FlexSim (v22.1)

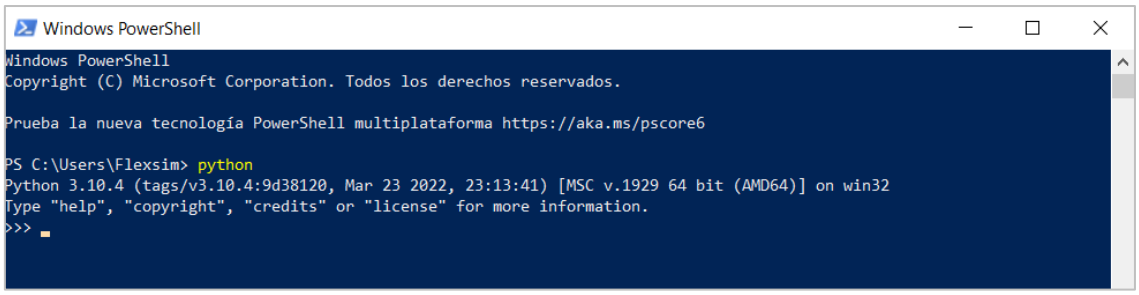
Implementación de algoritmos con Python

El objetivo de este tutorial es mostrar la conexión FlexSim-Python, y entender como aplicarla. En concreto, lo aplicaremos a un algoritmo de pathfinding en almacenes (encontrar la ruta más corta entre una serie de puntos).

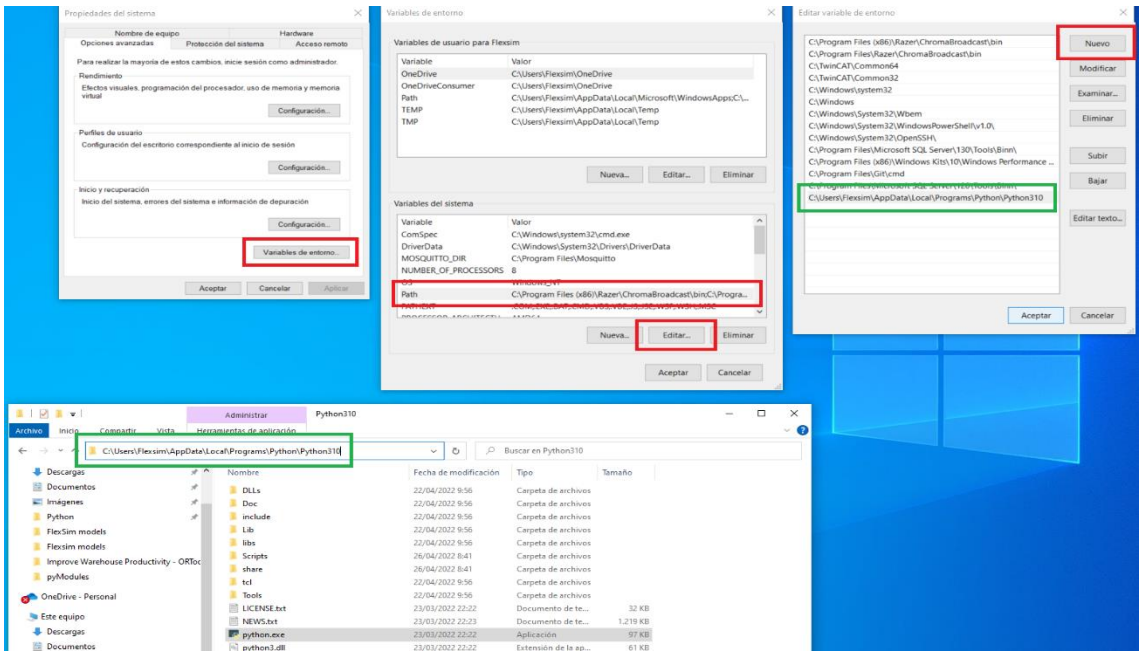
Instalación de programas

(Para evitar problemas desinstalaremos cualquier software "Python" anterior)

Instalaremos python 3.10 con el link "Python.org" o desde la tienda de Microsoft. Para comprobar que Windows lo ha instalado correctamente, abriremos "Windows PowerShell" y escribiremos "python". Si se ha instalado bien, en consola nos aparecerá:



(con la correspondiente versión). **Si no encuentra el programa**, nos debería llevar la tienda de Microsoft. En tal caso buscaremos "Editar las variables de entorno del sistema" y pincharemos sobre "Opciones Avanzadas > Variables de entorno…", se nos abrirá una nueva ventana. Buscamos "Variables del sistema > Path" y le damos a "Editar…", y de nuevo se abre otra ventana. Aquí pinchamos en el botón "Nuevo" y añadimos la ubicación de la carpeta que contiene el archivo "python.exe":



Podemos comporbar, como antes, si PowerShell reconoce el comando “python”. Para editar los scripts “.py” usaremos Visual Studio Code.

Si nos salta algún otro tipo de error, es probable que hayamos instalado python junto con un entorno de programación, o de alguna forma alternativa, y más adelante dé problemas de connectividad con FlexSim. En tal caso, se recomienda desinstalar todo el software python y reinstalar como se ha explicado.

Conexión FlexSim-python

Los primeros pasos consisten en probar la conexión con un modelo simple:

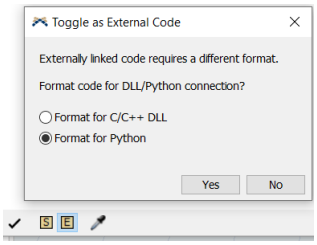
1. Necesitamos un script de Python a modo de test. Debe estar en la misma carpeta que nuestro modelo. Podemos usar el archivo de prueba “**PyMod.py**”, o copiar directamente:

```python
def PyFunc():
    return 20
```

2. Descargamos el archivo “**python-test.fsm**” o construimos el modelo siguiendo los pasos:

   - En FlexSim, para la conectividad con python, vamos a “File > Global Preferences > Code > Python Version” y seleccionamos la versión de python que hayamos instalado, en nuestro caso la 3.10 .

