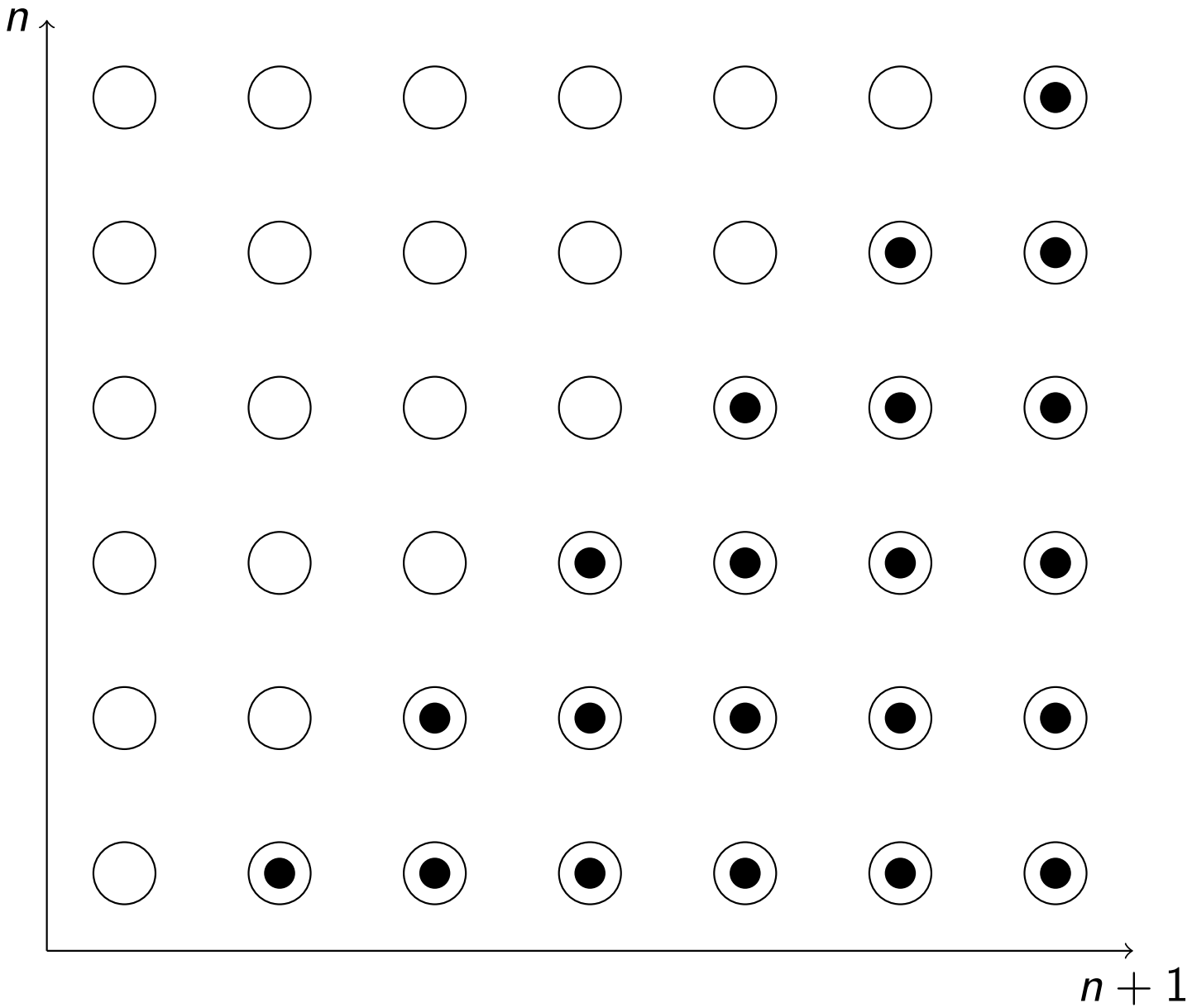
Algoritmos e inducción

Luis Sierra

Instituto de Computación
Facultad de Ingeniería
Universidad de la República

Febrero del 2012
Prácticas compartidas para la enseñanza de informática:
el aula y el trabajo
Universidad Nacional de Quilmes, Argentina

Outline


n
n + 1

Gauss

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Inducción como justificación

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Proof by *Mathematical Induction* follows this pattern: We want to verify that a formula, algebraic expression, holds true for all the values of the parameter, a whole number.

Inducción como justificación

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Proof by *Mathematical Induction* follows this pattern: We want to verify that a formula, algebraic expression, holds true for all the values of the parameter, a whole number.

- We verify that the formula holds true for the smallest possible value

Inducción como justificación

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Proof by *Mathematical Induction* follows this pattern: We want to verify that a formula, algebraic expression, holds true for all the values of the parameter, a whole number.

- We verify that the formula holds true for the smallest possible value
- If we know that the formula holds true for the numbers $< N$, then it also holds true for the number N.

Inducción como justificación

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Proof by *Mathematical Induction* follows this pattern: We want to verify that a formula, algebraic expression, holds true for all the values of the parameter, a whole number.

- We verify that the formula holds true for the smallest possible value
- If we know that the formula holds true for the numbers $< N$, then it also holds true for the number N.
- From that we conclude that the formula holds true for all the numbers.

Inducción como justificación

$(\forall n \in \mathbb{N} :: \mathcal{P}.n)$

$$\mathcal{P}.n \quad := \quad \sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Base: $\mathcal{P}.0$

Paso: $(\forall n \in \mathbb{N} : \mathcal{P}.n : \mathcal{P}.(n+1))$

Inducción como justificación

$(\forall n \in \mathbb{N} :: \mathcal{P}.n)$

$$\mathcal{P}.n \quad := \quad \sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Base: $\mathcal{P}.0$

$$\sum_{i=0}^{0} = \frac{0(0+1)}{2}$$
$$\Longleftarrow$$
$$0 = 0.$$

Paso: $(\forall n \in \mathbb{N} : \mathcal{P}.n : \mathcal{P}.(n+1))$

Inducción como justificación

$(\forall n \in \mathbb{N} :: \mathcal{P}.n)$

$$\mathcal{P}.n \quad := \quad \sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Base: $\mathcal{P}.0$

$$\sum_{i=0}^{0} = \frac{0(0+1)}{2}$$
$$\Longleftarrow$$
$$0 = 0.$$

Paso: $(\forall n \in \mathbb{N} : \mathcal{P}.n : \mathcal{P}.(n+1))$

$$\sum_{i=0}^{n+1} i = \frac{(n+1)(n+2)}{2}$$

Inducción como justificación

$(\forall n \in \mathbb{N} :: \mathcal{P}.n)$

$$\mathcal{P}.n \quad := \quad \sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Base: $\mathcal{P}.0$

$$\sum_{i=0}^{0} = \frac{0(0+1)}{2}$$
$$\Longleftarrow$$
$$0 = 0.$$

Paso: $(\forall n \in \mathbb{N} : \mathcal{P}.n : \mathcal{P}.(n+1))$

$$\sum_{i=0}^{n+1} i = \frac{(n+1)(n+2)}{2}$$
$$\Longleftarrow$$
$$n + 1 + \sum_{i=0}^{n} i = \frac{(n+1)(n+2)}{2}$$

