Centro de Investigación en Ciencias

Diplomado en Ciencia de Datos con Python

Introducción a la Computación

Módulo 1

Jorge Hermosillo Valadez

Universidad Autónoma del Estado de Morelos

Orígenes de la computación moderna

Orígenes de la computación moderna

ENIAC (Electronic Numerical Integrator And Calculator)

- Primera computadora electrónica (a base bulbos)
- Utilización del sistema binario
- Datos y Código en memoria RAM (Arquitectura von Neumann)

Orígenes de la computación moderna

ENIAC (Electronic Numerical Integrator And Calculator)

- Primera computadora electrónica (a base bulbos)
- Utilización del sistema binario
- Datos y Código en memoria RAM (Arquitectura von Neumann)

Microprocesador

- 4004 (1971) primer microprocesador de 4 bits; 2,300 transistores
- 80386 (1985) nace la arquitectura x86; 275 K transistores
- Core i7 (2008) 1,170 M de transistores

Orígenes de la computación moderna

ENIAC (Electronic Numerical Integrator And Calculator)

- Primera computadora electrónica (a base bulbos)
- Utilización del sistema binario
- Datos y Código en memoria RAM (Arquitectura von Neumann)

Microprocesador

- 4004 (1971) primer microprocesador de 4 bits; 2,300 transistores
- 80386 (1985) nace la arquitectura x86; 275 K transistores
- Core i7 (2008) 1,170 M de transistores

Lenguajes de programación

- Ada Lovelace (1843) primer algoritmo para una máquina
- Alan Turing (1936) Concepto de Máquina Universal
- Lenguaje Ensamblador (1949)

Orígenes de la computación moderna

ENIAC (Electronic Numerical Integrator And Calculator)

- Primera computadora electrónica (a base bulbos)
- Utilización del sistema binario
- Datos y Código en memoria RAM (Arquitectura von Neumann)

Microprocesador

- 4004 (1971) primer microprocesador de 4 bits; 2,300 transistores
- 80386 (1985) nace la arquitectura x86; 275 K transistores
- Core i7 (2008) 1,170 M de transistores

Lenguajes de programación

- Ada Lovelace (1843) primer algoritmo para una máquina
- Alan Turing (1936) Concepto de Máquina Universal
- Lenguaje Ensamblador (1949)

- Fortran (1957)
- BASIC (1964)
- PASCAL (1970)
- C (1972)

Origen de la computación moderna

ENIAC (Electronic Numerical Integrator And Calculator)

- Primera computadora electrónica (a base bulbos)
- Utilización del sistema binario
- Datos y Código en memoria RAM (Arquitectura von Neumann)

Microprocesador

- 4004 (1971) primer microprocesador de 4 bits; 2,300 transistores
- 80386 (1985) nace la arquitectura x86; 275 K transistores
- Core i7 (2008) 1,170 M de transistores

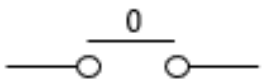
Lenguajes de programación

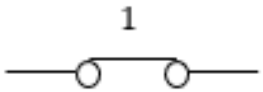
- Ada Lovelace (1843) primer algoritmo para una máquina
- Alan Turing (1936) Concepto de Máquina Universal
- Lenguaje Ensamblador (1949)
- Fortran (1957)
- BASIC (1964)
- PASCAL (1970)
- C (1972)
- C++ (1983)
- Python (1991)

Organización de una computadora

Unidades de información

- Bit (Binary Digit) 1 o 0





Unidades de información

- Bit (Binary Digit) 1 o 0

- Byte (8 bits)

MSB LSB

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

Unidades de información

- Bit (Binary Digit) 1 o 0

- Byte (8 bits)

MSB LSB

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0

- Unidades de medida

ACRÓNIMO	LECTURA	VALOR
KB	Kilo byte	1024 bytes
MB	Mega byte	10^6 bytes ≈ 1000 KB
GB	Giga byte	10^9 bytes ≈ 1000 MB
TB	Tera byte	10^{12} bytes ≈ 1000 GB
PB	Peta byte	10^{15} bytes ≈ 1000 TB

