Lenguajes y autómatas I

Manual para Estructurar Código LEGO y Python

Equipo:
CASTELLANOS HERRERA
JAHDIEL ELIEL
RODRIGUEZ CHAB CARMELO

Tabla de contenido

Estructura General

Inicio

LEGO	PYTHON
LEGO_INICIO	def main():
LEGO_FIN	if __name__ == '__main__':

Estructuras de Control

Condiciones

LEGO	PYTHON
REPETIR_LEGO	for
MIENTRAS_APILO	while

Bucles

LEGO	PYTHON
REPETIR_LEGO	for
MIENTRAS_APILO	while

Entrada/Salida

LEGO	PYTHON
MOSTRAR_LEGO	print
PEDIR_LEGO	Input

Declaración de Variables

LEGO	PYTHON
IDENTIFICADOR	Nombre de var
NUMBER	Número

Estructura General del Programa en LEGO

Formato

```
LEGO_INICIO
{
   // Código del programa
}
LEGO_FIN
```

Equivalente en Python

```
# Código generado desde LEGO
def main():
   # Código del programa

if __name__ == '__main__':
   main()
```

Declaraciones y Asignaciones

LEGO

```
a = 10;
b = 5;
suma = a AGREGAR_BLOQUE b;
MOSTRAR_LEGO("Resultado:");
MOSTRAR_LEGO(suma);
```

Python

```
a = 10
b = 5
suma = a + b
print("Resultado:")
print(suma)
```

Condicionales (SI/SINO)

LEGO

```
SI (x MAS_PEQUE_QUE 10) {
   MOSTRAR_LEGO("x es menor que 10");
} SINO {
   MOSTRAR_LEGO("x no es menor que 10");
}
```

Python

```
if x < 10:
   print("x es menor que 10")
else:
   print("x no es menor que 10")
```

Estructuras de Repetición

Repetir un Número de Veces

LEGO

```
REPETIR_LEGO(5) {
   MOSTRAR_LEGO("Iteración");
}
```

Python

```
for _ in range(5):
```

```
    print("Iteración")
```

Bucle Mientras

LEGO

```
MIENTRAS_APILO (contador MAS_PEQUE_QUE 3) {
    MOSTRAR_LEGO(contador);
    contador = contador AGREGAR_BLOQUE 1;
}
```

Python

```
while contador < 3:
    print(contador)
    contador += 1
```

Operadores Matemáticos

Operador LEGO	**Operador Python**
AGREGAR_BLOQUE	+
QUITAR_BLOQUE	-
APILAR_BLOQUES	*
DIVIDIR_BLOQUES	/
POTENCIAR_BLOQUES	**

Ejemplo

LEGO

```
resultado = a APILAR_BLOQUES b;
```

Python

```
resultado = a * b
```

Operadores Matemáticos

LEGO

Operador LEGO	Operador Python
MAS_GRANDE_QUE	>
MAS_PEQUE_QUE	<
ESTE_BLOQUE_IGUAL_A	==
DIFERENTE_A	!=

Ejemplo

LEGO

```
SI (a MAS_GRANDE_QUE b) {
   MOSTRAR_LEGO("a es mayor que b");
}
```

Python

```
if a > b:
   print("a es mayor que b")
```

Anidación de Estructuras

LEGO

```
REPETIR_LEGO(2) {
   MIENTRAS_APILO(contador MAS_PEQUE_QUE 3) {
      MOSTRAR_LEGO(contador);
      contador = contador AGREGAR_BLOQUE 1;
   }
}
```

Python

```
for _ in range(2):
   while contador < 3:
      print(contador)
      contador += 1
```

Programa con Múltiples Estructuras

LEGO

```
LEGO_INICIO
{
   contador = 0;
   REPETIR_LEGO(3) {
      MIENTRAS_APILO(contador MAS_PEQUE_QUE 2) {
         MOSTRAR_LEGO("Contador interno:");
         MOSTRAR_LEGO(contador);
         contador = contador AGREGAR_BLOQUE 1;
      }
      MOSTRAR_LEGO("Fin del ciclo interno.");
      contador = 0;
   }
   MOSTRAR_LEGO("Fin del programa.");
}
LEGO_FIN
```

Python

```
def main():
```

```
    contador = 0
    for _ in range(3):
        while contador < 2:
            print("Contador interno:")
            print(contador)
            contador += 1
        print("Fin del ciclo interno.")
        contador = 0
    print("Fin del programa.")
if __name__ == '__main__':
    main()
```