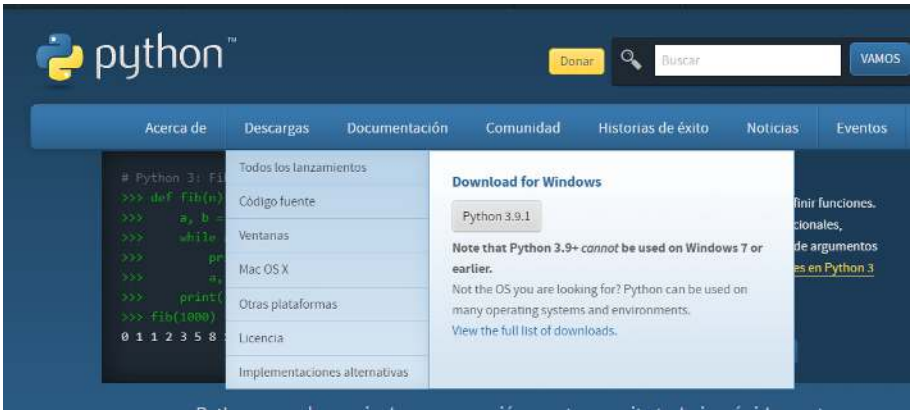
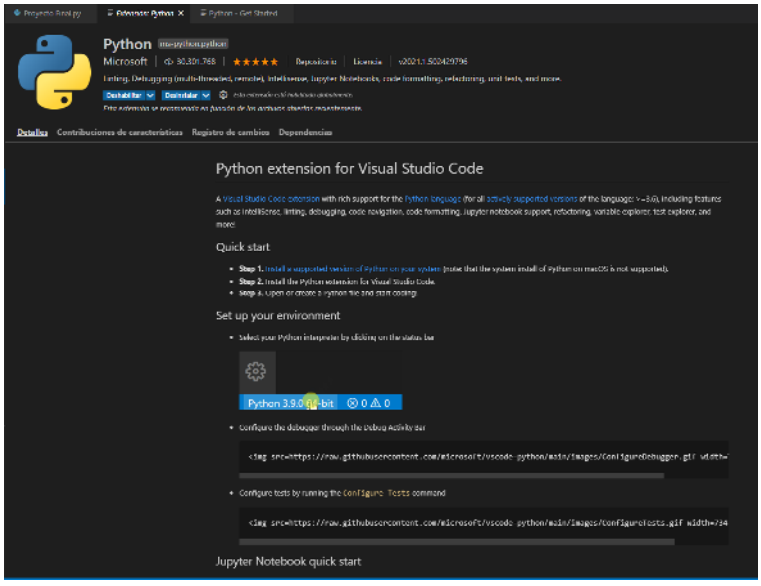
Instalación de Python

Se hace la descarga de Python en su página oficial y lo instalamos

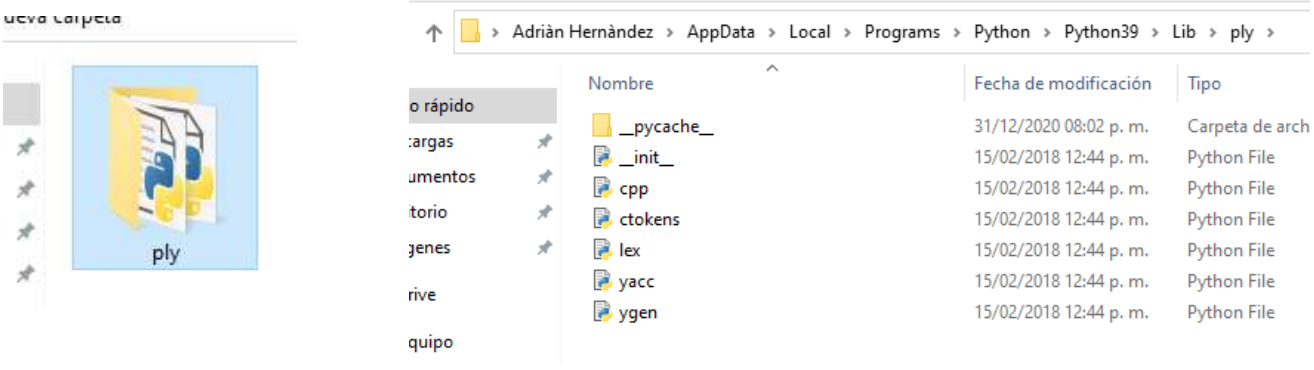


Para realizar el Análisis Léxico, primero tenemos que instalar la librería de Python en Visual Studio Code:



Después tenemos que descargar las librerías locales: lex, yacc, etc. Y colocar su carpeta dónde vienen incluidas en la siguiente dirección:

C:\Users\feli_\AppData\Local\Programs\Python\Python39\Lib



Programa – Analizador Léxico:

En primera instancia, tenemos que importar la librería local “lex” para el funcionamiento del compilador.

```
import ply.lex as lex
```

