


11 Manejo de excepciones.

Objetivo. ...

Funciones de Python: ... [Errors and Exceptions](#)

[MACTI-Algebra_Lineal_01](#) by [Luis M. de la Cruz](#) is licensed under [Attribution-ShareAlike 4.0 International](#) 

12 Tipos de errores:

En Python existen dos tipos de errores por los cuales un programa se detiene y no continua con su ejecución normal.

12.1 Errores de sintaxis.

Ocurren cuando no se escriben correctamente las expresiones y declaraciones, siguiendo la especificación de la interfaz de Python.

Por ejemplo:

```
# Escribimos a propósito 'printf' que es un nombre incorrecto.  
printf('Hola mundo!')
```

NameError: name 'printf' is not defined

- Observa que el tipo de error se imprime cuando éste ocurre.
- En el caso anterior el error fue de tipo **NameError**, por lo que hay que revisar que todo esté correctamente escrito.

12.2 Errores provocados por excepciones.

Son errores lógicos que detienen la ejecución de un programa aún cuando la sintaxis sea la correcta.

Por ejemplo:

```
def raizCuadrada(numero):  
    numero = float(numero)  
    print("La raíz cuadrada del número {} es {}".format(numero, numero ** 0.5))
```

```
# Ejemplo correcto que se ejecuta sin problemas.  
raizCuadrada(1)
```

La raíz cuadrada del número 1.0 es 1.0

```
# Ejemplo correcto, se calcula la raíz cuadrada de -1 en  
# El resultado es un número complejo. En este caso Python  
# se encarga de realizar las conversiones necesarias.  
raizCuadrada(-1)
```

La raíz cuadrada del número -1.0 es $(6.123233995736766e-17+1j)$

```
# Ejemplo incorrecto. No es posible calcular la raíz cuadrada  
# de un número complejo, es una operación no definida.  
raizCuadrada(1+1j)
```

`TypeError: float() argument must be a string or a real number, not 'complex'`

En el ejemplo anterior se produce un error de tipo `TypeError`, es decir hay incompatibilidad con los tipos de datos que se están manipulando.

```
# Ejemplo incorrecto. No se puede calcular la raíz cuadrada  
# de una cadena.  
raizCuadrada("hola")
```

`ValueError: could not convert string to float: 'hola'`

En el ejemplo anterior se produce un error de tipo `ValueError`, es decir es decir hay un problema con el contenido del objeto.

13 Manejo de excepciones con: `try`, `except`, `finally`

Los errores que se pueden manejar, son aquellos errores lógicos como los presentados anteriormente en donde es posible “predecir” el tipo de error que puede ocurrir de acuerdo con la implementación que estamos realizando.

Todas las excepciones en Python son ejemplos concretos de una clase (*instance*) que se derivan de la clase principal [BaseExcepcion](#). Más detalles se pueden consultar [aquí](#).

Las excepciones se pueden capturar y manejar adecuadamente. Para ello se tienen las siguientes herramientas:

- `try`
- `except`
- `else`
- `finally`

Cuando se identifica una sección de código susceptible de errores, ésta puede ser delimitada con la expresión `try`. Cualquier excepción que ocurra dentro de esta sección de código podrá ser capturada y gestionada.

La expresión `except` es la encargada de gestionar las excepciones que se capturan. Si se utiliza sin mayor información, ésta ejecutará el código que contiene para todas las excepciones que ocurran.

En el ejemplo de la función `raizCuadrada()` podemos manejar las excepciones como sigue:

```
def raizCuadrada(numero):
    """
    Función que calcula la raíz cuadrada de un número.

    Parameters
    -----
    numero: int o float
    Valor al que se le desea calcular la raíz cuadrada.

    """
    # Intenta realizar el cálculo que está dentro de try
    try:
        numero = float(numero)
        print(f"La raíz cuadrada del número {numero} es {numero**0.5}")

    # Si ocurre una excepción se captura en el except
    except:
        # No se hace nada con la excepción (por el momento)
        pass

    print('Gracias por usar Python!')
```

Usando la nueva versión de la función `raizCuadrada()` intentemos ejecutarla con los ejemplos anteriores:

```
raizCuadrada(1)
```

La raíz cuadrada del número 1.0 es 1.0
Gracias por usar Python!.

```
raizCuadrada(-1)
```

La raíz cuadrada del número -1.0 es (6.123233995736766e-17+1j)
Gracias por usar Python!.

```
raizCuadrada(1+1j)
```

Gracias por usar Python!.

```
raizCuadrada("hola")
```

Gracias por usar Python!.

Observa que ya no hay errores. Esto se debe a que la cláusula `except` captura todas las posibles excepciones, pero no hace nada, y aún con argumentos erróneos, no se sabe que hubo un error.