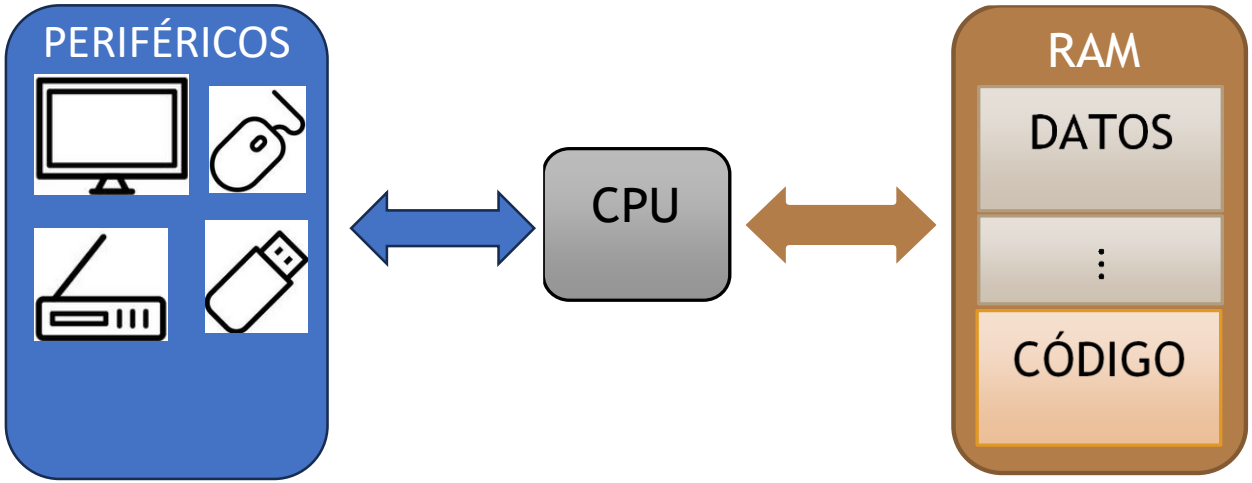
Bases Octal y Hexadecimal

- La base octal (base 8) admite los guarismos 0 al 7, y el sistema hexadecimal (base 16) cuenta con los siguientes guarismos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F.

Decimal	Binario (usando 6 dígitos)	Octal	Binario (usando 8 dígitos)	Hexadecimal
0	000|000	00	0000|0000	0x00
1	000|001	01	0000|0001	0x01
2	000|010	02	0000|0010	0x02
3	000|011	03	0000|0011	0x03
⋮	⋮	⋮	⋮	⋮
9	001|001	11	0000|1001	0x09
10	001|010	12	0000|1010	0x0A
15	001|111	17	0000|1111	0x0F

Noción de arquitectura

- La arquitectura de una computadora se refiere a la forma en que están organizados los distintos dispositivos que cumplen con funciones esenciales en la operación de una computadora.
  - Memoria RAM
  - Unidad Central de Procesamiento (CPU)
  - Periféricos (pantalla, teclado, ratón, impresoras, internet, ...)

