**Recursos útiles:**

* Archivo **TutorialPython3.pdf**. página 53

**Algunas excepciones incorporadas:**

**Exceptions**: clase raíz para todas las excepciones.

**AttributeError**: se dispara cuando falla una referencia a atributo o una asignación.

**IOError**: Ocurre cuando se intenta abrir un archivo que no existe (entre otras cosas)

**IndexError**: Ocurre cuando no existe un índice sobre una secuencia

**KeyError**: Disparado cuando no existe una llave sobre un mapping (diccionarios)

**NameError**: Disparado cuando un nombre o variable no existe

**SyntaxError**: Disparado cuando el codigo esta mal formado.

**TypeError**: Se dispara cuando una operación o función incorporada es aplicado a un objeto de tipo equivocado.

**ValueError**: Disparada cuando una operación o función incorporada es aplicada a un objeto de tipo correcto pero con el valor inapropiado.

**ZeroDivisionError**: Disparado cuando el segundo argumento de una división es cero.

**Actividades**:

1. Realiza una función, que lea un número y lo devuelva si el número leído es entero. Sólo se saldrá de la función con el número entero leído. En caso de no teclear un valor correcto, mostrar mensaje y seguir leyendo (en la función.

Teclea un número: sdfdf

Debe ser número entero

Teclea un número: 6,6

Debe ser número entero

Teclea un número: 6.7

Debe ser número entero

Teclea un número: 6

6

1. Haz una función que lea un número entero y devuelva dicho número si es entero, convertido en entero, y si no es entero (real o string)que devolverá el error de ejecución que se haya provocado. Controlar dicho error en la función principal o que le ha llamado. Seguir pidiendo el número mientra no sea entero (desde la función llamadora)

Teclea un número: asdfasdf

Debe ser un número entero

Teclea un número: 6,7

Debe ser un número entero

Teclea un número: 6.7

Debe ser un número entero

Teclea un número: 6

6

1. Realiza un programa que lea una opción de Altas, Bajas o Modificaciones ("Ingresa A (Altas) - B (Bajas) - M (Modificaciones): "). En caso de teclear un valor diferente se levantará un error de ejecución. Se tratará dicho error en el programa principal (leer hasta correcto). Busca cómo utilizar la instrucción assert para tratar validaciones de usuario como errores de ejecución.

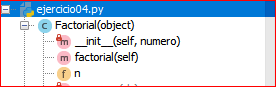
Ingresa A (Altas) - B (Bajas) - M (Modificaciones): h

Opción no es A-B-M

Ingresa A (Altas) - B (Bajas) - M (Modificaciones): a

A

1. Realiza un programa que lea un número y escriba su factorial. Realiza la siguiente clase para resolverlo:



A la clase llegará el número validado (entero y mayor que cero). La validación hay que hacerla en el programa principal. Se capturarán dos errores de ejecución diferentes:

Número Entero >0: -3

El número debe ser mayor o igual a 0

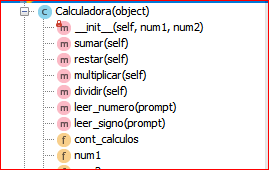
Número Entero >0: 5.6

Debe ser número entero

Número Entero >0: 5

Factorial: 120

1. Realiza un programa que pida un número, un signo (+,-,\*,/) y un segundo número, realice el cálculo y muestre el resultado. Terminar de realizar cálculos cuando en el primer operando teclea un 0. Al final mostrar cuántos cálculos ha realizado. Para ello utiliza la siguiente clase:



Los métodos leer\_numero y leer\_signo, son **métodos estáticos** que leen el dato y validan o leen hasta teclear un valor correcto, número o signo. Se crea el objeto con los datos correctos. Puede darse el caso que como divisor se teclee el 0, en ese caso el método dividir devolverá el error de ejecución, mostramos el error y el programa continúa.

Primer operando (0:Terminar): 3

Signo (+, -, \*, /): -

Segundo operando: 4

Resultado resta: -1.00

Primer operando (0:Terminar): 4

Signo (+, -, \*, /): /

Segundo operando: 0

**divisor es cero**

Primer operando (0:Terminar): f

Primer operando (0:Terminar): 7.4

Signo (+, -, \*, /): aa

Signo (+, -, \*, /): \*

Segundo operando: 1

Resultado multiplicación: 7.40

Primer operando (0:Terminar): 0

Total operaciones: 2.00