[**Compiladores**] Código: [**3307**]

Proyecto 2. Valor 2%

Temas de Estudio

- Tema 4. Análisis sintáctico.
- Tema 5. Traducción orientada por la sintaxis.
- Tema 6. Generación de código intermedio.

Objetivo

Aplicar los conocimientos adquiridos en el curso, para que realice un análisis sintáctico de código Python, identificando tokens y componentes léxicos para verificar su correcto uso y estructuras sintácticas, implementando una traducción orientada por la sintaxis, así como comprender el uso de la generación de código intermedio.

Software de Desarrollo

El proyecto debe ser realizado en el lenguaje de programación Java específicamente el tipo de proyecto Java with Ant, utilizando como IDE NetBeans en modo carácter. No se debe usar el modo gráfico. Si se entrega en otro tipo de proyecto el mismo no será aceptado.

Desarrollo

- Este proyecto 2 es continuación del proyecto 1, por lo que es importante que los puntos no funcionales del proyecto 1, estén resueltos para poder dar continuidad a este.

- No se permite el uso de librerías de terceros, que son fragmentos de código desarrollados por otras personas y encapsulados en componentes que podrían utilizarse como referencia en la solución del problema a través del código fuente.

- Se utilizará el mismo ejemplo de código corresponde al juego tradicional "**El Gato**" y está desarrollado en el lenguaje programación Python, e implementa la lógica para jugar contra la computadora (CPU).

- El archivo del código Python, se encuentra disponible para descargar en la plataforma "Moodle" no tiene ningún error y lo puede ejecutar online en el siguiente sitio que interpreta código python https://www.programiz.com/python-programming/online-compiler/

- Usted puede trabajar a partir de dicho código para realizar lo que se le solicita, pero tenga en cuenta que el código, será modificado por el tutor (a), para agregar errores según se mencionan en la rúbrica para poder comprobar si el estudiante realizó la validación que se le solicitó.

#	Requerimientos a evaluar
1	**Proyecto 1:** Realiza todas las correcciones que se le indicó para el proyecto 1.
2	**Independencia física en la ejecución** El archivo compilado de java .jar es ejecutado en cualquier máquina y en cualquier carpeta, así como para la lectura del archivo .py no depende de configuraciones específicas del estudiante, tanto en rutas como en nombre de los archivos, la ejecución del ejecutable .jar debe poder realizarse desde la consola CMD o símbolo del sistema.

3	**Archivo de errores**
	Los errores son agregados en el archivo de errores según las especificaciones determinadas:
	• Se deberá indicar el error utilizando el siguiente formato:
	  ○ Número del error: la tipificación es a criterio del desarrollador
	  ○ Descripción: corresponde a un error particular según lo que debió hacerse
	• Otra manera de agregar los errores es al final del archivo, pero debe de indicar el número de línea donde encontró el error y respetar lo anterior, ejemplo:
	  Error 200. Línea 20. El import no se encuentra al inicio del código
	• Es importante que se determinen errores específicos tal y como se indica en el ejemplo anterior, evitar que sean genéricos.

4	**Comando while**
	La sintaxis de un bucle while en Python es la siguiente:
	`while condición:`
	`  # Bloque de código que se ejecutará mientras la condición sea verdadera`
	`  declaración_1`
	`  declaración_2`
	`  ...`
	Explicación de la sintaxis:
	• while: Palabra clave que inicia el bucle.
	• condición: Una expresión que se evalúa como verdadera (True) o falsa (False). Mientras esta condición sea verdadera, el bloque de código dentro del while se ejecutará repetidamente.
	• : (dos puntos): Indica el comienzo del bloque de código que pertenece al bucle while.
	• Bloque de código: Las declaraciones o líneas de código que se ejecutarán mientras la condición sea verdadera. Este bloque debe estar indentado (tab) con respecto al while. Al menos debe haber una línea de código entre la línea del comando while y su cierre.
	• El cierre de la estructura while es cuando la línea que siga no está identado consecuentemente al while, es decir la línea de comando está al mismo nivel del while, en la siguiente imagen se muestra el ejemplo de como se cierra el while.

```python
contador = 0  # Inicializamos la variable contador

while contador < 5:  # La condición es que contador sea menor que 5
    print("El contador es:", contador)  # Imprimimos el valor actual de contador
    contador += 1  # Incrementamos contador en 1 en cada iteración

print("El bucle ha terminado")  # Este mensaje se imprime cuando el bucle termina
```

Nota: cualquier incumplimiento de los puntos mencionados ya debe considerarse un error porque no se estaría cumpliendo con la sintaxis.