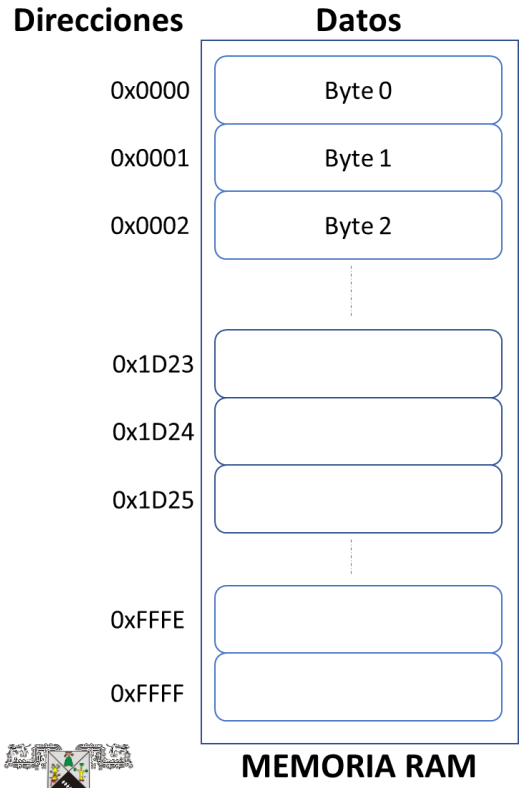


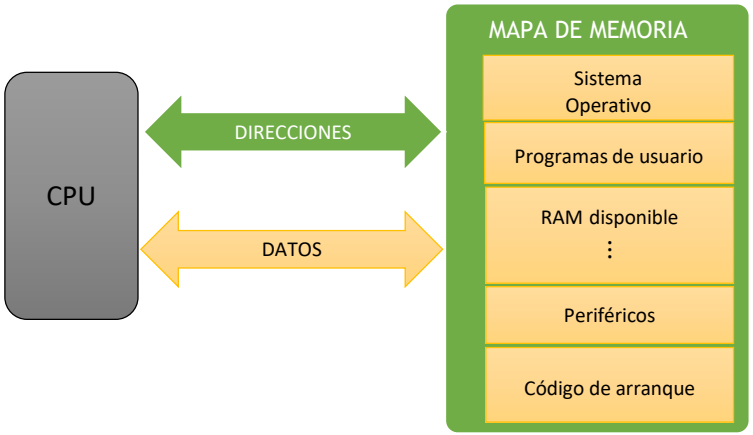
	STUDENT NAME	MATHS	ENGLISH	BIOLOGY	PHYSICS	CHEMISTRY
3	ANURAG KUMAR	87	57	77	63	87
4	SAPTARSHI MONDAL	98	88	58	85	90
5	SARTHAK GHOSH	85	95	45	90	81
6	NISCHAY	32	62	39	98	62
7	AKASH SHARMA	66	46	73	66	76
8	DEEPESH	72	12	53	79	72
9	PRATEEK	56	76	94	66	80
10	PRATISH	98	66	43	87	44
11	SHIVANI	92	52	62	91	77
12	SHRUTI	59	49	72	49	34
13	SHREYA	47	60	31	87	17

```
requests.get(url)
# Checking response.status_code (if you get 502, try
if response.status_code != 200:
    print(f"Status: {response.status_code} - Try
else:
    print(f"Status: {response.status_code}\n")
```

Memoria RAM

- La memoria RAM es donde se almacenan físicamente los datos y programas en una computadora. Es donde esencialmente reside toda la información que procesa el CPU, o microprocesador.





Nociones de programación

Algoritmo vs programa

- Un algoritmo es:
  - una secuencia ordenada de pasos individuales (paso 1, 2, 3)
  - definidos (una acción a la vez)
  - finitos (siempre terminan)
  - para resolver un problema (tener claro el problema).

- Un programa es una **estructura de datos** y un **algoritmo** escrito en un lenguaje de programación para resolver un problema.

Noción de variable

- Recordemos que una computadora tiene memoria, por lo que podemos evitar la pérdida de información si recurrimos a ella. Para ello, usamos variables.

Noción de variable

- Recordemos que una computadora tiene memoria, por lo que podemos evitar la pérdida de información si recurrimos a ella. Para ello, usamos variables.
- Una variable es un contenedor de información que la computadora mantiene en algún lugar físico de la memoria.

Noción de variable

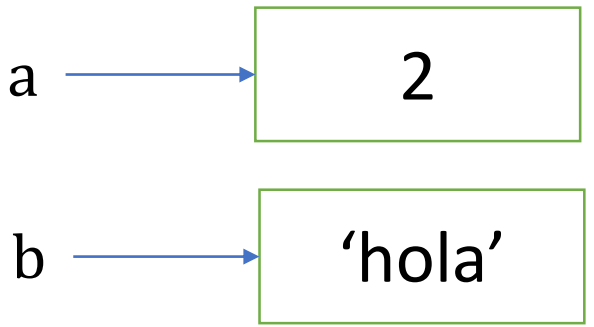
- Recordemos que una computadora tiene memoria, por lo que podemos evitar la pérdida de información si recurrimos a ella. Para ello, usamos variables.
- Una variable es un contenedor de información que la computadora mantiene en algún lugar físico de la memoria.
- Un programa puede escribir información en el contenedor, lo que se conoce como asignación a la variable, y el contenido de ésta permanece guardado mientras no haya otra asignación.

