M04-UF2

Ian Nogueira

XML

En XML es necesario comenzar agregando \<?xml version="1.0" encoding="UTF-8" ?>, esto es una cabecera.
Gracias a esta etiqueta el lenguaje puede funcionar correctamente.
En todo archivo XML es necesario que haya una etiqueta raíz.
La sintaxis básica de XML se basa en etiquetas y elementos. Un elemento se compone de una etiqueta de apertura y una etiqueta de cierre, y puede contener otros elementos, texto u otros tipos de datos.
XML también permite el uso de atributos para proporcionar datos adicionales a los elementos. Los atributos se definen dentro de las etiquetas de apertura de los elementos.
Existen varios tipos de etiquetas:
Pares
Impares
Booleanas

PARES: *Estas etiquetas son las que tienen una etiqueta para abrir y otra para cerrar. EJEMPLO:<name>Eustaquio\</name>*
IMPARES: *Estas etiquetas solo necesitan una etiqueta que cierre. EJEMPLO: <age years="197" />*
BOOLEANAS: *Con esta etiqueta puedes determinar el estado de una variable por defecto, si es true o false. EJEMPLO: <tienelaeso />*

Para que todos los archivos xml sigan una estructura concreta es necesario que haya un archivo que determine dicha estructura. El lenguaje de estas estructuras pueden ser DTD o XSD.

Código básico de XML:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE character SYSTEM "character.dtd">
<character id_character="1">
    <name>Eustaquio</name>
    <surname>Mendoza</surname>
    <age years="197" />
    <race>Enano</race>
    <class>Artificiero</class>
    <gender abbrev="N">Non-Binary</gender>
    <height cm="130" />
    <weight kg="80" />
    <language                                    abbrev="prt"
prefix="+32">Portugues</language>
    <tienelaeso />
    <weapons>
        <weapon id_weapon="1"/>
        <weapon id_weapon="3"/>
        <weapon id_weapon="4"/>
        <weapon id_weapon="7"/>
    </weapons>

</character>
```

XML se puede utilizar como un formato de archivo para almacenar y organizar la información relacionada con los guardados de un juego. Por ejemplo, puedes usar elementos XML para representar información como el nombre del jugador, el progreso en el juego, los logros desbloqueados, entre otros datos relevantes.
XML también sirve, de la misma forma que para los guardados, para almacenar diferentes tipos de datos para distintos tipos de empresas, como bancos o censos. Para esto hay que utilizar librerías especiales con las que crear diccionarios de los datos de dichos archivos XML. Con estos diccionarios luego podemos sacar la información deseada en un script en shell o en python.

DTD

```
<!ELEMNT character (name, surname, age, race,
 class,gender, height,
 weight, language, tienelaeso?, weapons?)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT surname (#PCDATA)>
<!ELEMENT age EMPTY>
<!ELEMENT race (#PCDATA)>
<!ELEMENT class (#PCDATA)>
<!ELEMENT gender (#PCDATA)>
<!ELEMENT height EMPTY>
<!ELEMENT weight EMPTY>
<!ELEMENT language (#PCDATA)>
<!ELEMENT tienelaeso EMPTY>
<!ELEMNT weapons (weapon*)>
<!ELEMENT weapon EMPTY>

<!ATTLIST age years CDATA #REQUIRED>
<!ATTLIST gender abbrev CDATA #REQUIRED>
```