5 **Validación estructura de las funciones**:

Las funciones en Python sirven para varios propósitos importantes, para la reutilización de código que permiten escribir un bloque de código una vez y reutilizarlo tantas veces como sea necesario, por lo que reduce la redundancia y hace que el código sea más limpio y fácil de mantener. Permite modularidad, que ayudan a dividir un programa grande en partes más pequeñas y manejables. Cada función puede realizar una tarea específica, lo que facilita la comprensión y el mantenimiento del código.

```python
def saludo(nombre):
    mensaje = "Hola, " + nombre  # Esta línea está indentada y pertenece a la función
    return mensaje

print(saludo("María"))
```

La sintaxis de una función es la siguiente:

```python
def nombre_de_la_funcion(parametros):
    # Cuerpo de la función
    instrucciones
    return valor_de_retorno
```

Explicación de la sintaxis:

- **def** es una palabra clave que indica el comienzo de una función.
- **nombre_de_la_funcion** es el nombre que le das a la función.
- **parametros** son los valores que la función toma como entrada, para este proyecto se asumirá que estarán correctos por lo que esté entre paréntesis no debe validarlos. Solo debe validar que tenga los paréntesis redondos de abrir y cerrar.
- **Dos puntos:** se debe validar que existan los dos puntos a la par del paréntesis de cierre

- **El cuerpo de la función** contiene las instrucciones que se ejecutarán cuando se llame a la función.
- **return** es una palabra clave que indica el valor que la función devolverá, a veces puede ser que no esté, por lo que debe permitirlo en ambos escenarios.
- **Identación:** debe considerar que las líneas de código siguientes a la línea **def**, deben estar identadas o tabulación de lo contrario, daría error como: IndentationError: expected an indented block. Ejemplo de sintaxis incorrecta de identación:

```
def saludo(nombre):
mensaje = "Hola, " + nombre  # Esta línea no está indentada correctamente
return mensaje
```

En el ejemplo, anterior, la línea que inicia con “mensaje” y return están incorrectas porque les falta identación.

6 **Llamado a una función**

La sintaxis para llamar a una función en Python es bastante sencilla. Simplemente necesitas usar el nombre de la función seguido de paréntesis redondos, y dentro de los paréntesis redondos colocar los argumentos necesarios (si los hay).

- El llamado se determina de la siguiente manera nombre_de_la_funcion(argumentos)
- La función debe existir definida correctamente

Se requiere verificar que función exista, es decir que esté definida correctamente según su estructura del punto 5.

- nombre_de_la_funcion: es el nombre que le das a la función
- Paréntesis: validar que se encuentre abierto y cerrado
- Parámetros: se puede asumir que son correctos no los debe validar

En caso que la función no esté definida se estaría haciendo un llamado incorrecto por lo que deberá dar error.

7 **Salida de datos**

La función print() en Python se utiliza para mostrar información en la pantalla o realizar salto de lineas, y la sintaxis básica se conforma de las siguientes maneras:

- print("texto entre comillas")
  - Sirve para imprimir en pantalla un texto. Se compone del comando print, abre paréntesis redondo, abre comilla, hay texto, cierra comillas dobles y cierra paréntesis redondo.
- print()
  - Sirve para hacer un salto de línea, y su sintaxis es el comando print, luego abre y cierra paréntesis redondos.
- print(variable)

- Sirve para imprimir el resultado de una variable y su sintaxis es, comando print, abre paréntesis redondo, sigue la variable, cierra paréntesis redondo. Además, se debe validar que la variable exista es decir que esté declarada.

Es importante aclarar que en Python hay otras maneras de utilizar la función print, sin embargo, para este proyecto solo validaremos las tres formas de sintaxis mencionadas. La excepción para el proyecto, será no validar las siguientes líneas de código y se asumirá que estarán correctas, esto debido que tienen otra sintaxis no mencionada:

- print(f" {tablero[i][j]} ", end="")
- print("|", end="")
- print(f"\nTurno del jugador {jugador_usuario if jugador_actual == 'X' else jugador_cpu} ({jugador_actual})")

8 **Manejo errores try/except**

El bloque try / except en Python se utiliza para manejar excepciones, que son errores que ocurren durante la ejecución de un programa. Este mecanismo permite que tu programa continúe ejecutándose incluso si ocurre un error, en lugar de terminar abruptamente.

```
try:
    # Bloque de código que puede causar una excepción
    instrucciones
except TipoDeExcepcion:
    # Bloque de código que se ejecuta si ocurre la excepción
    manejo_de_error
```

Explicación de la sintaxis, que para este proyecto nos basaremos en aspectos básicos de la sintaxis para evaluar y debe respetar lo que se indica a continuación, cualquier aspecto diferente a lo indicado, aunque en el lenguaje de python sea válido para nuestro proyecto sería un error por el incumplimiento de las indicaciones.