Noción de variable

- Asignación de una variable

$a = 2$

$b = \text{‘hola’}$



Noción de variable

- **Lectura de una variable**

a

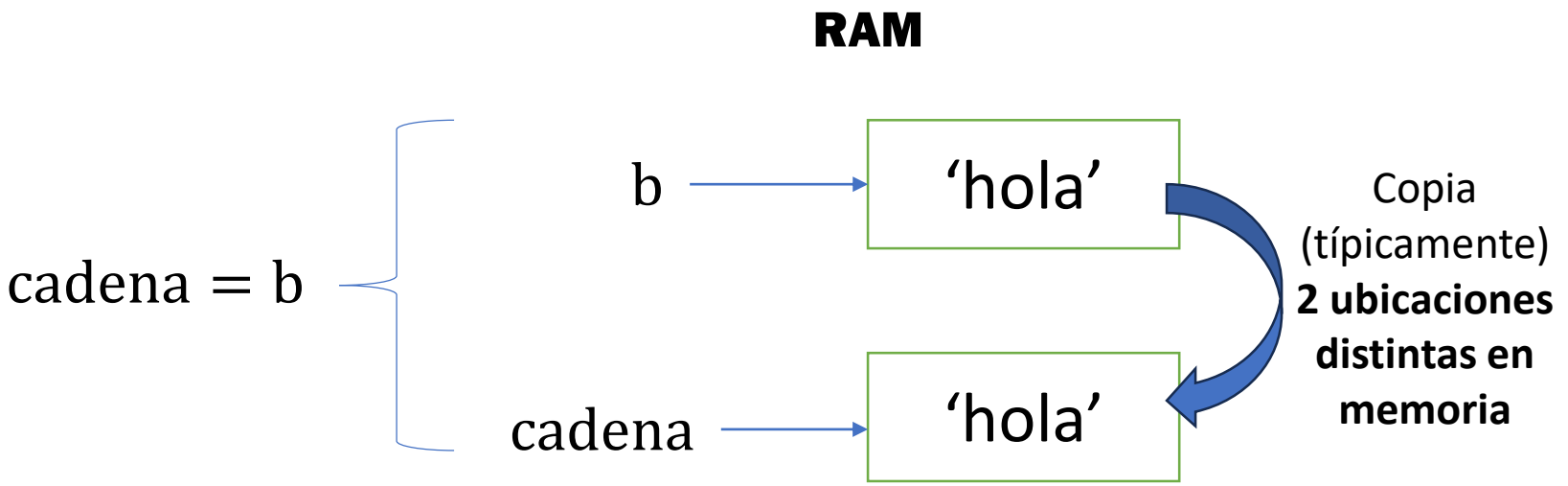
El valor 2 se imprime en pantalla o “no pasa nada”

Noción de variable

- **Lectura de una variable**

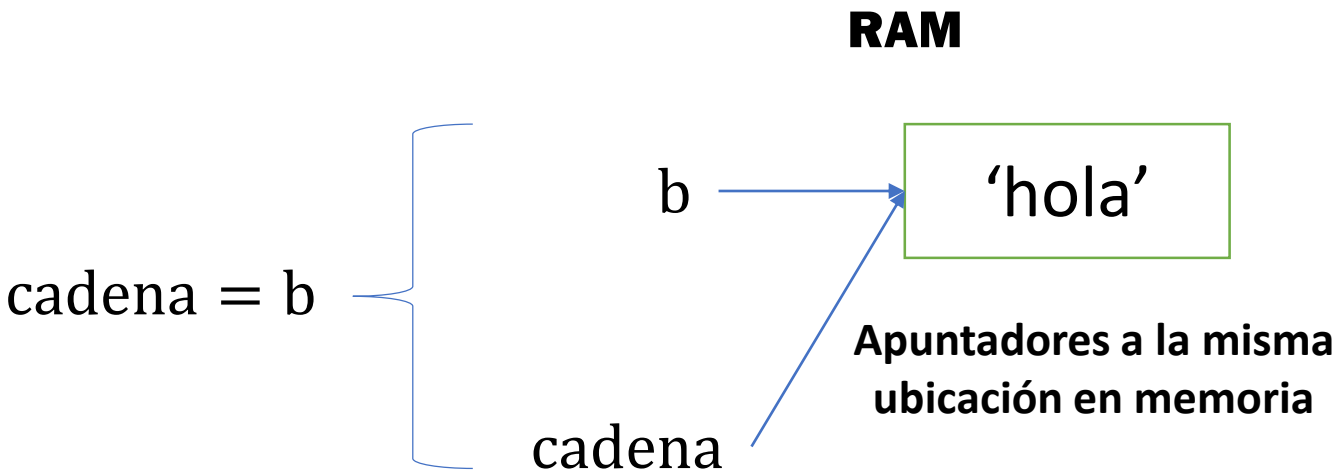
a

El valor 2 se imprime en pantalla o "no pasa nada"



Noción de variable

- Lectura de una variable (en Python)



Lógica booleana (lógica binaria)

- Una variable lógica, o booleana, es una entidad en memoria que representa un valor lógico de verdadero o falso, pero no ambos a la vez.

Lógica booleana (lógica binaria)

- Una variable lógica, o booleana, es una entidad en memoria que representa un valor lógico de verdadero o falso, pero no ambos a la vez.
- Al igual que los operadores aritméticos, los operadores lógicos actúan sobre una o más variables para producir un resultado: se dice que el operador recibe valores de entrada y produce valores de salida.
- Negación: NO (NOT)

entrada	salida
A	$\text{not}(A)$
V	F
F	V

Lógica booleana (lógica binaria)

- Conjunción: Y (AND)

entrada	salida
A B	A and B
V V	V
V F	F
F V	F
F F	F

- Disyunción: O-inclusivo (OR)

entrada	salida
A B	A or B
V V	V
V F	V
F V	V
F F	F

Composición de operadores lógicos

- NO-O (NOR)

