Universidad de Oviedo

Otros lenguajes OOP

Ignacio Alvarez García

Sept - 2024

Otros lenguajes OOP

- Muchos lenguajes de programación admiten OOP:
  - Lenguajes compilados:
    - El compilador genera el código máquina de golpe a partir del fuente
    - Hay que compilar versión para cada tipo de máquina destino
    - En tiempo de ejecución el fuente no es usado, sólo requiere librerías ya compiladas
    - El ejecutable es pequeño y rápido
    - Los cambios requieren recompilación
  - Lenguajes interpretados:
    - El intérprete genera el código de máquina a partir del fuente según se avanza en la ejecución
    - En tiempo de ejecución es necesario el código fuente, el intérprete, y las librerías (muchas en forma de fuente, otras ya compiladas)
    - El resultado es más grande y lento
    - Se pueden hacer cambios sobre la marcha
  - Seguridad (en combinación con S.O.):
    - Lenguajes seguros: hacen comprobaciones internas que impiden determinados accesos por parte del programa
    - Lenguajes inseguros: no hacen comprobaciones, el programa tiene acceso a todo lo que el S.O. le permita

Otros lenguajes OOP

- Muchos lenguajes de programación admiten OOP:
  - **C++**: de propósito general, compilado, inseguro, utilizable en casi todo tipo de dispositivos destino
  - **Python**: de propósito general, interpretado, semi-seguro, utilizable en casi todo tipo de dispositivos destino (excepto los muy pequeños)
  - **Java**: de propósito general, compilado+interpretado, seguro, utilizable en menos dispositivos destino (típico en Android)
  - **Javascript**: orientado a navegadores web, interpretado, seguro
  - **Matlab**: para cálculo matemático, interpretado, semi-seguro, requiere dispositivos potentes
  - **C#** (Csharp): similar a Java, compilado, orientado a aplicaciones web/gui en MS Windows
  - **Otros**: Ruby, R, PHP, Perl, …

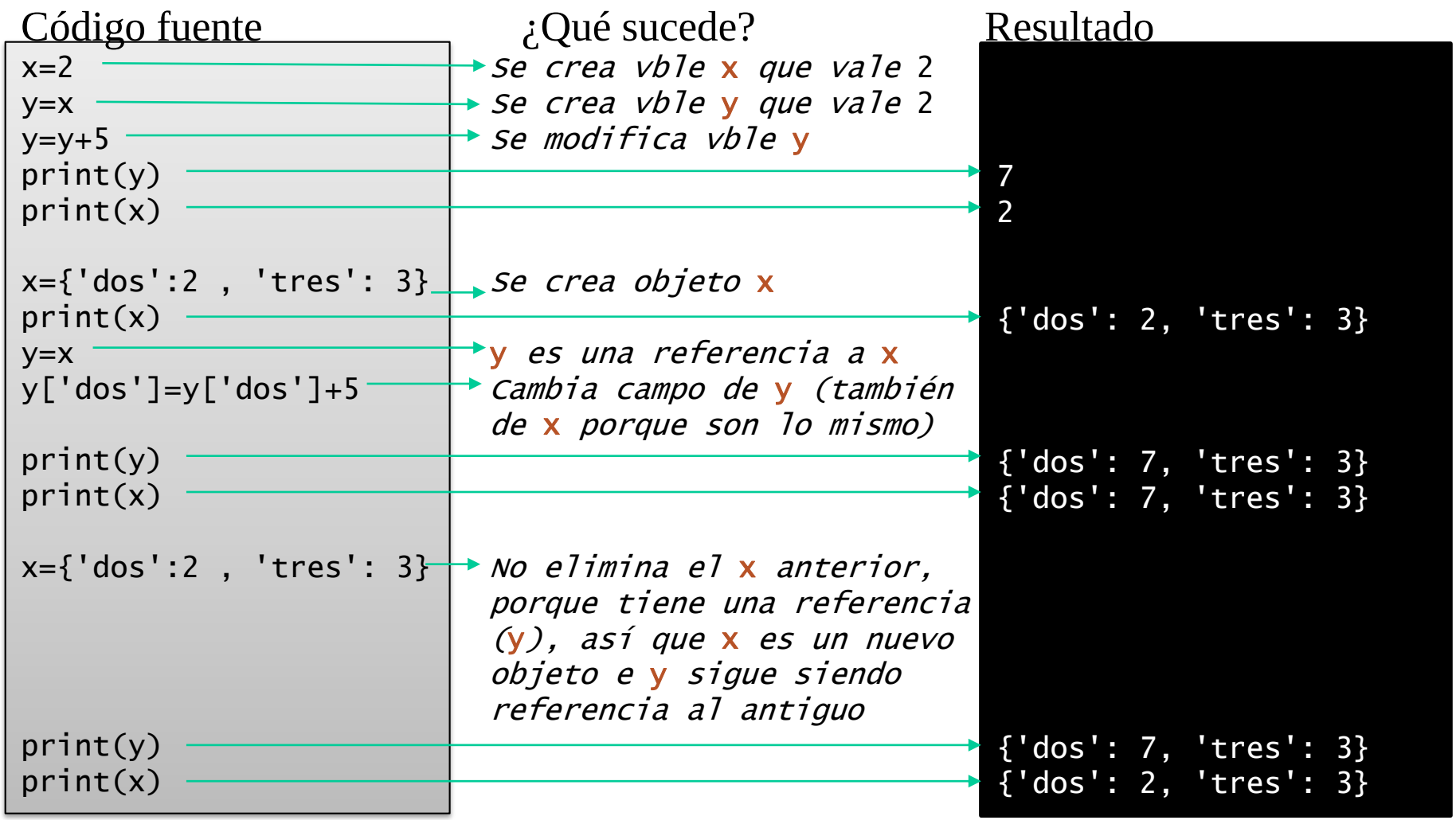
Muchas empresas imponen el lenguaje de programación a utilizar, según el destino de la aplicación, su experiencia o conocimientos previos, etc.