Inducción como justificación

$(\forall n \in \mathbb{N} :: \mathcal{P}.n)$

$$\mathcal{P}.n \quad := \quad \sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Base: $\mathcal{P}.0$

$$\sum_{i=0}^{0} = \frac{0(0+1)}{2}$$
$$\Longleftarrow$$
$$0 = 0.$$

Paso: $(\forall n \in \mathbb{N} : \mathcal{P}.n : \mathcal{P}.(n+1))$

$$\sum_{i=0}^{n+1} i = \frac{(n+1)(n+2)}{2}$$
$$\Longleftarrow$$
$$n + 1 + \sum_{i=0}^{n} i = \frac{(n+1)(n+2)}{2}$$
$$\Longleftarrow (*)$$
$$n + 1 + \frac{n(n+1)}{2} = \frac{(n+1)(n+2)}{2}$$

Inducción como justificación

$(\forall n \in \mathbb{N} :: \mathcal{P}.n)$

$$\mathcal{P}.n \quad := \quad \sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

Base: $\mathcal{P}.0$

$$\sum_{i=0}^{0} = \frac{0(0+1)}{2}$$
$$\Longleftarrow$$
$$0 = 0.$$

Paso: $(\forall n \in \mathbb{N} : \mathcal{P}.n : \mathcal{P}.(n+1))$

$$\sum_{i=0}^{n+1} i = \frac{(n+1)(n+2)}{2}$$
$$\Longleftarrow$$
$$n+1+\sum_{i=0}^{n} i = \frac{(n+1)(n+2)}{2}$$
$$\Longleftarrow (*)$$
$$n+1+\frac{n(n+1)}{2} = \frac{(n+1)(n+2)}{2}$$
$$\Longleftarrow$$
$$2(n+1)+n(n+1) = (n+1)(n+2).$$

Observación

Al probar los casos inductivos correspondientes a algún principio de inducción suele seguirse el siguiente mecanismo:

1. Se despliega la expresión de $\mathcal{P}.(n+1)$ para que aparezcan elementos de $\mathcal{P}.n$
2. Se aplica la hipótesis inductiva
3. Se realizan operaciones simples

Inducción como descubrimiento

$$\mathcal{P}.n \quad := \quad \sum_{i=0}^{n} i = an^2 + bn + c$$

Base: $\mathcal{P}.0$

Paso: $(\forall n \in \mathbb{N} : \mathcal{P}.n : \mathcal{P}.(n+1))$

Inducción como descubrimiento

$$\mathcal{P}.n \quad := \quad \sum_{i=0}^{n} i = an^2 + bn + c$$

Base: $\mathcal{P}.0$

$\sum_{i=0}^{0} = a0^2 + b0 + c$

Paso: $(\forall n \in \mathbb{N} : \mathcal{P}.n : \mathcal{P}.(n+1))$

Inducción como descubrimiento

$$\mathcal{P}.n \quad := \quad \sum_{i=0}^{n} i = an^2 + bn + c$$

Base: $\mathcal{P}.0$

$\sum_{i=0}^{0} = a0^2 + b0 + c$
$\Longleftarrow$
$0 = c.$

Paso: $(\forall n \in \mathbb{N} : \mathcal{P}.n : \mathcal{P}.(n+1))$

Inducción como descubrimiento

$$\mathcal{P}.n \quad := \quad \sum_{i=0}^{n} i = an^2 + bn + c$$

Base: $\mathcal{P}.0$

$\sum_{i=0}^{0} = a0^2 + b0 + c$
$\Longleftarrow$
$0 = c.$

Paso: $(\forall n \in \mathbb{N} : \mathcal{P}.n : \mathcal{P}.(n+1))$

$\sum_{i=0}^{n+1} i = a(n+1)^2 + b(n+1)$
$\Longleftarrow$

Inducción como descubrimiento

$$\mathcal{P}.n \quad := \quad \sum_{i=0}^{n} i = an^2 + bn + c$$

Base: $\mathcal{P}.0$

$\sum_{i=0}^{0} = a0^2 + b0 + c$
$\Longleftarrow$
$0 = c.$

Paso: $(\forall n \in \mathbb{N} : \mathcal{P}.n : \mathcal{P}.(n+1))$

$\sum_{i=0}^{n+1} i = a(n+1)^2 + b(n+1)$
$\Longleftarrow$
$n + 1 + \sum_{i=0}^{n} i = an^2 + (2a+b)n + a + b$
$\Longleftarrow$

Inducción como descubrimiento

$$\mathcal{P}.n \quad := \quad \sum_{i=0}^{n} i = an^2 + bn + c$$

Base: $\mathcal{P}.0$

$\sum_{i=0}^{0} = a0^2 + b0 + c$
$\Longleftarrow$
$0 = c.$

Paso: $(\forall n \in \mathbb{N} : \mathcal{P}.n : \mathcal{P}.(n+1))$

$\sum_{i=0}^{n+1} i = a(n+1)^2 + b(n+1)$
$\Longleftarrow$
$n + 1 + \sum_{i=0}^{n} i = an^2 + (2a+b)n + a + b$
$\Longleftarrow (*)$
$n + 1 + an^2 + bn = an^2 + (2a+b)n + a + b$
$\Longleftarrow$

Inducción como descubrimiento

$$\mathcal{P}.n \quad := \quad \sum_{i=0}^{n} i = an^2 + bn + c$$

Base: $\mathcal{P}.0$

$\sum_{i=0}^{0} = a0^2 + b0 + c$
$\Longleftarrow$
$0 = c.$

Paso: $(\forall n \in \mathbb{N} : \mathcal{P}.n : \mathcal{P}.(n+1))$

$\sum_{i=0}^{n+1} i = a(n+1)^2 + b(n+1)$
$\Longleftarrow$
$n + 1 + \sum_{i=0}^{n} i = an^2 + (2a+b)n + a + b$
$\Longleftarrow (*)$
$n + 1 + an^2 + bn = an^2 + (2a+b)n + a + b$
$\Longleftarrow$
$1 = 2a \text{ y } 1 = a + b$
$\Longleftarrow$

Inducción como descubrimiento

$$\mathcal{P}.n \quad := \quad \sum_{i=0}^{n} i = an^2 + bn + c$$

Base: $\mathcal{P}.0$

$\sum_{i=0}^{0} = a0^2 + b0 + c$
$\Longleftarrow$
$0 = c.$

Paso: $(\forall n \in \mathbb{N} : \mathcal{P}.n : \mathcal{P}.(n+1))$

$\sum_{i=0}^{n+1} i = a(n+1)^2 + b(n+1)$
$\Longleftarrow$
$n + 1 + \sum_{i=0}^{n} i = an^2 + (2a+b)n + a + b$
$\Longleftarrow (*)$
$n + 1 + an^2 + bn = an^2 + (2a+b)n + a + b$
$\Longleftarrow$
$1 = 2a \text{ y } 1 = a + b$
$\Longleftarrow$
$a = \frac{1}{2} \text{ y } b = \frac{1}{2}.$

Justificación vs descubrimiento

Pruebe por inducción

- $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$
- $\sum_{i=0}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$
- $\sum_{i=0}^{n} i^3 = \frac{n^2(n+1)^2}{4}$

Encuentre los coeficientes

- $\sum_{i=0}^{n} i = an^2 + bn + c$
- $\sum_{i=0}^{n} i^2 = an^3 + bn^2 + cn + d$
- $\sum_{i=0}^{n} i^3 = an^4 + bn^3 + cn^2 + dn + e$

Observación

Estamos habituados a que la inducción se use en un contexto de justificación. Dado un resultado, se pide al estudiante que verifique el mismo mediante una prueba inductiva.
Me resulta más divertido encontrar un resultado a partir de una prueba inductiva.

