Conceptos básicos
Fundamentos de la programación

Salvador Sánchez, Miguel A. Sicilia

Universidad de Alcalá

Septiembre de 2015

Licencia

Resolver problemas

- **Problema**: enunciado $\rightarrow$ estrategia de resolución $\rightarrow$ solución

Resolver problemas

- **Problema**: enunciado $\rightarrow$ estrategia de resolución $\rightarrow$ solución
- Un algoritmo es un método para resolver un problema

Resolver problemas

- **Problema**: enunciado $\rightarrow$ estrategia de resolución $\rightarrow$ solución
- Un algoritmo es un método para resolver un problema
  - Al-Khwārizmī: matemático persa (siglo IX) que enunció reglas paso a paso para la resolución de operaciones aritméticas de números decimales

Resolver problemas

- **Problema**: enunciado $\rightarrow$ estrategia de resolución $\rightarrow$ solución
- Un algoritmo es un método para resolver un problema
  - Al-Khwārizmī: matemático persa (siglo IX) que enunció reglas paso a paso para la resolución de operaciones aritméticas de números decimales
- Uno de los **objetivos fundamentales** de esta asignatura es aprender a diseñar algoritmos

Algoritmo

- Una receta es un algoritmo, pues describe cómo llevar a cabo un cierto trabajo, en este caso, la elaboración de una **Tortilla francesa**
  - Cascar un huevo de gallina en un plato
  - Batirlo con un tenedor
  - Calentar 10cc de aceite de oliva en una sartén
  - Cuando el aceite esté caliente, verter el contenido del plato
  - Darle la forma deseada volteando el huevo con un tenedor
  - Quitar la sartén del fuego una vez terminada

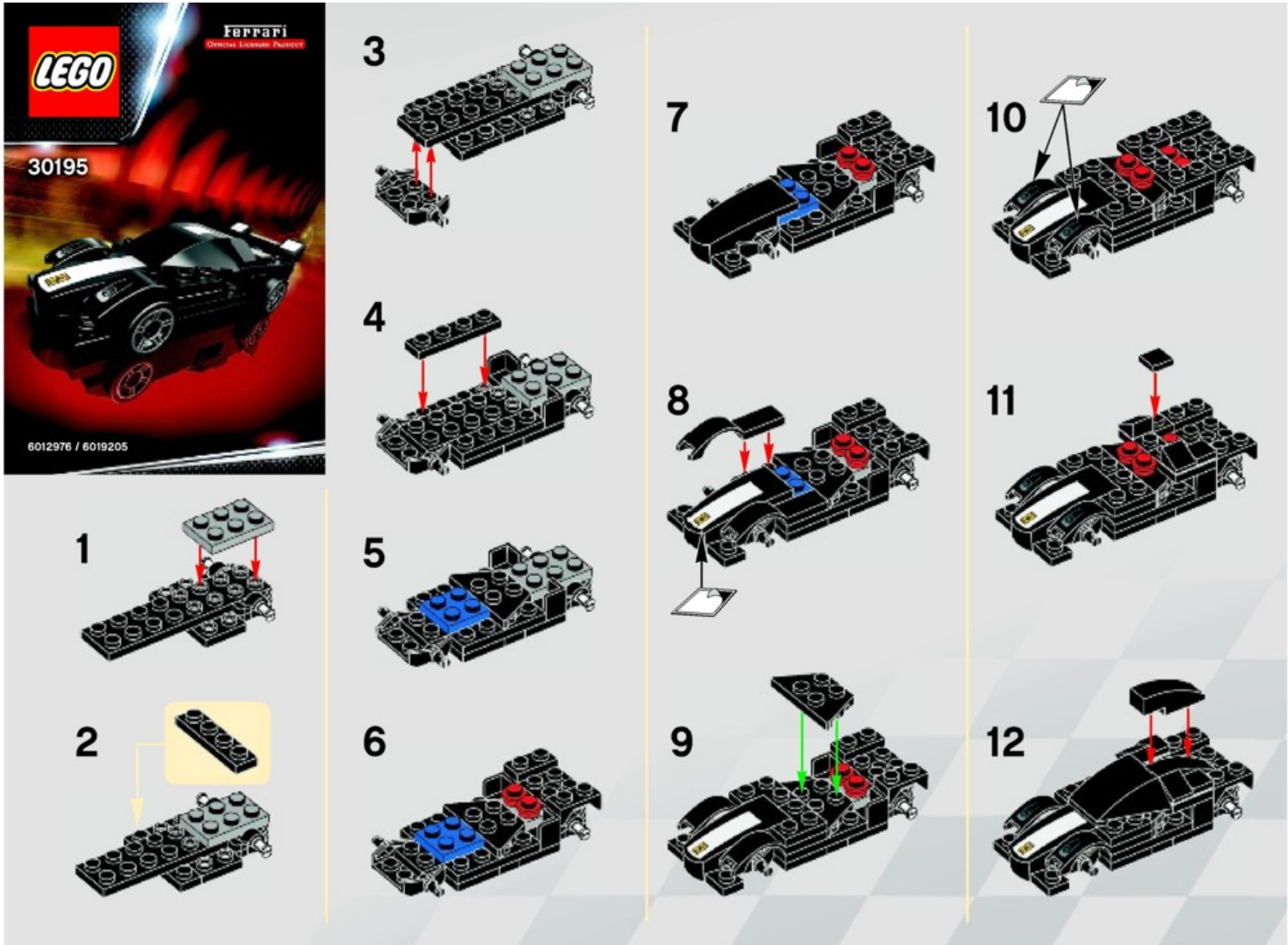
Algoritmo

- Una receta es un algoritmo, pues describe cómo llevar a cabo un cierto trabajo, en este caso, la elaboración de una **Tortilla francesa**
  - Cascar un huevo de gallina en un plato
  - Batirlo con un tenedor
  - Calentar 10cc de aceite de oliva en una sartén
  - Cuando el aceite esté caliente, verter el contenido del plato
  - Darle la forma deseada volteando el huevo con un tenedor
  - Quitar la sartén del fuego una vez terminada

Algoritmos para humanos

- Otro algoritmo (muy popular)...
  - Estirar el brazo izquierdo
  - Estirar el brazo derecho
  - Tocar el hombro derecho con brazo izquierdo
  - Tocar el hombre izquierdo con bazo derecho
  - Poner la mano izquierda en la nuca
  - Poner la mano derecha en la nunca
  - Poner la mano izquierda en la cadera
  - Poner la mano derecha en la caldera
  - Giro completo de la cadera
  - Saltar y girar 90° a la derecha

