo'
    assert y<0, 'y debe ser negativo'
    return y**x
```

Cuestiones adicionales

- Todos los errores son excepciones pero no todas las excepciones son errores. Pueden ser señales o avisos

while True:

```
    try: line = input()
    except EOFError: break
    else: # procesa siguiente línea
```

- Se puede enviar una señal con **raise** para distinguir si es correcto o erróneo
- Trazar errores

try:

```
    ... # ejecutar programa
```

```
except: import sys; print sys.exc_type,sys.exc_value
```

Ejemplo

```
import sys
try:
    f = open('enteros.txt')
    s = f.readline()
    i = int(s.strip())
except IOError as (errno, strerror):
    print "I/O error({0}): {1}".format(errno, strerror)
except ValueError:
    print "No valid integer in line."
except:
    print "Unexpected error:", sys.exc_info()[0]
    raise
```

Prof.Miguel García Silvente

164

Cuestiones de diseño

- Se usa try en operaciones que usualmente pueden fallar como apertura de ficheros y llamadas a sockets
- Es posible que interese que el programa muera en ciertas situaciones
- Se usa *try/finally* para garantizar su ejecución.
- Es preferible usar una única instrucción try con varios casos en lugar de varios try.
- La vida de una excepción termina cuando es capturada.

Prof.Miguel García Silvente

165

Información sobre la excepción

try:

```
    raise NotImplementedError("No error")
```

except Exception as e:

```
    exc_type, exc_obj, exc_tb = sys.exc_info()
```

```
    fname =
```

```
os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
```

```
    print(exc_type, fname, exc_tb.tb_lineno)
```

Unicode

Unicode

- Los strings en ACSII tienen 7 bits. No es suficiente para expresar toda la información en distintos idiomas.
- Existen distintas codificaciones, en particular se usan uno o más bytes.
- Los strings Unicode proporcionan una codificación común a todos los idiomas.
- Es posible convertir información entre Unicode y otras codificaciones

Prof.Miguel García Silvente

168

Ejemplo de Unicode

```
>>> s = u'blå' # Carácter noruego å
>>> s.encode() # convierte a ASCII
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
UnicodeEncodeError: 'ascii' codec can't encode
  character u'\xe5' in position 2: ordinal not in range(128)
>>> s.encode('utf-8') # convierte aUTF-8
'b\xc3\xa5'
>>> s.encode('latin-1')
'b\xe5'
>>> s.encode('utf-16')
'\xff\xfeb\x00\x00\xe5\x00'
```

Prof.Miguel García Silvente

169

Ejemplo de Unicode y ficheros

```
import codecs  
fi = codecs.open(filename, 'r', encoding='utf-16')  
fo = codecs.open(filename, 'w', encoding='utf-8')  
for line in fi:  
    # line es un string unicode  
    fo.write(line) # convierte automáticamente a UTF-8
```

Prof.Miguel García Silvente

170

Anotaciones en funciones

- Ejemplo:

```
def posint(n: int) -> bool:  
    return n > 0
```

- Se asocia un atributo llamado `__annotations__` que corresponde al diccionario

```
{'n': <class 'int'>, 'return': <class 'bool'>}
```

- En PEP (Python Enhancement Proposal) aparecen casos de uso que incluyen chequeo de tipos.

Prof.Miguel García Silvente

171

Cálculo simbólico

```
>>> from sympy import *
>>> x = symbols('x')
>>> a = Integral(cos(x)*exp(x), x)
>>> Eq(a, a.doit())
Integral(exp(x)*cos(x), x) == exp(x)*sin(x)/2 +
exp(x)*cos(x)/2
```