NOR		
entrada		salida
A	B	$\neg(A \vee B)$
V	V	F
V	F	F
F	V	F
F	F	V

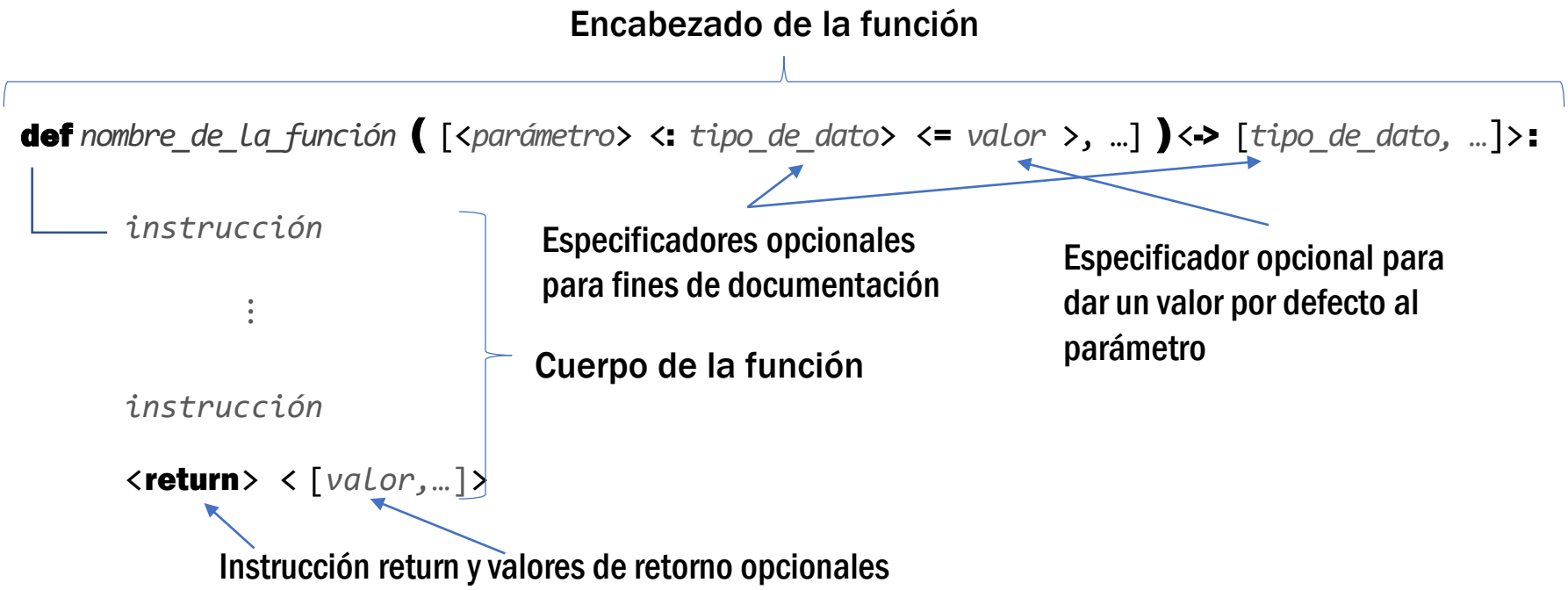
- NO-Y (NAND)

NAND		
entrada		salida
A	B	$\neg(A \wedge B)$
V	V	F
V	F	V
F	V	V
F	F	V

Noción de función

- Una función es una rutina o subprograma que ejecuta una tarea específica y puede, o no, devolver un valor de algún tipo de dato.
- El propósito de una función es *encapsular* la ejecución de esa tarea específica, para que pueda ser utilizada tantas veces como sea necesario en un programa, y en cualquier programa.

Declaración de una función en Python



Uso de una función en Python

```
def mi_funcion1 ( a ):
    b = a*2
    return b
```

```
def mi_funcion2 ( a = 3 ):
    b = a*2
    return b
```

```
def mi_funcion3 ( a:int = 3 ) ->int:
    b = a*2
    return b
```

- Para usar una función, simplemente se invoca o llama usando su nombre, e incluyendo los parámetros necesarios o deseados.

`c = mi_funcion1()`	Produce error: mi_funcion1 requiere un parámetro faltante
`c = mi_funcion2()`	c contiene el valor 3*2=6, se utiliza el valor asignado por defecto
`c = mi_funcion2(4)`	c contiene el valor 4*2=8, se utiliza el valor del parámetro
`c = mi_funcion3(4)`	c contiene el valor 4*2=8, igual al anterior