Práctica 1 - Introducción a Python
Ingeniería de Software 2024-2

Erick Martínez Piza
ermarp@ciencias.unam.mx

Rogelio Alcantar Arenas
rogelio-aa@ciencias.unam.mx

Luis Fernando Yang Fong Baeza
fernandofong@ciencias.unam.mx

Valeria García Landa
vale.garcia.landa@gmail.com

Francisco Valdés Souto
fvaldes@ciencias.unam.mx

8 de Enero 2024

1. Introducción

El motivo de esta práctica es para que el alumno aprenda el uso correcto de Python en un entorno muy parecido a desarrollo profesional, así como distintas bibliotecas y el uso de entornos virtuales, así como ventajas y desventajas, además de ser una introducción amigable para el alumno partiendo del entendido que se tienen conocimientos o nociones básicas de algún otro lenguaje de programación y fundamentos de estructuras de datos y análisis de algoritmos.

2. Información

Python es un lenguaje de programación interpretado y muy versatil, puesto que cuenta con muchos componentes de distintos paradigmas como el paradigma orientado a objetos, el paradigma imperativo o paradigma declarativo. Por este motivo, su popularidad ha crecido bastante y hoy en día es uno de los lenguajes más usados en la industria y en muchas otras áreas.

Para este curso (incluyendo el desarrollo de los casos de uso) hemos decidido usar este lenguaje ya que permite el uso de entornos virtuales, haciendo los desarrollos mucho más sencillos sin necesidad de tener una gran instalación y/o arquitectura. Un entorno virtual es la creación de un intérprete en una versión específica pero que no cuenta con bibliotecas preinstaladas, es decir, un intérprete sin instalaciones de bibliotecas auxiliares, todo esto sin afectar el intérprete instalado en la memoria física de la computadora, sin importar la versión tampoco, por ejemplo, es posible que en la versión de la computadora se cuente con la versión 3.8.5 del intérprete de Python y en un entorno virtual con la versión 3.9.0. Esto facilita las actividades de desarrollo de cualquier software o sistema, puesto que de esta manera el entorno virtual ahora forma parte del proyecto también y puede ser ejecutado en cualquier computadora.

Python consta de una sintaxis bastante intuitiva que para cualquier persona que cuente con un conocimiento relativamente básico del idioma inglés y conceptos básicos de programación, le sea muy fácil aprender a programar en este lenguaje, sin mencionar que cuenta con todo tipo de bibliotecas para distintos propósitos como puede ser, cómputo científico, cómputo para servidores, cómputo para aplicaciones o hasta incluso ejecutar código en una Raspberry Pi, además de que Python cuenta con el instalador de paquetes conocido como `pip` que facilita la búsqueda e instalación de paquetes, así como el saber qué paquetes ya se encuentran instalados.

En esta práctica se aprenderá el uso del lenguaje Python desde la sintaxis, estructuras de control, clases, objetos y bibliotecas externas, en esta práctica NO se espera el uso de cómputo concurrente ni es acreedor a

puntos extras el implementarlo de esta manera.

3. Desarrollo

La práctica consta de 4 reactivos con un valor de 2.5 puntos cada uno. Para que el alumno sea acreedor de los puntos de cada reactivo, se tienen que cumplir con los requisitos especificados para cada reactivo, dichos reactivos son:

1. Creación de un entorno virtual.
2. Script de sintaxis básica de python.
3. Script de manipulación de objetos y funciones.
4. Script de instalación y utilización de bibliotecas.

3.1. Creación de un entorno virtual

El alumno deberá de crear un entorno virtual, activarlo y usarlo para el resto de los puntos de la práctica, al entregar este entorno virtual, el alumno ya es acreedor a 2.5 puntos. El nombre del entorno virtual es a libre criterio con moderación.

3.2. Script de sintaxis básica de python

Este script debe de ir acompañado con lógica para poder ser acreedor de los 2.5 puntos completos. Este script deberá de emular el comportamiento de un marcador de un partido de tenis con las siguientes reglas: El mejor de 3 sets, los juegos con ventaja.

En caso de que el alumno no conozca las reglas, aquí se explican a continuación. Para que un partido de tenis tenga ganador, siempre se tiene que definir primero el número de sets que se va a jugar, el jugador que haga el mayor número de sets, será el ganador, en cuanto se alcance el número que determina la mayoría de sets, el partido se termina (puesto que no tiene sentido jugar el resto de los sets), por esta misma razón, el número de sets siempre tiene que ser impar puesto que no se permite el empate, por ejemplo, si se dice que el partido es al mejor de 5 sets, en cuanto uno de los jugadores marque 3 sets, se termina, de manera que el marcador en sets puede ser 3 sets a 0, 3 sets a 1 o 3 sets a 2.