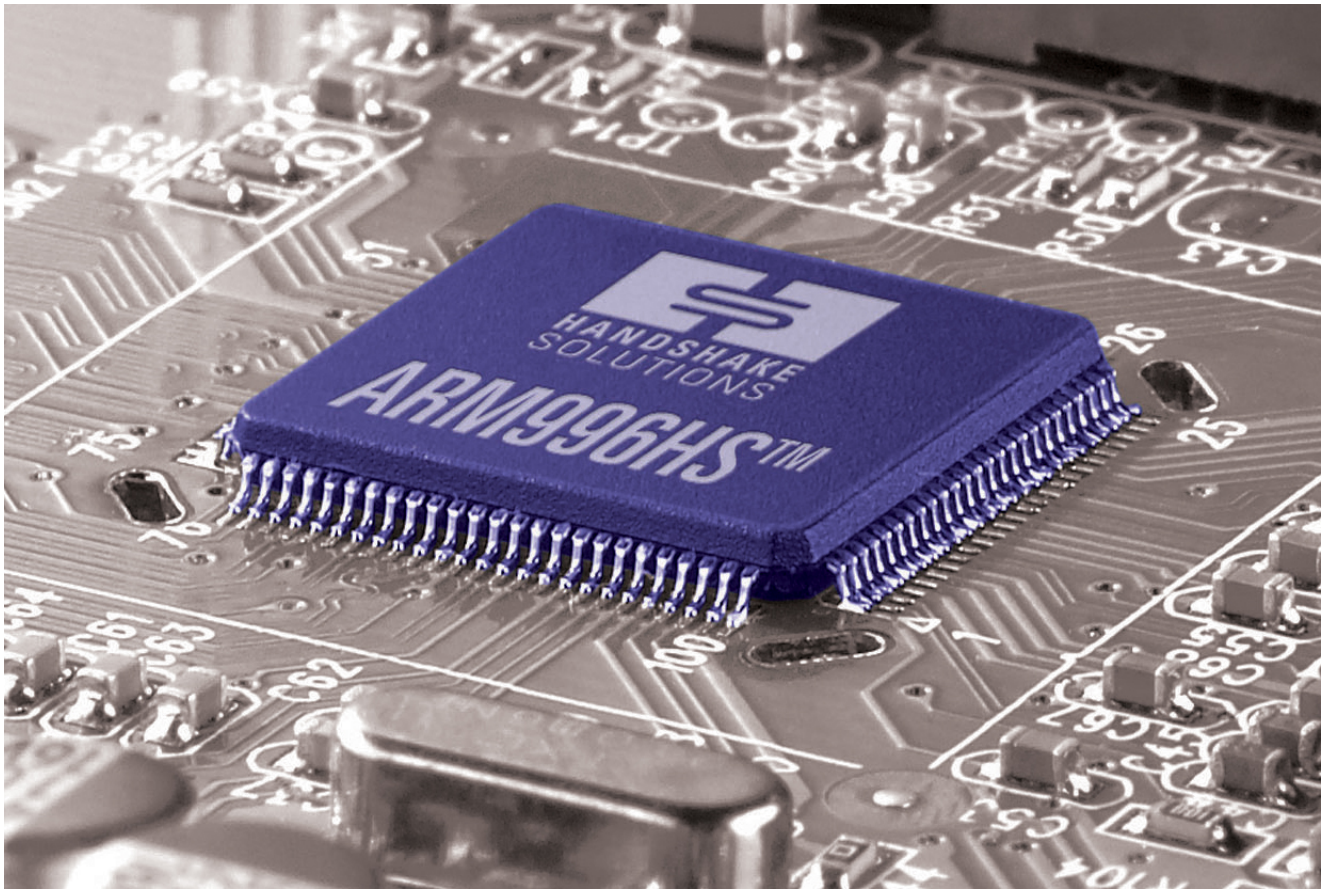
Algoritmos para humanos



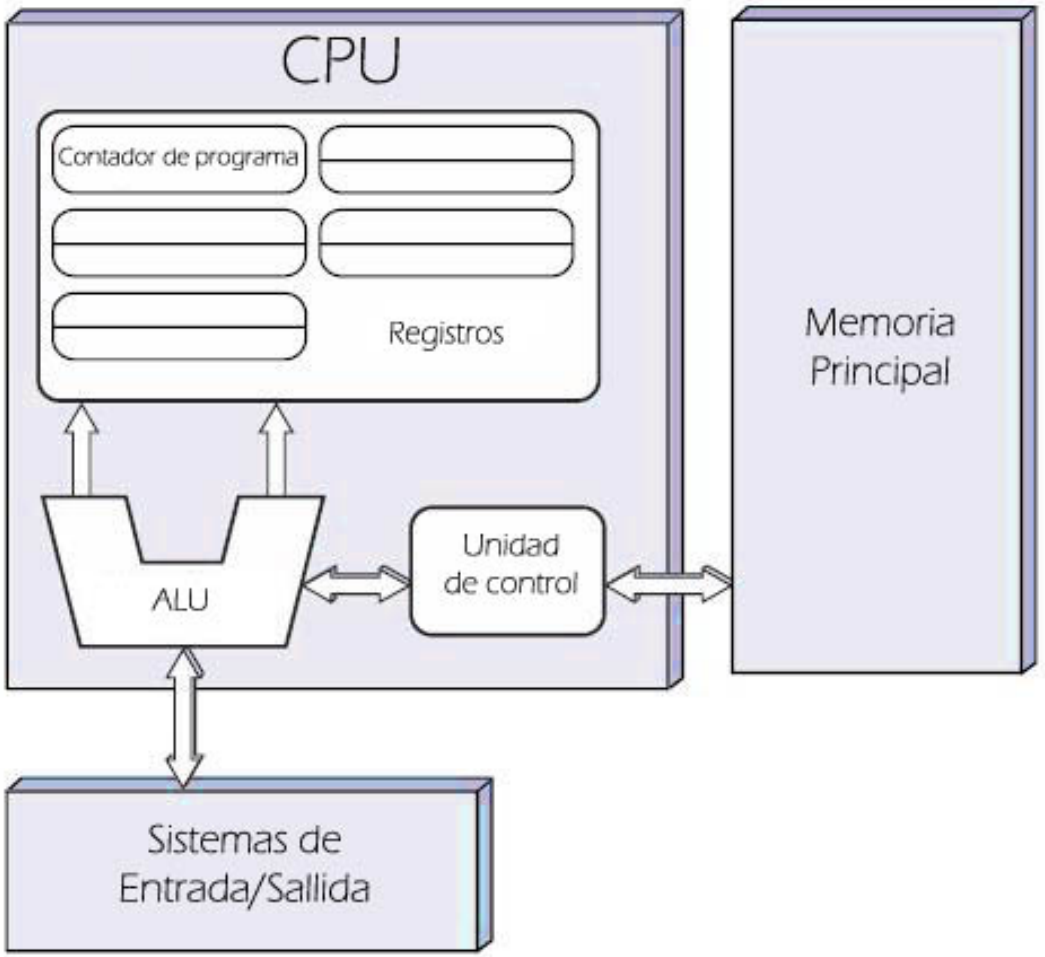
Procesadores

- Un **procesador** es toda entidad capaz de comprender un algoritmo y llevar a cabo el trabajo indicado en el mismo
  - En los dos primeros ejemplos, una persona que sepa leer español y que disponga de los utensilios necesarios
  - En el ejemplo de Lego, cualquiera que sepa interpretar los diagramas
- Un algoritmo puede expresarse en diferentes lenguajes
- Si se desea que lo procese un ordenador, hay que emplear un lenguaje especial, llamado lenguaje de programación
- El **microprocesador** de un ordenador es un procesador capaz de ejecutar instrucciones mediante operaciones simples y accesos a la memoria del ordenador.

