

► Manal Técnico

Laboratorio de Lenguajes Formales y de Programación
Sección B-

Derek Francisco Orellana Ibáñez ► 202001151

Clases

Abstracto.py

Define una interfaz para las subclases, teniendo como parámetro fila y columna.

```
1 class Abstracto(ABC):  
2     def __init__(self, fila, columna):
```

def getValor(self, arbol)

Función que permite devuelve el valor del objeto.

```
1 @abstractmethod  
2     def getValor(self, arbol):
```

def getFila(self)

Función que devolverá la fila

```
1     @abstractmethod  
2     def getFila(self):
```

def getColumna(self)

Función que devolverá la columna

```
1     @abstractmethod  
2     def getColumna(self):
```

Error.py

Define una clase que hereda la clase abstracta, teniendo como parámetros, el lexema, tipo de error, la fila y columna donde se encontró el error.

```
1 class Error(Abstracto):  
2     def __init__(self, lexema, tipo, fila, columna):
```

Al ser una subclase de la clase Abstracta, tendrá las mismas funciones de getValor, getColumna y getFila.

```
def getTipo(self)
```

Función que permite obtener el tipo de error que ha sido ingresado al objeto.

```
1 def getTipo(self):
```

Lexema.py

Clase que crea un lexema heredando de la clase Abstracto teniendo así sus funciones principales.

```
1 class Lexema(Abstracto):  
2     def __init__(self, lexema, fila, columna):
```

```
def setValor(self, valor)
```

Función que modifica el valor del lexema

```
1 def setValor(self, valor):
```

Numero.py

Clase que crea un numero heredando de la clase Abstracto teniendo así sus funciones principales.

```
1 class Numero(Abstracto):  
2     def __init__(self, valor, fila, columna):
```

```
def setValor(self, valor)
```

Función que modifica el valor del lexema

```
1 def setValor(self, valor):
```

Aritmetica.py

Clase que resuelve una operación aritmética que hereda de la clase Abstracto, teniendo sus funciones principales y tomando como parámetros el tipo de operación, valor 1 y valor 2, nodo_valor1, nodo_valor2, fila y columna.

```
1 class Aritmetica(Abstracto):  
2     def __init__(self, operacion, valor1, valor2, nodo_valor1, nodo_valor2, fila, columna):
```

```
def getNodo_valor1(self)
```

Obtiene el objeto de clase Nodo_Aritmetico o Nodo_Trigonometrico si el valor 1 es una operación.

```
1     def getNodo_valor1(self):
```

```
def getNodo_valor2(self)
```

Obtiene el objeto de clase `Nodo_Aritmetico` o `Nodo_Trigonometrico` si el valor 2 es una operación.

```
1 def getNodo_valor2(self):
```

```
def getOperacion(self, arbol)
```

Devuelve el tipo de operación aritmética

```
1 def getOperacion(self, arbol):
```

```
def getValor1(self, arbol)
```

Devuelve el valor1, si este valor es otra operación, devuelve el resultado de dicha operación.

```
1 def getValor1(self, arbol):
```

```
def getValor2(self, arbol)
```

Devuelve el valor2, si este valor es otra operación, devuelve el resultado de dicha operación.

```
1 def getValor2(self, arbol):
```

Trigonometricas.py

Clase que resuelve una operación trigonométrica que hereda de la clase Abstracto, teniendo sus funciones principales y tomando como parámetros el tipo de operación, valor 1, nodo_valor1, fila y columna.

```
1 class Trigonometricas(Abstracto):
2     def __init__(self, operacion, valor1, nodo_valor1, fila, columna):
```

```
def getOperacion(self, arbol)
```

Devuelve el tipo de operación aritmética

```
1     def getOperacion(self, arbol):
```

```
def getValor1(self, arbol)
```

Devuelve el valor1, si este valor es otra operación, devuelve el resultado de dicha operación.

```
1     def getValor1(self, arbol):
```

```
def getNodo_valor1(self)
```

Obtiene el objeto de clase Nodo_Aritmetico o Nodo_Trigonometrico si el valor 1 es una operación.

```
1     def getNodo_valor1(self):
```