Ejercicio de Matemática Discreta

Demuestre que $7^n - 2^n$ *es divisible por 5, para todo* $n \in \mathbb{N}$.

Ejercicio de Matemática Discreta

Demuestre que $7^n - 2^n$ *es divisible por 5, para todo* $n \in \mathbb{N}$.

Reformulo el enunciado para poner de manifiesto un programa.

Demuestre que para todo $n \in \mathbb{N}$ *se cumple*

$$(\exists k \in \mathbb{N} :: 7^n = 2^n + 5k)$$

Ejercicio de Matemática Discreta

Demuestre que $7^n - 2^n$ es divisible por 5, para todo $n \in \mathbb{N}$.

Reformulo el enunciado para poner de manifiesto un programa.

Demuestre que para todo $n \in \mathbb{N}$ se cumple

$$(\exists k \in \mathbb{N} :: 7^n = 2^n + 5k)$$

Base: $\mathcal{P}.0$

$$7^0 = 2^0 + 5k$$
$$\Longleftrightarrow$$
$$1 = 1 + 5k$$
$$\Longleftrightarrow$$
$$0 = k.$$

Ejercicio de Matemática Discreta

Demuestre que $7^n - 2^n$ es divisible por 5, para todo $n \in \mathbb{N}$.

Reformulo el enunciado para poner de manifiesto un programa.

Demuestre que para todo $n \in \mathbb{N}$ se cumple

$$(\exists k \in \mathbb{N} :: 7^n = 2^n + 5k)$$

Base: $\mathcal{P}.0$

$$7^0 = 2^0 + 5k$$
$$\Longleftrightarrow$$
$$1 = 1 + 5k$$
$$\Longleftrightarrow$$
$$0 = k.$$

Programa

```
function ejercicio(n){
  if (n == 0) return 0;
  else ...
}
```

Ejercicio de Matemática Discreta: caso paso

Demuestre que para todo $n \in \mathbb{N}$ *se cumple*

$$(\exists k \in \mathbb{N} :: 7^n = 2^n + 5k)$$

Ejercicio de Matemática Discreta: caso paso

Demuestre que para todo $n \in \mathbb{N}$ *se cumple*

$$(\exists k \in \mathbb{N} :: 7^n = 2^n + 5k)$$

Paso: $\mathcal{P}.n \Longrightarrow \mathcal{P}.(n+1)$

$$
\begin{aligned}
& 7^{n+1} = 2^{n+1} + 5k \\
& \Longleftarrow \\
& 7 \times 7^n = 2 \times 2^n + 5k \\
& \Longleftarrow (*) \\
& 7 \times (2^n + 5k_H) = 2 \times 2^n + 5k \\
& \Longleftarrow \\
& 5 \times 2^n + 35k_H = 5k \\
& \Longleftarrow \\
& 2^n + 7k_H = k.
\end{aligned}
$$

Ejercicio de Matemática Discreta: caso paso

Demuestre que para todo $n \in \mathbb{N}$ *se cumple*

$$(\exists k \in \mathbb{N} :: 7^n = 2^n + 5k)$$

Paso: $\mathcal{P}.n \Longrightarrow \mathcal{P}.(n+1)$

$$
\begin{array}{l}
7^{n+1} = 2^{n+1} + 5k \\
\Longleftarrow \\
7 \times 7^n = 2 \times 2^n + 5k \\
\Longleftarrow (*) \\
7 \times (2^n + 5k_H) = 2 \times 2^n + 5k \\
\Longleftarrow \\
5 \times 2^n + 35k_H = 5k \\
\Longleftarrow \\
2^n + 7k_H = k.
\end{array}
$$

Programa

```
function ejercicio(n){
  if (n == 0) return 0;
  else {
    kh = ejercicio (n-1);
    return
      7 * kh + pow (2, n-1);
  }
}
```

Observación

En informática nos interesan soluciones fáciles de generalizar. La elegancia matemática no está en encontrar casos particulares interesantes, sino en encontrar esquemas generales aburridos.

¿Qué es sumar?

¿Qué es sumar?

$$\begin{array}{rr} & 11 \\ & 4321 \\ + & 9876 \\ \hline & 14197 \end{array}$$

¿Qué es sumar?

$$\begin{array}{rr} & 11 \\ & 4321 \\ + & 9876 \\ \hline & 14197 \end{array} \qquad \begin{array}{rr} & 9876543 \\ + & 0123457 \\ \hline & \end{array}$$

¿Qué es sumar?

$$\begin{array}{rr} & 11 \\ & 4321 \\ + & 9876 \\ \hline & 14197 \end{array} \qquad \begin{array}{rr} & 9876543 \\ + & 0123457 \\ \hline & 10000000 \end{array}$$

¿Qué es sumar?

$$\begin{array}{r} \ 11 \\ \ 4321 \\ +\quad 9876 \\ \hline 14197 \end{array} \qquad \begin{array}{r} \ 9876543 \\ +\quad 0123457 \\ \hline 10000000 \end{array} \qquad \begin{array}{r} \ 111111 \\ +\quad 666666 \\ \hline \end{array}$$

¿Qué es sumar?

$$\begin{array}{r} \ 11 \\ \ 4321 \\ +\quad 9876 \\ \hline 14197 \end{array} \qquad \begin{array}{r} \ 9876543 \\ +\quad 0123457 \\ \hline 10000000 \end{array} \qquad \begin{array}{r} \ 111111 \\ +\quad 666666 \\ \hline 777777 \end{array}$$

¿Qué es sumar?

$$\begin{array}{r} 11 \\ 4321 \\ +\quad 9876 \\ \hline 14197 \end{array} \qquad \begin{array}{r} 9876543 \\ +\quad 0123457 \\ \hline 10000000 \end{array} \qquad \begin{array}{r} 111111 \\ +\quad 666666 \\ \hline 777777 \end{array}$$