Un microprocesador



Esquema simplificado de un ordenador



Resolución de problemas con ordenador

- Ayudado por un ordenador, la resolución de un problema consiste en:

Resolución de problemas con ordenador

- Ayudado por un ordenador, la resolución de un problema consiste en:
  - Analizar el problema
  - Diseñar una estrategia de resolución o algoritmo
  - Resolver el algoritmo en el ordenador

Resolución de problemas con ordenador

- Ayudado por un ordenador, la resolución de un problema consiste en:
  - Analizar el problema
  - Diseñar una estrategia de resolución o algoritmo
  - Resolver el algoritmo en el ordenador
- Se trata de un proceso iterativo e incremental
- No termina con la entrega del programa: mantenimiento

Resolución de problemas con ordenador

- **Análisis**: ¿qué datos necesito para resolverlo? ¿qué información ha de producir su resolución?

Resolución de problemas con ordenador

- **Análisis**: ¿qué datos necesito para resolverlo? ¿qué información ha de producir su resolución?
- **Diseño**: Descomponer el problema en problemas más simples. Especificar los pasos para cada subproblema.

Resolución de problemas con ordenador

- **Análisis**: ¿qué datos necesito para resolverlo? ¿qué información ha de producir su resolución?
- **Diseño**: Descomponer el problema en problemas más simples. Especificar los pasos para cada subproblema.
- **Resolución**: Codificación del algoritmo en un lenguaje de programación. Ejecución del programa y comprobación.

Lenguajes de programación

- Un **lenguaje de programación** es un lenguaje formal concebido para comunicar instrucciones a un ordenador.

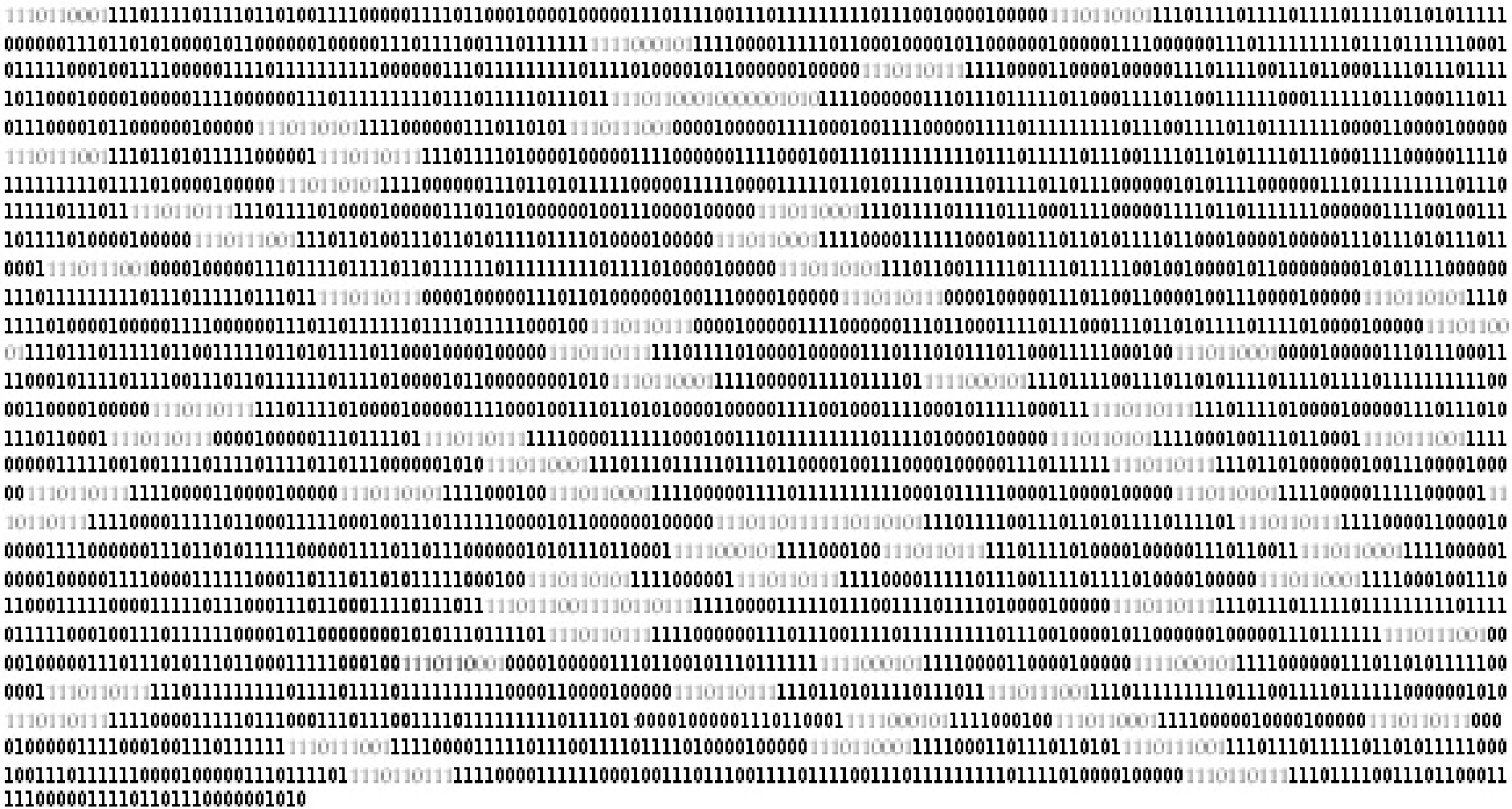
Lenguajes de programación

- Un **lenguaje de programación** es un lenguaje formal concebido para comunicar instrucciones a un ordenador.
- Un **programa** es la representación de un algoritmo en un lenguaje de programación que permite ejecutar dicho algoritmo en un ordenador.

