

ESTRUCTURAS DE CONTROL BUCLES

Dentro de un bucle `for` o `while`:

- **Break:** para **interrumpir el bucle** en función de una **condición determinada**.
- **Continue:** salta a la siguiente iteración **ignorando el código** que siga a `continue`

Diferencia entre break y continue: con break la ejecución sale inmediatamente del bucle, continue sigue dentro del bucle pero salta a la siguiente iteración.

- **Pass:** no hace **nada**. Marcador de posición para **indicar** que el bloque de **código** en el que se encuentra **aún no se ha implementado**.

Diferencia entre pass y continue: el continue ignora esa línea de código y sigue ejecutando el bucle y el pass ejecuta esa línea de código

.isdigit(): chequea si los valores que tenemos en el **string** **son números o no**.

Control de errores: programación defensiva

En programación se escriben programas que anticipan y manejan posibles errores o problemas que pueden surgir durante la ejecución.

Se usa para crear programas que puedan funcionar en una gran variedad de situaciones.

Try... except: técnica para controlar los errores que podrían ocurrir en un programa y dar solución a esos posibles problemas.

Sintaxis: try: código que puede generar error. except código para manejar el error

Errores:

- `TypeError`: se produce cuando una función o operación se **aplica a un objeto de tipo inapropiado**.
- `ValueError`: se produce cuando una función o método recibe un argumento de tipo correcto pero con un **valor inapropiado**.
- `IndexError`: se produce cuando se intenta **acceder a un índice** que está **fuerza del rango** de una lista o secuencia.

- `KeyError`: se produce cuando se intenta acceder a una **clave** que **no existe** en un diccionario.
- `AttributeError`: se produce cuando se intenta acceder a un **atributo** que **no existe** en un objeto.
- `IOError`: se produce cuando se intenta acceder a un **archivo que no existe o no se puede abrir**.
- `ZeroDivisionError`: se produce cuando se intenta **dividir un número por cero**.
- `ImportError`: se produce cuando **no se puede importar** un módulo.
- `KeyboardInterrupt`: se produce **cuando el usuario interrumpe la ejecución** del programa.

Try... except... except...: lo mismo que try except pero se puede **repetir el except para distintos tipos de error**.

Sintaxis: Try: Código que se desea intentar ejecutar

Except ExceptionType1: Código que se ejecutará si se produce una excepción de tipo ExceptionType1

Except ExceptionType2: Código que se ejecutará si se produce una excepción de tipo ExceptionType2

Except ExceptionTypeN: Código que se ejecutará si se produce una excepción de tipo ExceptionTypeN

Try... except... else...: se ejecuta el bloque try y si no hay excepciones (no se cumple ningún except) se ejecuta el bloque else.

Sintaxis: try: Código que se puede lanzar una excepción

except ExceptionTipo1: Manejo de la excepción Tipo 1

except ExceptionTipo2: Manejo de la excepción Tipo 2

else: Código a ejecutar si no se produce ninguna excepción

` Try... except... else... finally...: finally se ejecuta siempre, haya excepciones o no