Seminario de lenguajes
Opción Python

Práctica 4 - 2018

Excepciones

1. Dada la lista de colores del Ejercicio 1 de la Práctica 2, realice una función que ingrese coordenadas por teclado y retorne el color asociado al mismo. Maneje con excepciones si las coordenadas ingresadas no existen en la lista, informándolo y requiriendo que las ingrese nuevamente.

2. Dado el archivo utilizado con datos de jugadores del Ejercicio 3 de la Práctica 3, implemente una función que reciba el nombre del archivo como parámetro y maneje con excepciones el caso que el archivo no exista, informando dicha situación.

3. Una manera de evitar problemas con la versión de Python para el ingreso de datos por teclado es definiendo la siguiente función:

```
def ingresar(texto):
      import sys
      if sys.version > '3':
            return input(texto)
      else:
            return raw_input(texto)
```

   ¿Cómo haría para tener el mismo resultado con manejo de excepciones y sin usar una función aparte?

4. Lea de un archivo (miarchivo.txt) números, maneje a través de excepciones la posibilidad de que no exista el archivo y también que los datos leídos no sean números.

5. Defina dos funciones que recibe una cantidad variable de argumentos, la función sumar, puede llegar a recibir hasta 30 números como parámetros (*args) y debe devolver la suma total de los mismos. La otra función que debe definirse, recibe parámetros con nombre(**kwargs), se deberá imprimir los datos que reciba(usar excepciones para determinar qué campos faltan como parámetros). De antemano no se sabe cuáles de los tres **posibles** parámetros se reciben:

   - nombre
   - apellido
   - sexo

Programación Orientada Objetos

6. Implemente una clase Punto que represente una coordenada (x, y). Debe poseer los siguientes métodos:

   - **calcularDistancia()**: Recibe como parámetro otro punto y retorna la distancia entre ambos.

- **esIgual()**: Recibe como parámetro otro punto y retorna si ambos puntos son iguales.
- **sumarX()**: Recibe como parámetro un número entero, y lo suma a la coordenada x, retornando el nuevo punto generado.
- **sumarY()**: Recibe como parámetro un número entero, y lo suma a la coordenada y, retornando el nuevo punto generado.

7. Modelice las clases Puntaje y Jugador. La clase Puntaje tendrá como atributos nivel, puntos y tiempo. La clase Jugador tendrá como información su nombre y una lista de Puntajes, con los siguientes métodos:

   - **nivelMaximo()**: Retorna el nivel máximo alcanzado por el jugador.
   - **cantidadTotalPuntajes()**: Retorna la cantidad de puntajes almacenados del jugador.
   - **puntajeMaximo()**: Recibe como parámetro un nivel y retorna el máximo puntaje del jugador para ese nivel.
   - **menorTiempo()**: Recibe como parámetro un nivel y retorna el menor tiempo de juego del jugador para ese nivel.

8. En base al Ejercicio 6 de la Práctica 3, implemente una clase Exportar que permita exportar archivos de un formato a otro (CSV a JSON y viceversa). Implemente los métodos **setearFormatos()**, que setea los formatos de entrada y salida en base a una tupla recibida como parámetro, y el método **exportar()**, que recibe un archivo con el formato de entrada y retorna el mismo con el formato de salida.