Juanjo Conti
(basado en una charla de Facundo Batista)

Python para no programadores

Objetivo

Tener una idea de

qué es programar

Hablarle a la máquina

- Usamos un lenguaje

- Le decimos a la computadora lo que queremos que haga

Lenguaje de Programación

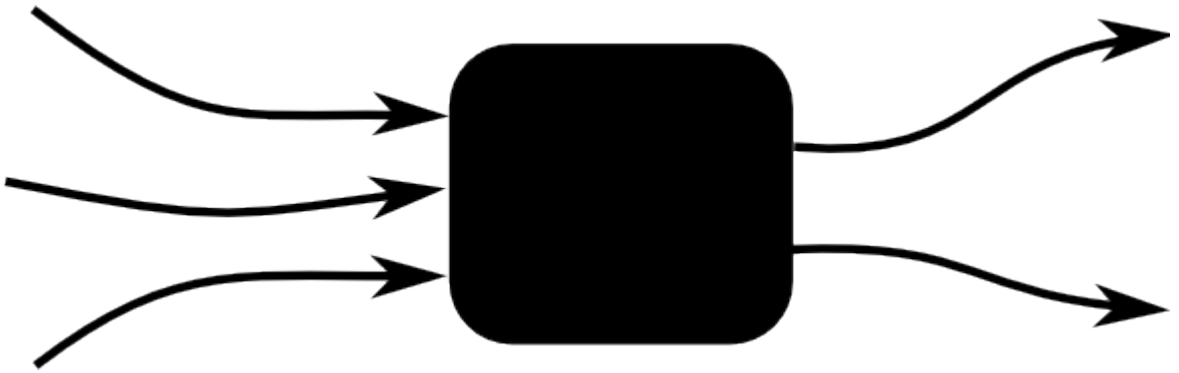
Es un lenguaje para decirle a la computadora que haga cosas

¿Cómo le decimos?

- Escribimos un programa
- Luego lo ejecutamos
- ¿Pero a quién le decimos?

Caja negra

- Tiene entradas
- Hace cosas
- Tiene salidas



Entradas y Salidas

- Input / Output
- Archivos, red
- Terminal e Interfaces gráficas
- Teclado, mouse
- Dispositivos especiales

Niveles de lenguajes

- Multiplicamos dos números

ASM

```
  .file    "mult.c"
  .text
.globl main
  .type    main, @function
main:
  leal    4(%esp), %ecx
  andl    $-16, %esp
  pushl    -4(%ecx)
  pushl    %ebp
  movl    %esp, %ebp
  pushl    %ecx
  subl    $16, %esp
  movl    $5, -16(%ebp)
  movl    $10, -12(%ebp)
  movl    -16(%ebp), %eax
  imull    -12(%ebp), %eax
  movl    %eax, -8(%ebp)
  addl    $16, %esp
  popl    %ecx
  popl    %ebp
  leal    -4(%ecx), %esp
  ret
  .size    main, .-main
  .section    .note.GNU-stack,"",@progbits
```

Python

```
a = 5
b = 10
c = a * b
```

- Python es de Muy Alto Nivel (VHLL)

Empezando

El intérprete interactivo es el mejor amigo

del hombre (y de la mujer)

Objetos

- Dando nombres
- Operando con números

Más objetos

- Trabajando con `cadenas`

- `print`

Nuestro primer programa

- Como escribirlo
- Como ejecutarlo
- Hola mundo!