- try es la palabra reservada que se requiere para iniciar el bloque de manejo de errores
- dos puntos : es necesario que esté posterior junto al comando try
- debe existir al menos una línea de código que se requiere que se maneje la excepción
- except es la palabra reservada para indicar el tipo de excepción que se requiere atrapar, para nuestro proyecto solo validaremos que sean estas tres:
  - ArithmeticError
  - ZeroDivisionError
  - ValueError

  Cualquier otra palabra reservada que, aunque sea permitada en Python, debe reportarse como error en nuestro proyecto.
- por último, una línea que permita manejar el error por lo general es indicarle por medio de un mensaje a la persona usuaria que algo sucedió. Para nuestros efectos del proyecto validaremos que tengan una línea del comando print, si no se encuentra debe dar error.
- Considere que debe utilizarse la identación o un tab, tanto lo que esté debajo del try como del except, tal y como se mostró en la imagen anterior, si esto no se cumple hay un error.

Un ejemplo funcional de su sintaxis:

```
try:
    resultado = 10 / 0
except ZeroDivisionError:
    print("Error: No se puede dividir por cero.")
```

Honestidad Académica

https://audiovisuales.uned.ac.cr/play/player/23048

Nota Importante

Cada estudiante es responsable del contenido que entrega, si no es el archivo correcto, no podrá entregarlo posterior a la fecha establecida.

Si el contenido del archivo coincide con algún otro estudiante, o se comprueba que no es de su autoría, se aplicaría lo indicado en la plataforma en el documento Lineamientos ante casos de plagio

Indicaciones Importantes

- Es obligatorio que incluya todo el directorio donde se encuentra el Proyecto 2.
- Este proyecto debe estar desarrollado en NETBEANS que es la herramienta oficial de la asignatura.
- El programa debe ser modular, utilizando de la mejor manera funciones definidas por usted.
- Los trabajos deben realizarse en forma individual. Dentro del código del programa debe de indicar la documentación que explique cómo fue realizado el programa.
- Si utiliza código de algún ejemplo del libro, o de otra fuente que no sea de su autoría, debe de indicarlo.
- Comprima todos los archivos en un solo archivo .zip o .rar.
- Nombre del archivo que envía: debe ser nombre y primer apellido del estudiante, y nombre del proyecto. Ejemplo: JuanRojas-Proyecto2
- La entrega de Proyecto 2 en las fechas establecidas en la plataforma de aprendizaje en línea Moodle en el apartado que se indique.

- Si no concluyó a tiempo este proyecto, debe entregar lo que pudo hacer e incluir una carta explicando las razones por las cuales no finalizó.

Rúbrica de Evaluación

Criterio	Cumple a satisfacción lo indicado en la evaluación	Cumple medianamente en lo indicado en la evaluación	Cumple en contenido y formato, pero los aportes no son significantes	No cumple o no presenta lo solicitado
1. Proyecto 1: Realiza todas las correcciones que se le indicó para el proyecto 1.	5	2	1	0
2. Independencia física en la ejecución El archivo compilado de java .jar es ejecutado en cualquier máquina y en cualquier carpeta, así como para la lectura del archivo .py no depende de configuraciones específicas del estudiante, tanto en rutas como en nombre de los archivos, la ejecución del ejecutable .jar debe poder realizarse desde la consola CMD o símbolo del sistema.	5	2	1	0
3. Archivo de errores, los errores son agregados en el archivo de errores según las especificaciones determinadas	15	6	2	0
4. Cumple con la sintaxis del comando while	20	10	5	0
5.Cumple con validación estructura de las funciones	20	10	5	0
6. En caso que la función no esté definida se estaría haciendo un llamado incorrecto por lo que deberá dar error.	10	5	3	0
7. Válida correctamente la sintaxis básica de las tres maneras indicadas: a. print("texto entre comillas") b. print() c. print(variable)	15	10	5	0
8. Manejo errores considerando la sintaxis de los comandos try/except	10	5	3	0
TOTAL	**100**	**50**	**25**	**0**