Manejo de Excepciones

Manejo de Excepciones

Jonatan Gomez Perdomo, Ph. D.
jgomezpe@unal.edu.co

Arles Rodríguez, Ph.D.
aerodriguezp@unal.edu.co

Camilo Cubides, Ph.D. (c)
eccubidesg@unal.edu.co

Carlos Andrés Sierra, M.Sc.
casierrav@unal.edu.co

Research Group on Articial Life
Grupo de investigación en vida artificial (Alife)
Computer and System Department
Engineering School
Universidad Nacional de Colombia

Agenda

Hechos QUE CONECTAN

UNIVERSIDAD NACIONAL DE COLOMBIA un

Definición I

En general los programas usan librerías, operaciones preexistentes o funciones nuevas que son especificadas para que funcionen con un cierto tipo de datos, con unos ciertos valores, o bajo unas ciertas condiciones.

En algunas ocasiones este tipo de especificaciones no se pueden garantizar. Por ejemplo, cuando se lee un archivo, es posible que el dispositivo externo falle y no se permita la lectura del mismo, o que al realizar una división, el divisor resulte aproximado a 0 y no se pueda realizar dicha operación, o querer obtener un valor de una lista que este más allá de su tamaño. Algunas veces estas situaciones son derivadas del mal uso que realizan los usuarios del programa al ingresar valores inválidos.

Definición II

Este tipo de situaciones, que son consideradas **excepcionales**, pueden afectar el flujo "normal" del programa interrumpiendo en muchos casos su ejecución.

Un programador puede considerar estas situaciones inesperadas de forma anticipada y darle un **manejo excepcional** a las mismas para que no interrumpan la ejecución del programa.

Agenda

División por cero I

Ejemplo

Para el programa.

```
def division(a, b):
    coc = a//b
    res = a % b
    return(coc, res)

division(4,5)
    print(division(10, 0))
    print(division(1024, 10))
```


División por cero II

Ejemplo (continuación)

La salida obtenida es:

```
ZeroDivisionError  Traceback (most recent call last)
<ipython-input-24-af58c2d4968e> in <module>
      6 division(4,5)
      7
----> 8 print(division(10, 0))
      9 print(division(1024,10))
<ipython-input-24-af58c2d4968e> in division(a, b)
      1 def division(a, b):
----> 2   coc = a//b
      3   res = a % b
      4   return (coc, res)
      5
      ZeroDivisionError: integer division or modulo by zero
```

Agenda

Sintaxis

El manejo de excepciones, si se quiere uno para cada tipo de excepción (opcional), tiene la siguiente forma:

```
try:
    código que puede generar alguna excepción a manejar
except Exception_name1: # Primer tipo de excepción
    código que maneja la excepción Exception_name1
except Exception_name2: # Segundo tipo de excepción (opcional)
    código que maneja la excepción Exception_name2
    ...
except Exception_nameN: # N-ésimo tipo de excepción (opcional)
    código que maneja la excepción Exception_nameN
else: #opcional
    código extra por si no se presenta una excepción
```

Manejo de la excepción generada por la división por cero I

Ejemplo

Para el programa.

```
def division(a, b):
    try:
        coc = a // b
        res = a % b
        return(coc, res)
    except:
        print("Error en la división de", a, "entre", b)
        return ""
print(division(10, 0))
print(division(1024, 10))
```

Manejo de la excepción generada por la división por cero II

Ejemplo (continuación)

La salida obtenida es:

```
Error en la división de 10 entre 0

(102, 4)
```

Agenda

Hechos QUE CONECTAN

UNIVERSIDAD NACIONAL DE COLOMBIA un

Valor no apropiado I

Ejemplo

En el siguiente programa se pueden generar errores cuando se ingresa un texto en la variable num o en div.

```
def division(a, b):
    try:
        coc = a//b
        res = a % b
        return(coc, res)
    except:
        print("Error en la división de", a, "entre", b)
def main():
    num = int(input("digite el dividendo: "))
    div = int(input("digite el divisor: "))
    print(division(num, div))
main()
```

Valor no apropiado II

Ejemplo (continuación)

La salida obtenida cuando se digita el texto hola, es:

```
digite el dividendo: hola
---------------------------------------------------------------
ValueError                    Traceback (most recent call last)
<ipython-input-11-2cc363a40bbb> in <module>()
     10    div = int(input("digite el divisor: "))
     11    print(division(num, div))
---> 12 main()
<ipython-input-11-2cc363a40bbb> in main()
      7        print("Error en la división de", a, "entre", b)
      8 def main():
----> 9    num = int(input("digite el dividendo: "))
     10    div = int(input("digite el divisor: "))
     11    print(division(num, div))
ValueError: invalid literal for int() with base 10: 'hola'
```

Manejo de la excepción generada por un valor no apropiado I

Ejemplo

Para el programa

```
def division(a, b):
    try:
        coc = a // b
        res = a % b
        return(coc, res)
    except:
        print("Error en la división de", a, "entre", b)
```


Manejo de la excepción por un valor no apropiado II

Ejemplo

```
def main():
    try:
        num = int(input("digite el dividendo: "))
        div = int(input("digite el divisor: "))
        print(division(num, div))
    except ValueError:
        print("El valor digitado no es un número.")
main()
```

La salida obtenida es

```
digite el dividendo: diez
El valor digitado no es un número.
```

Agenda

Hechos QUE CONECTAN

UNIVERSIDAD NACIONAL DE COLOMBIA un

El bloque finalmente (`finally`)

