

SINTÁXIS



Sintáxis y Semántica

Un lenguaje de programación es una notación formal para describir algoritmos a ser ejecutados en una computadora

- Lenguaje de programación
 - Sintaxis
 - Semántica



Sintáxis y Semántica

■ Definiciones.

- Sintáxis: Conjunto de reglas que definen como componer letras, dígitos y otros caracteres para formar los programas
- Semántica: Conjunto de reglas para dar significado a los programas sintácticamente válidos.

v: array [1..10] of integer; ----- en Pascal
y
int v[10]; ----- en C



Sintáxis y Semántica

- ¿Cuál es la utilidad de definir y conocer la sintáxis y la semántica de un lenguaje? ¿Quiénes se benefician?
 - Programadores
 - Implementador (Compilador)
- La definición de la sintáxis y la semántica de un lenguaje de programación proporcionan mecanismos para que una persona o una computadora pueda decir:
 - Si el programa es válido y
 - Si lo es, qué significa



Sintáxis

■ **Características de la sintáxis**

- La sintáxis debe ayudar al programador a escribir programas correctos sintácticamente
- La sintáxis establecen reglas que sirven para que el programador se comuniquen con el procesador
- La sintáxis debe contemplar soluciones a características tales como:
 - Legibilidad
 - Verificabilidad
 - Traducción
 - Falta de ambigüedad



Sintáxis

La sintáxis establece reglas que definen cómo deben combinarse las componentes básicas, llamadas "**word**", para formar sentencias y programas.

- **Elementos de la sintáxis**
 - Alfabeto o conjunto de caracteres
 - identificadores
 - Operadores
 - Palabra clave y palabra reservada
 - Comentarios y uso de blancos



Alfabeto o conjunto de caracteres

Latin-1 (ISO-8859-1: Western European)

El código ASCII

Cyrillic (ISO-8859-5)

Arabic (ISO-8859-6)

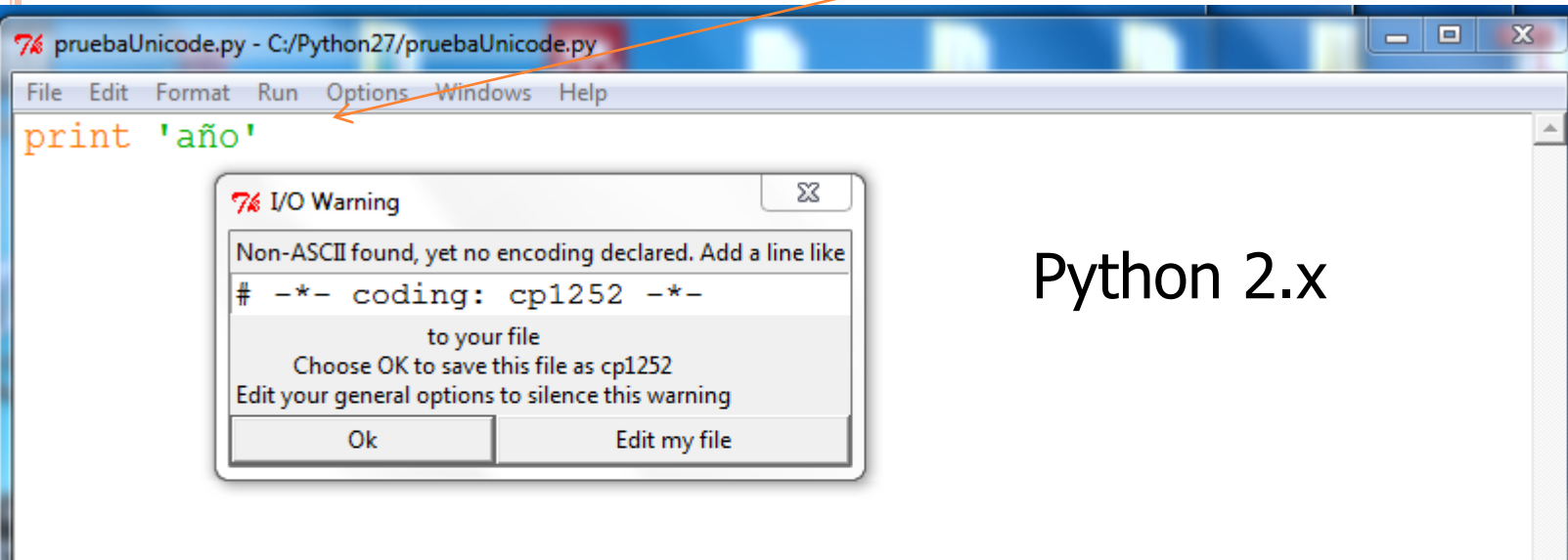
Greek (ISO-8859-7)

