**Excepciones integradas**

Te mostraremos una breve lista de las excepciones más útiles. Si bien puede sonar extraño llamar "útil" a una cosa o un fenómeno que es un signo visible de una falla o retroceso, como sabes, errar es humano y si algo puede salir mal, saldrá mal.

Las excepciones son tan rutinarias y normales como cualquier otro aspecto de la vida de un programador.

Para cada excepción, te mostraremos:

* Su nombre.
* Su ubicación en el árbol de excepciones.
* Una breve descripción.
* Un fragmento de código conciso que muestre las circunstancias en las que se puede generar la excepción.

Hay muchas otras excepciones para explorar: simplemente no tenemos el espacio para revisarlas todas aquí.

**ArithmeticError**

**Ubicación:**

BaseException ← Exception ← ArithmeticError

**Descripción:**

Una excepción abstracta que incluye todas las excepciones causadas por operaciones aritméticas como división cero o dominio inválido de un argumento.

**AssertionError**

**Ubicación:**

BaseException ← Exception ← AssertionError

**Descripción:**

Una excepción concreta generada por la instrucción de aserción cuando su argumento se evalúa como False (falso), None (ninguno), 0, o una cadena vacía.

**Código :**

from math import tan, radians

angle = int(input('Ingresa el angulo entero en grados: '))

# debemos estar seguros de ese angulo != 90 + k \* 180

assert angle % 180 != 90

print(tan(radians(angle)))

**BaseException**

**Ubicación:**

BaseException

**Descripción:**

La excepción más general (abstracta) de todas las excepciones de Python: todas las demás excepciones se incluyen en esta; se puede decir que las siguientes dos excepciones son equivalentes: except: y except BaseException:.

**IndexError**

**Ubicación:**

BaseException ← Exception ← LookupError ← IndexError

**Descripción:**

Una excepción concreta que surge cuando se intenta acceder al elemento de una secuencia inexistente (por ejemplo, el elemento de una lista).

**Código:**

# el codigo muestra una forma extravagante

# de dejar el bucle

lista = [1, 2, 3, 4, 5]

ix = 0

doit = True

while doit:

try:

print(lista[ix])

ix += 1

except IndexError:

doit = False

print('Listo')

## KeyboardInterrupt

**Ubicación:**

BaseException ← KeyboardInterrupt

**Descripción:**

Una excepción concreta que surge cuando el usuario usa un atajo de teclado diseñado para terminar la ejecución de un programa (*Ctrl-C* en la mayoría de los Sistemas Operativos); si manejar esta excepción no conduce a la terminación del programa, el programa continúa su ejecución. Nota: esta excepción no se deriva de la clase Exception. Ejecuta el programa en IDLE.

**Código:**

# este código no puede ser terminado

# presionando Ctrl-C

from time import sleep

seconds = 0

while True:

try:

print(seconds)

seconds += 1

sleep(1)

except KeyboardInterrupt:

print("¡No hagas eso!")

## LookupError

**Ubicación:**

BaseException ← Exception ← LookupError

**Descripción:**

Una excepción abstracta que incluye todas las excepciones causadas por errores resultantes de referencias no válidas a diferentes colecciones (listas, diccionarios, tuplas, etc.).

## MemoryError

**Ubicación:**

BaseException ← Exception ← MemoryError

**Descripción:**

Se produce una excepción concreta cuando no se puede completar una operación debido a la falta de memoria libre.

**Código:**

# este código causa la excepción MemoryError

# advertencia: ejecutar este código puede ser crucial

# para tu sistema operativo

# ¡no lo ejecutes en entornos de producción!

string = 'x'

try:

while True:

string = string + string

print(len(string))

except MemoryError:

print('¡Esto no es gracioso!')

## OverflowError

**Ubicación:**

BaseException ← Exception ← ArithmeticError ← OverflowError

**Descripción:**

Una excepción concreta que surge cuando una operación produce un número demasiado grande para ser almacenado con éxito.

**Código:**

# el código imprime los valores subsequentes

# de exp(k), k = 1, 2, 4, 8, 16, ...

from math import exp

ex = 1

try:

while True:

print(exp(ex))

ex \*= 2

except OverflowError:

print('El número es demasiado grande.')

## ImportError

**Ubicación:**

BaseException ← Exception ← StandardError ← ImportError

**Descripción:**

Se produce una excepción concreta cuando falla una operación de importación.

**Código:**

# una de estas importaciones fallará, ¿cuál será?

try:

import math

import time

import abracadabra

except:

print('Una de sus importaciones ha fallado.')

## KeyError

**Ubicación:**

BaseException ← Exception ← LookupError ← KeyError

**Descripción:**

Una excepción concreta que surge cuando intentas acceder al elemento inexistente de una colección (por ejemplo, el elemento de un diccionario).

**Código:**

# como abusar del diccionario

# y cómo lidiar con ello

dict = { 'a' : 'b', 'b' : 'c', 'c' : 'd' }

ch = 'a'

try:

while True:

ch = dict[ch]

print(ch)

except KeyError:

print('No existe tal clave:', ch)

Hemos terminado con excepciones por ahora, pero volverán cuando discutamos la programación orientada a objetos en Python. Puedes usarlos para proteger tu código de accidentes graves, pero también tienes que aprender a sumergirte en ellos, explorando la información que llevan.

De hecho, las excepciones son objetos; sin embargo, no podemos decirle nada sobre este aspecto hasta que te presentemos clases, objetos y similares.

Por el momento, si deseas obtener más información sobre las excepciones por tu cuenta, consulta la Biblioteca estándar de Python en <https://docs.python.org/3.6/library/exceptions.html>.