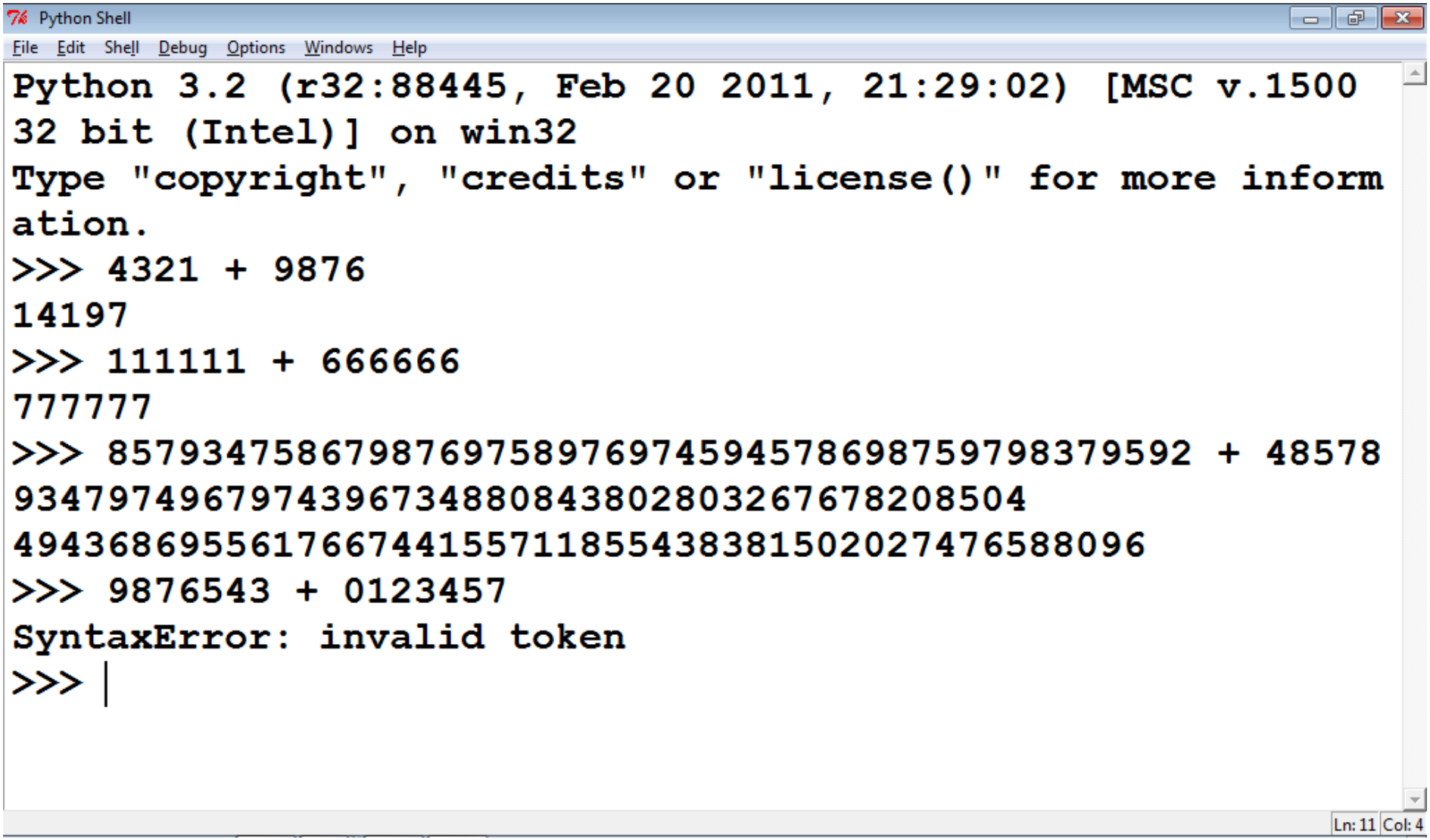
$$\begin{array}{r} 857934758679876975897697459457869875979837959 2 \\ +\quad 485789347974967974396734880843802803267678208504 \\ \hline \end{array}$$

¿Qué es sumar?

$$\begin{array}{r} 11 \\ 4321 \\ +\quad 9876 \\ \hline 14197 \end{array} \qquad \begin{array}{r} 9876543 \\ +\quad 0123457 \\ \hline 10000000 \end{array} \qquad \begin{array}{r} 111111 \\ +\quad 666666 \\ \hline 777777 \end{array}$$

$$\begin{array}{r} 857934758679876975897697459457869875979837959 2 \\ +\quad 485789347974967974396734880843802803267678208504 \\ \hline ? \end{array}$$

¿Qué es sumar para Python?



Observaciones

Hechos

- Las computadoras nos permiten realizar sumas muy grandes
- Las computadoras no manejan la misma escritura que nosotros

Preguntas

- ¿Podemos confiar en una computadora para sumar?
- ¿Tenemos que sumar como las computadoras?
- ¿Las computadoras tienen que sumar como nosotros?
- ¿Cómo sabemos si las computadoras suman bien?
- ¿Qué es la suma?

¿Qué es la suma?

$$\begin{array}{rr} & 4321 \\ + & 9876 \\ \hline \end{array}$$

¿Qué es la suma?

$$\begin{array}{rr} & 4321 \\ + & 9876 \\ \hline & 14197 \end{array}$$

¿Qué es la suma?

$$\begin{array}{rr} & 4321 \\ + & 9876 \\ \hline & 14197 \end{array} \qquad \begin{array}{rr} & \mathit{MMMMCCCXXI} \\ + & \mathit{MMMMMMMMMDCCCLXXVI} \\ \hline & \end{array}$$

¿Qué es la suma?

$$\begin{array}{rr} & 4321 \\ + & 9876 \\ \hline & 14197 \end{array} \qquad \begin{array}{rr} & \mathit{MMMMCCCXXI} \\ + & \mathit{MMMMMMMMMDCCCLXXVI} \\ \hline & ? \end{array}$$

¿Qué es la suma?

$$\begin{array}{rr} & 4321 \\ + & 9876 \\ \hline & 14197 \end{array} \qquad \begin{array}{rr} & MMMMCCCXXI \\ + & MMMMMMMMMDCCCLXXVI \\ \hline & ? \end{array}$$

Wikipedia

La suma o adición es la operación básica por su naturalidad, que se combina con facilidad matemática de composición que consiste en combinar o añadir dos números o más para obtener una cantidad final o total. La suma también ilustra el proceso de juntar dos colecciones de objetos con el fin de obtener una sola colección.
[...]
En términos más formales, la suma es una operación aritmética definida sobre conjuntos de números [...], y también sobre estructuras asociadas a ellos, como espacios vectoriales con vectores cuyas componentes sean estos números o funciones que tengan su imagen en ellos.

Pequeña encuesta

- ¿Sabemos hacer una suma?

Pequeña encuesta

- ¿Sabemos hacer una suma?
- ¿Sabemos qué es una suma?

Pequeña encuesta

- ¿Sabemos hacer una suma?
- ¿Sabemos qué es una suma?
- ¿Sabemos hacer un proyector?

Pequeña encuesta

- ¿Sabemos hacer una suma?
- ¿Sabemos qué es una suma?
- ¿Sabemos hacer un proyector?
- ¿Sabemos qué es un proyector?