Lenguajes de programación

- Un **lenguaje de programación** es un lenguaje formal concebido para comunicar instrucciones a un ordenador.
- Un **programa** es la representación de un algoritmo en un lenguaje de programación que permite ejecutar dicho algoritmo en un ordenador.
- La representación se puede hacer a varios niveles:
  - **Lenguaje máquina** (código binario: por ejemplo el conjunto de instrucciones para los microprocesadores de la familia x86).
  - **Lenguaje ensamblador** (por ejemplo, la sintaxis Intel para el conjunto de instrucciones x86).
  - **Lenguajes de alto nivel** (por ejemplo, C, Java, Python, etc.).

Código máquina



Ensamblador

```
008548ED: jmp    854927H
008548EF: cmp    esi,18H
008548F2: jne    8548E8H
008548F4: cmp    byte [edi][48H],0
008548F8: jne    854927H
008548FA: mov    eax,edi
008548FC: call   near 855118H
00854901: mov    eax,[eax][33H]
00854904: je     854927H
00854907: and    eax,00000200
0085490C: cmp    eax,0
0085490F: je     854927H
00854911: push   0
00854913: lea    ebx,[edi]
00854915: lea    ecx,[edi][4]
00854918: mov    eax,00000000
```

Lenguajes de programación de alto nivel

- Dentro de los de alto nivel, hay diferentes formas de clasificarlos.
  - http://en.wikipedia.org/wiki/Category:Programming_language_classification

Lenguajes de programación de alto nivel

- Dentro de los de alto nivel, hay diferentes formas de clasificarlos.
  - http://en.wikipedia.org/wiki/Category:Programming_language_classification
- Desde la aparición de Fortran en 1954, se han contado al menos 2300 lenguajes de alto nivel. Sus relaciones se pueden consultar aquí: http://www.digibarn.com/collections/posters/tongues/

Lenguajes de programación de alto nivel

- Dentro de los de alto nivel, hay diferentes formas de clasificarlos.
  - http://en.wikipedia.org/wiki/Category:Programming_language_classification
- Desde la aparición de Fortran en 1954, se han contado al menos 2300 lenguajes de alto nivel. Sus relaciones se pueden consultar aquí: http://www.digibarn.com/collections/posters/tongues/
- Muchos de ellos están "en peligro de extinción".

Un ejemplo

```c
/* Un programa sencillo en lenguaje C */
#include <stdio.h>
int main(void)
{
    printf("En un lugar de la Mancha, de cuyo nombre no
        quiero acordarme...");
    return 0;
}
```

Un ejemplo

```
#Un programa sencillo en python
print ("En un lugar de la Mancha,")
print ("de cuyo nombre no quiero acordarme,")
print ("no ha mucho tiempo que vivia un hidalgo...")
```

Cómo se expresan los algoritmos

- Lenguaje natural.
  - Problema: Calcular el máximo común divisor de dos enteros positivos.

Cómo se expresan los algoritmos

- Lenguaje natural.
  - Problema: Calcular el máximo común divisor de dos enteros positivos.
  - Algoritmo (Euclides): *Tomar los dos enteros, y calcular un nuevo par formado por el menor de ellos y el resto de la división entera de dividir el mayor entre el menor. Repetir el proceso hasta que los números en el par sean el mismo o el resto de la división sea cero. El resultado será el divisor de la última división*.

Cómo se expresan los algoritmos

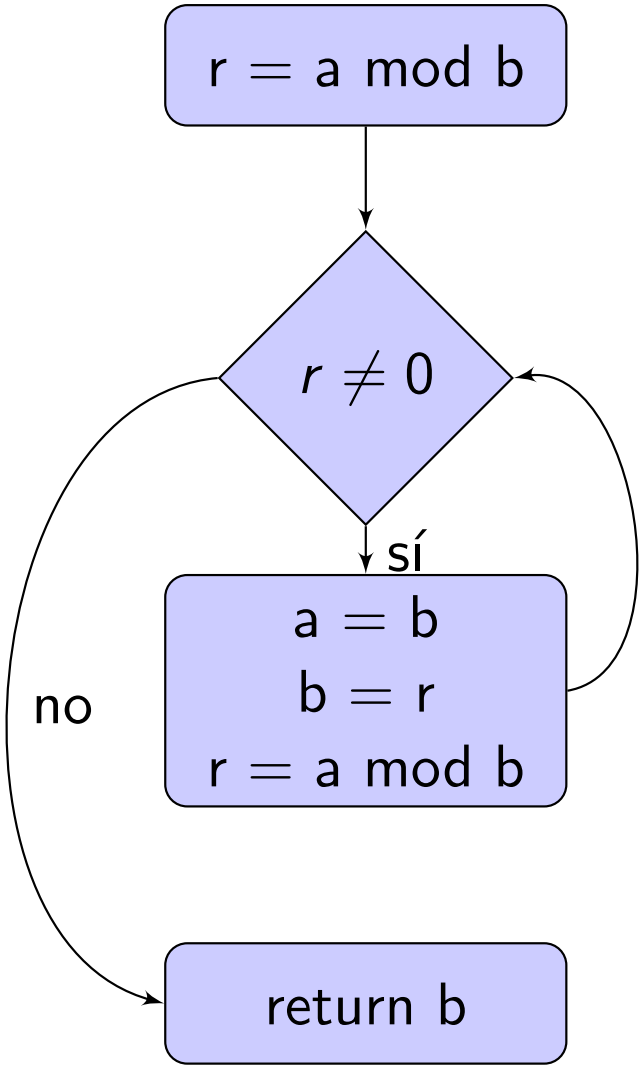
- Lenguaje natural.
  - Problema: Calcular el máximo común divisor de dos enteros positivos.
  - Algoritmo (Euclides): *Tomar los dos enteros, y calcular un nuevo par formado por el menor de ellos y el resto de la división entera de dividir el mayor entre el menor. Repetir el proceso hasta que los números en el par sean el mismo o el resto de la división sea cero. El resultado será el divisor de la última división*.
- Pseudocódigo.
- Diagramas de flujo.
- En un lenguaje de programación.
- Otros modos: tablas de control, lenguajes formales.