Para marcar un set, se tienen que marcar 6 juegos siempre y cuando la diferencia de juegos con el otro jugador sea mayor o igual que dos, el primero en lograr dicho número de juegos, marca el set, por ejemplo, si el marcador es 6 juegos a 4, entonces el jugador con 6 juegos ha marcado un set, sin embargo, si el marcador es 5-5 y cualquier jugador hace un juego, esto no es un set puesto que $|6 - 5| < 2$, entonces para que pueda hacer un set el jugador con 6 juegos, tiene que realizar uno más y quedar 7-5 lo cual también sería un set, pero esto no impide al otro jugador de llevarse el set, sin embargo, tendría que marcar 3 juegos de manera consecutiva para que el marcador sea 6-8 que también cumple con la regla de la diferencia de 2, existe en internet la regla de que si en el set se llega a 6-6, entonces se jugará una muerte súbita pero para propósitos de este script, esta regla deberá de ser ignorada y mantener el conteo como se explicó hasta que se cumpla la regla de diferencia de dos juegos.

Por último, para que un jugador pueda marcar un juego, se tienen que hacer puntos de acuerdo al siguiente conteo:

1. 15
2. 30
3. 40
4. *game* o juego

Esto quiere decir que un jugador debe de hacer un punto por el 15, otro para estar en 30 y un último para alcanzar el 40, si estando en 40 ese mismo jugador hace otro punto, entonces se lleva el juego, como el otro jugador también puede marcar un punto (no al mismo tiempo), puede ser el caso que ambos jugadores estén en 40 y de acuerdo con las reglas, se tienen que hacer dos puntos seguidos para ser ganador del juego, al primer punto de estos dos, se le conoce como *ventaja* (*advantage* en inglés). De manera que un ejemplo de un marcadores válidos podrían ser: 15-0, 30-0, 40-0, 15-15, 30-15, 40-15, 30-30, 40-30, 40-40, Adv. - 40, 40 - Adv., 30-40, 15-40 y 0-40, si cualquiera de los jugadores hace un punto entonces se cumplirá alguno de estos marcadores o el jugador será acreedor a un juego.

Como reglas complemetales, está la regla de saque, esta regla dice que el mismo jugador saca durante todo un juego, hasta que alguien lo gane y después le toca sacar al otro jugador sin importar si el set termina o no, bajo ninguna circunstancia se permite que un mismo jugador saque en dos juegos seguidos aunque sean de distinto set y en segundo lugar, la regla del cambio de cancha, en la cual, siempre se realiza cambio de cancha si el número de juegos jugados en el set es impar, por ejemplo, si el marcador en el set es 4-1, $4 + 1 = 5$ que es impar, entonces se realiza cambio de cancha, si el set es 4-0, $4 + 0 = 4$, no es impar, entonces se debe de jugar el juego 5 en la misma cancha, aquí el set sí influye, puesto que si el set termina 6-3, entonces corresponde al cambio de cancha y cuando sea 1-0 del siguiente set, se tiene que hacer el cambio de cancha de nuevo.

Todas estas reglas deben de ser implementadas en el marcador que lleva el script, con el propósito de trabajar la lógica y las estructuras de control utilizando Python, así como el manejo de entradas y operaciones básicas. La entrada del programa es a libre formato pero debe de ser especificado dicho formato en un archivo `readme_script1`, la salida debe de ser la salida estándar (Terminal), siempre y cuando se cumpla que: El usuario pueda ingresar los nombres de los dos jugadores y el script especifique quién es el ganador del punto. El nombre de este script deberá de ser `Script1.py`

Se realiza el desglose de requisitos para este script, mismo que se utilizará para calificar posteriormente la práctica:

- Ejecución sin errores (Funcionamiento correcto al momento de ingresar la entrada tal cual se especificó en el readme) = 1.25pts
- Lógica correcta en el conteo del marcador = 1 pt
- Especificación correcta del cambio de cancha y jugador que saque. (Para el inicio del saque, se puede tomar un jugador al azar pero después se debe de alternar correctamente) = 0.25pts

Por último, para este script también se tiene que incluir al menos una estructura de control `try-except` para comprender su funcionamiento para hacer el script resistente a fallas (mala entrada del usuario), dando la capacidad de reingresar a dónde el usuario cometió un error.

3.3. Estructuras de datos y objetos

Python como cualquier lenguaje cuenta con estructuras de datos integradas y además permite la creación de las mismas a manera de objetos, así como también la creación de funciones auxiliares que se necesiten sin necesariamente pertenecer a una clase, emulando el comportamiento de un lenguaje imperativo (Como C/C++).