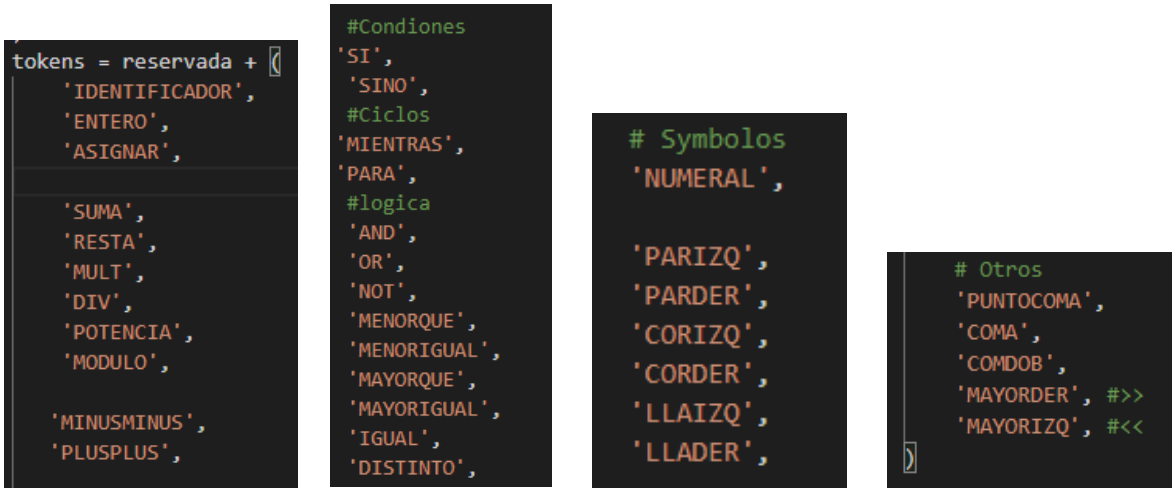
Tenemos que declarar la variable donde se va a guardar el resultado del análisis teniendo en cuenta el lexema ingresado, en este caso el lexema es la cadena de caracteres o valor numérico aceptados por el token.

```
# resultado del analisis
resultado_lexema = []
```

En seguida tenemos que declarar cada uno de los tokens, empezamos por las palabras reservadas:

```
reservada = (
    # Palabras Reservadas
    'INCLUDE',
    'USING',
    'NAMESPACE',
    'STD',
    'COUT',
    'CIN',
   'GET',
   'CADENA',
  'RETURN',
   'VOID',
    'INT',
    'ENDL',
)
```

Seguimos creando los tokens, pero ahora con las palabras que definen las operaciones aritméticas, las condiciones, los ciclos, las compuertas lógicas, los símbolos (corchetes, llaves, etc.), y los signos de puntuación:



Posteriormente pasamos a la declaración de las Expresiones Regulares, definiremos los símbolos de aceptación que tomará cada token al momento de ingresar los lexemas para su análisis:

```
# Reglas de Expresiones Regualres para token de Contexto simple

t_SUMA = r'\+'
t_RESTA = r'-'
t_MINUSMINUS = r'\-\-'
t_PUNTO = r'\.'
t_MULT = r'\*'
t_DIV = r'/'
t_MODULO = r'\%'
t_POTENCIA = r'(\*{2} | \^)'

t_ASIGNAR = r'='
# Expresiones Logicas
t_AND = r'\&\&'
t_OR = r'\|{2}'
t_NOT = r'\!'
t_MENORQUE = r'<'
t_MAYORQUE = r'>'
t_PUNTOCOMA = ';'
t_COMA = r','
t_PARIZQ = r'\('
t_PARDER = r'\)'
t_CORIZQ = r'\['
t_CORDER = r'\]'
t_LLAIZQ = r'{'
t_LLADER = r'}'
t_COMDOB = r'\"'
```