Pseudocódigo

Algorithm 1 Algoritmo de Euclides

Requiere: $a \geq 0$ and $b \geq 0$ ▷ Precondición
  procedimiento EUCLIDES(a, b) ▷ Calcula el m.c.d. de a y b
    $r \leftarrow a$ mód b
    mientras $r \neq 0$ **hacer** ▷ Hemos terminado si r es 0
      $a \leftarrow b$
      $b \leftarrow r$
      $r \leftarrow a$ mód b
    fin mientras
    devolver b ▷ El m.c.d es b
  fin procedimiento

Diagrama de flujo



Lenguaje de alto nivel (Python)

```
# Calculo del MCD mediante el algoritmo de Euclides
a = 42
b = 56
r = a % b
while r <> 0 :
    a = b
    b = r
    r = a % b
print b
```

Pulsa aquí para ejecutar en PythonTutor

Errores en los algoritmos

- Revisando el algoritmo visto anteriormente...
  - Estirar el brazo izquierdo
  - Estirar el brazo derecho
  - Tocar el hombro derecho con brazo izquierdo
  - Tocar el hombre izquierdo con bazo derecho
  - Poner la mano izquierda en la nuca
  - Poner la mano derecha en la nunca
  - Poner la mano izquierda en la cadera
  - Poner la mano derecha en la caldera
  - Giro completo de la cadera
  - Saltar y girar 90° a la derecha

Errores en los algoritmos

- Revisando el algoritmo visto anteriormente...
  - Estirar el brazo izquierdo
  - Estirar el brazo derecho
  - Tocar el hombro derecho con brazo izquierdo
  - Tocar el hombr**e** izquierdo con **ba**zo derecho
  - Poner la mano izquierda en la nuca
  - Poner la mano derecha en la nu**n**ca
  - Poner la mano izquierda en la cadera
  - Poner la mano derecha en la ca**l**dera
  - Giro completo de la cadera
  - Saltar y girar 90° a la derecha

Errores en los algoritmos

- Revisando el algoritmo visto anteriormente...
  - Estirar el brazo izquierdo
  - Estirar el brazo derecho
  - Tocar el hombro derecho con brazo izquierdo
  - Tocar el hombro izquierdo con brazo derecho
  - Poner la mano izquierda en la nuca
  - Poner la mano derecha en la nuca
  - Poner la mano izquierda en la cadera
  - Poner la mano derecha en la cadera
  - Giro completo de la cadera
  - Saltar y girar 90° a la derecha

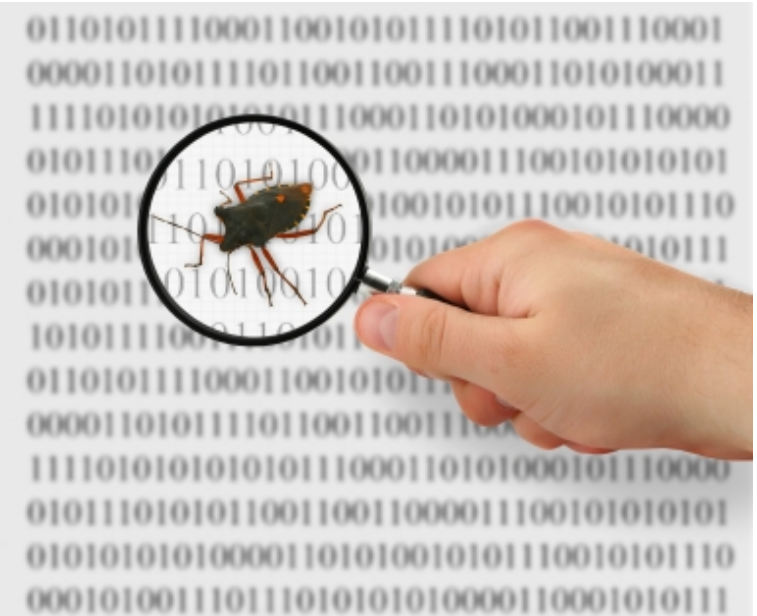
Errores en los algoritmos

- Revisando el algoritmo visto anteriormente...
  - Estirar el brazo izquierdo
  - Estirar el brazo derecho
  - Tocar el hombro derecho con brazo izquierdo
  - Tocar el hombro izquierdo con brazo derecho
  - Poner la mano izquierda en la nuca
  - Poner la mano derecha en la nuca
  - Poner la mano izquierda en la cadera
  - Poner la mano derecha en la cadera
  - Giro completo de la cadera
  - Saltar y girar 90° a la derecha
- Se trata de errores que se detectan en **tiempo de compilación**

Bug o error

Definition

Bug: error o fallo en un programa de computador o sistema de software que desencadena un resultado indeseado [de Wikipedia]



Errores en los algoritmos

- Revisando el algoritmo visto anteriormente...
  - Estirar el brazo izquierdo
  - Estirar el brazo derecho
  - Tocar el hombro derecho con brazo izquierdo
  - Tocar el hombro izquierdo con brazo derecho
  - Poner la mano izquierda en la nuca
  - Poner la mano derecha en la nuca
  - Poner la mano izquierda en la cadera
  - Poner la mano izquierda en la cadera
  - Giro completo de la cadera
  - Saltar y girar 90° a la derecha
- Se trata de errores que se detectan en **tiempo de ejecución** (bugs)

Programador vs. usuario

- El **programador** escribe un programa en un lenguaje de programación siguiendo una estrategia de resolución (algoritmo)

Programador vs. usuario

- El **programador** escribe un programa en un lenguaje de programación siguiendo una estrategia de resolución (algoritmo)
- Para un **usuario** todo es más fácil: solo tiene que elegir entre los varios programas que alguien puso a su alcance listos para usarse

Pensar computacionalmente

- Los ordenadores son **máquinas que ejecutan programas**.

Pensar computacionalmente

- Los ordenadores son **máquinas que ejecutan programas**.
- Los programas resuelven **problemas** utilizando **algoritmos**.

Pensar computacionalmente

- Los ordenadores son **máquinas que ejecutan programas**.
- Los programas resuelven **problemas** utilizando **algoritmos**.
- Pensar **computacionalmente** implica saber expresar la solución a los problemas en una forma que permite su resolución con ordenadores.
- Al traducir un algoritmo a un programa, aparecen **limitaciones prácticas**, como el valor máximo de un número entero:
  - En C el máximo valor positivo de un entero (`MAX_INT`) depende de la plataforma, si bien como mínimo es 32.767 (16 bits).
  - En Java, es en todos los casos 2.147.483.647 (32 bits).

Qué es un programa

- Una secuencia de instrucciones para un ordenador escrita en un cierto lenguaje de programación.

Qué es un programa

- Una secuencia de instrucciones para un ordenador escrita en un cierto lenguaje de programación.
  - Las instrucciones de un programa y los datos necesarios para ejecutarlo se cargan en la memoria del ordenador.

Qué es un programa

- Una secuencia de instrucciones para un ordenador escrita en un cierto lenguaje de programación.
  - Las instrucciones de un programa y los datos necesarios para ejecutarlo se cargan en la memoria del ordenador.
  - El microprocesador va realizando los cálculos correspondientes a las diferentes instrucciones según los datos disponibles.