UNICODE

[illegible]

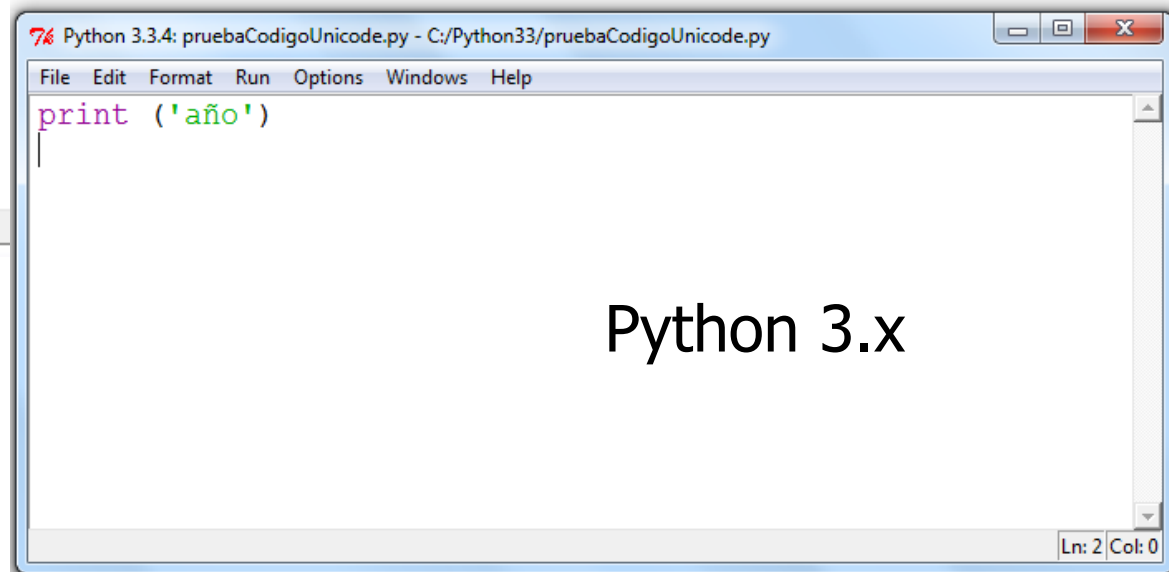
Sintáxis

-*- coding: utf-8 -*-



Python 2.x

caracteres
hora de
rácter la



Python 3.x

Sintáxis

■ Identificadores

- Elección más ampliamente utilizada: Cadena de letras y dígitos, que deben comenzar con una letra
- Si se restringe la longitud se pierde legibilidad

■ Operadores

- Con los operadores de suma, resta, etc. la mayoría de los lenguajes utilizan +, -. En los otros operadores no hay tanta uniformidad

■ Comentarios

- Hacen los programas más legibles

"El código es leído muchas más veces de lo que es escrito". Guido Van Roussen.