Nodos.py

Archivo donde están creados los 3 tipos de nodos, Nodo_Texto, Nodo_Aritmetico y Noto_Trigonometrico

Nodo_Texto()

Crea un nodo que obtiene los parámetros de los lexemas finales para obtener los textos y formas para la grafica.

```
1 class Nodo_Texto():
2     def __init__(self, texto, color_fondo, color_fuente, forma):
```

def getText(self)

Devuelve el valor del texto para el título de la gráfica.

```
1     def getTexto(self):
```

def getColorFondo(self)

Devuelve el color de fondo de los nodos de la gráfica.

```
1     def getColorFondo(self):
```

def getColorFuente(self)

Devuelve el color de fuente de la gráfica.

```
1     def getColorFuente(self):
```

```
def getForma(self)
```

Devuelve la forma que tomara la gráfica.

```
1 def getForma(self):
```

```
Nodo_Aritmetico()
```

Clase que almacena los valores, el tipo de operación, el resultado y si los valores son el resultado de otra operación, nodo_valor1 y nodo_valor2 almacenaran el tipo de Nodo, ya sea Nodo_Aritmetico o Nodo_Trigonometrico

```
1 class Nodo_Aritmetico():
2     def __init__(self, operacion, valor1, valor2, nodo_valor1, nodo_valor2, resultado):
```

```
def getOperacion(self)
```

Devuelve el tipo de operacion

```
1 def getOperacion(self):
```

```
def getValor1(self)
```

Devuelve el valor 1 o el resultado si el valor1 es otra operación.

```
1 def getValor1(self):
```

```
def getValor2(self)
```

Devuelve el valor 1 o el resultado si el valor2 es otra operación.

```
1 def getValor2(self):
```



```
def getNodo_valor1(self)
```

Devuelve un nodo tipo `Nodo_Aritmetico` o `Nodo_Trigonometrico`, dependiendo si el `valor1` es una operación, de lo contrario, retornara `None`

```
1 def getNodo_valor1(self):
```

```
def getNodo_valor2(self)
```

Devuelve un nodo tipo `Nodo_Aritmetico` o `Nodo_Trigonometrico`, dependiendo si el `valor2` es una operación, de lo contrario, retornara `None`

```
1 def getNodo_valor2(self):
```

```
def getResultado(self)
```

Devuelve el resultado final de la operación entre el `valor1` y `valor2`

```
1 def getResultado(self):
```


```
Nodo_Trigonometrico()
```

Clase que almacena los valores, el tipo de operación, el resultado y si el valor es el resultado de otra operación, `nodo_valor1` almacenaran el tipo de `Nodo`, ya sea `Nodo_Aritmetico` o `Nodo_Trigonometrico`

```
1 class Nodo_Trigonometrico():
2     def __init__(self, operacion, valor1, nodo_valor1, resultado):
```

```
def getOperacion(self)
```


Devuelve el tipo de operacion



```
1 def getOperacion(self):
```

```
def getValor1(self)
```

Devuelve el valor 1 o el resultado si el valor1 es otra operación.



```
1 def getValor1(self):
```

```
def getNodo_valor1(self)
```

Devuelve un nodo tipo Nodo_Aritmetico o Nodo_Trigonometrico, dependiendo si el valor1 es una operación, de lo contrario, retornara None



```
1 def getNodo_valor1(self):
```

```
def getResultado(self)
```

Devuelve el resultado final de la operación entre el valor1 y valor2



```
1 def getResultado(self):
```

Analizador.py

```
def analizar_cadena(cadena)
```

Función que analiza el contenido del archivo leyendo carácter por carácter y creando los lexemas.

```
1 def analizar_cadena(cadena):
```

```
def es_numero(numero)
```

Función que analiza si el numero que se envia es tipo int o tipo float

```
1 def es_numero(numero):
```

```
def armar_lexema(cadena)
```

Función que analiza la cadena enviada y arma el lexema retornando la cadena modificada y el lexema encontrado.

```
1 def armar_lexema(cadena):
```


```
def armar_numero(cadena)
```

Función que analiza la cadena enviada y arma el lexema retornando la cadena modificada y el numero encontrado.

```
1 def armar_numero(cadena):
```

```
def realizar_operacion(es_hijo=False)
```


