Tutorial 04/05/2018

Listas y strings

Introducción

En este tutorial revisaremos algunos conceptos clave para comprender el uso de listas y strings y el acceso a archivos usando Python, y luego veremos algunos ejercicios prácticos.

Revisión de Conceptos

1. ¿Que es una sublista en Python?
2. ¿Qué pasa con una lista si se modifica un elemento en una sublista creada a partir de ella con un rango especificado con `[i:j]`, suponiendo `i`, `j` índices válidos de la lista?
3. ¿Cómo se determina el largo de una lista que está dentro de otra?
4. ¿Como se puede leer un elemento que está dentro de otra?
5. ¿Cuántos niveles o dimensiones de sub listas se pueden tener?
6. Tener dos listas anidadas sirve para generar matrices NxM. Sin embargo, ¿Se pueden crear listas anidadas de otras formas o geometrías?
7. ¿Qué diferencia hay entre un string y una *lista de caracteres* en Python?

Problema 1

Primero comenzaremos con un pequeño problema introductorio sobre listas anidadas. Cree una función llamada $multiplicacion\_por\_escalar(escalar, matriz)$, la cual recibe como parámetros un número entero llamado escalar y una matriz representada por una listas de listas. Esta función retorna una lista de listas del mismo tamaño que la matriz original, en donde cada elemento está multiplicado por el escalar. Guíese por el ejemplo:

```
In [1]: m = [[0,1,4],[2,2,3]]

In [2]: multiplicacion_por_escalar(3,m)
Out[2]: [[0, 3, 12], [6, 6, 9]]

In [3]: multiplicacion_por_escalar(-2,m)
Out[3]: [[0, -2, -8], [-4, -4, -6]]
```

Problema 2

Desarrolle una función llamada $encontrar\_substring(string, substring)$, la cual busca un string dentro de otro y entrega una lista con los índices donde comienza dicho string. Veamos unos ejemplos:

```
In [1]: encontrar_substring("odio los mensajes de odio","odio")
Out[1]: [0, 21]

In [2]: encontrar_substring("deberían censurar la censura","censura")
Out[2]: [9, 21]
```

Problema 3

Desarrolle una función llamada $censurar(mensaje, palabra)$, la cual censura la palabra pasada como parámetro dentro del mensaje, colocando asteriscos en vez de ella. Notar que la cantidad de asteriscos es la misma que la longitud de la palabra. A continuación veamos un ejemplo:

```
In [1]: censurar("odio los mensajes de odio","odio")
Out[1]: '**** los mensajes de ****'

In [2]: censurar("deberían censurar la censura","censura")
Out[2]: 'deberían *******r la *******'
```

Problema 4

El juego del combate naval consiste en un juego donde cada contrincante tiene un tablero, originalmente de 10×10 y 5 barcos, los cuales pueden ser portaaviones, acorazado, submarino, crucero y destructor. Cada barco tiene un tamaño predefinido y las piezas deben ser ubicadas en el tablero antes de comenzar una partida. Sabemos que el portaaviones tiene tamaño 5, el acorazado tamaño 4, el submarino tamaño 3, el crucero tamaño 3 y el destructor tamaño 2.
En el presente problema iremos construyendo por partes una aplicación que permita inicializar un tablero de $n \times n$ de combate naval para m barcos.
El código de esta aplicación es el siguiente:

Listing 1: Código del programa principal

```
1  tipos = [["portaaviones",5],["acorazado",4],["crucero",3],["submarino",3],["destructor",2]]
2  d = int(input("Ingrese el tamaño del tablero:"))
3  n = int(input("Ingrese cantidad de barcos:"))
4  t = crear_tablero(d)
5  k = 1
6  while k <= n:
7      flag = True
8      while flag:
9          print("----------------------")
10         print("Datos barco Nro "+str(k))
11         print("----------------------")
12         tipo = input("Ingrese tipo del barco:")
13         o = input("Ingrese orientación [h/v]:")
14         i = int(input("Ingrese fila:"))
15         j = int(input("Ingrese columna:"))
16         if cabe_barco(tipo,o,i,j,t) and not colisiona(tipo,o,i,j,t):
17             t = ingresar_barco(tipo, o, i, j, t)
18             flag = False
19         else:
20             print("No se ha podido ingresar el barco")
21     k += 1
22 mostrar_tablero(t)
```

Problema 4 continúa en siguiente página...