Qué es un programa

- Una secuencia de instrucciones para un ordenador escrita en un cierto lenguaje de programación.
  - Las instrucciones de un programa y los datos necesarios para ejecutarlo se cargan en la memoria del ordenador.
  - El microprocesador va realizando los cálculos correspondientes a las diferentes instrucciones según los datos disponibles.
- Las instrucciones o sentencias se estructuran en líneas

Qué es un programa

- Una secuencia de instrucciones para un ordenador escrita en un cierto lenguaje de programación.
  - Las instrucciones de un programa y los datos necesarios para ejecutarlo se cargan en la memoria del ordenador.
  - El microprocesador va realizando los cálculos correspondientes a las diferentes instrucciones según los datos disponibles.
- Las instrucciones o sentencias se estructuran en líneas
- Algunas instrucciones ocupan una línea, otras más:

Qué es un programa

- Una secuencia de instrucciones para un ordenador escrita en un cierto lenguaje de programación.
  - Las instrucciones de un programa y los datos necesarios para ejecutarlo se cargan en la memoria del ordenador.
  - El microprocesador va realizando los cálculos correspondientes a las diferentes instrucciones según los datos disponibles.
- Las instrucciones o sentencias se estructuran en líneas
- Algunas instrucciones ocupan una línea, otras más:

```
a = 4          # linea 1: sentencia de asignacion
b = a + 12     # linea 2: asignacion con expresion
if a < b:      # lineas 3 a 6: sentencia alternativa (if)
   print "a es mayor"
else:
   print "b es mayor"
```

Flujo de un programa

- Durante su ejecución, un programa va pasando por los diferentes pasos que le ha especificado el programador.

Flujo de un programa

- Durante su ejecución, un programa va pasando por los diferentes pasos que le ha especificado el programador.
- Algunos pasos son secuenciales: el programa pasa de un paso al siguiente.

Flujo de un programa

- Durante su ejecución, un programa va pasando por los diferentes pasos que le ha especificado el programador.
- Algunos pasos son secuenciales: el programa pasa de un paso al siguiente.
- Otras veces se hacen o no en función de una condición

Flujo de un programa

- Durante su ejecución, un programa va pasando por los diferentes pasos que le ha especificado el programador.
- Algunos pasos son secuenciales: el programa pasa de un paso al siguiente.
- Otras veces se hacen o no en función de una condición
- Hay pasos que se repiten varias veces

Elementos de un programa

- **Constante**. Valores fijos que no cambian.

Elementos de un programa

- **Constante**. Valores fijos que no cambian.
- **Variables**. Elementos cuyo valor puede cambiar.

Elementos de un programa

- **Constante**. Valores fijos que no cambian.
- **Variables**. Elementos cuyo valor puede cambiar.
- **Expresiones**. Combinaciones de operandos y operadores que se evalúan dando como resultado un valor.

Elementos de un programa

- **Constante**. Valores fijos que no cambian.
- **Variables**. Elementos cuyo valor puede cambiar.
- **Expresiones**. Combinaciones de operandos y operadores que se evalúan dando como resultado un valor.
- **Palabras reservadas**: Palabras con un significado especial en el lenguaje de programación utilizado (ej. `while`, `if`, `for`...)

Elementos de un programa

- **Constante**. Valores fijos que no cambian.
- **Variables**. Elementos cuyo valor puede cambiar.
- **Expresiones**. Combinaciones de operandos y operadores que se evalúan dando como resultado un valor.
- **Palabras reservadas**: Palabras con un significado especial en el lenguaje de programación utilizado (ej. `while`, `if`, `for`...)
- **Comentarios**: Texto complementario al programa y que el ordenador no interpreta como instrucciones (no desencadenan ninguna acción).

Constante

Definition

Elemento cuyo valor no cambia durante el desarrollo del algoritmo (durante la ejecución del programa).

Constante

Definition

Elemento cuyo valor no cambia durante el desarrollo del algoritmo (durante la ejecución del programa).

- Ejemplos: PI = 3.141592... , E = 2.71828, lado = 10, etc.

Constante

Definition

Elemento cuyo valor no cambia durante el desarrollo del algoritmo (durante la ejecución del programa).

- Ejemplos: PI = 3.141592... , E = 2.71828, lado = 10, etc.
- No sólo numéricas, también textos (`string`):
  - Warning = 'Prohibido el paso', saludo = "Hola"
  - Más de una línea: poniendo una barra inclinada (\) tras cada parte de la cadena: "En un lugar \ de la Mancha \ de cuyo nombre..."

Constante

Definition

Elemento cuyo valor no cambia durante el desarrollo del algoritmo (durante la ejecución del programa).

- Ejemplos: PI = 3.141592... , E = 2.71828, lado = 10, etc.
- No sólo numéricas, también textos (`string`):
  - Warning = 'Prohibido el paso', saludo = "Hola"
  - Más de una línea: poniendo una barra inclinada (\) tras cada parte de la cadena: "En un lugar \ de la Mancha \ de cuyo nombre..."

Variable

Definition

Elemento con nombre cuyo valor puede cambiar

Variable

Definition

Elemento con nombre cuyo valor puede cambiar

- Dicho nombre (identificador) puede utilizarse para almacenar un dato y luego recuperarlo

Variable

Definition

Elemento con nombre cuyo valor puede cambiar

- Dicho nombre (identificador) puede utilizarse para almacenar un dato y luego recuperarlo
- El nombre de las variables lo elige el programador

Variable

Definition

Elemento con nombre cuyo valor puede cambiar

- Dicho nombre (identificador) puede utilizarse para almacenar un dato y luego recuperarlo
- El nombre de las variables lo elige el programador
- El dato almacenado puede modificarse

Variable

Definition

Elemento con nombre cuyo valor puede cambiar

- Dicho nombre (identificador) puede utilizarse para almacenar un dato y luego recuperarlo
- El nombre de las variables lo elige el programador
- El dato almacenado puede modificarse

Experimentemos un poco...

Vamos a buscar juntos el mayor número de una serie de varios...

Expresiones

Definition

Una **expresión** es una combinación de constantes, variables, símbolos de operación, paréntesis, funciones especiales y otros elementos que se evalúa produciendo un valor.

- Numéricas: (3+4)*9, x - 1, 2**5

Expresiones

Definition

Una **expresión** es una combinación de constantes, variables, símbolos de operación, paréntesis, funciones especiales y otros elementos que se evalúa produciendo un valor.

- Numéricas: (3+4)*9, x - 1, 2**5
- Texto: 'Hola' + 'amigo'

Expresiones

Definition

Una **expresión** es una combinación de constantes, variables, símbolos de operación, paréntesis, funciones especiales y otros elementos que se evalúa produciendo un valor.

- Numéricas: (3+4)*9, x - 1, 2**5
- Texto: 'Hola' + 'amigo'
- Lógicas: $(a > 5)$ and $(x < 0)$
- Otros tipos de expresiones (más adelante...)
- Las expresiones se interpretan de acuerdo a unas reglas (de precedencia y asociación) específicas de cada lenguaje

Expresiones (cont.)

