Programación II

Excepciones

Lic. Federico Severino Guimpel
LCC - FCEIA - UNR

Excepciones

Una excepción, en términos de programación, representa un error en tiempo de ejecución.

Por ejemplo, supongamos que tenemos el siguiente código, en Python:

```
a = int(input("Ingrese un numero: "))
print(a)
```

Vamos a ver el resultado con diferentes ejecuciones.

Si ingresamos un número, como nos solicita el cartel, el programa funciona con normalidad:

```
Ingrese un numero: 8
8
```

Si por el contrario en lugar de un número se ingresa otro tipo de dato. Obtenemos una salida similar a esta:

```
Ingrese un numero: hola
Traceback (most recent call last):
  File "D:/Fede/Facultad/Programacion II/Python/ejemplos-excepciones.py", line 2, in <module>
    a = int(input("Ingrese un numero: "))
ValueError: invalid literal for int() with base 10: 'hola'
```

Como podemos ver, la excepción nos da información sobre:

- el tipo de error producido;
- dónde se produjo.

En nuestro ejemplo, la excepción nos indica que estamos intentando convertir a int a un dato que no lo es y, que esto pasa en la línea 2 del archivo.

```
Ingrese un numero: hola
Traceback (most recent call last):
  File "D:/Fede/Facultad/Programacion II/Python/ejemplos-excepciones.py", line 2, in <module>
    a = int(input("Ingrese un numero: "))
ValueError: invalid literal for int() with base 10: 'hola'
```

Pero entonces, ¿qué alternativas se tienen para que una excepción no rompa el programa?

Esa es la pregunta que se va a intentar responder a continuación. La respuesta de la misma depende del lenguaje de programación elegido, pero el mecanismo es similar.

Excepciones

La gestión de excepciones, en el caso de Python, viene de la mano del par: try - except.

¿En qué consiste su uso? El bloque try contiene una o más sentencias que pueden contener excepciones.

Por ejemplo, podríamos poner:

```
try:
    a = int(input("Ingrese un numero: "))
```

Excepciones

A continuación del bloque try vienen uno o más bloques except. En cada uno de ellos se indica qué tipo de excepción se va a capturar.
Las sentencias dentro de dicho bloque representan el código a ejecutar en caso que el bloque try haya generado dicha excepción.

Vamos a aplicar esto a nuestro ejemplo. La excepción que se había generado era del tipo ValueError.
Por lo que nuestro código sería:

```
try:
    a = int(input("Ingrese un numero: "))
except ValueError:
    print("No ingresó un número!")
```

Excepciones

Ahora bien, supongamos que escribimos el siguiente código:

```
try:
    a = int(input("Ingrese un numero: "))
except ValueError:
    print("No ingresó un número!")
print(a)
```

¿Cuál se imaginan que es el comportamiento del mismo?

Si se ejecuta con una entrada correcta, el programa funciona adecuadamente:

```
Ingrese un numero: 10
10
```

Sin embargo, si se ingresa una cadena la salida que se tiene es:

```
Ingrese un numero: hola
No ingresó un número!
Traceback (most recent call last):
  File "D:/Fede/Facultad/Programacion II/Python/ejemplos-excepciones.py", line 6, in <module>
    print(a)
NameError: name 'a' is not defined
```

¿Qué significa esto?

La explicación de esta excepción (Sí, esta es una nueva excepción) es la siguiente:

```
try:
    a = int(input("Ingrese un numero: "))
except ValueError:
    print("No ingresó un número!")
print(a)
```

El bloque try cuenta con una única sentencia en la que se asigna a una variable a, el resultado de la conversión a entero del dato ingresado por teclado. Si, como en este caso, esa línea generó una excepción, la misma no se ejecutó. Por lo que la variable *a* no se encuentra declarada.

La explicación de esta excepción (sí, esta es una nueva excepción) es la siguiente:

```
try:
    a = int(input("Ingrese un numero: "))
except ValueError:
    print("No ingresó un número!")
print(a)
```

La sentencia print final, al estar fuera de todo bloque, se ejecuta siempre. Esto es, sin importar si el bloque try generó una excepción o no.

Ahora bien, lo que se quisiera es que se imprima el valor de la variable *a* sólo si no se generó una excepción. Para esto, se cuenta con un bloque (opcional) else.

Usándolo el código quedaría así:

```
try:
    a = int(input("Ingrese un numero: "))
except ValueError:
    print("No ingresó un número!")
else:
    print(a)
```

Si se ejecuta, se pueden ver las siguientes salidas:

```
Ingrese un numero: 10
10

Ingrese un numero: hola
No ingresó un valor numérico
```

Excepciones

Ya tenemos resuelto el problema usando los bloques try - except - else.

Lo que se puede uno preguntar es: ¿existirá otra forma de resolver esto sin el uso del bloque else? Es decir, sólo usando try - except.

La respuesta es que sí. Vamos a analizar el siguiente código:

```
try:
    a = int(input("Ingrese un numero: "))
    print(a)
except ValueError:
    print("No ingresó un número!")
```

En este ejemplo, se puede ver el comportamiento del bloque try. Si la primera sentencia se ejecuta sin problemas, se continúan ejecutando las siguientes sentencias. Es decir, si se pudo leer y convertir correctamente entonces se va a imprimir el dato. Este comportamiento era el buscado.

```
Ingrese un numero: 6
6
```

```
Ingrese un numero: hola
No ingresó un número!
```