Otros lenguajes OOP

- Las diferencias fundamentales:
  - **Formato**: cada lenguaje requiere un formato específico
  - **Declaración variables**: los lenguajes interpretados no suelen requerir declaración previa de variables
  - **Tipos de datos**: muchos lenguajes asumen tipos por defecto para los datos según la asignación (a veces, no son lo que parecen), e incluso permiten que los datos cambien de tipo al realizar nuevas asignaciones
  - **Referencias**: muchos lenguajes trabajan por defecto con referencias (no copias) al usar clases y objetos
  - **Punteros**: la mayoría de lenguajes no admiten punteros
  - **Librerías**: cada lenguaje dispone de librerías estándar de funciones/clases, y otras muchas aportadas por desarrolladores

Python

- Ejemplo básico:



HTML y JavaScript

- Ejemplo básico:

```
<!DOCTYPE HTML>
<html>
<head>
  <title>TEST JAVASCRIPT</title>
</head>

<body>
    <h2>TEST JAVASCRIPT</h2>
    <div>
        Value 1: <input id="value1" type="number" min="0" max="99" step="1" value="8" size="2" style="width:40px">
        Value 2: <input id="value2" type="number" min="0" max="99" step="1" value="6" size="2" style="width:40px">
        <button id="id_Button_Add" onclick="Add()">ADD</button>
        <button id="id_Button_Sub" onclick="Sub()">SUB</button>
    </div>
    <div>
        <p><b>Resultado (mal):</b><input id="result_wrong" type="number" size="4" style="width:40px" readonly></p>
        <p><b>Resultado (bien):</b><input id="result_ok" type="number" size="4" style="width:40px" readonly></p>
    </div>

    <script>
        var v1=document.getElementById('value1');
        var v2=document.getElementById('value2');
        var r_ok=document.getElementById('result_ok');
        var r_fail=document.getElementById('result_wrong');

        function Add()    {
            r_fail.value=v1.value+v2.value;                    // Resultado incorrecto: suma cadenas de caracteres
            r_ok.value=Number(v1.value)+Number(v2.value);      // Resultado correcto: suma números

        }
        function Sub()   {
            r_fail.value=v1.value-v2.value;    // Ahora asume números (la resta no existe para cadenas)
            r_ok.value=Number(v1.value)-Number(v2.value);
        }
    </script>
</body>
```

Test

Otros lenguajes OOP

- Conclusiones:
  - **Elegir** (si nos permiten) el lenguaje más apropiado para la aplicación, según su destino, las librerías disponibles, etc.
  - **Aprender** el formato específico del lenguaje
  - **Comprobar** cómo trabaja el lenguaje con datos, tipos, etc. (no fiarse de lo que parece, salvo en C++)