PYTHON

El código fuente de Python es el mismo para todos los sistemas operativos. El código compilado es diferente, sin embargo, es específico para el dispositivo diseñado.

Lenguaje interpretado: programas que hacen la interconversión automáticamente. R y Python, por ejemplo. No hace falta indicarle al programa el tipo de variable (número, texto...), él propio lo entiende.

Elementos

* Variables

Todas tienen un nombre específico (identificadores). Siempre dar nombres representativos a las variables.

En Python siempre hay que empezar con letras el nombre de variable.

Operador; por ejemplo, el = es un asignador de 10 a la variable x (x=10).

* Números enteros (int)
* Números fraccionales (float)
* Texto (string)
* Tipos booleanos: verdadero o falso. Palabra clave o reservada.

\*En negro lo que tecleamos, en azul lo que aparece en la pantalla

\*No se puede nombrar una variable igual que un una palabra reservada.

Python puede sumar cadenas de texto: viva + luis = vivaluis

Viva + espacio + Luis = viva Luis

Type () invoca una función; pequeños programas dentro de Python que realiza tareas

En Python 3 las divisiones siempre se interpretan como decimales, aunque sea una división exacta.

Internamente se trabaja en números binarios.

* Operadores

Asignación (=) no es lo mismo que la comparación (==)

Not niega lo que viene entre paréntesis.

Con letras se utiliza también alfabeto numérico: a<b 🡪 True

Operador or con que uno sea verdadero vale.

En código interno True es 1 y False es 0

Input es un operador de asignación y siempre asume que lo que pones entre paréntesis es texto.

In es un operador lógico que devuelve una entrada booleana.

* Funciones y métodos

Miniprogramas que permiten realizar determinadas tareas. Cada función necesita un parámetro distinto para funcionar.

Los métodos son funciones que se aplican a determinadas funciones.

Int es una función que convierte el argumento a un número entero.

Métodos

Upper 🡪 Convertir argumento a mayúsculas

El índice más pequeño es 0; siempre empieza por 0 para contar. Importante para el método “find”. En R empieza por 1.

Listas/vectores/arrays

Una variable que contiene todo el listado de os números que me interesan. Ayuda a almacenar todos los datos que necesito.

Se hace referencia por el orden de introducción.

Diccionarios

Una única variable que contiene todos los datos que nos interesan pero se asigna a cada valor un nombre.

\t 🡪 Tabulador

End=”” 🡪 Hace que Python no salte de línea cuando imprime (ejecuta print)

CONTROL DE FLUJO: USO DE CONDICIONALES

Con el uso de condicionales se pueden introducir bifurcaciones. “If” y a continuación la condición que queremos evaluar, seguido de “:”. El bloque de código se identifica entre los márgenes que metes.

Si queremos que genere una respuesta si la condición es falsa: “Else” + el bloque de código que corresponda + “:”

LOS COMANDOS ELIF SE EJECUTAN DE FORMA SECUENCIAL

BUCLES

Los bucles for s ejecuta n veces (las que indica la cláusula for).

Los bucles while se repite un número de veces, pero no está determinado de antemano. Después del bloque se vuelve a ejecutar whil hasta que sea false. Ojo porque se corre el riesgo de entrar en un bucle infinito.

Alt +tabulador = cambio de pantalla tiki.

Rnge genera una lista, que por defecto, va desde cero hasta el número que pongas entre paréntesis (excluido de ella).

En Python los strings se consideran como listas.

.split (“”) 🡪 Devuelve los caracteres dividos de uno en uno. Otra opción sería list(pepelu) 🡪 p,e,p,e,l,u

Append añade elementos a una lista de forma natural

\n = cambio de línea (retorno de carro)

.csv =comma separated value (extensión de texto)

Readlines vs. Readlines 🡪 Leer todas las líneas a la vez o una cada vez

En el segundo caso hay que utilizar bucles (for o while not)

strip = eliminación del último carácter no visible de la lista (normalmente elimina el \n, también tabulador \t)

print (MyFields [0]) 🡪 sólo muestra el primer carácter de cada lista

cd .. 🡪 volver a la carpeta anterior

../../ 🡪 Volver dos carpetas hacia arriba

Introduciendo módulos podemos añadir funcionalidades adicionales a Python.

FUNCIONES

Funcionan como un programa independiente.

Modulan el programa 🡪 Mejoran su lectura, manejo y reciclaje.

Return (lo que queremos que nos devuelva) y se acaba la ejecución de la función. Sólo es necesario introducirlo cuando no se ha introducido print en la función.

Cada tarea unitaria debe asignarse a una función determinada 🡪 Construcción de un script modular

Asignación por posición.

Importante nombrar diferente a las variables de dentro de la función de las del cuerpo del script (variables locales vs globales). Es una receta para evitar el desastre.

Import sys: esencial para poder ejecutar el comando de la siguiente línea. Permite introducir funciones adicionales.

[sys.argv[1]] 🡪 Comando que permite introducir el argumento al script

El 1 lo ponemos para que tome como argumento el primer elemento después de la invocación¡n al script (que sería el elemnto 0).

try/except 🡪 Evitas que el programa aborte si el comando detrás de try diera error.

Utilizando triple comillas “”””” podemos decirle a Python como funciona una determinada función