

Compyler. Un traductor de lenguaje natural a Python para listas

Vladislav Nikolov Vasilev
Nazaret Román Guerrero

Variables y alias

```
char * procesado;
char * elem;
int i = 0;
int j = 0;
int comparar = 0;
char * comparacion;
%}

letra      [a-zA-Z]
digito     [0-9]
espacio    [ ]
entero     \-?[digito]+
numero     {entero}(\.{digito})+
delimitador  "[^\t\n]+"
cadena     "{letra}|{digito}|{espacio}*"
variable   ({letra}|{digito}|_)+
crear      "crear "{variable}" "{(numero)|{cadena}|{espacio}}+?"
longitud   "longitud "{variable}
imprimir   "imprimir "{variable}
recorrer    "recorrer "{variable}" "{delimitador}?
insertar    "insertar "{variable}" "{(cadena)|{numero}}
borrar      "borrar "{variable}" "{(cadena)|{numero}}
obtener     "obtener "{variable}" "{entero}
copiar      "copiar "{variable}" "{variable}
concatenar  "concatenar "{variable}" "{variable}" "{variable}
ordenar     "ordenar "{variable}" reves"?
suma        {variable}" mas "{variable}" (" mas "{variable})*
operador    "igual"|"diferente"|"menor"|"menor igual"|"mayor"|"mayor igual"
expresion   {longitud}|{suma}|{obtener}|{variable}|{ordenar}
comparar     "comparar "{expresion}" "{operador}" "{expresion}
```

Funcionalidades (I)

```
{crear}

{procesado = yytext + 6;
char * lista = malloc(strlen(procesado));
elem = malloc(strlen(procesado) * 2);
for (i = 0; i < strlen(procesado) && procesado[i] != ' '; i++)
    lista[i] = procesado[i];
if (i < strlen(procesado)) {
    procesado += i + 1;
    j = 0;
    int cambiar_espacio_coma = 0;
    for (i = 0; i < strlen(procesado); i++) {
        if (procesado[i] == ' ') {
            if (cambiar_espacio_coma != 0) {
                elem[j] = ',';
                j++;
                elem[j] = ' ';
                j++;
                cambiar_espacio_coma = 0;
            }
        } else {
            elem[j] = procesado[i];
            j++;
            cambiar_espacio_coma = 1;
        }
    }
    printf("%s = [%s]", lista, elem);
}

{longitud}
{procesado = yytext + 9;
if (comparar > 0)
    comparar--;
printf("len(%s)", procesado);}
```

Figure 1: Expresiones para crear una lista y obtener la longitud.

Funcionalidades (II)

```
{imprimir}                                     {procesado = yytext + 9; printf("print(%s)", procesado);}
{recorrer}                                     {procesado = yytext + 9;
                                             char * lista = malloc(strlen(procesado));
                                             char * delim;
                                             int usar_delim = 0;
                                             for (i = 0; i < strlen(procesado) && procesado[i] != ' '; i++)
                                                 lista[i] = procesado[i];
                                             if (strlen(procesado) >= i + 1)
                                                 usar_delim = 1;
                                             procesado += i + 1;
                                             delim = procesado;
                                             printf("for item in %s:\n    print(item", lista);
                                             if (usar_delim)
                                                 printf(", end = %s", delim);
                                             printf(")");
                                             }
}
```

Figure 2: Expresiones para imprimir y recorrer una lista.

Funcionalidades (III)

```
{insertar}                                {procesado = yytext + 9;  
                                           char * lista = malloc(strlen(procesado));  
                                           for (i = 0; i < strlen(procesado) && procesado[i] != ' '; i++)  
                                             lista[i] = procesado[i];  
                                           procesado += i + 1;  
                                           elem = procesado;  
                                           printf("%s.append(%s)", lista, elem);}  
  
{borrar}                                {procesado = yytext + 7;  
                                           char *lista = malloc(strlen(procesado));  
                                           for(i = 0; i < strlen(procesado) && procesado[i] != ' '; i++)  
                                             lista[i] = procesado[i];  
                                           procesado += i + 1;  
                                           elem = procesado;  
                                           printf("%s.remove(%s)", lista, elem);}  
  
{obtener}                                {procesado = yytext; procesado += 8;  
                                           char *lista = malloc(strlen(procesado));  
                                           for(i = 0; i < strlen(procesado) && procesado[i] != ' '; i++)  
                                             lista[i] = procesado[i];  
                                           procesado += i + 1;  
                                           elem = procesado;  
                                           if(comparar > 0) {  
                                             printf("%s[%s]", lista, elem);  
                                             comparar--;  
                                           }  
                                           else  
                                             printf("print(%s[%s])", lista, elem);}
```

