Curso Python. Sintaxis Básica I.

Promt: nos marca el punto exacto dentro de la consola donde introduciremos nuestras instrucciones.

# Curso Python. Sintaxis Básica III Funciones

Las funciones son líneas de código que funcionan como unidad realizando una tarea especifica.

Las funciones pueden devolver valores.

Las funciones pueden tener parámetros/argumentos.

A las funciones también se les denomina “métodos” cuando se encuentran definidas dentro de una clase.

Su función es la reutilización de código cuando sea necesario o si es necesario.

Sintaxis:

Def nombre\_funcion(parametros):

Instrucciones de la función

Return(opcional)

Ejecutar una función:

Nombre\_funcion(parametros)

Lanzando excepciones con raise

Raise TypeError(mensaje personalizado)

Puede ser cualquier tipo de excepción (type error contemplado por la biblioteca de python)

TUPLAS

Las tuplas son listas inmutable, es decir, no se pueden modificar después de su creación.

No permiten añadir, eliminar, mover elementos etc (no append, extend, remove)

Si permiten extraer porciones, pero el resultado de la extracción es una tupla nueva.

Si permiten comprobar si un elemento se encuentra en la tupla.

Utilidad o ventaja que tienen respecto a los arreglos

Mas rapidas

Menos espacio (mayor optimizacion)

Formatean strings

Pueden utilizarse como claves en un diccionario. (las listas no)

SINTAXIS:

Sus elementos van entre paréntesis pero es opcional si los lleva o no

Nombrearreglo=(elem1,elem2,elem3)

EXCEPCIONES

Las excepciones son errores que ocurren durante la ejecución del programa. La sintaxis del código es correcta pero durante la ejecución ha ocurrido “algo inesperado”.

Este tipo de errores de ejecución se pueden controlar para que la ejecución del programa continúe. Es lo que se conoce como manejo o control de excepciones.

Sintaxis:

Try:

Código a ejecutar

Except: (descripción del error):

Instrucción a mostrar

Retornar un valor

Finally:

Lo que esta dentro del finally siempre se ejecuta

Un try siempre debe de llevar except o finally

PROGRAMACION ORIENTADA A OBJETOS

Programación orientada a procedimientos

* Ejemplos de lenguajes: Fortran, Cobol, Basic.

Desventajas:

* Unidades de código muy grandes en aplicaciones complejas.
* En aplicaciones complejas el código resultaba difícil de descifrar.
* Poco reutilizable.
* Si existe fallo en alguna línea, es muy probable que el programa caiga.
* Aparición frecuente de código espagueti.
* Difícil de depurar por otros programadores en caso de necesidad o error.

Ventajas:

* Programas divididos en partes, módulos o clases(Modularización).
* Muy reutilizable(Herencia: reutiliza codigo de una aplicación en otra).
* Si existe fallo en alguna línea del código, el programa continuara con su funcionamiento. Tratamiento de excepciones.
* Encapsulamiento.

Programación orientada a objetos

¿En qué consiste?

* Trasladar la naturaleza de los objetos de la vida real al código de programación

¿Cuál es la naturaleza de un objeto de la vida real?

* Los objetos tienen un estado (como se encuentra un objeto), un comportamiento (¿Qué puede hacer?), y unas propiedades.

Ejemplo: El objeto coche

* ¿Cuál es el estado de un coche? Un coche puede estar parado, circulando, estacionado etc.
* ¿Qué propiedades tiene un coche? Un coche tiene un color, un peso, un tamaño etc.
* ¿Qué comportamiento tiene un coche? Un coche puede arrancar, frenar, acelerar, girar etc.

Tiene propiedades(atributos):

* Color
* Peso
* Alto
* Largo

Tiene un comportamiento (¿Qué es capaz de hacer?):

* Arrancar
* Frenar
* Girar
* Acelerar

Vocabulario de la POO

* Clase
* Objeto
* Ejemplar de clase. Instancia de clase.Ejemplarizar una clase. Instanciar una clase.
* Modularizacion
* Encapsulamiento / Encapsulacion.
* Herencia
* Polimorfismo