13.0.1 Gestión general de las excepciones

Ya que sabemos como capturar las excepciones, veamos cómo pueden ser tratadas para dar retroalimentación al usuario.

```
def raizCuadrada(numero):
    """
    Función que calcula la raíz cuadrada de un número.

    Parameters
    -----
    numero: int o float
    Valor al que se le desea calcular la raíz cuadrada.

    """
    # Variable Booleana para manejar las excepciones.
    ocurre_error = False

    # Intenta realizar el cálculo que está dentro de try
    try:
        numero = float(numero)
        print("La raíz cuadrada del número {} es {}".format(numero, numero ** 0.5))

    # Si ocurre una excepción se captura en el except
    except:
        ocurre_error = True

    # Cuando ocurre un error se hace lo siguiente:
    if ocurre_error:
        print("Cuidado, hubo una falla en el programa, no se pudo realizar el cálculo")
    else:
        print('Gracias por usar Python!')
```

```
raizCuadrada(1)
```

La raíz cuadrada del número 1.0 es 1.0
Gracias por usar Python!.

```
raizCuadrada(-1)
```

La raíz cuadrada del número -1.0 es (6.123233995736766e-17+1j)
Gracias por usar Python!.

```
raizCuadrada(1+1j)
```

Cuidado, hubo una falla en el programa, no se pudo realizar el cálculo

```
raizCuadrada("hola")
```

Cuidado, hubo una falla en el programa, no se pudo realizar el cálculo

Observa que ahora se avisa al usuario que hubo un error al ejecutar la función por lo que el cálculo no se realizó. Sin embargo hace falta más información.

13.0.2 Gestión de las excepciones por su tipo.

La expresión `except` puede ser utilizada de forma tal que ejecute código dependiendo del tipo de error que ocurra. En este caso sabemos que pueden ocurrir dos tipos de errores: `TypeError` y `ValueError`. Entonces la nueva versión de la función `raizCuadrada()` es como sigue:

```
def raizCuadrada(numero):
    """
    Función que calcula la raíz cuadrada de un número.

    Parameters
    -----
    numero: int o float
    Valor al que se le desea calcular la raíz cuadrada.

    """
    # Variable Booleana para manejar las excepciones.
    ocurre_error = False

    # Intenta realizar el cálculo que está dentro de try
    try:
        numero = float(numero)
        print("La raíz cuadrada del número {} es {}".format(numero, numero ** 0.5))

    # En esta sección se trata la excepción de tipo TypeError
    except TypeError:
        ocurre_error = True
        print("Ocurrió un error de tipo: TypeError, verifique que los tipos sean")

    # En esta sección se trata la excepción de tipo ValueError
    except ValueError as detalles:
        ocurre_error = True
        print("Ocurrió un error de tipo ValueError, verifique el contenido de los")

    # En esta sección se tratan todas las otras posible excepciones
    except:
        ocurre_error = True
        print("Ocurrió algo misterioso")

    # Cuando ocurre un error se hace lo siguiente:
    if ocurre_error:
        print("Hubo una falla en el programa, no se pudo realizar el cálculo")
    else:
        print('Gracias por usar Python!.')
```

```
raizCuadrada(1)
```

La raíz cuadrada del número 1.0 es 1.0
Gracias por usar Python!.

```
raizCuadrada(-1)
```

La raíz cuadrada del número -1.0 es (6.123233995736766e-17+1j)
Gracias por usar Python!.

```
raizCuadrada(1+4j)
```

Ocurrió un error de tipo: TypeError, verifique que los tipos sean compatibles.
Hubo una falla en el programa, no se pudo realizar el cálculo

```
raizCuadrada("hola")
```

Ocurrió un error de tipo ValueError, verifique el contenido de los argumentos.
Hubo una falla en el programa, no se pudo realizar el cálculo

Observa que ya se da mayor información sobre el tipo de error que ocurrió y el usuario puede saber que hacer como corregir los errores.

Todos los tipos de errores que existen en Python se pueden consulta en [Concrete exceptions](#).

13.0.3 Información del error

Se puede capturar toda la información del error para pasarla al usuario. Esto se hace como sigue:

```
def raizCuadrada(numero):  
    """  
    Función que calcula la raíz cuadrada de un número.  
  
    Parameters  
    -----  
    numero: int o float  
    Valor al que se le desea calcular la raíz cuadrada.  
  
    """  
    # Variable Booleana para manejar las excepciones.  
    ocurre_error = False  
  
    # Intenta realizar el cálculo que está dentro de try  
    try:  
        numero = float(numero)  
        print("La raíz cuadrada del número {} es {}".format(numero, numero ** 0.5))  
  
    # En esta sección se trata la excepción de tipo TypeError y se obtienen los c  
    except TypeError as info:  
        ocurre_error = True  
        print("Ocurrió un error (TypeError):", info)
```

```
# En esta sección se trata la excepción de tipo ValueError y se obtienen los
except ValueError as info:
    ocurre_error = True
    print("Ocurrió un error (ValueError):", info)

# En esta sección se tratan todas las otras posible excepciones
except:
    ocurre_error = True
    print("Ocurrió algo misterioso")

# Cuando ocurre un error se hace lo siguiente:
if ocurre_error:
    print("Hubo una falla en el programa, no se pudo realizar el cálculo")
else:
    print('Gracias por usar Python!')
```

```
raizCuadrada(1)
```