Función que entra en recursividad al momento de realizar las operaciones trigonométricas o aritméticas, retornando una lista de nodos.



```
1 def realizar_operacion(es_hijo=False):
```

```
def obtener_respuestas()
```


Función que llama a la función realizar_operacion() para realizar la operación, retornando la lista de nodos y la lista de errores.



```
1 def obtener_respuestas():
```

```
def analizar_errores(palabra)
```

Función que analiza y compara la palabra enviada con cada palabra de la lista de lexemas reservados, si esta mal escrita la corrige y crea un nodo de tipo error retornando la palabra corregida.



```
1 def analizar_errores(palabra):
```

MainWindow.py

Clase principal donde se crea la ventana principal de la interfaz gráfica.

```
1 class MainWindow(tk.Tk):  
2     def __init__(self):
```

```
def crear_menu(self)
```

Función que crea el menú de opciones en la parte superior de la ventana.

```
1     def crear_menu(self):
```

```
def abrir_manual_usuario(self)
```

Función que permite abrir el manual de usuario con el lector pdf predeterminado.

```
1     def abrir_manual_usuario(self):
```

```
def abrir_manual_tecnico(self)
```

Función que permite abrir el manual técnico usuario con el lector pdf predeterminado.

```
1     def abrir_manual_tecnico(self):
```

```
def crear_widgets(self)
```

Función que crea los componentes dentro de la ventana, creando un frame para el título y dos frames mas para el lado izquierdo y derecho, agregando textarea y botones.

```
1 def crear_widgets(self):
```

```
def ventana_info(self, nombre, carnet, seccion)
```

Función que crea una nueva ventana mostrando la información del estudiante.

```
1 def ventana_info(self, nombre, carnet, seccion):
```

```
def mostrar_ventana_info(self)
```

Función que abre la ventana de información.

```
1 def mostrar_ventana_info(self):
```

```
def abrir_archivo(self)
```

Función que abre el archivo y lo agrega al textarea de la izquierda.

```
1 def abrir_archivo(self):
```

```
def leerArchivo(self, ruta)
```

Función que se encarga de leer el archivo que se selecciono al momento de abrir para leer su contenido y agregarlo al textarea.

```
1 def leerArchivo(self, ruta):
```

```
def guardar_archivo(self)
```

Guarda el archivo abierto.

```
1 def guardar_archivo(self):
```

```
def guardar_archivo_como(self)
```

Guarda el archivo actual como un nuevo archivo en una ubicación que el usuario seleccione.

```
1 def guardar_archivo_como(self):
```

```
def guardar_archivo_errores(self)
```

Función que guarda el contenido del textarea de los errores en un archivo Json en una ubicación seleccionada.

```
1 def guardar_archivo_errores(self):
```

```
def analizar(self)
```

Función que manda a llamar a la función de `analizar_cadena()` y `obtener_respuestas` para analizar la cadena y generar los errores.

```
1 def analizar(self):
```

```
def graficar_resultados(self)
```

Función que manda a llamar a la función `graficar` y `generar_errores` si no se han generado.

```
1 def graficar_resultados(self):
```

```
def graficar(self, lista_nodos)
```

Función que genera la grafica de los resultados.

```
1 def graficar(self, lista_nodos):
```

```
def imprimir_nodos(self, nodo, ruta, n_padre=0, hijo='')
```

Función que crea y escribe en un archivo `.dot` temporal la estructura de los nodos.

```
1 def imprimir_nodos(self, nodo, ruta, n_padre=0, hijo=''):
```



```
def enlazar_nodos(self, nodo, ruta, n_padre=0, hijo='')
```

Función que lee el archivo .dot temporal y agrega la estructura para enlazar los nodos.

```
1 def enlazar_nodos(self, nodo, ruta, n_padre=0, hijo='')
```

```
def generar_errores(self)
```

Función que escribe en el textarea de los errores, los errores encontrados al analizar la cadena.

```
1 def generar_errores(self):
```

```
def mostrar_errores_anteriores(self)
```

Función que agrega al textarea de errores, el archivo de los errores que se abrió o guardo anteriormente.

```
1 def mostrar_errores_anteriores(self):
```