Para esto, se le ha solicitado al alumno la creación de un script en el cual se creen dos cosas: Una función que resuelva un problema (descrito a continuación) con los parámetros del enunciado del problema como entrada y un árbol binario ordenado a manera de clase de python.

Desglosando primero el enunciado del problema para la función es; Un caminante ha decidido salir a una montaña aleatoria a dar una vuelta y quiere contar sus montañas y valles en su recorrido, una montaña es cuando el recorrido lo lleva menos un paso sobre el nivel del mar y hasta que regrese al mismo, el valle es análogo pero cuando se hace debajo del nivel del mar, de manera que se puede suponer que la caminata empieza en el nivel del mar. Dada una lista conformada únicamente por letras Ú'y 'D'representando a Ú'como paso hacia arriba y 'D'paso hacia abajo, se debe de regresar el número de valles que se recorrió. Solo se cuenta con la lista o la cadena de entrada, no se puede contar con que la función reciba la longitud del recorrido y este problema

tendrá 3 posibles calificaciones, 0, 1.25 o 0.625, 0 si no funciona en ningún caso, 1.25 si funciona en todos los casos o 0.625 si falla en al menos un caso de prueba, obviamente estos casos de prueba son ocultos para los alumnos.

En el caso del árbol binario ordenado (y para los alumnos que desconozcan esta estructura de datos), un árbol binario es una estructura recursiva definida por `nil` o un elemento con un subárbol izquierdo y un subárbol derecho. Esta definición, aunque es correcta, no es tan tangible para una implementación, de manera que otra manera de hacerlo es mediante un objeto `nodo` que conste de un elemento y un nodo izquierdo y un nodo derecho, así como un nodo padre para poder subir y bajar por el árbol y tener una referencia al nodo raíz o el primer nodo del árbol, sin embargo, la versatilidad de Python permite tener las 2 implementaciones a diferencia de otros lenguajes como por ejemplo Java.

Ahora, un árbol binario ordenado es un árbol binario cuyos elementos cumplen que; para cualquier elemento en el árbol, todos los elementos en su subárbol izquierdo son menores o iguales que él y los elementos en su subárbol derecho son mayores que él, de manera que cualquiera de las dos implementaciones que se escoja, se tiene que seguir esta regla.

Al estar tratando con un objeto, se espera que para el caso del árbol binario ordenado el comportamiento mínimo es el siguiente:

- Agregar elementos respetando la propiedad de árboles binarios ordenados.
- Recorrido pre-orden (raíz, izquierdo, derecho) regresando una lista con los elementos.
- Recorrido in-orden (derecho, raíz, izquierdo) regresando una lista con los elementos.
- Recorrido post-orden (derecho, izquierdo, raíz) regresando una lista con los elementos.

Valiendo exactamente $\frac{1,25}{4}$ cada inciso. Para el caso de los recorridos se debe de regresar una lista con todos los elementos en el árbol y en el orden correcto. El nombre de este script deberá de ser `Script2.py`.

3.4. Importación de bibliotecas externas

Utilizando el instalador de paquetes `pip`, deberán de instalar la biblioteca de matplotlib y crear un script que utilice esta biblioteca para graficar una función de la elección del alumno, puede ser cualquier tipo de función pero es responsabilidad del alumno, aprender como es que funciona la sintaxis de matplotlib.

Para ser acreedor a los 2.5 puntos, la gráfica debe de poder ser ejecutada sin problemas cuando el ambiente esté activo, no será válido argumentar que en la computadora personal funcionó de manera correcta puesto que es el propósito de dicha práctica, en caso de que esto sea completamente cierto y el entorno de quien califique está fallando, eso será deliberado por los ayudantes. El nombre de este script deberá de ser `Script3.py`.

4. Formato de entrega

La entrega es individual por cada alumno, no es válido entregar las prácticas en parejas, tercias ni cualquier cantidad mayor de integrantes.

Para que esta práctica se considere como entregada, tiene que realizarse un fork al repositorio oficial del curso, (Leer práctica 0) y realizar un fork al repositorio oficial, crear una nueva rama cuyo nombre solo tiene que empezar con `1-`, por ejemplo `1-practica-intro-python`, en esta rama se tiene que trabajar utilizando `git` y hacer `commit/push` una vez que se haya terminado la práctica.

Dentro de la rama del repositorio también debe de estar el entorno virtual con las bibliotecas instaladas, de manera que si una persona hace un `git clone` de su repositorio, activando el ambiente virtual, sea capaz de ejecutar el código sin problemas.

La entrega es para el 15 de Febrero, antes de las 23:59:59.