   - A continuación, añadimos un User Command desde “Tools” y lo llamamos “**pyConnect**”. Al editar el código, buscamos la opción de External code, marcada con una “E” encuadrada (si paramos el ratón encima aparece la nota: Make code externally linked). Seleccionamos **Format for Python** y le damos a aceptar:

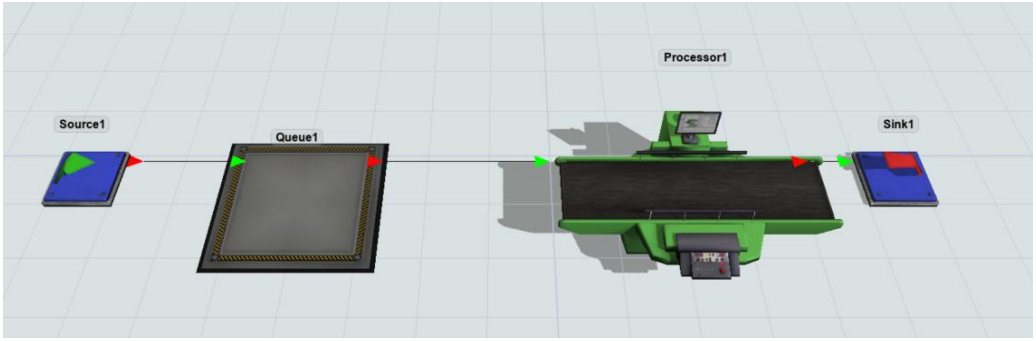


Nos generará un código con dos strings: La primera (`"PyMod"`) es el nombre del script de python sin la extensión “.py” y la segunda (`"PyFunc"`) el nombre de la función que hemos programado (Obviamente los nombres son completamente arbitrarios).

```
/**external python: */ /**/"PyMod"/**/
/** \nfunction name:*/ /**/"PyFunc"/**/
```

Este código puede modificarse, pero necesita las palabras “external python” al principio.

- Para comprobar que python devuelve el número 20 con éxito, construimos en el 3d:



- En "*Processor1 > Processor > Process Time*" abrimos el editor de código y copiamos:

```
Object current = ownerobject(c);
Object item = param(1);
double process_time = pyConnect();
print("Return from Python:", process_time);
return /**/process_time/**direct*/;
```

3. Ejecutamos el modelo. El processor1 debería tardar 20 segundos en procesar cada caja si todo está bien.

Implementación del algoritmo de pathfinding

La extrapolación a algoritmos más complejos es fácil con este procedimiento. De modo que podemos tener un algoritmo muy complejo y ejecutarlo con una línea desde FlexSim.

Problema

Nuestro problema consiste en optimizar el tiempo que está un operario o AGV viajando en un almacén. Es decir, encontrar el camino más corto entre una serie de picks, limitado por los obstáculos físicos que no nos dejan cruzar en línea recta (Racks).

Algoritmo

Esta limitación física se añade al algoritmo desde FlexSim. De modo que necesitamos una **función de cálculo de distancia** entre dos puntos. Esta dependerá de la distribución particular del almacén y será equivalente a la distancia más corta andando.

Además, desde FlexSim construimos la **matriz de distancias** que debemos darle como input al algoritmo. Esta se construye con ayuda de la función anterior de la siguiente manera:

> A partir de un conjunto ordenado de puntos (picks) de una lista. Cada posición de la matriz viene dada por las coordenadas [fila, columna] = [inicio, fin]. Y su valor es la distancia entre el punto de la lista con el índice *inicio* y el punto de la lista con el índice *fin*.

Con el siguiente ejemplo se ve más claro:

Tenemos tres puntos en una lista [a, b, c], con sus correspondientes posiciones físicas (x,y). De modo que el índice 1 de la lista corresponde al punto “a”, el 2 al “b” y el 3 al “c”.

filas \ columnas	1	2	3
1	0	55	33
2	55	0	44
3	33	44	0

La tabla nos indica las distancias entre los elementos de las filas y los de las columnas, de modo que tiene sentido que en la diagonal haya ceros y que la matriz sea simétrica. La matriz en sí es solo la parte coloreada de verde.

Esto es todo lo que necesita el algoritmo para darnos una solución. El output de dicho algoritmo son los nodos ordenados con el camino más corto. En el caso de ejemplo, el algoritmo devuelve `[0,2,1]` (Python empieza a contar en el `0`, a diferencia de C++).

Además, cabe decir que el algoritmo va a colocar siempre el nodo 0 al principio y el 1 al final (esto es fácil de cambiar mirando el código). De modo que la solución dada para el ejemplo solo sirve como comprobante de que dicha condición de inicio y fin está siendo respetada. Ahora quedaría transmitir la información del orden de la ruta a los operarios, pero eso forma parte de programación en FlexSim.

Conclusiones

De modo que, con este ingrediente y una buena programación de nuestro modelo, podemos conseguir una implementación relativamente sencilla de una aplicación muy potente. Aun así, esto es solo una prueba de concepto, y el caso real conlleva problemas no tratados por el algoritmo, como el tráfico, la estabilidad y tamaño de los ítems, etc.

Este tutorial viene con un caso de estudio y el algoritmo de Python.