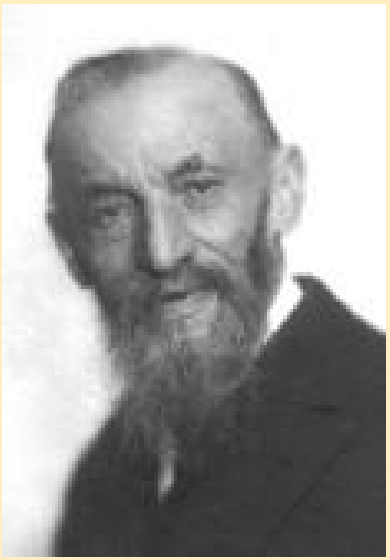
La suma

$$\begin{aligned} 0 + m &:= m \\ succ.n + m &:= n + succ.m \end{aligned}$$

La suma

$$
\begin{array}{lcl}
0 + m & := & m \\
succ.n + m & := & n + succ.m
\end{array}
$$

Rivista di matematica, 1895



$$
\begin{array}{ll}
1 & 0 \in N_0 \\
2 & n \in N_0 . \supset . n+ \in N_0 \\
3 & \alpha \in \mathrm{Cls} . 0 \in \alpha : n \in \alpha . \supset_n . n+ \in \alpha : \supset . N_0 \subset \alpha \\
4 & n, b \in N_0 . n+ = b+ . \supset . n = b \\
5 & n \in N_0 . \supset . n+ \neq 0
\end{array}
$$

Concreto vs abstracto

MMXII

1000101

algoritmos

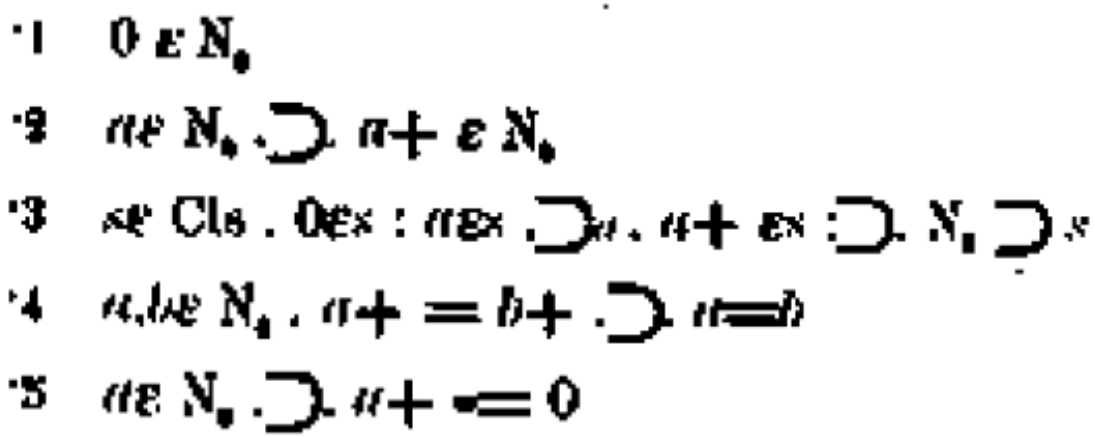
Observación

En informática trabajamos en dos niveles

- a veces buscamos que los objetos concretos se reflejen adecuadamente en los abstractos
- a veces buscamos trabajar solamente en lo concreto, que es lo único que puede hacer una computadora

Los objetos concretos que entiende una computadora son listas de ceros y unos.

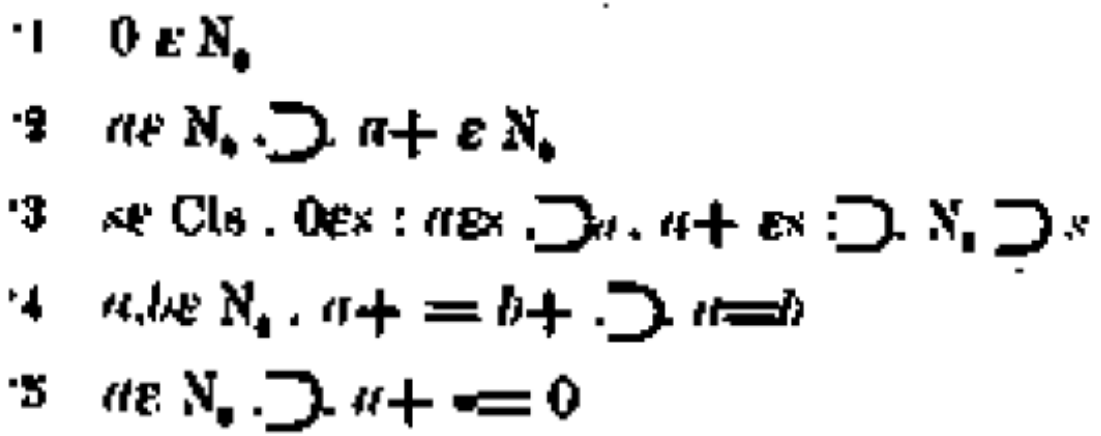
Los numerales



Definición

1. $0 \in Num$
2. $1 \in Num$
3. $(\forall w \in Num :: w0 \in Num)$
4. $(\forall w \in Num :: w1 \in Num)$

Los numerales



Definición

1. $0 \in Num$
2. $1 \in Num$
3. $(\forall w \in Num :: w0 \in Num)$
4. $(\forall w \in Num :: w1 \in Num)$

Inducción

Sea $\mathcal{P}$ una propiedad sobre Num tal que:

1. $\mathcal{P}.0$
2. $\mathcal{P}.1$
3. $(\forall w \in Num : \mathcal{P}.w : \mathcal{P}.w0)$
4. $(\forall w \in Num : \mathcal{P}.w : \mathcal{P}.w1)$

Entonces, $(\forall w \in Num :: \mathcal{P}.w)$.

Semántica de *Num*

$$[0] \quad :=$$

Semántica de *Num*

$$[0] \quad := \quad 0$$

Semántica de *Num*

$$
\begin{array}{lll}
[0] & := & 0 \\
[1] & := &
\end{array}
$$

Semántica de *Num*

$$
\begin{array}{lcl}
[0] & := & 0 \\
[1] & := & 1
\end{array}
$$

Semántica de *Num*

$$
\begin{array}{lll}
[0] & := & 0 \\
[1] & := & 1 \\
[w0] & := &
\end{array}
$$

Semántica de *Num*

$$
\begin{array}{lcr}
[0] & := & 0 \\
[1] & := & 1 \\
[w0] & := & 2\,[w]
\end{array}
$$

Semántica de *Num*

$$
\begin{array}{lcr}
[0] & := & 0 \\
[1] & := & 1 \\
[w0] & := & 2\,[w] \\
[w1] & := &
\end{array}
$$

Semántica de *Num*

$$
\begin{array}{lcr}
[0] & := & 0 \\
[1] & := & 1 \\
[w0] & := & 2\,[w] \\
[w1] & := & 1 + 2\,[w]
\end{array}
$$

succ: programa dependiente de la semántica

El programa *succ* sobre *Num* debe cumplir:

$$(\forall w \in Num :: [succ.w] = succ.[w])$$

succ: programa dependiente de la semántica

El programa *succ* sobre *Num* debe cumplir:

$$(\forall w \in Num :: [succ.w] = succ.[w])$$

- *succ.w* es el *Num* que resulta de aplicar el programa *succ* al numeral *w*

succ: programa dependiente de la semántica

El programa *succ* sobre *Num* debe cumplir:

$$(\forall w \in Num :: [succ.w] = succ.[w])$$

- $succ.w$ es el *Num* que resulta de aplicar el programa *succ* al numeral w
- $[w]$ es la semántica del numeral w

succ: programa dependiente de la semántica

El programa *succ* sobre *Num* debe cumplir:

$$(\forall w \in Num :: [succ.w] = succ.[w])$$

- $succ.w$ es el *Num* que resulta de aplicar el programa *succ* al numeral w
- $[w]$ es la semántica del numeral w
- $[succ.w]$ es la semántica del numeral $succ.w$

succ: programa dependiente de la semántica

El programa *succ* sobre *Num* debe cumplir:

$$(\forall w \in Num :: [succ.w] = succ.[w])$$

- $succ.w$ es el *Num* que resulta de aplicar el programa *succ* al numeral w
- $[w]$ es la semántica del numeral w
- $[succ.w]$ es la semántica del numeral $succ.w$
- $succ.[w]$ es el resultado de aplicar el sucesor de los naturales, el de Peano, el de Platón, el que no sabemos si existe, a $[w]$

