Desc.

Uso de funciones y buenas prácticas en código.

Sintaxis

```
def myFunction(params)
        print("Hello")
# Llamada
myFucntion()
```

Sin valor de retorno

La función ejecuta un bloque de código pero no devuelve ningún tipo de dato.

Con valor de retorno

Haremos uso de la palabra clave **return** para retornar contenido generado en la función.

```
def conReturnFunction(params)
        return params

num = 10
conReturnFunction(num)
```

Manejo de errores

Pass

indica que no se realizará ninguna acción en un bloque de código.

```
try:
    # Código que puede lanzar una excepción
except ValueError:
    # Maneja la excepción
    pass
```

try catch | except

Controlador de errores, es utilizado siempre que estemos operando con datos problemáticos o sensibles:

- **try:** Encierra un bloque de código donde espera que ocurra una excepción.
- **except:** En este bloque se manejan los errores capturados especificando que tipo de excepciones a manejar y como operar con esta.

```
try:
    # Código que puede lanzar una excepción
    resultado = 10 / 0
except ZeroDivisionError:
    # Código para manejar la excepción ZeroDivisionError
    print("No se puede dividir por cero.")
```

Calculadora puntos de experiencia

Los enemigos poseen de varias dificultades, haciendo uso de funciones deberemos ser capaces de averiguar cual es la experiencia ganada cada vez que nos enfrentemos de uno en uno teniendo en cuenta si dificultad y nuestro nivel.

Procedimiento

- Uso de función **calcular_experiencia**
- Dificultades:
  - fácil
  - media
  - difícil
  - nightmare
- Uso múltiple de la función

Ejercicio

```
# definir la función


# Asignar nivel usuario y uso de función

# Mostrado por pantalla resultado
```

Gestión de inventario

Nuestro jugador dispondrá de un inventario limitado con capacidad de 15 items, el usuario al principio dispone de 5 items e irá encontrando diferentes items, el jugador decidirá si cambia el item encontrado por uno a su elección de su inventario, los items tendrán el siguiente esquema:

- Calidad
- Tipo

Ejercicio

```
# Creación de funciones

# Creación del inventario del jugador

# Cambio de items

# Mostramos el inventario
```

Resultado

Ejercicio 1

```
def calcular_experiencia(nivel_jugador, dificultad_enemigo):
    # Definir la cantidad base de puntos de experiencia
    puntos_base = 100

    # Ajustar la cantidad de puntos de experiencia según la dificultad
    if dificultad_enemigo == "fácil":
        puntos_extra = 50
    elif dificultad_enemigo == "media":
        puntos_extra = 100
    elif dificultad_enemigo == "difícil":
        puntos_extra = 150
    elif dificultad_enemigo == "nightmare":
        puntos_extra = 200
    else:
```

```
        print("Dificultad no reconocida.")
        return 0

    # Calcular la experiencia total
    experiencia_total = puntos_base + puntos_extra

    return experiencia_total

# Ejemplo de uso
nivel = 10
dificultad = "nightmare"
experiencia_ganada = calcular_experiencia(nivel, dificultad)
print(f"El jugador de nivel {nivel} ganó {experiencia_ganada} puntos de experiencia
al derrotar a un enemigo de dificultad {dificultad}.")
```

Ejercicio 2

```
def mostrar_inventario(inventario):
    print("Inventario actual:")
    for i, item in enumerate(inventario, start=1):
        calidad, tipo = inventario[item]
        print(f"{i}. {item}: Calidad {calidad}, Tipo {tipo}")

def actualizar_inventario(inventario, item_nuevo, calidad_nuevo, tipo_nuevo):
    if len(inventario) < 15:
        inventario[item_nuevo] = (calidad_nuevo, tipo_nuevo)
        print(f"Se ha añadido '{item_nuevo}' al inventario.")
    else:
        print("¡El inventario está lleno! No se puede añadir más items.")

def cambiar_item(inventario, indice_cambio, item_nuevo, calidad_nuevo, tipo_nuevo):
    if 1 <= indice_cambio <= len(inventario):
        item_antiguo = list(inventario.keys())[indice_cambio - 1]
        inventario[item_nuevo] = (calidad_nuevo, tipo_nuevo)
        del inventario[item_antiguo]
        print(f"Se ha cambiado '{item_antiguo}' por '{item_nuevo}' en el
inventario.")
    else:
        print("Índice de inventario inválido.")

# Inventario inicial del jugador
inventario_jugador = {
    "espada": (80, "arma"),
    "pociones": (10, "consumible"),
    "armadura": (70, "equipo"),
```

```
    "amuleto": (90, "accesorio"),
    "botas": (75, "equipo")
}

print("¡Bienvenido al juego!")

while True:
    opcion = input("¿Deseas encontrar un nuevo item? (s/n): ").lower()

    if opcion == 'n':
        print("Fin del juego.")
        break

    elif opcion == 's':
        # Simulación de encontrar un nuevo item
        item_nuevo = input("Introduce el nombre del nuevo item: ")
        calidad_nuevo = int(input("Introduce la calidad del nuevo item (0-100): "))
        tipo_nuevo = input("Introduce el tipo del nuevo item: ")

        mostrar_inventario(inventario_jugador)

        decision = input("¿Deseas cambiar este item por uno de tu inventario?
(s/n): ").lower()
        if decision == 's':
            indice_cambio = int(input("Introduce el número del item que deseas
cambiar: "))
            cambiar_item(inventario_jugador, indice_cambio, item_nuevo,
calidad_nuevo, tipo_nuevo)
        else:
            actualizar_inventario(inventario_jugador, item_nuevo, calidad_nuevo,
tipo_nuevo)

        mostrar_inventario(inventario_jugador)

    else:
        print("Opción no válida. Por favor, introduce 's' para sí o 'n' para no.")
```