En seguida definiremos las funciones (def) necesarias de los tokens para reconocer los números, los identificadores, los símbolos, etc., tomando como referencia que son Expresiones Regulares avanzadas:

```
def t_INCLUDE(t):
    r'include'
    return t

def t_USING(t):
    r'using'
    return t

def t_NAMESPACE(t):
    r'namespace'
    return t

def t_STD(t):
    r'std'
    return t

def t_COUNT(t):
    r'cout'
    return t

def t_CIN(t):
    r'cin'
    return t

def t_GET(t):
    r'get'
    return t

def t_ENDL(t):
    r'endl'
    return t

def t_SINO(t):
    r'else'
    return t

def t_SI(t):
    r'if'
    return t

def t_RETURN(t):
    r'return'
    return t

def t_VOID(t):
    r'void'
    return t
```

```
def t_MIENTRAS(t):
    r'while'
    return t

def t_PARA(t):
    r'for'
    return t

def t_ENTERO(t):
    r'\d+'
    t.value = int(t.value)
    return t

def t_IDENTIFICADOR(t):
    r'\w+(_\d\w)*'
    return t

def t_CADENA(t):
    r'\"?(\w+ \ *\w*\d* \ *)\"?'
    return t

def t_NUMERAL(t):
    r'\#'
    return t

def t_PLUSPLUS(t):
    r'\+\+'
    return t

def t_MENORIGUAL(t):
    r'<='
    return t

def t_MAYORIGUAL(t):
    r'>='
    return t

def t_IGUAL(t):
    r'=='
    return t

def t_MAYORDER(t):
    r'<<'
    return t

def t_MAYORIZQ(t):
    r'>>'
    return t
```

```
def t_DISTINTO(t):
    r'!='
    return t

def t_newline(t):
    r'\n+'
    t.lexer.lineno += len(t.value)

def t_comments(t):
    r'/\*(.|\n)*?\*/'
    t.lexer.lineno += t.value.count('\n')
    print("Comentario de multiple linea")

def t_comments_ONELine(t):
    r'\/\/(.)*\n'
    t.lexer.lineno += 1
    print("Comentario de una linea")
t_ignore =' \t'
```

Es de suma importancia agregar la función de error, esto es para que el análisis detecte errores cuando un carácter ingresado por el usuario no es válido:

```
def t_error( t):
    global resultado_lexema
    estado = "** Token no valido en la Linea {:4} Valor {:16} Posicion {:4}".format(str(t.lineno), str(t.value),
                                                                                    str(t.lexpos))
    resultado_lexema.append(estado)
    t.lexer.skip(1)
```

Casi para terminar, se agrega la función para hacer pruebas de ingreso de los lexemas. Si se cumplen las condiciones declaradas anteriormente en los tokens, se imprimirá un resultado exitoso a través de la variable mencionada anteriormente:

A continuación, se explica el significado de los indicadores Línea, Tipo, Valor, Posición:

- **Línea:** Se refiere al renglón en que fue ingresado el lexema.
- **Tipo:** El analizador considera el tipo de dato ingresado por el usuario (Identificador, Número, Suma, Resta, etc., etc.) que hacen referencia a los tokens.
- **Valor:** La función de este indicador es, sencillamente, repetir el lexema ingresado.
- **Posición:** Es un contador donde se considera la posición del cursor en que se ingresa el valor.

```
# Prueba de ingreso
def prueba(data):
    global resultado_lexema

    analizador = lex.lex()
    analizador.input(data)

    resultado_lexema.clear()
    while True:
        tok = analizador.token()
        if not tok:
            break
        # print("lexema de "+tok.type+" valor "+tok.value+" linea "tok.lineno)
        estado = "Linea {:4} Tipo {:16} Valor {:16} Posicion {:4}".format(str(tok.lineno),str(tok.type) ,str(tok.value), str(tok.lexpos) )
        resultado_lexema.append(estado)
    return resultado_lexema
```

Posteriormente, se instancia el analizador léxico para imprimir la indicación “ingrese” que permitirá al usuario ingresar los datos y mediante ello realizará todo el análisis valorando a través de cada uno de los tokens, como se indica en el paso anterior.

Por último, se manda llamar a la variable del resultado, dónde se alojará y mostrará el resultado obtenido del análisis léxico.

```
# instanciamos el analizador lexico
analizador = lex.lex()

if __name__ == '__main__':
    while True:
        data = input("ingrese: ")
        prueba(data)
        print(resultado_lexema)
```

Para concluir, realizamos una prueba, para demostrar el funcionamiento de este analizador léxico:

```
ingrese: hola
['Linea 1    Tipo IDENTIFICADOR    Valor hola            Posicion 0   ']
ingrese: cómo estás
['Linea 1    Tipo IDENTIFICADOR    Valor cómo            Posicion 0   ', 'Linea 1    Tipo IDENTIFICADOR    Valor estás           Posicion 5   ']
ingrese: Ok, amigo
['Linea 1    Tipo IDENTIFICADOR    Valor Ok              Posicion 0   ', 'Linea 1    Tipo COMA             Valor ,               Posicion 2   ', 'Linea 1    Tipo IDENTIFICADOR    Valor amigo           Posicion 4   ']
ingrese: 15.2
['Linea 1    Tipo ENTERO           Valor 15              Posicion 0   ', 'Linea 1    Tipo PUNTO            Valor .               Posicion 2   ', 'Linea 1    Tipo ENTERO           Valor 2               Posicion 3   ']
ingrese: 2 + 5 = 7
['Linea 1    Tipo ENTERO           Valor 2               Posicion 0   ', 'Linea 1    Tipo SUMA             Valor +               Posicion 2   ', 'Linea 1    Tipo ENTERO           Valor 5               Posicion 4   ', 'Line
a 1    Tipo ASIGNAR          Valor =               Posicion 6   ', 'Linea 1    Tipo ENTERO           Valor 7               Posicion 8   ']
```