holamundo.py

Algo útil

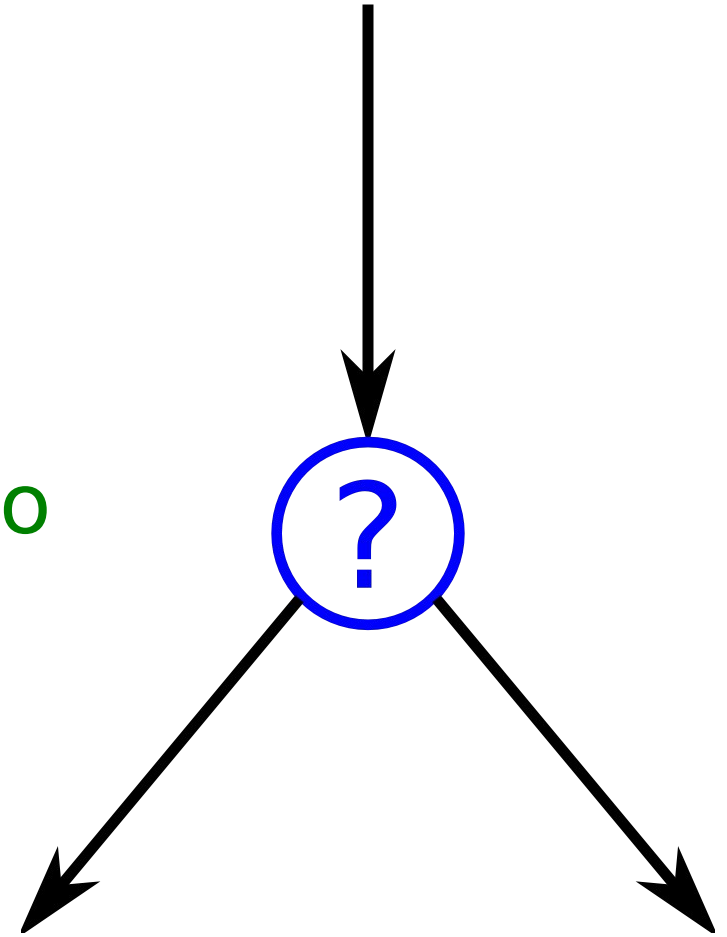
- Calculamos el cuadrado

- `input`

`cuadrado.py`

Decidiendo

- Condiciones
- `if / else`
- Todo es Verdadero o Falso



Sintaxis del if

```
<antes>

if <expresión>:
    <bloque de código>
else:
    <bloque de código>

<después>
```

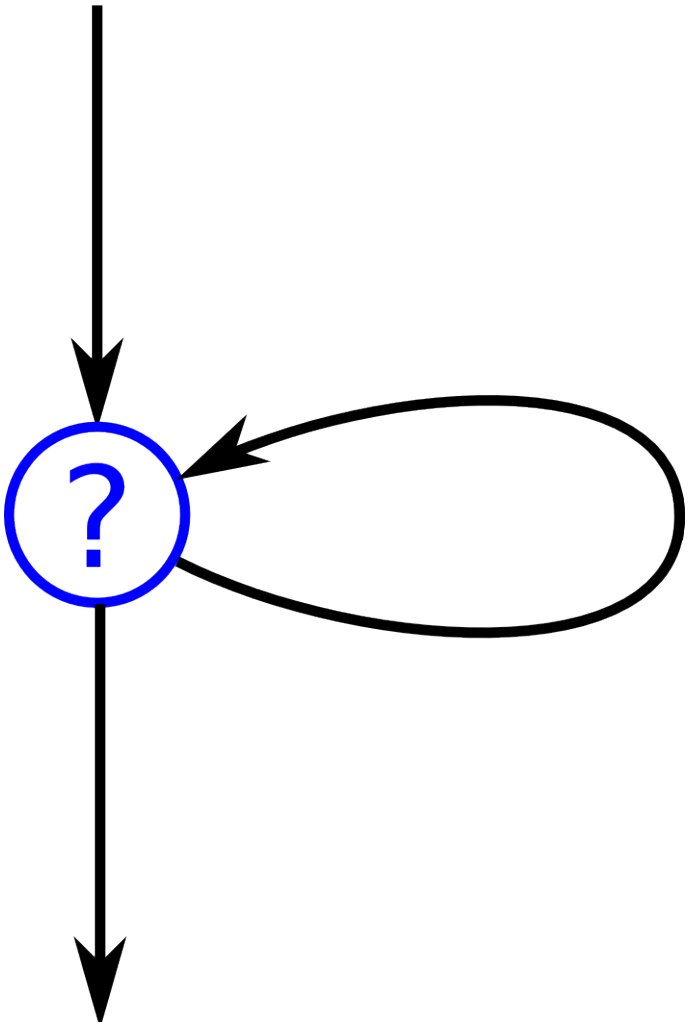
Programa más complejo

- Si > 10, ganaste
- Sino, contestamos n+1, y perdiste

ganame.py

Bucles

- Mientras algo se cumpla
- Damos vueltas y vueltas en el aire



Sintaxis del while

```
<antes>

while <expresión>:
    <bloque de código>

<después>
```

Adiviná el número I

- Versión aburrida, número prefijado
- Bucle hasta que adivina

adivina1.py

Adiviná el número II

- Calculamos uno al azar
- import, ¿qué importamos?

adivina2.py

Te ayudo a adivinar

- Calculamos también al azar
- Vamos ayudando, damos pistas
- Decimos si menor o mayor