Figure 3: Expresiones para insertar, borrar y obtener un elemento de una lista.

Funcionalidades (IV)

```
{copiar}                                {procesado = yytext + 7;  
                                        char *orig = malloc(strlen(procesado));  
                                        char *dest = malloc(strlen(procesado));  
                                        for(i = 0; i < strlen(procesado) && procesado[i] != ' '; i++)  
                                            orig[i] = procesado[i];  
                                        procesado += i + 1;  
                                        dest = procesado;  
                                        printf("%s = %s", dest, orig);}  
  
{concatenar}                            {procesado = yytext + 11;  
                                        char *l1 = malloc(strlen(procesado));  
                                        char *l2 = malloc(strlen(procesado));  
                                        char *dest = malloc(strlen(procesado));  
                                        for(i = 0; i < strlen(procesado) && procesado[i] != ' '; i++)  
                                            l1[i] = procesado[i];  
                                        procesado += i + 1;  
                                        for(i = 0; i < strlen(procesado) && procesado[i] != ' '; i++)  
                                            l2[i] = procesado[i];  
                                        procesado += i + 1;  
                                        dest = procesado;  
                                        printf("%s = %s + %s", dest, l1, l2);}  
  
{ordenar}                              {procesado = yytext + 8;  
                                        char *lista = malloc(strlen(procesado));  
                                        if (comparar > 0)  
                                            comparar--;  
                                        for(i = 0; i < strlen(procesado) && procesado[i] != ' '; i++)  
                                            lista[i] = procesado[i];  
                                        if (strlen(procesado) >= i + 1)  
                                            printf("%s.sort(reverse=True)", lista);  
                                        else  
                                            printf("%s.sort()", lista);}
```

Figure 4: Expresiones para copiar, concatenar y ordenar listas.

Funcionalidades (V)

```
{suma}                                {procesado = yytext;
                                     char * suma = malloc(strlen(procesado));
                                     int salta_palabra = 0;
                                     j = 0;

                                     if (comparar > 0)
                                         comparar--;
                                     for(i = 0; i < strlen(procesado); i++) {
                                         if (procesado[i] == ' ') {
                                             salta_palabra = !salta_palabra & 0x1;
                                             if(salta_palabra) {
                                                 suma[j] = ' ';
                                                 j++;
                                                 suma[j] = '+';
                                                 j++;
                                                 suma[j] = ' ';
                                                 j++;
                                             }
                                         } else {
                                             if (!salta_palabra) {
                                                 suma[j] = procesado[i];
                                                 j++;
                                             }
                                         }
                                     }
                                     printf("%s", suma);}
{comparar}                            {comparar = 2; printf("comparacion = "); yyless(9);}
```

Figure 5: Expresiones para sumar y comparar listas.

Funcionalidades (VI)

```
{operador}                                {procesado = yytext;
                                           i = 0;
                                           if(procesado[i] == 'i')
                                               printf(" == ");
                                           else if(procesado[i] == 'd')
                                               printf(" != ");
                                           else if(procesado[i] == 'm') {
                                               if(procesado[i+1] == 'e' && yyleng == 5)
                                                   printf(" < ");
                                               else if(procesado[i+1] == 'e' && yyleng > 5)
                                                   printf(" <=" );
                                               else if(procesado[i+1] == 'a' && yyleng == 5)
                                                   printf(" > ");
                                               else
                                                   printf(" >=" );}}
{variable}                                {if (comparar > 0)
                                           comparar--;
                                           printf("%s", yytext);}
"mostrar comparacion"                    {printf("print(comparacion)");}
                                           {}
```

Figure 6: Expresiones para procesar un operador, una variable, mostrar el resultado de una comparación y regla por defecto.