Operation	Result
`x + y`	sum of *x* and *y*
`x - y`	difference of *x* and *y*
`x * y`	product of *x* and *y*
`x / y`	quotient of *x* and *y*
`x // y`	floored quotient of *x* and *y*
`x % y`	remainder of `x / y`
`-x`	*x* negated
`+x`	*x* unchanged
`abs(x)`	absolute value or magnitude of *x*
`int(x)`	*x* converted to integer
`float(x)`	*x* converted to floating point
`complex(re, im)`	a complex number with real part *re*, imaginary part *im*. *im* defaults to zero.
`c.conjugate()`	conjugate of the complex number *c*
`divmod(x, y)`	the pair `(x // y, x % y)`
`pow(x, y)`	*x* to the power *y*
`x ** y`	*x* to the power *y*

Tipo de datos

Definition

Un **tipo de datos** es un conjunto de datos cuyos valores tienen características comunes y predefinidas.

- El tipo determina los valores posibles, las operaciones permitidas, el significado de los datos y la forma en que se almacenan los valores en la memoria del ordenador.
- Tipos básicos:
  - Numéricos: enteros, reales, complejos...
  - Texto: caracter, cadenas de caracteres (texto)
  - Lógicos: booleanos
- Tipos derivados: arrays, listas, tuplas, registros, diccionarios, clases...
- Python: int, float, string...

Tipos de datos en Python

- El tipo se tiene en cuenta cuando se solicita una operación, y si la misma no está permitida el intérprete emite un error:

```
a = ''Hola''
b = a + 10
# Respuesta:
#>>>Traceback (most recent call last):
#>>>  File "<stdin>", line 1, in <module>
#>>> TypeError: Can't convert 'int' object to str
    implicitly
```

- La función "type" permite conocer el tipo de una variable o expresión

```
x = 100
type(x)
#respuesta:
# <class 'int'>
```

Conversiones de tipo

- Internamente todas las operaciones se realizan entre datos del mismo tipo: a menudo es necesario convertir un dato a otro tipo distinto.
- Conversiones implícitas:
  - Promoción: las conversiones que se hacen para hacer concordar los tipos dentro de una expresión.
  - Las lleva a cabo el compilador, no el programador
- Conversiones explícitas (*casting*):
  - A través de mecanismos del lenguaje: En python las funciones `int()` y `float()` permiten convertir texto en números o números entre sí
  - Indicadas por el programador

Conversiones de tipo

En general, en las expresiones se promocionan los tipos hacia el que tiene un rango más amplio.

```
>>> 5 / 2
2
>>> 5 / 2.0
2.5
>>> 5 / float(2)
2.5
```

Precedencia de operadores

Definition

Orden en el que un determinado lenguaje lleva a cabo las operaciones

Ejemplo:

```
9 + 4 * 2
```

Precedencia de operadores

Definition

Orden en el que un determinado lenguaje lleva a cabo las operaciones

Ejemplo:

```
9 + 4 * 2
```

- Imprime 17 porque el $*$ tiene más **precedencia** que el operador + y ambos **asocian** de izquierda a derecha.
- Los paréntesis se pueden utilizar para forzar otra interpretación.

Precedencia de operadores

Definition

Orden en el que un determinado lenguaje lleva a cabo las operaciones

Ejemplo:

```
9 + 4 * 2
```

- Imprime 17 porque el * tiene más **precedencia** que el operador + y ambos **asocian** de izquierda a derecha.
- Los paréntesis se pueden utilizar para forzar otra interpretación.

```
(9 + 4) * 2
```

- Imprime 26 porque el paréntesis tiene mayor **precedencia**.

Uso de variables lógicas

Utilizar las constantes "True" y "False" sólo para inicializar, evitando escribir algo como:

```
if (ready == True) ...
while (empty == False) ...
```

expresiones que pueden siempre reemplazarse por:

```
if (ready) ...
while (not empty) ...
```

... lo cual tiene sentido al traducirse y leerse en voz alta
Evitar nombres "negativos" para facilitar la legibilidad. Así, usar:

```
if (completo) ...
if (not completo) ...
```

... evitando los más confusos:

```
if (not noCompleto) ...
if (noCompleto) ...
```

Uso de variables lógicas

Utilizar las variables booleanas para recoger el valor de una expresión, sin usar ifs innecesarios. Así, en lugar de:

```
if condicion:
  booleano = True
else:
  booleano = False
```

debe escribirse:

```
booleano = condicion
```

Nombres de variable en Python

Python permite utilizar **cualquier conjunto de letras, números y subrayados "_"** como nombre de variable, con algunas limitaciones:

- Deben comenzar por letra o subrayado: `contador`, `_index`, etc.

Nombres de variable en Python

Python permite utilizar **cualquier conjunto de letras, números y subrayados "_"** como nombre de variable, con algunas limitaciones:

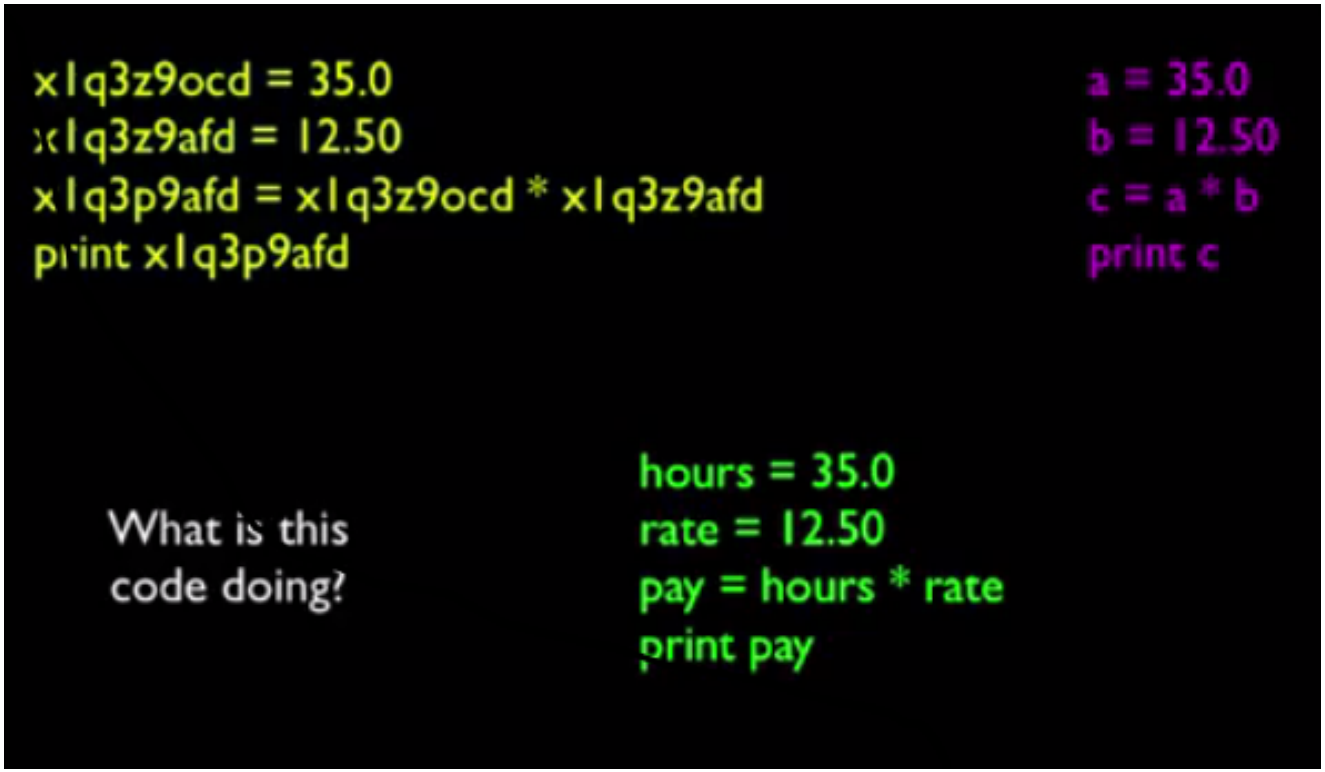
- Deben comenzar por letra o subrayado: `contador`, `_index`, etc.
- Python distingue entre mayúsculas y minúsculas: no es la misma variable "`Valor`" que "`valor`"