Se especifica el bloque `finally` para determinar acciones que se deben ejecutar sin importar si se produce una excepción o no

```
try:
    Código que puede generar alguna excepción a manejar
except:
    # Manejo de excepciones (pueden ser varias)
finally:
    Código que se ejecuta al final si o si
```

El orden de ejecución de las excepciones es el siguiente:

$$\texttt{try} \Longrightarrow \texttt{except} \Longrightarrow \texttt{else} \Longrightarrow \texttt{finally}$$

Ejemplo: El bloque finalmente (`finally`)

Ejemplo

Al ejecutar el siguiente programa, el texto `El programa termina!`, se mostrará independientemente de que se produzca una excepción o no:

```
try:
    num = int(input("Ingrese un número "))
    re = 100/num
except:
    print("Algo está mal")
else:
    print("El resultado es ",re)
finally:
    print("El programa termina!")
```

Agenda

Capturar e identificar varios tipos de excepciones I

Para determinar el tipo de excepción se puede utilizar una línea de código:

```
try:
    num = int(input("Ingrese un número: "))
    re = 100/num # Generar excepción si se digitó 0
    print(re)
except Exception as e:
    print(e, "\n", type(e))
```

Capturar e identificar varios tipos de excepciones II

- Si el usuario digita cero (0), se tiene como salida

```
Ingrese un número: 0
division by zero
 <class 'ZeroDivisionError'>
```

- Si el usuario digita hola, se tiene como salida

```
Ingrese un número: hola
invalid literal for int() with base 10: 'hola'
 <class 'ValueError'>
```

Hechos QUE CONECTAN

UNIVERSIDAD NACIONAL DE COLOMBIA un

Agenda

Lanzar una excepción (raise) I

Es posible lanzar una excepción (cuando sea necesario) utilizando el comando `raise`. Para el código:

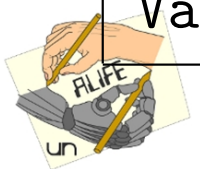
```
raise ValueError("error de división por cero")
```

La salida es:

```
ValueError                Traceback (most recent call last)

<ipython-input-6-496f78253296> in <module>()
----> 1 raise ValueError("error de división por cero")

ValueError: error de división por cero
```



Lanzar una excepción (raise) II

Ejemplo

Para el siguiente código se lanza y se atrapa una excepción

```
def division(a, b):
    if b == 0:
        raise ValueError("!Error de división por cero¡")
    else:
        coc = a // b
        res = a % b
        return(coc, res)
try:
    print(division(10, 0))
except Exception as e:
    print(e, "\n", type(e))
```


Lanzar una excepción (raise) III

Ejemplo (continuación)

La salida es:

```
!Error de división por cero¡
 <class 'ValueError'>
```

Ejercicios I

Problemas

1. Capture la excepción que evita que el usuario acceda a posiciones que no se encuentran definidas en la lista dada y muestre el mensaje `Intenta acceder a una posición que no está en la lista`:

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

Si se ejecuta sin el manejo de la excepción se produce la siguiente salida:

```
IndexError      Traceback (most recent call last)
<ipython-input-42-64245f71fd49> in <module>
      1 lista = [1, 2, 3, 4]
----> 2 lista[5]
IndexError: list index out of range
```

Ejercicios II

Problemas (continuación)

2. Capture la excepción para evitar que un programador sume una cadena de texto a un número y muestre el mensaje
   `Los tipos de datos no cuadran para hacer la operación`:

```
def operar(a, b):
    return a + b

def main():
    a = int(input())
    b = "hola"
    operar(a, b)

main()
```

Ejercicios III

Problemas (continuación)

Si se ejecuta el anterior programa, sin el manejo de la excepción, se produce la siguiente salida (con cualquier número dado por el usuario):

```
---------------------------------------------------------------
TypeError                 Traceback (most recent call last)
<ipython-input-43-63f2edb5e1e0> in <module>
      7     operar(a, b)
      8
----> 9 main()

<ipython-input-43-63f2edb5e1e0> in main()
      5     a = int(input())
      6     b = "hola"
----> 7     operar(a, b)
      8
      9 main()
...
```

Ejercicios IV

Problemas (continuación)

Si se ejecuta el anterior programa, sin el manejo de la excepción, se produce la siguiente salida (con cualquier número dado por el usuario):

```
...
<ipython-input-43-63f2edb5e1e0> in operar(a, b)
      1 def operar(a, b):
----> 2     return a + b
      3
      4 def main():
      5     a = int(input())

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

Ayuda: La instrucción `type(b) == str` permite verificar si la variable `b` es una cadena.

Ejercicios V

Problemas (continuación)

3. Capture la excepción cuando se trata de obtener una llave que no se encuentra en un diccionario y muestre el mensaje `Intenta acceder una llave que no está en el diccionario`:

```
def main():
    dict = {"James":"Java", "Dennis":"C", "Das":"Python"}
    print(dict["Ada"])

main()
```

Ejercicios VI

Problemas

Si se ejecuta el anterior programa, sin el manejo de la excepción, se produce la siguiente salida:

```
---------------------------------------------------------
KeyError    Traceback (most recent call last)
<ipython-input-45-174134c9fede> in <module>
      3     print(dict["Ada"])
      4
----> 5 main()

<ipython-input-45-174134c9fede> in main()
      1 def main():
      2     dict = {"James": "Java", "Dennis" : "C", "Das":"Python"}
----> 3     print(dict["Ada"])
      4
      5 main()

KeyError: 'Ada'
```