Ejemplo: salida por defecto

```
hazret@hazret-GE63-T8D:~/Escritorio/ETSIIIT_comp/3*/MC/P2/ugr_modelos_computacion/src/P25 cat ejempl
ejemplo.txt
crear listas 'esto' 'es' 'la' 'primera' 'lista'
crear lista2
copiar lista1 lista2
imprimir lista1
recorrer lista2
comparar lista1 igual lista2
mostrar comparacion
crear listas 1 2 3 4
crear listas 4 3 2 1
concatenar lista1 lista4 lista_larga
ordenar lista_larga revers
recorrer lista_larga len
comparar lista3 menor lista4
mostrar comparacion
comparar lista1 diferente lista4
mostrar comparacion
comparar lista1 igual ordenar lista4
mostrar comparacion
longitud lista_larga
crear listas 5 6
comparar lista3 mas lista5 igual lista4 mas lista5 mas lista_larga
mostrar comparacion
comparar longitud lista1 igual longitud lista2
mostrar comparacion
comparar obtener lista1 0 diferente obtener lista2 2
mostrar comparacion
obtener lista_larga 1
crear lista_nueva
insertar lista_nueva 1 3
insertar lista_nueva 123.45
imprimir lista_nueva
recorrer lista_nueva len
imprimir lista_nueva
hazret@hazret-GE63-T8D:~/Escritorio/ETSIIIT_comp/3*/MC/P2/ugr_modelos_computacion/src/P25

hazret@hazret-GE63-T8D:~/Escritorio/ETSIIIT_comp/3*/MC/P2/ugr_modelos_computacion/src/P25 ./compyle
ejemplo_completo.txt
lista1 = ['esto', 'es', 'la', 'primera', 'lista']
lista2 = []
lista3 = lista1
print(lista1)
for item in lista1:
    print(item)
for item in lista2:
    print(item, end = ' ')
comparacion = lista1 == lista2
print(comparacion)
lista3 = [1, 2, 3, 4]
lista4 = [4, 3, 2, 1]
lista_larga = lista3 + lista4
lista_larga.sort(reverse=True)
for item in lista_larga:
    print(item, end = ' ')
comparacion = lista3 == lista4
print(comparacion)
comparacion = lista3 != lista4
print(comparacion)
comparacion = lista3 == lista4.sort()
print(comparacion)
len(lista_larga)
lista5 = [1, 4]
comparacion = lista3 + lista5 == lista4 + lista5 + lista_larga
print(comparacion)
comparacion = len(lista1) == len(lista2)
print(comparacion)
comparacion = lista1[0] != lista2[2]
print(comparacion)
print(lista_larga[1])
lista_nueva = []
lista_nueva.append(1.2)
lista_nueva.append(123.45)
lista_nueva.append(15)
print(lista_nueva)
lista_nueva.remove(15)
print(lista_nueva)
hazret@hazret-GE63-T8D:~/Escritorio/ETSIIIT_comp/3*/MC/P2/ugr_modelos_computacion/src/P25
```

Figure 7: Salida por pantalla de la funcionalidad.

Ejemplo: salida a un archivo .py

```
hazaret@nazaret-GE63-7RD:~/Escritorio/ETSIIT_comp/3º/MC/P2/ugr_modelos_computacion/src/P2$ ./compiler ejemplo-completo.txt > ejemplo_salida.py
hazaret@nazaret-GE63-7RD:~/Escritorio/ETSIIT_comp/3º/MC/P2/ugr_modelos_computacion/src/P2$ python3 ejemplo_salida.py
esto
es
la
primera
lista
esto_es_la_primera_lista_True

LaTeX

true
true
false
false
true
true

1.1, -123.45, 15]
1.1, -123.45]
```



Figure 8: Salida redirigida a un archivo .py y ejecutado en python.