**Sintaxis y Semántica del Lenguaje**

**Práctica Nº 5**

**GRAMATICAS FORMALES**

**Esta práctica deberá resolverse en la semana del 03/06/2019**

**2° Parte - Gramáticas libres de contexto**

En Python los tokens:

Newline indican salto de línea

Indent la tabulación de 4 espacios

Dedent la salida de la indentación

1. Definir la gramática para la siguiente sentencia Python: <iden>= <expresión>;

<iden>::= <iden><letra>| <iden><dig> |<iden>\_<letra>| <iden>\_<dig>| <letra>

puede tener guiones

<expresión>::=<expresión>+<término>| <expresión> - <término>|<término>

<termino>::= <termino> \* <factor> | <termino> / <factor> | <factor>

<factor>::=<iden> |<num>

<iden>::= <letra> | <iden>\_<letra>| <iden>\_<dig>|<iden><letra>| <iden><dig>

<letra>::=a|b|c|d…..…..….|z

<dig>::=0|1|2 …..…..….|9

<num>::=<num><dig>|<dig>

1. Escribir la gramática que permita validar la sentencia while de Python
2. Sin tener en cuenta la alineación

<while>::= while <condiciones>: <lista sentencias>

<condiciones>::= <condiciones><conector><unacond>|<unacond>

<unacond>::= <cond> | not <cond>

<cond>::= <expresión> <comparador> <expresión>

<comparador>::= <|>|== | != | >= | <=

<conector>::= and | or

<lista sentencias>::= <listasentencias><unasent> | <unasent>

<unasent>::= <while> | <if> | <if-elif>| …..

<expresión>::=<expresión>+<término>| <expresión> - <término>|<término>

<termino>::= <termino> \* <factor> | <termino> / <factor> | <factor>

<factor>::=<iden> |<num>

<iden>::= <letra> | <iden>\_<letra>| <iden>\_<dig>|<iden><letra>| <iden><dig>

<letra>::=a|b|c|d…..…..….|z

<dig>::=0|1|2 …..…..….|9

<num>::=><num><dig>|<dig>

1. Teniendo en cuenta la alineación

<while>::= while <condiciones>: <tabulación><lista sentencias> dedent

<tabulación>::= newline indent 🡪 son tokens (terminales) del lenguaje y los identifica como tal

<condiciones>::= <condiciones><conector><unacond>|<unacond>

<unacond>::= <cond> | not <cond>

<cond>::= <expresión> <comparador> <expresión>

<comparador>::= <|>|== | != | >= | <=

<conector>::= and | or

<lista sentencias>::= <listasentencias><unasent>|<unasent>

<unasent>::= <while> | <if> | <if-elif>| …..

<expresión>::=<expresión>+<término>| <expresión> - <término>|<término>

<termino>::= <termino> \* <factor> | <termino> / <factor> | <factor>

<factor>::=<iden> |<num>

<iden>::= <letra> | <iden>\_<letra>| <iden>\_<dig>|<iden><letra>| <iden><dig>

<letra>::=a|b|c|d…..…..….|z

<dig>::=0|1|2 …..…..….|9

<num>::=><num><dig>|<dig>

1. Escribir una gramática para validar la definición de una función en Python.(cabecera y cuerpo)
2. Sin tener en cuenta la alineación

<defineFun>::= def <iden>: <cuerpo>|

def <iden>(<param>): <cuerpo>

<cuerpo>::=<listasentencias> return <expresión>

<param>::= <param>,<unpar>| <unpar>

<unpar>::= <iden> | <iden>=<num>

<lista sentencias>::= <listasentencias><unasent>|<unasent>

<unasent>::= <while> | <if> | <if-elif>| …..

<expresión>::=<expresión>+<término>| <expresión> - <término>|<término>

<termino>::= <termino> \* <factor> | <termino> / <factor> | <factor>

<factor>::=<iden> |<num>

<iden>::= <letra> | <iden>\_<letra>| <iden>\_<dig>|<iden><letra>| <iden><dig>

<letra>::=a|b|c|d…..…..….|z

<dig>::=0|1|2 …..…..….|9

<num>::=><num><dig>|<dig>

1. Teniendo en cuenta la alineación

<defineFun>::= def <iden>: <tabulación> <cuerpo> dedent |

def <iden>(<param>): <tabulación> <cuerpo> dedent

<cuerpo>::=<listasentencias> return <expresión>

<param>::= <param>,<unpar>| <unpar>

<unpar>::= <iden> | <iden>=<num>

<listasentencias>::= <listasentencias><unasent>|<unasent>

<unasent>::= <while> | <if> | <if-elif>| …..