Programando *succ*: casos base

$$\mathcal{P}.w := [succ.w] = succ.[w]$$

Programando *succ*: casos base

$$\mathcal{P}.w := [succ.w] = succ.[w]$$

$\mathcal{P}.0$

$$[succ.0] = succ.[0]$$
$$\Longleftarrow$$
$$[succ.0] = succ.0$$
$$\Longleftarrow$$
$$[succ.0] = 1.$$

Programando *succ*: casos base

$$\mathcal{P}.w := [succ.w] = succ.[w]$$

$\mathcal{P}.0$

$$[succ.0] = succ.[0]$$
$$\Longleftarrow$$
$$[succ.0] = succ.0$$
$$\Longleftarrow$$
$$[succ.0] = 1.$$

$\mathcal{P}.1$

$$[succ.1] = succ.[1]$$
$$\Longleftarrow$$
$$[succ.1] = succ.1$$
$$\Longleftarrow$$
$$[succ.1] = 2.$$

Programando *succ*: casos base

$$\mathcal{P}.w := [succ.w] = succ.[w]$$

$\mathcal{P}.0$

$$[succ.0] = succ.[0]$$
$$\Longleftarrow$$
$$[succ.0] = succ.0$$
$$\Longleftarrow$$
$$[succ.0] = 1.$$

$\mathcal{P}.1$

$$[succ.1] = succ.[1]$$
$$\Longleftarrow$$
$$[succ.1] = succ.1$$
$$\Longleftarrow$$
$$[succ.1] = 2.$$

$$\begin{aligned} succ.0 &= 1 \\ succ.1 &= 10 \end{aligned}$$

Programando *succ*: casos paso

$$\mathcal{P}.w := [succ.w] = succ.\,[w]$$

Programando *succ*: casos paso

$$\mathcal{P}.w := [succ.w] = succ.\,[w]$$

$\mathcal{P}.w0$

$$
\begin{aligned}
& [succ.w0] = succ.\,[w0] \\
\Longleftarrow\ & \\
& [succ.w0] = succ.(2\,[w]) \\
\Longleftarrow\ & \\
& [succ.w0] = 1 + 2\,[w] \\
\Longleftarrow\ & \\
& [succ.w0] = [w1]\,.
\end{aligned}
$$

Programando *succ*: casos paso

$$\mathcal{P}.w := [succ.w] = succ.\,[w]$$

$\mathcal{P}.w0$

$$\begin{array}{l} [succ.w0] = succ.\,[w0] \\ \Longleftarrow \\ [succ.w0] = succ.(2\,[w]) \\ \Longleftarrow \\ [succ.w0] = 1 + 2\,[w] \\ \Longleftarrow \\ [succ.w0] = [w1]\,. \end{array}$$

$\mathcal{P}.w1$

$$\begin{array}{l} [succ.w1] = succ.\,[w1] \\ \Longleftarrow \\ [succ.w1] = succ.(1 + 2\,[w]) \\ \Longleftarrow \\ [succ.w1] = 2(1 + [w]) \\ \Longleftarrow \\ [succ.w1] = 2\,[succ.w]\,. \end{array}$$

Programando *succ*: casos paso

$$\mathcal{P}.w := [succ.w] = succ.\,[w]$$

$\mathcal{P}.w0$

$$\begin{array}{l} [succ.w0] = succ.\,[w0] \\ \Longleftarrow \\ [succ.w0] = succ.(2\,[w]) \\ \Longleftarrow \\ [succ.w0] = 1 + 2\,[w] \\ \Longleftarrow \\ [succ.w0] = [w1]\,. \end{array}$$

$\mathcal{P}.w1$

$$\begin{array}{l} [succ.w1] = succ.\,[w1] \\ \Longleftarrow \\ [succ.w1] = succ.(1 + 2\,[w]) \\ \Longleftarrow \\ [succ.w1] = 2(1 + [w]) \\ \Longleftarrow \\ [succ.w1] = 2\,[succ.w]\,. \end{array}$$

$$\begin{array}{lcl} succ.w0 & = & w1 \\ succ.w1 & = & \text{let } u := succ.w \text{ in } u0 \end{array}$$

Programa *succ*

Se deben cumplir las siguientes igualdades:

$$
\begin{array}{lcl}
succ.0 & = & 1 \\
succ.1 & = & 10 \\
succ.w0 & = & w1 \\
succ.w1 & = & \text{let } u := succ.w \text{ in } u0
\end{array}
$$

Programa *succ*

Se deben cumplir las siguientes igualdades:

$$\begin{array}{lcr} succ.0 & = & 1 \\ succ.1 & = & 10 \\ succ.w0 & = & w1 \\ succ.w1 & = & \text{let } u := succ.w \text{ in } u0 \end{array}$$

Se infiere el siguiente programa:

$$\begin{array}{lcr} succ.0 & := & 1 \\ succ.1 & := & 10 \\ succ.w0 & := & w1 \\ succ.w1 & := & \text{let } u := succ.w \text{ in } u0 \end{array}$$

pred: programa independiente de la semántica

El programa *pred* sobre *Num* debe cumplir:

$$(\forall w \in Num :: pred.succ.w = w)$$

$\mathcal{P}.0$

$$pred.succ.0 = 0$$
$$\Longleftarrow$$
$$pred.1 = 0.$$

$\mathcal{P}.1$

$$pred.succ.1 = 1$$
$$\Longleftarrow$$
$$pred.10 = 1.$$

pred: programa independiente de la semántica

El programa *pred* sobre *Num* debe cumplir:

$$(\forall w \in Num :: pred.succ.w = w)$$

$\mathcal{P}.0$

$$pred.succ.0 = 0$$
$$\Longleftarrow$$
$$pred.1 = 0.$$

$\mathcal{P}.1$

$$pred.succ.1 = 1$$
$$\Longleftarrow$$
$$pred.10 = 1.$$

$$\begin{array}{lcl} pred.1 & = & 0 \\ pred.10 & = & 1 \end{array}$$

Programando *pred*: casos paso

$$\mathcal{P}.w := pred.succ.w = w$$

$\mathcal{P}.w0$

$$pred.succ.w0 = w0$$
$$\Longleftarrow$$
$$pred.w1 = w0.$$

$\mathcal{P}.w1$

$$pred.succ.w1 = w1$$
$$\Longleftarrow (u = succ.w)$$
$$pred.u0 = w1.$$

Programando *pred*: casos paso

$$\mathcal{P}.w := pred.succ.w = w$$

$\mathcal{P}.w0$

$$pred.succ.w0 = w0$$
$$\Longleftarrow$$
$$pred.w1 = w0.$$

$\mathcal{P}.w1$

$$pred.succ.w1 = w1$$
$$\Longleftarrow (u = succ.w)$$
$$pred.u0 = w1.$$

$$\begin{array}{lcl} pred.u0 & = & \text{let } w := pred.u \text{ in } w1 \\ pred.w1 & = & w0 \end{array}$$