Sintáxis

■ Palabra clave y palabra reservada

Array do else if

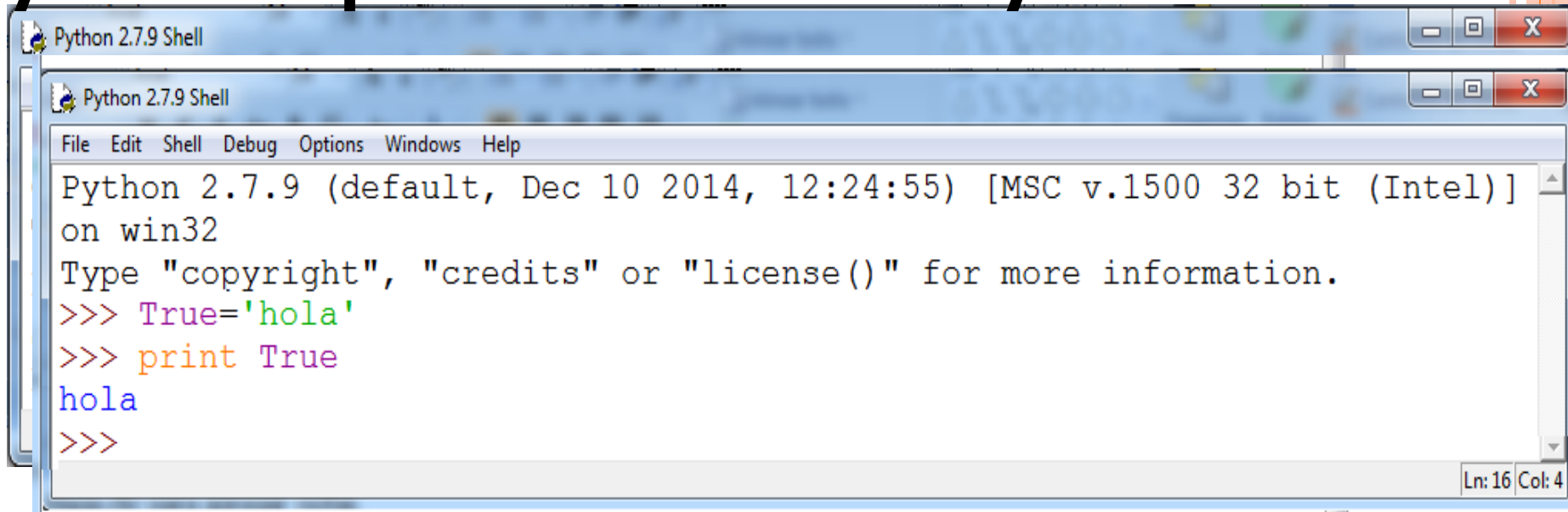
- Palabra **clave** o **keywords**, son palabras claves que tienen un significado dentro de un contexto.
- Palabra **reservada**, son palabras claves que además no pueden ser usadas por el programador como identificador de otra entidad.
- Ventajas de su uso:
 - Permiten al compilador y al programador expresarse claramente
 - Hacen los programas más legibles y permiten una rápida traducción
- Soluciones para evitar confusión entre palabras claves e identificadores
 - Usar palabras reservadas

Ejemplos de lenguajes con uso de palabras reservadas:

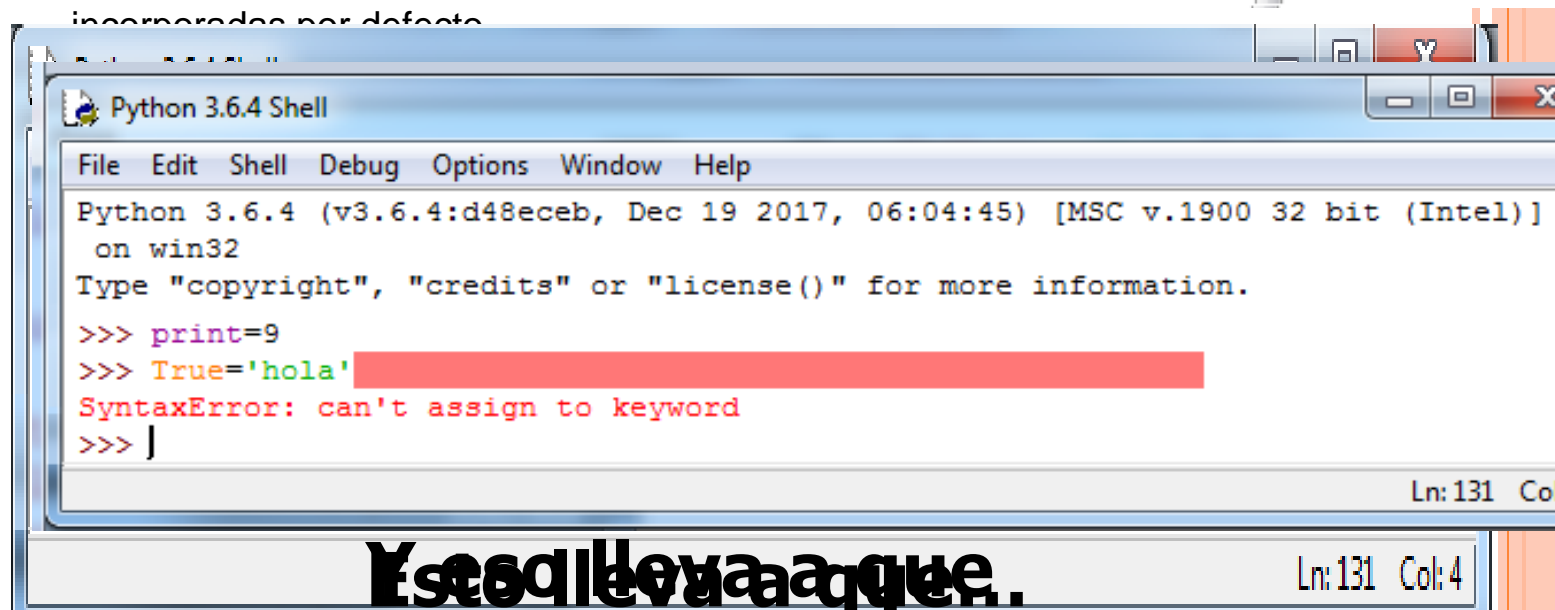
- **C** ej.: auto, break, case, char, const, continue, default, do, double, else, enum, extern, float, for, goto, if, int, etc
- **Pascal** ej.: absolute, and, array, begin, const, div, do, downto, else, if, in, label, mod, not, of, packed, procedure, record, set, shr, then, to, unit, uses, var, while, xor, etc