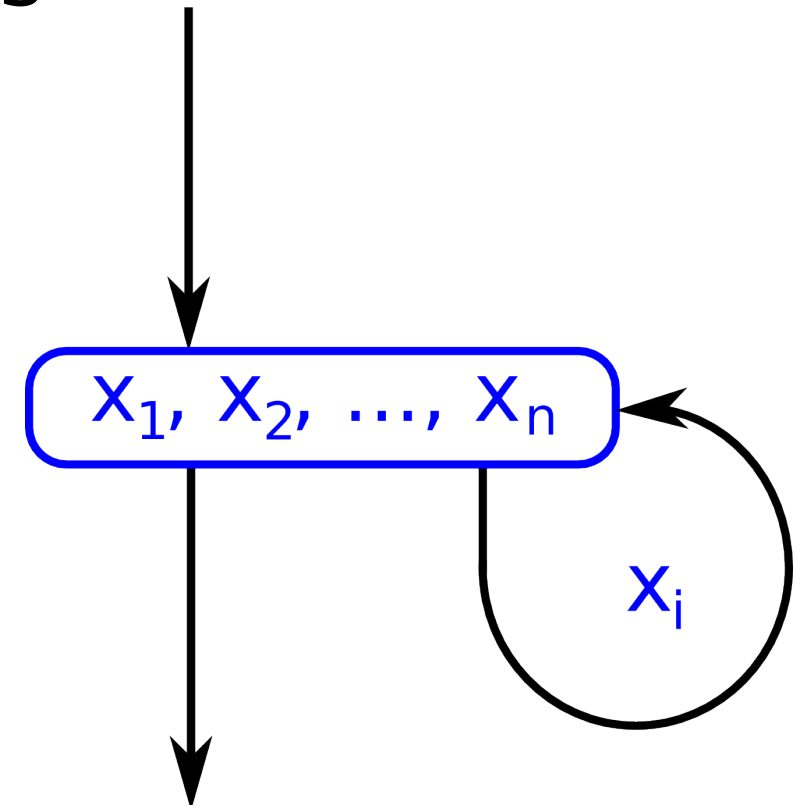
ayudamos.py

Más tipos de datos

- Punto flotante
- Arrays
- Conjuntos
- Diccionarios
- …

Recorriendo estructuras

- Por cada uno de estos
- Hacemos algo



Sintaxis del for

```
<antes>

for <nombre> in <iterable>:
    <bloque de código>

<después>
```

Mostramos los intentos

- Guardamos cada intento
- Al terminar, los mostramos

mostramos.py

Funciones

- Permite agrupar código
- Usable varias veces pero no lo repetimos
- Se le pasa parámetros
- Devuelve un resultado

Sintaxis de la función

- Definimos:

```
def <nombre>(<parámetros>):
    <bloque de código>
    return <resultado>
```

- Llamamos:

```
<nombre> = <nombrefunción>(<parámetros>)
```

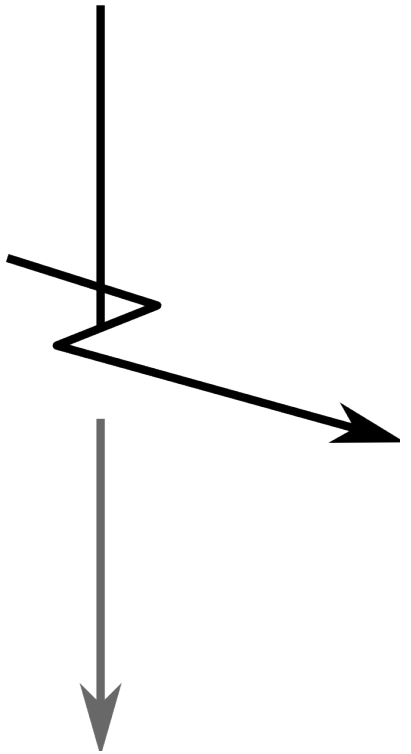
Adivinando difícil

- Que haya distintos niveles de dificultad
- Preguntamos qué nivel se quiere
- No repetimos código

dificil.py

Excepciones

- ¿Qué pasa si tenemos un problema?
- Podemos manejarlo
- O dejarlo pasar



Sintaxis de excepciones

```
<antes>

try:
    <bloque de código>
except:
    <bloque de código>
else:
    <bloque de código>

<después>
```

Somos más robustos

- Soportamos errores del usuario
- Capturamos la excepción
- Tomamos una acción correspondiente

¡Muchas gracias!

¿Preguntas?

¿Sugerencias?

Juanjo Conti

jjconti@gmail.com - http://www.juanjoconti.com.ar