Programa *pred*

Se deben cumplir las siguientes igualdades:

$$
\begin{array}{lcl}
pred.1 & = & 0 \\
pred.10 & = & 1 \\
pred.u0 & = & \text{let } w := pred.u \text{ in } w1 \\
pred.w1 & = & w0
\end{array}
$$

Programa *pred*

Se deben cumplir las siguientes igualdades:

$$\begin{array}{lcl} pred.1 & = & 0 \\ pred.10 & = & 1 \\ pred.u0 & = & \text{let } w := pred.u \text{ in } w1 \\ pred.w1 & = & w0 \end{array}$$

¿Se infiere el siguiente programa?

$$\begin{array}{lcl} pred.0 & := & ? \\ pred.1 & := & 0 \\ pred.w0 & := & \text{let } u := pred.w \text{ in if } w = 1 \text{ then } 1 \text{ else } u1 \\ pred.w1 & := & w0 \end{array}$$

Observación

A veces se programa tomando en cuenta la semántica. Las justificaciones se basan en el conocimiento del dominio semántico.
A veces se programa sobre la estructura de datos. Las justificaciones pueden obtenerse a partir de la manipulación concreta.

Un programa para sumar: independiente de la codificación

$$\begin{aligned} 0 + m &:= m \\ succ.n + m &:= n + succ.m \end{aligned}$$

Un programa para sumar: independiente de la codificación

$$\begin{array}{lcl} 0 + m & := & m \\ succ.n + m & := & n + succ.m \end{array}$$

$$suma.w.u \quad := \quad \text{if } w = 0 \text{ then } u \text{ else } suma.(pred.w).(succ.u)$$

Un programa para sumar: independiente de la codificación

$$\begin{array}{lcl} 0 + m & := & m \\ succ.n + m & := & n + succ.m \end{array}$$

$$suma.w.u \quad := \quad \text{if } w = 0 \text{ then } u \text{ else } suma.(pred.w).(succ.u)$$

$$7 + 4 = 6 + 5 = 5 + 6 = 4 + 7 = 3 + 8 = 2 + 9 = 1 + 10 = 0 + 11 = 11$$

Un programa para sumar: independiente de la codificación

$$\begin{array}{lcl} 0 + m & := & m \\ succ.n + m & := & n + succ.m \end{array}$$

$$suma.w.u \quad := \quad \text{if } w = 0 \text{ then } u \text{ else } suma.(pred.w).(succ.u)$$

$$7 + 4 = 6 + 5 = 5 + 6 = 4 + 7 = 3 + 8 = 2 + 9 = 1 + 10 = 0 + 11 = 11$$

¿Funciona el programa?

$$\begin{array}{lcl} [suma.0.w] & = & [w] \\ [suma.(succ.u).w] & = & [suma.u.(succ.w)] \end{array}$$

Un programa para sumar: independiente de la codificación

$$\begin{array}{lcl} 0 + m & := & m \\ succ.n + m & := & n + succ.m \end{array}$$

$$suma.w.u \quad := \quad \text{if } w = 0 \text{ then } u \text{ else } suma.(pred.w).(succ.u)$$

$$7 + 4 = 6 + 5 = 5 + 6 = 4 + 7 = 3 + 8 = 2 + 9 = 1 + 10 = 0 + 11 = 11$$

¿Funciona el programa?

$$\begin{array}{lcl} [suma.0.w] & = & [w] \\ [suma.(succ.u).w] & = & [suma.u.(succ.w)] \end{array}$$

Pero ... ¿qué devuelve $suma.00.0$?

El programa escolar: dependiente de la codificación

$$
\begin{array}{lcl}
0 + m & := & m \\
succ.n + m & := & n + succ.m
\end{array}
$$

El programa escolar: dependiente de la codificación

$$
\begin{array}{lcl}
0 + m & := & m \\
succ.n + m & := & n + succ.m
\end{array}
$$

$$
\begin{array}{lcl}
suma.0.w & := & w \\
suma.1.w & := & succ.w \\
suma.u0.0 & := & u0 \\
suma.u0.1 & := & u1
\end{array}
$$

El programa escolar: dependiente de la codificación

$$
\begin{array}{lcl}
0 + m & := & m \\
succ.n + m & := & n + succ.m
\end{array}
$$

$$
\begin{array}{lcl}
suma.0.w & := & w \\
suma.1.w & := & succ.w \\
suma.u0.0 & := & u0 \\
suma.u0.1 & := & u1 \\
suma.u0.w0 & := & \text{let } v := suma.u.w \text{ in } v0 \\
suma.u0.w1 & := & \text{let } v := suma.u.w \text{ in } v1
\end{array}
$$

El programa escolar: dependiente de la codificación

$$
\begin{array}{lcl}
0 + m & := & m \\
succ.n + m & := & n + succ.m
\end{array}
$$

$$
\begin{array}{lcl}
suma.0.w & := & w \\
suma.1.w & := & succ.w \\
suma.u0.0 & := & u0 \\
suma.u0.1 & := & u1 \\
suma.u0.w0 & := & \text{let } v := suma.u.w \text{ in } v0 \\
suma.u0.w1 & := & \text{let } v := suma.u.w \text{ in } v1 \\
suma.u1.0 & := & u1 \\
suma.u1.1 & := & (succ.u)0 \\
suma.u1.w0 & := & \text{let } v := suma.u.w \text{ in } v1 \\
suma.u1.w1 & := & \text{let } v := suma.u.w \text{ in } (succ.v)0
\end{array}
$$

¿Funciona el programa?

$$
\begin{array}{lcl}
[suma.0.w] & = & [w] \\
[suma.(succ.u).w] & = & [suma.u.(succ.w)]
\end{array}
$$

¿Funciona el programa?

