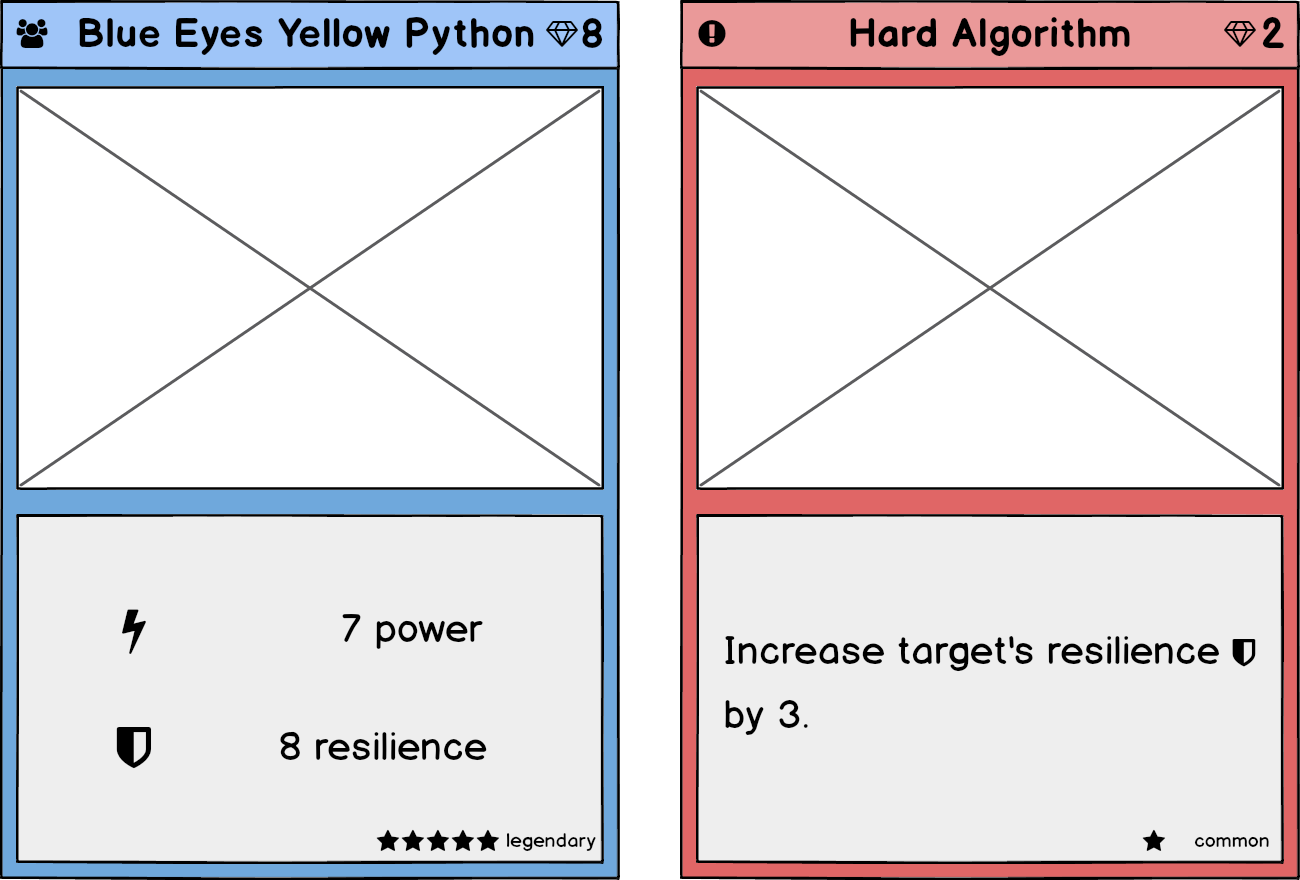
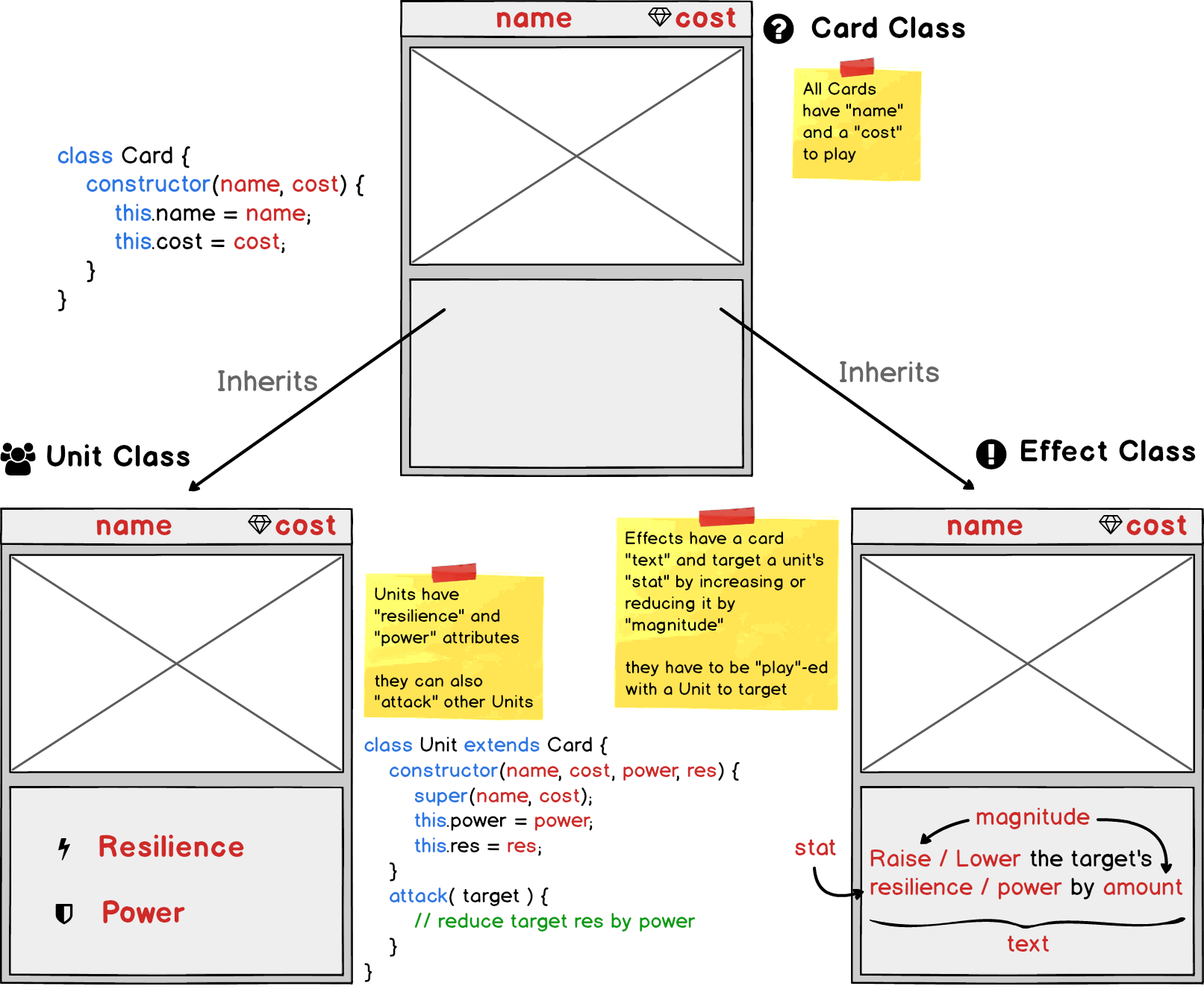
# ENUNCIADO HACKATHON 2021 FULLSTACK PYTHON