A continuación se muestra un ejemplo de ejecución de esta aplicación:

```
Ingrese el tamaño del tablero:10
Ingrese cantidad de barcos:5
----------------------
Datos barco Nro 1
----------------------
Ingrese tipo del barco:portaaviones
Ingrese orientación [h/v]:h
Ingrese fila:0
Ingrese columna:0
----------------------
Datos barco Nro 2
----------------------
Ingrese tipo del barco:crucero
Ingrese orientación [h/v]:v
Ingrese fila:0
Ingrese columna:7
----------------------
Datos barco Nro 3
----------------------
Ingrese tipo del barco:submarino
Ingrese orientación [h/v]:h
Ingrese fila:8
Ingrese columna:8
No se ha podido ingresar el barco
----------------------
Datos barco Nro 3
----------------------
Ingrese tipo del barco:submarino
Ingrese orientación [h/v]:h
Ingrese fila:8
Ingrese columna:5
----------------------
Datos barco Nro 4
----------------------
Ingrese tipo del barco:destructor
Ingrese orientación [h/v]:v
Ingrese fila:1
Ingrese columna:1
----------------------
Datos barco Nro 5
----------------------
Ingrese tipo del barco:acorazado
Ingrese orientación [h/v]:h
Ingrese fila:4
Ingrese columna:4
X X X X X     X
  X           X
  X           X
```

```
X X X X


  X X X
```

Para que este código funcione deberá implementar las funciones que se detallan a continuación:

- **crear_tablero(tamaño)**: función que recibe un parámetro denominado tamaño, el cual indica las dimensiones del tablero. La función retorna una lista de listas, en donde cada lista tiene como longitud el tamaño indicado. Cada elemento de las sublistas serán ceros. A continuación un ejemplo de ejecución:

```
>>> crear_tablero(3)
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
>>> crear_tablero(5)
[[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]
>>> crear_tablero(10)
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
```

- **mostrar_tablero(tablero)**: función que recibe como parámetro una lista de listas, como la generada por la función anterior. Cada vez que se lea en la sublista un 0, la función imprime dos espacios, mientras que si lee un 1, imprime una X y un espacio. A continuación un ejemplo de ejecución:

```
>>> t = crear_tablero(3)
>>> t[0][1] = 1
>>> t[0][2] = 1
>>> t[1][2] = 1
>>> t[2][2] = 1
>>> t[2][0] = 1
>>> mostrar_tablero(t)
  X X
    X
X   X
```

- **ingresar_barco(tipo,orientacion,i,j,tablero)**: función que recibe cinco parámetros y permite ingresar un barco en el tablero. El tipo representa el tipo de barco. La orientación puede ser h si el barco está horizontal y v si el barco está vertical. Los valores enteros i y j representan el cuadrante inicial dónde irá el barco dentro del tablero. A continuación unos ejemplos de ejecución:

```
>>> tipos = [["portaaviones",5],["acorazado",4],["crucero",3],["submarino",3],["destructor",2]]
>>> t = crear_tablero(5)
>>> t = ingresar_barco("portaaviones","h",0,0,t)
>>> mostrar_tablero(t)
X X X X X
```

```
>>> t = ingresar_barco("acorazado","v",1,0,t)
>>> mostrar_tablero(t)
X X X X X
X
X
X
X
```

- **cabe_barco(tipo,orientacion,i,j,tablero)**: función que dado los atributos anteriormente explicados, determine si un barco cabe o no dentro del tablero (esto es, si no se escapa del tamaño predefinido). Si cabe, la función retorna True, y si no, retorna False. A continuación un ejemplo de su utilización:

```
>>> t = crear_tablero(5)
>>> cabe_barco("portaaviones","h",4,0,t)
True
>>> cabe_barco("portaaviones","v",4,0,t)
False
>>> cabe_barco("portaaviones","h",0,3,t)
False
>>> cabe_barco("portaaviones","h",0,0,t)
True
```

- **colisiona(tipo,orientacion,i,j,tablero)**: función que, dado los parámetros explicados anteriormente, determina si un barco colisiona con los demás barcos que ya se encuentran en el tablero. La función retorna True en caso de que el barco colisione con otro, y False en caso contrario.A continuación un ejemplo de su utilización:

```
>>> t = crear_tablero(5)
>>> t = ingresar_barco("portaaviones","h",0,0,t)
>>> colisiona("acorazado","v",1,0,t)
False
>>> t = ingresar_barco("acorazado","v",1,0,t)
>>> mostrar_tablero(t)
X X X X X
X
X
X
X
>>> colisiona("portaaviones","h",2,0,t)
True
>>> colisiona("acorazado","h",2,1,t)
False
>>> t = ingresar_barco("acorazado","h",2,1,t)
>>> mostrar_tablero(t)
X X X X X
X
```

```
X X X X X
X
X
```