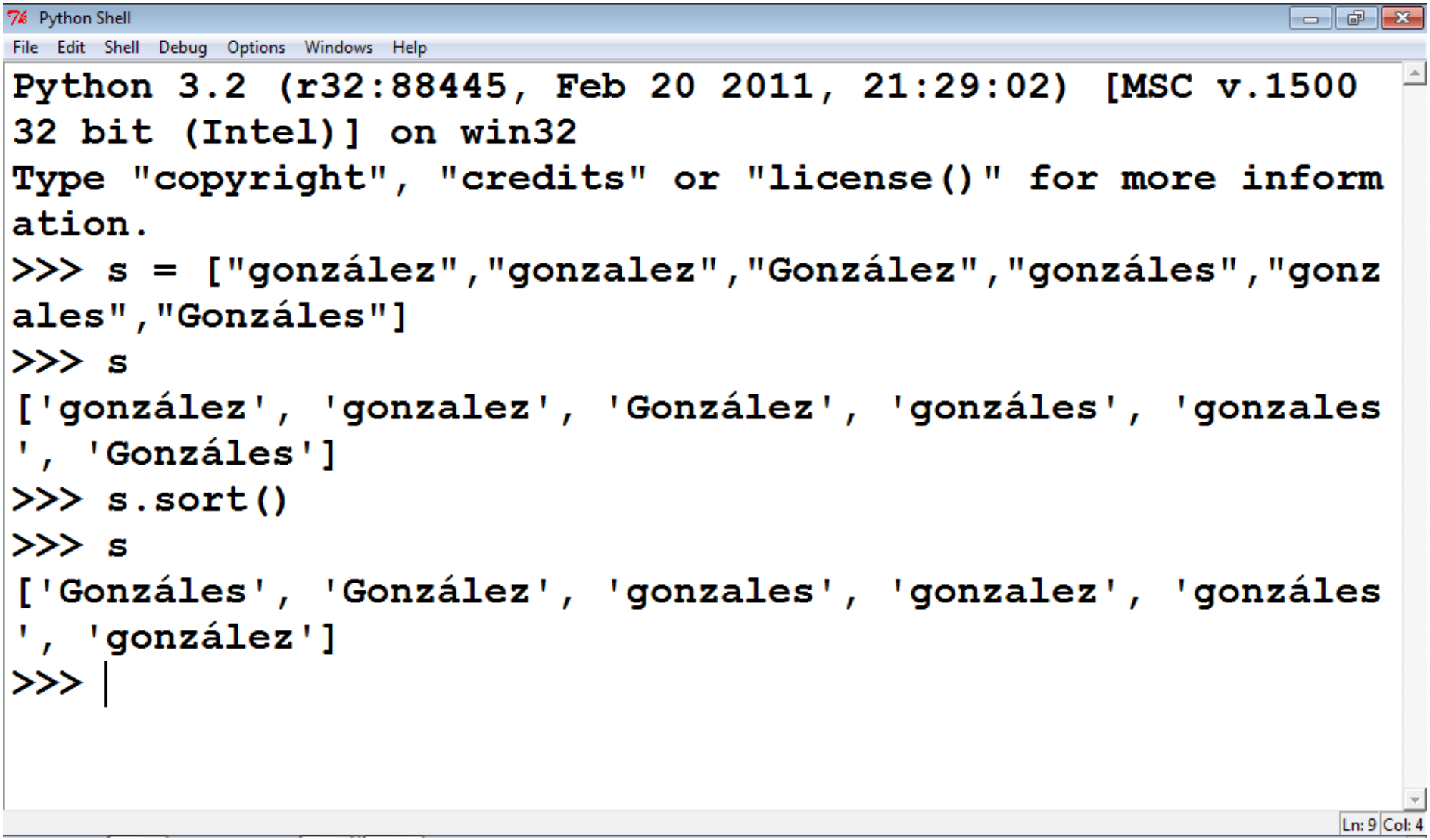
$$
\begin{array}{lcl}
[suma.0.w] & = & [w] \\
[suma.(succ.u).w] & = & [suma.u.(succ.w)]
\end{array}
$$

$$
\begin{array}{l}
[suma.(succ.u1).w1] = [suma.u1.(succ.w1)] \\
\Longleftarrow \\
[suma.((succ.u)0).w1] = [suma.u1.((succ.w)0)] \\
\Longleftarrow (v = suma.(succ.u).w, v' = suma.u.(succ.w)) \\
[v1] = [v'1]
\end{array}
$$

Observación

La programación centrada en la semántica puede resultar ineficiente. Y nunca puede ser independiente de la codificación.
La programación centrada en la codificación puede resultar complicada.
Pero hay una herramienta conocida que ayuda en esa tarea: la inducción.

Los ?onz?le?


Python Shell
File Edit Shell Debug Options Windows Help
Python 3.2 (r32:88445, Feb 20 2011, 21:29:02) [MSC v.1500
32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more inform
ation.
>>> s = ["González","gonzalez","González","gonzáles","gonz
ales","Gonzáles"]
>>> s
['González', 'gonzalez', 'González', 'gonzáles', 'gonzales
', 'Gonzáles']
>>> s.sort()
>>> s
['Gonzáles', 'González', 'gonzales', 'gonzalez', 'gonzáles
', 'González']
>>>
Ln: 9 Col: 4

Observación

En informática hay dos objetos que entender

- el lenguaje formal
- la interpretación de ese lenguaje formal

Si no entendemos alguno de ellos, o no sabemos escribir, o no sabemos leer.
Es muy popular el aprender a escribir, pero no saber leer.

Un último ejercicio

Considere un tablero cuadriculado de 2^n cuadrados por lado al cual le falta un cuadradito en algún lugar. Demuestre que se puede cubrir dicho tablero con piezas en forma de L formadas por 3 cuadraditos cada una.

Observaciones finales

- La inducción puede servir para descubrir resultados matemáticos

Observaciones finales

- La inducción puede servir para descubrir resultados matemáticos
- Podemos razonar inductivamente sobre objetos diferentes a los naturales

Observaciones finales

- La inducción puede servir para descubrir resultados matemáticos
- Podemos razonar inductivamente sobre objetos diferentes a los naturales
- Las pruebas inductivas pueden convertirse en algoritmos

Observaciones finales

- La inducción puede servir para descubrir resultados matemáticos
- Podemos razonar inductivamente sobre objetos diferentes a los naturales
- Las pruebas inductivas pueden convertirse en algoritmos
- La inducción es una buena excusa para centrarse en la enseñanza de algoritmos

Observaciones finales

- La inducción puede servir para descubrir resultados matemáticos
- Podemos razonar inductivamente sobre objetos diferentes a los naturales
- Las pruebas inductivas pueden convertirse en algoritmos
- La inducción es una buena excusa para centrarse en la enseñanza de algoritmos
- Lamentablemente, este uso de la inducción no se explota tanto como sería deseable

Observaciones finales

- La inducción puede servir para descubrir resultados matemáticos
- Podemos razonar inductivamente sobre objetos diferentes a los naturales
- Las pruebas inductivas pueden convertirse en algoritmos
- La inducción es una buena excusa para centrarse en la enseñanza de algoritmos
- Lamentablemente, este uso de la inducción no se explota tanto como sería deseable
- ... por mí, claro.

Algunos recursos usados

- Roland Backhouse "Program Construction"
- Luis Sierra. "Tal vez le erre" (blog). http://pkanota.wordpress.com
- Matemática Discreta 1. IMERL, Facultad de Ingeniería. http://imerl.fing.edu.uy/matdisc1
- Gauss' childhood problem. http://www.csun.edu/~ac53971/pump/20081014_basic.pdf
- Caricatura de Gauss. Thiago Castor, http://www.toonpool.com/user/4296/files/c_f_gauss_574659.jpg
- Caricatura de Platón. Gary Brown, http://www.sciencephoto.com/image/90379/large/C0026127-Plato,_caricature-SPL.jpg
- Retrato de Peano. http://upload.wikimedia.org/wikipedia/commons/thumb/3/3a/Giuseppe_Peano.jpg/220px-Giuseppe_Peano.jpg