## Introducción

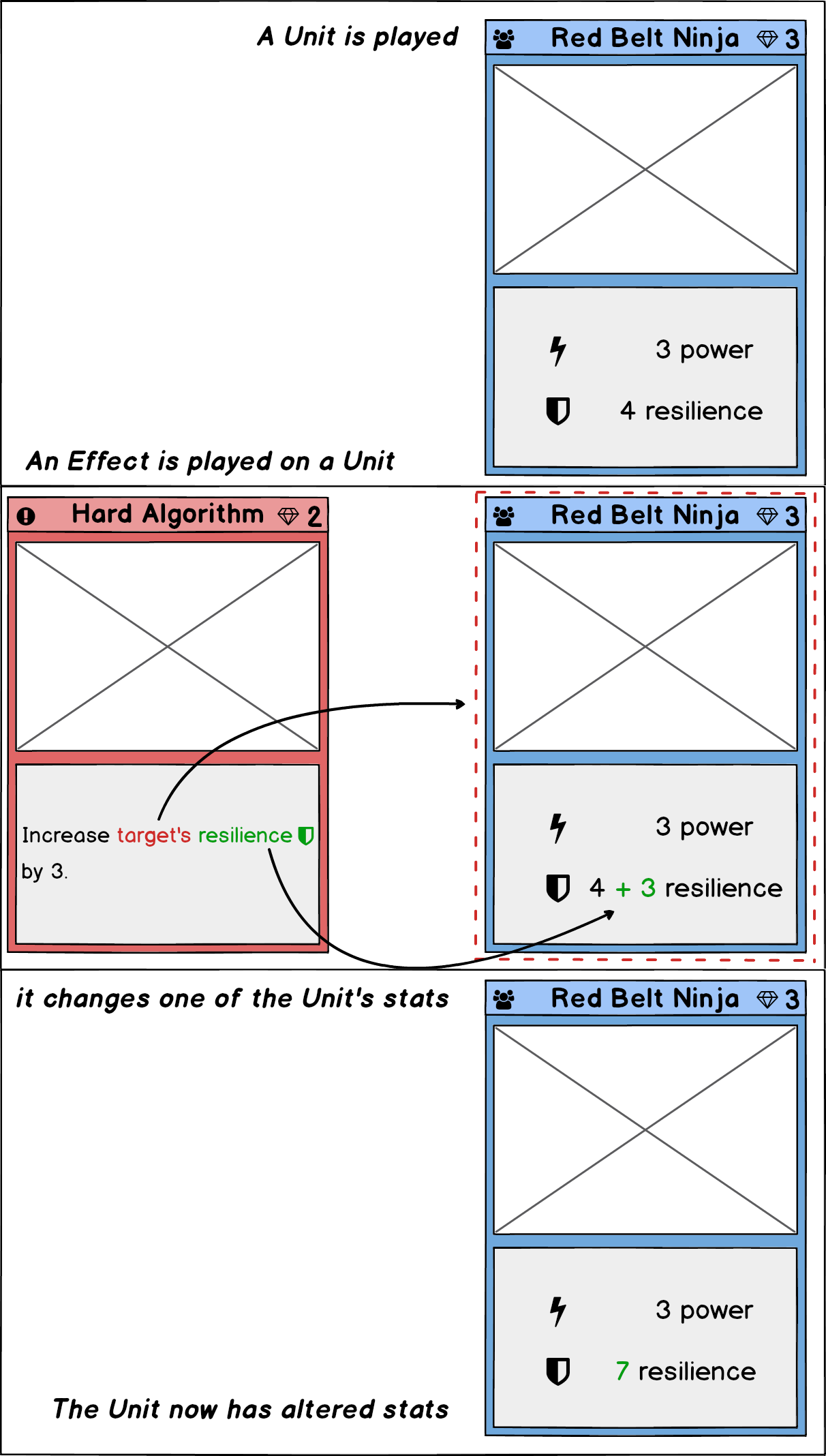
En este desafío diseñarás un juego de cartas coleccionables al estilo de Hearthstone, Yu-Gi-Oh !, o Magic: The Gathering. Si no estás familiarizado con este tipo de juegos, no te preocupes, te explicaremos lo que necesitas saber sobre ellos.