Nombres de variable en Python

Python permite utilizar **cualquier conjunto de letras, números y subrayados "_"** como nombre de variable, con algunas limitaciones:

- Deben comenzar por letra o subrayado: `contador`, `_index`, etc.
- Python distingue entre mayúsculas y minúsculas: no es la misma variable "`Valor`" que "`valor`"
- Prohibido el uso de tildes, y otros símbolos, así como las número de palabras denominadas "reservadas': and, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try y while.

Nombres significativos de variable



- *Interesante lectura al respecto:*
  http://computationaltales.blogspot.com.es/2011/03/importance-of-variable-names.html

Reglas de estilo en Python

- Se deben usar nombres de variables descriptivos, no ambiguos y que definan claramente lo que contienen.

Reglas de estilo en Python

- Se deben usar nombres de variables descriptivos, no ambiguos y que definan claramente lo que contienen.
  - Por ejemplo para almacenar el número de alumnos no uses `n = 20`, sino, `num_alumnos = 20`.
  - Una alternativa común es el uso de mayúsculas iniciales: `numAlumnos`, `tasaInteranual`, etc.
- Pueden usarse mayúsculas y minúsculas, pero por convenio se suelen usar las minúsculas

Reglas de estilo en Python

- Se deben usar nombres de variables descriptivos, no ambiguos y que definan claramente lo que contienen.
  - Por ejemplo para almacenar el número de alumnos no uses `n = 20`, sino, `num_alumnos = 20`.
  - Una alternativa común es el uso de mayúsculas iniciales: `numAlumnos`, `tasaInteranual`, etc.
- Pueden usarse mayúsculas y minúsculas, pero por convenio se suelen usar las minúsculas
- Se suele usar el guión bajo "_" para separar palabras: `alumnos_aprobados`

Reglas de estilo en Python

- Se deben usar nombres de variables descriptivos, no ambiguos y que definan claramente lo que contienen.
  - Por ejemplo para almacenar el número de alumnos no uses `n = 20`, sino, `num_alumnos = 20`.
  - Una alternativa común es el uso de mayúsculas iniciales: `numAlumnos`, `tasaInteranual`, etc.
- Pueden usarse mayúsculas y minúsculas, pero por convenio se suelen usar las minúsculas
- Se suele usar el guión bajo "_" para separar palabras: `alumnos_aprobados`

Entrada/Salida en Python

- La función `input(‘‘apunte de entrada’’)` permite solicitar un valor al usuario interativamente.
- Lo habitual es almacenar dicho valor en una variable

```
nombre = input(‘‘Introduce tu nombre: ‘‘)
```

- La función habitual de salida es `print()`
- Para mostrar varios elementos, separar por comas:

```
print(’’Encantado,’’, nombre, ’’yo soy python :)’’)
```

- Operaciones con cadenas: + (concatena), * (concatenación múltiple)

```
print(’’Hola ’’*3) #La salida es Hola Hola Hola
```

Comentarios

Información que el compilador o intérprete ignorará, pero importante para:

- documentar y clarificar nuestro código
- identificar el autor, fecha, ultimas modificaciones, etc.
- desactivar líneas de código (tal vez sólo temporalmente)

```
# Get the name of the file and open it
name = raw_input("Enter file:")
handle = open(name, "r")
text = handle.read()
words = text.split()

# Count word frequency
counts = dict()
for word in words:
  counts[word] = counts.get(word,0) + 1

# Find the most common word
bigcount = None
bigword = None
for word,count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

# All done
print bigword, bigcount
```

Comentarios

En python hay diversas formas de poner comentarios. Un convenio bastante extendido es:

- “ “ “ para documentar código al inicio de la pieza o cuando los comentarios ocupen más de una línea
- “ “ para dejar inactivo un código
- # Para pequeños comentarios interiores

Resumen

- Un algoritmo es una secuencia de pasos para obtener un resultado a partir de entradas bien definidas.
- Un programa es una representación de un algoritmo en un lenguaje de programación.
- Los programas trabajan con valores constantes y variables.
- Los valores tienen un tipo asociado.
- Las expresiones tienen que tener en cuenta qué tipos de datos se pueden combinar.

Referencias y reconocimientos

- Algunos de los contenidos de esta presentación han sido adaptados de los materiales del curso de "Programming for Everybody (Python)", creado por Charles Severance y disponible en `https://www.coursera.org/course/pythonlearn`.
- También se han utilizado ejemplos y adaptado partes del libro ''Fundamentos de la Programación, 2ed" de Luis Joyanes Aguilar, con permiso expreso del autor.

Lecturas y vídeos recomendados

- Capítulo 1 del libro ''Python for Informatics" de Charles Severance.
- Capítulo 1 del libro ''Fundamentos de la Programación, 2ed" de Luis Joyanes Aguilar.
- Ver el vídeo "Computer Programming — A short interesting film"
- TED talk "You Should Learn to Program"

Para experimentar

Visitar y probar los siguientes enlaces:

- Área de principiantes de programmingbasics.org
- One hour of Code
- Codepad