```
<!ATTLIST height cm CDATA #REQUIRED>
<!ATTLIST weight kg CDATA #REQUIRED>
<!ATTLIST language abbrev CDATA #REQUIRED>
<!ATTLIST language prefix CDATA>
<!ATTLIST id_weapon CDATA #REQUIRED>
```

Gracias a esto tenemos un estándar para los archivos de characters en XML.

Los archivos dtd son principalmente las estructuras de archivos XML, como la cantidad de veces que se repite una cosa o su contenido mismo. Gracias a estas estructuras podemos comprobar si un archivo que nos han mandado sigue la estructura predefinida del resto de archivos XML.

HISTORIA XML

Lenguaje base de lenguajes de marcas. Viene de un lenguaje de los años 70 que se llama SGML. Utilizaba etiquetas como la y demás. Word es un texto binario con texto enriquecido. Los lenguajes de marcas tienen como objetivo remarcar palabras. En los años 70 no había gráfica, por lo que se usaba este lenguaje de marcas para printar este tipo de texto. Lo que hacía era colocar estas etiquetas etc para que la impresora parseara estas etiquetas e imprimiera las palabras como se quería.
En los 80 con las gráficas era posible usar interfaces gráficas. HTTP (Hyper Text Transfer Protocol) este protocolo conseguía transformar el texto en tres dimensiones en vez de dos. La tercera dimensión son los enlaces, gracias a esto existe la capacidad de poner enlaces en las palabras. En los 90 SGML se transformó en HTML. Gracias a esto se consiguió enlazar páginas web que estén en servidores HTTP. El primer browser se llamó

Mosaic creado por el mismo que creó HTML. Apareció una empresa llamada NetScape que creó un navegador gratuito que interpretaba lenguaje HTML. IRC es un protocolo de chat que usa twitch. Microsoft compró internet explorer. La expresión de navegar viene de netScape y Microsoft intentó implementar el término surfear. Microsoft intentó imponer Internet Explorer haciendo que sea del propio núcleo y haciendo que el explorador de archivos se vea como una web de internet explorer. Internet Explorer aplicaba mal el lenguaje HTML y como era el navegador predominante todo el mundo programaba páginas webs de la forma que lo interpretaba Internet Explorer. Esto consiguió acabar con todos los navegadores. W3C era el consorcio de navegadores. Al morir netscape liberó el código fuente con el nombre del proyecto mozilla. El consorcio quería delimitar los estándares en las estructuras de las páginas web con html.

Después de html4 se creó XTML, se hizo para determinar cómo eran las etiquetas y para determinar las normas. Con XTML se podrían crear otros lenguajes de marcas, no impone visualización, lo que impone es una estructura.

En los años 99 al 2008 estaba XHTML e hicieron que todo pasará por las normas de XTML.

Gracias a la estructura impuesta comenzaron a demandar a Microsoft ya que no seguía ninguna estructura acordada. Así que Microsoft comenzó a decir que lo intentaría.

PYTHON

Python es un lenguaje de programación interpretado, de alto nivel y de propósito general. Python es ampliamente utilizado en

diversos campos, como desarrollo web, análisis de datos, inteligencia artificial y automatización de tareas.
Python contiene una sintaxis legible, tipado dinámico, estructuras predefinidas, orientada a objetos, muchas librerías con distintos usos, comunidad activa. Python es muy útil en nuestro caso para hacer scripts rápidos con los que por ejemplo, gestionar una base de datos o para hacer scripts con los que atacar o defenderse de posibles intrusos. Python es un lenguaje donde indentar es su razón de existir.

CONDICIONALES

***IF** condicion:*

    Codigo

EJEMPLO:

```
if edad >= 18:
    permiso=True
```

***ELIF** condicion:*

    Codigo

EJEMPLO:

```
elif edad <= 21:
    permiso=True
```

ELSE:

    Codigo

EJEMPLO:

```
else:
    permiso=False
```

BUCLES

FOR *elemento* ***IN*** *cosa que iterar:*

```
    Codigo
```

EJEMPLO:

```
lista= ["Mariano", "Mariana", "Eustaquio", "Eustaquia"]
for nombre in lista:
    print(nombre)
```

Aquí se printarian los nombres de la lista empezando por el primero.

WHILE *condición a cumplir:*

```
    Codigo
```

EJEMPLO:

```
numero=0
while numero < 5:
    print(numero)
    numero+=1
```

FUNCIONES

DEF *nombre****():***

```
    Codigo
```

EJEMPLO:

```
def saludo():
    print("Mamahuevo")
saludo()
```

DEF *suma****():***

```
    Codigo
```

EJEMPLO:

```
def suma(val1, val2):
    return val1+val2
```

```
suma(1, 1)
```

CLASES

CLASS nombre:

```
    Codigo
```

EJEMPLO:

```
class Alarmas:
    def alarma1(self):
        print("DESPIERTA MAMAHUEVO")
alarma()
```

En otra clase:

```
alarmas=Alarmas()
alarmas.alarma1()
```

GENERAL

En python la jerarquía va:
Variables en minúscula y con barras bajas.
Funciones con minúscula y con barras bajas.
Clases capitalizadas.

En las clases hay un __main__. Si ponemos esto fuera de nuestra clase lo que hará será ejecutar lo que haya en ese main si es que el archivo se ha ejecutado de primeras. Si la clase es llamada desde otro archivo no se ejecutará nada de lo que haya en el main. Existe un __init__, este init es el constructor de la clase. En este constructor se ejecutarán las partes del código que son esenciales para el programa y que solamente hacen falta ejecutar una vez. En las funciones siempre hay una

variable llamada self. Esta variable sirve para llamar a los valores de la propia clase que han sido iniciados en el constructor o en la propia función. Indentar mal conlleva a que no funcione nada del código. Las bibliotecas sirven para añadir funcionalidades que tendríamos que programar nosotros pero que otra persona ya ha hecho. Un ejemplo de esto es la librería de xmltodict. Esta librería sirve para leer, escribir, eliminar y muchas otras cosas a archivos xml. Python también tiene una terminal propia que sirve para hacer código rápido si es que es necesario hacer un script justo en el momento. La cabecera es ***#!/usr/bin/python3****. Con esta cabecera no hará falta poner el comando de python3 todo el rato si es que queremos darle permisos de ejecución al archivo.*