Sintaxis

Python: las palabras reservadas y sus versiones...



```
Python 2.7.9 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.9 (default, Dec 10 2014, 12:24:55) [MSC v.1500 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>> True='hola'
>>> print True
hola
>>>
```



```
Python 3.6.4 Shell
File Edit Shell Debug Options Window Help
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>> print=9
>>> True='hola'
SyntaxError: can't assign to keyword
>>> |
```

Esquema de...

Ln: 131 Col: 4

Sintáxis

■ Estructura sintáctica

■ **Vocabulario o words**

- Conjunto de caracteres y palabras necesarias para construir expresiones, sentencias y programas. Ej: identificadores, operadores, palabras claves, etc.

Las words no son elementales se construyen a partir del alfabeto

■ **Expresiones**

- Son funciones que a partir de un conjunto de datos devuelven un resultado.
- Son bloques sintácticos básicos a partir de los cuales se construyen las sentencias y programas

■ **Sentencias**

- Componente sintáctico más importante.
- Tiene un fuerte impacto en la facilidad de escritura y legibilidad
- Hay sentencias simples, estructuradas y anidadas.



Sintáxis

■ Reglas léxicas y sintácticas.

- Diferencias entre mayúsculas y minúsculas
- Símbolo de distinto. En C != en Pascal <>
- Reglas léxicas: Conjunto de reglas para formar las "**word**", a partir de los caracteres del alfabeto
- Reglas sintácticas: Conjunto de reglas que definen como formar las "**expresiones**" y "**sentencias**"

•El If en C no lleva "then", en Pascal si

La diferencia entre léxico y sintáctico es arbitrario, dan la apariencia externa del lenguaje



Sintáxis

■ Tipos de Sintáxis

■ **ABSTRACTA**

- Se refiere básicamente a la estructura

■ **CONCRETA**

- Se refiere básicamente a la parte léxica

■ **PRAGMÁTICA**

- Se refiere básicamente al uso práctico



Sintáxis

Ejemplo de sintáxis concreta y abstracta:.

while (x != y)

Uso de paréntesis

{

Forma de
encerrar un
bloque

(En C)

while x <> y do

begin

Símbolo de distinto

end

(En Pascal)

- Son diferentes respecto a la **sintáxis concreta**, porque existen diferencias léxicas entre ellas
- Son iguales respecto a la **sintáxis abstracta**, ya que ambas tienen la misma estructura

while condición
bloque



Sintáxis

Ejemplo de sintáxis pragmática:.

Ej1.

<> es mas legible que **!=**

Ej2.