La raíz cuadrada del número 1.0 es 1.0
Gracias por usar Python!.

```
raizCuadrada(-1)
```

La raíz cuadrada del número -1.0 es (6.123233995736766e-17+1j)
Gracias por usar Python!.

```
raizCuadrada(1+4j)
```

Ocurrió un error (TypeError): float() argument must be a string or a real number, not 'complex'
Hubo una falla en el programa, no se pudo realizar el cálculo

```
raizCuadrada("hola")
```

Ocurrió un error (ValueError): could not convert string to float: 'hola'
Hubo una falla en el programa, no se pudo realizar el cálculo

Observa que ahora además de conocer el tipo de error, también se muestra toda la información del error para que el usuario tome las acciones pertinentes.

13.0.4 finally

Esta sección se ejecuta siempre, sin importar si hubo una excepción o no.

```
def raizCuadrada(numero):
    """
    Función que calcula la raíz cuadrada de un número.
```

```

Parameters
-----
numero: int o float
Valor al que se le desea calcular la raíz cuadrada.

"""
# Variable Booleana para manejar las excepciones.
ocurre_error = False

# Intenta realizar el cálculo que está dentro de try
try:
    numero = float(numero)
    print("La raíz cuadrada del número {} es {}".format(numero, numero ** 0.5))

# En esta sección se trata la excepción de tipo TypeError y se obtienen los detalles
except TypeError as info:
    ocurre_error = True
    print("Ocurrió un error (TypeError):", info)

# En esta sección se trata la excepción de tipo ValueError y se obtienen los detalles
except ValueError as info:
    ocurre_error = True
    print("Ocurrió un error (ValueError):", info)

# En esta sección se tratan todas las otras posibles excepciones
except:
    ocurre_error = True
    print("Ocurrió algo misterioso")

# Cuando ocurre un error se hace lo siguiente:
finally:
    if ocurre_error:
        print("Hubo una falla en el programa, no se pudo realizar el cálculo")
    else:
        print('Gracias por usar Python!.')

```

```
raizCuadrada(1)
```

La raíz cuadrada del número 1.0 es 1.0
Gracias por usar Python!.

```
raizCuadrada(-1)
```

La raíz cuadrada del número -1.0 es (6.123233995736766e-17+1j)
Gracias por usar Python!.

```
raizCuadrada(1+4j)
```


Ocurrió un error (TypeError): float() argument must be a string or a real number, not 'complex'

Hubo una falla en el programa, no se pudo realizar el cálculo

```
raizCuadrada("hola")
```

Ocurrió un error (ValueError): could not convert string to float: 'hola'

Hubo una falla en el programa, no se pudo realizar el cálculo

13.0.5 Lanzar excepciones controladas.

Es posible presentar toda la información que genera la excepción y agregarle notas para el usuario. Para agregar notas usamos el método `add_note()` y para lanzar la excepción una vez controlada usamos `raise`. La siguiente versión de la función `raizCuadrada()` tiene al final una cláusula `else`, la cual se ejecuta cuando no ocurre ninguna excepción. En este caso, dentro del `try` realizamos el cálculo de la raíz cuadrada y en el `else` hacemos la impresión del resultado.

```
def raizCuadrada(numero):
    """
    Función que calcula la raíz cuadrada de un número.

    Parameters
    -----
    numero: int o float
    Valor al que se le desea calcular la raíz cuadrada.

    """

    # Intenta realizar el cálculo que está dentro de try
    try:
        numero_cuadrado = float(numero) ** 0.5

    # En esta sección se trata la excepción de tipo TypeError y se obtienen los c
    except TypeError as info:
        info.add_note("\n" + "-"*20)
        info.add_note(f"raizCuadrada{numero}: Para calcular una raíz cuadrada, el
        info.add_note("-"*20)
        raise # Lanzamos la excepción con toda la información

    # En esta sección se trata la excepción de tipo ValueError y se obtienen los
    except ValueError as info:
        info.add_note("\n" + "-"*20)
        info.add_note(f"raizCuadrada('{numero}')": Para calcular una raíz cuadrada
        info.add_note("-"*20)
        raise # Lanzamos la excepción con toda la información

    # En esta sección se tratan todas las otras posible excepciones
    except:
        print("Ocurrió algo misterioso")
```

```
else:  
    print("La raíz cuadrada del número {} es {}".format(numero, numero_cuadrada))
```

```
raizCuadrada(1)
```

La raíz cuadrada del número 1 es 1.0

```
raizCuadrada(-1)
```

La raíz cuadrada del número -1 es (6.123233995736766e-17+1j)

```
raizCuadrada(1+4j)
```

TypeError: float() argument must be a string or a real number, not 'complex'

```
raizCuadrada("hola")
```

ValueError: could not convert string to float: 'hola'