En este juego habrá dos tipos diferentes de Cartas: **Unidades y Efectos**. Las unidades se jugarán para obtener el control del tablero y lucharán con las unidades jugadas por un oponente. Los efectos requieren que una Unidad sea apuntada para poder jugar.



Los efectos requerirán un "objetivo" cuando se juegan. Aumentan o disminuyen el poder o la resistencia de la "Unidad" a la que apuntan. Del mismo modo, las "Unidades" pueden atacar a otras "Unidades", cuando lo hacen disminuyen la "resistencia" del objetivo por el "poder" del atacante.



Si queremos asegurarnos de que el objetivo del ataque de una Unidad o un Efecto se esté jugando en una Unidad, podemos usar instance para verificar que el objetivo sea una "Unidad". Si no es así, es posible que queramos lanzar un error para alertar al usuario o al programador de que la acción no podrá continuar.

def play( target ) {

    if( target instance Unit ) {

        # implementa el texto de carta aquí

    } else {

        raise Error( "Target must be a unit!" );

    }

}

## Especificación de las Cartas

#### Tarjetas de unidad

|  |  |  |  |
| --- | --- | --- | --- |
| **nombre** | **costo** | **poder** | **Resiliencia** |
| Ninja Cinturón Rojo | 3 | 3 | 4 |
| Ninja Cinturón Negro | 4 | 5 | 4 |

#### Cartas de efectos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **nombre** | **costo** | **texto** | **stat** | **magnitud** |
| Algoritmo Difícil | 2 | aumentar la resistencia del objetivo en 3 | Resiliencia | +3 |
| Rechazo de promesa no manejado | 1 | reducir la resistencia del objetivo en 2 | Resiliencia | -2 |
| Programación en pareja | 3 | aumentar el poder del objetivo en 2 | poder | +2 |

#### 

#### Juega el siguiente escenario

|  |  |
| --- | --- |
| **turno** | **acción** |
| 1 | El jugador 1 convoca a "Ninja Cinturón Rojo" |
| 1 | El jugador 1 juega "Algoritmo duro" en "Ninja Cinturón Rojo" |
| 2 | El jugador 2 convoca a "Ninja Cinturón Negro" |
| 2 | El jugador 2 juega "Rechazo de promesa no controlada" en "Ninja Cinturón Rojo" |
| 3 | El jugador 1 juega "Programación en pareja" en "Ninja Cinturón Rojo" |
| 3 | El jugador 1 tiene el ataque "Ninja Cinturón Rojo" "Ninja Cinturón Negro" |

## Objetivo

Debes recrear el escenario descrito anteriormente en HTML, usando la librería Jinja2. Puedes usar los archivos “server.py” y “templates/index.html” como moldes para poder iniciar tu proyecto. Se recomienda leer brevemente la documentación de Jinja2.

Debes crear las clases y subclases que consideres necesarias. Recuerda primero activar el ambiente virtual antes de comenzar ☺.

Mucha suerte!!!!

## BONUS

Puede usar los requests y response de flask para ir ejecutando las jugadas una a una.