<expresión>::=<expresión>+<término>| <expresión> - <término>|<término>

<termino>::= <termino> \* <factor> | <termino> / <factor> | <factor>

<factor>::=<iden> |<num>

<iden>::= <letra> | <iden>\_<letra>| <iden>\_<dig>|<iden><letra>| <iden><dig>

<letra>::=a|b|c|d…..…..….|z

<dig>::=0|1|2 …..…..….|9

<num>::=><num><dig>|<dig>

1. Escribir la gramática que valide la sentencia import de Python

<import>::= import <listamodulos>

<listamodulos>::=<unmodulo> | <unmodulo>,<listamodulos>

<unmodulo>::= os|sys|math| <iden>| ……

<iden>::= <letra> | <iden>\_<letra>| <iden>\_<dig>|<iden><letra>| <iden><dig>

<letra>::=a|b|c|d…..…..….|z

<dig>::=0|1|2 …..…..….|9

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Fuera de práctica

<importar>::=from <iden> import <varios>

<varios>::=\*| <listamodulos>

<listamodulos>::=<unmodulo> | <unmodulo>,<listamodulos>

<unmodulo>::= os|sys|math| <iden>| ……

<iden>::= <letra> | <iden>\_<letra>| <iden>\_<dig>|<iden><letra>| <iden><dig>

<letra>::=a|b|c|d…..…..….|z

<dig>::=0|1|2 …..…..….|9

<for>::= for <iden> in range (<rango\_valores>):<lista\_sentencias>

<lista\_sentencias>::=<sentencia><lista\_sentencias> | <sentencia>

<sentencia>::=<if>|<if-elif>|<for>|<while>|....

<rango\_valores>::=<expresion>|<expresion>,<expresion>|<expresion>,<expresion>,<expresion>

<expresión>::=<expresión>+<término>| <expresión> - <término>|<término>

<termino>::= <termino> \* <factor> | <termino> / <factor> | <factor>

<factor>::=<iden> |<num>

<iden>::= <letra> | <iden>\_<letra>| <iden>\_<dig>|<iden><letra>| <iden><dig>

<letra>::=a|b|c|d…..…..….|z

<dig>::=0|1|2 …..…..….|9

<num>::=><num><dig>|<dig>

<if>::=if <condiciones>:<lista\_sentencias> |

If <condiciones>:<lista\_sentencias>else:<lista\_sentencias>

<listasentencias>::= <listasentencias><unasent>|<unasent>

<unasent>::= <while> | <if> | <if-elif>| …..

<expresión>::=<expresión>+<término>| <expresión> - <término>|<término>

<termino>::= <termino> \* <factor> | <termino> / <factor> | <factor>

<factor>::=<iden> |<num>

<iden>::= <letra> | <iden>\_<letra>| <iden>\_<dig>|<iden><letra>| <iden><dig>

<letra>::=a|b|c|d…..…..….|z

<dig>::=0|1|2 …..…..….|9

<num>::=><num><dig>|<dig>

<if-elif>::= if <condiciones>:<lista\_sentencias><varios\_elif>else:<lista\_sentencias>

<varios\_elif>::=<unelif>|<varios\_elif><unelif>

<unelif>::= elif <condiciones>:<lista\_sentencias>

<lista\_sentencias>::=<sentencia><lista\_sentencias> |<sentencia>

<sentencia>::=<if>|<if-elif>|<for>|<while>|....

<condiciones>::= <condiciones><conector><unacond>|<unacond>

<unacond>::= <cond> | not <cond>

<cond>::= <expresión> <comparador> <expresión>

<comparador>::= <|>|== | != | >= | <=

<conector>::= and | or

<expresión>::= <expresión> + <termino> | <expresión> - <termino> | <termino>

<termino>::= <termino> \* <factor> | <termino> / <factor> | <factor>

<factor>::= <ID> | <numero>

<numero>::= <digito><numero> | <digito>

<ID>::= <ID><letra> | <ID><digito> | <letra>

<letra>::= a | b | ... | z

<digito>::= 0 | 1 | ... | 9