En C y Pascal {} o begin-end pueden omitirse si el bloque esta compuesto por una sola sentencia

while (x!=y) x=y+1

En Modula:

If x=y then

end

Pragmáticamente puede considerarse que si se necesitara agregar una sentencia debe agregarse el begin end o las {}.

Sintáxis

■ **Cómo definir la sintáxis**

- Se necesita una descripción finita para definir un conjunto infinito (conjunto de todos los programas bien escritos)
- Formas para definir la sintaxis:
 - Lenguaje natural. Ej.: Fortran
 - Utilizando la gramática libre de contexto, definida por Backus y Naun: BNF. Ej: Algol
 - Diagramas sintácticos son equivalentes a BNF pero mucho mas intuitivos



Sintáxis

■ BNF (Backus Naun Form)

- Es una notación formal para describir la sintaxis
- Es un metalenguaje
- Utiliza metasímbolos
 - $< > ::= |$
- Define las reglas por medio de “producciones”

Ejemplo:

$< \text{digito} > ::= 0|1|2|3|4|5|6|7|8|9$

No terminal

Se define como

Metasímblo

Terminales



Sintáxis

■ Gramática

- Conjunto de reglas finita que define un conjunto infinito de posibles sentencias válidas en el lenguaje.
- Una gramática esta formada por una 4-tupla

$$\mathbf{G} = (\mathbf{N}, \mathbf{T}, \mathbf{S}, \mathbf{P})$$

Conjunto de
símbolos no
terminales

Conjunto de
símbolos
terminales

Símbolo distinguido
de la gramática que
pertenece a N

Conjunto de
producciones



Sintáxis

■ Árboles sintácticos

“Juan un canta manta”

- Es una oración sintácticamente incorrecta
- No todas las oraciones que se pueden armar con los terminales son válidas
- Se necesita de un **Método de análisis (reconocimiento)** que permita determinar si un string dado es valido o no en el lenguaje:
Parsing.
- El **parse**, para cada sentencia construye un **“árbol sintáctico o árbol de derivación”**

Sintáxis

■ Árboles sintácticos

■ Dos maneras de construirlo:

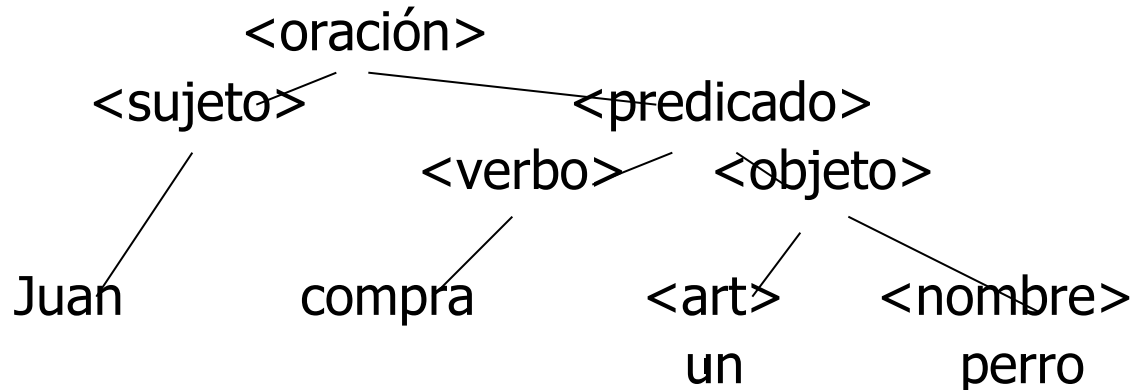
■ Método bottom-up

- De izquierda a derecha
- De derecha a izquierda

■ Método top-down

- De izquierda a derecha
- De derecha a izquierda

Ejemplo: árbol sintáctico de "oración". Top-down de izquierda a derecha



Sintáxis

■ Árbol de derivación:

- Ejemplo top-down de izquierda a derecha

<oración>	=>	<sujeto><predicado>
	=>	Juan <predicado>
	=>	Juan <verbo><objeto>
	=>	Juan compra <objeto>
	=>	Juan compra art><sustan>
	=>	Juan compra un <sustan>
	=>	Juan compra un perro

- Los compiladores utilizan el parse canónico que es el bottom-up de izquierda a derecha