TEMA 1
INTRODUCCIÓN

TEMA 1. INTRODUCCIÓN

- **1.1 Informática básica**
- **1.2 Organización física de un ordenador electrónico digital**

  HARDWARE
- **1.3 Sistemas Operativos**

  SOFTWARE

1.1 Informática básica

- Datos → Magnitudes numéricas, nombres, conjuntos de símbolos, imágenes, etc …
- Información → Datos elaborados para proporcionar conocimiento, adoptar decisiones.
- INFORMATICA → Acrónimo de INFORMación automÁTICA.

  Es la ciencia del tratamiento automático de la información.

  Necesitamos:
  - Representarla eficiente y automatizablemente.
  - Transmitirla sin errores ni pérdidas.
  - Almacenarla para acceder a ella y recuperarla tanto como sea preciso.
  - Procesarla para obtener nuevas informaciones y más útiles.

1.1 Informática básica

"Los ordenadores son máquinas capaces de aceptar datos a través de un medio de entrada, procesarlos automáticamente bajo el control de un programa previamente almacenado y proporcionar la información resultante mediante un dispositivo de salida."

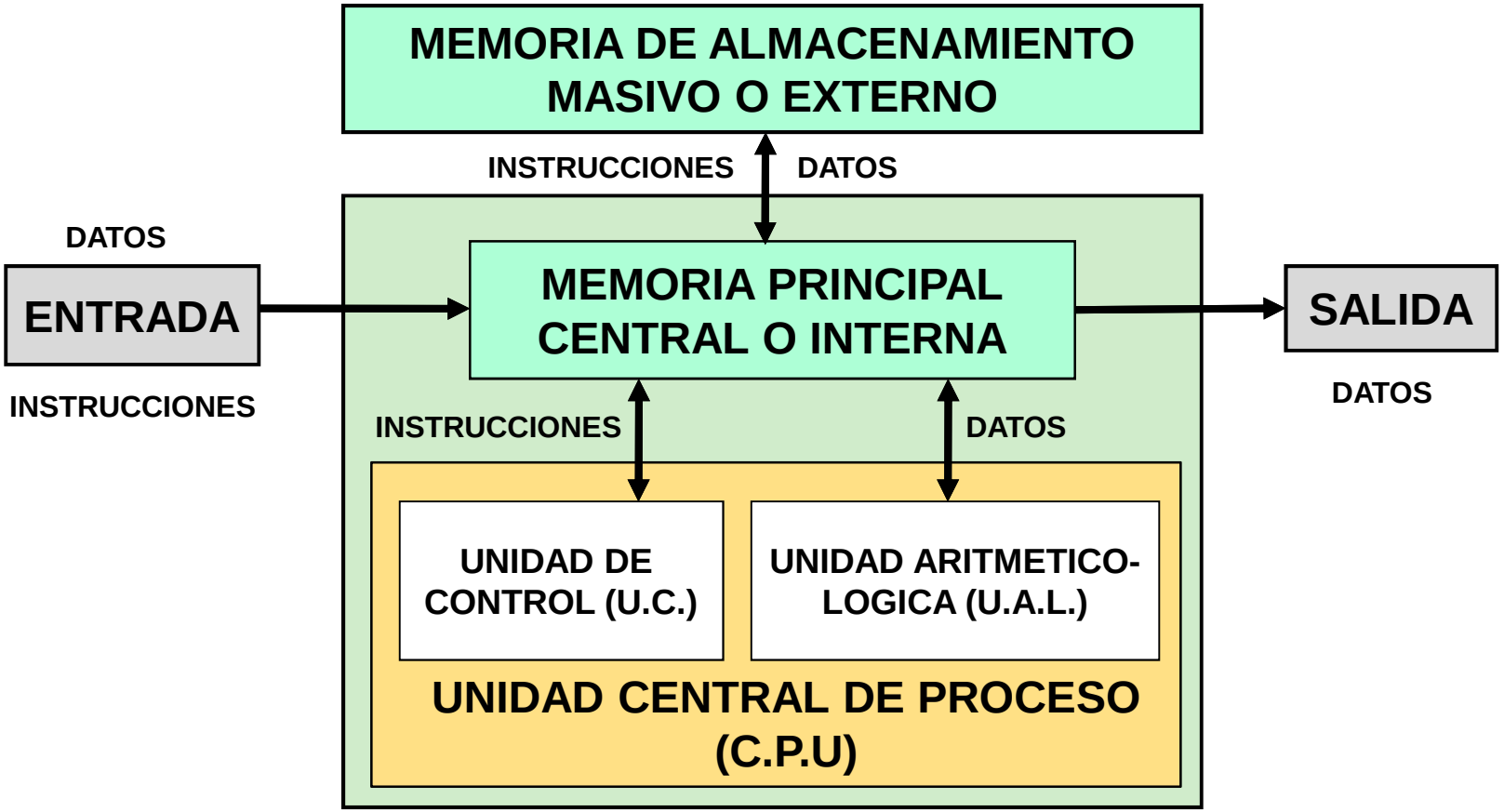
1.1 Informática básica

- **800** El matemático persa Musa al-Juarismi inventa el algoritmo,
- **1645** Pascal inventa la pascalina, una de las primeras calculadoras mecánicas
- **1837** Charles Babbage describe la máquina analítica, computador moderno de propósito general. Padre de las Computadoras Modernas
- **1843** Ada Augusta Lovelace se convierte en la primera programadora.
- **1936** Alan Turing describe la máquina de Turing, formalizando el uso del algoritmo.
- **1942** John Atanasoff diseña una calculadora de propósito especial para resolver sistemas de ecuaciones lineales simultáneas.
- **1944** Se desarrolla en la Universidad de Harvard la Mark I primer ordenador electromecánico.

1.1 Informática básica

- **1ª Generación**
  - Válvulas de vacío
  - Tarjetas perforadas
  - 1945 ENIAC
  - 1951 UNIVAC
- **2ª Generación**
  - Transistores
  - Sistemas de almacenamiento
  - COBOL, FORTRAN
  - 1962 ATLAS
- **3ª Generación**
  - Circuitos integrados (Chip)
  - Software BASIC, PASCAL
  - Compatibilidad
  - 1964 IBM 360
- **4ª Generación**
  - Microprocesador
  - Multitarea
  - MS DOS
  - 1971 IBM 370
- **5ª Generación?**

1.2 Organización física de un ordenador electrónico digital, HARDWARE



- Dispositivos de entrada:
  - Tarjetas perforadas
  - Teclado
  - Ratón
  - Lápiz electrónico, joystick
  - Lectores de CD, DVD
  - Micrófono

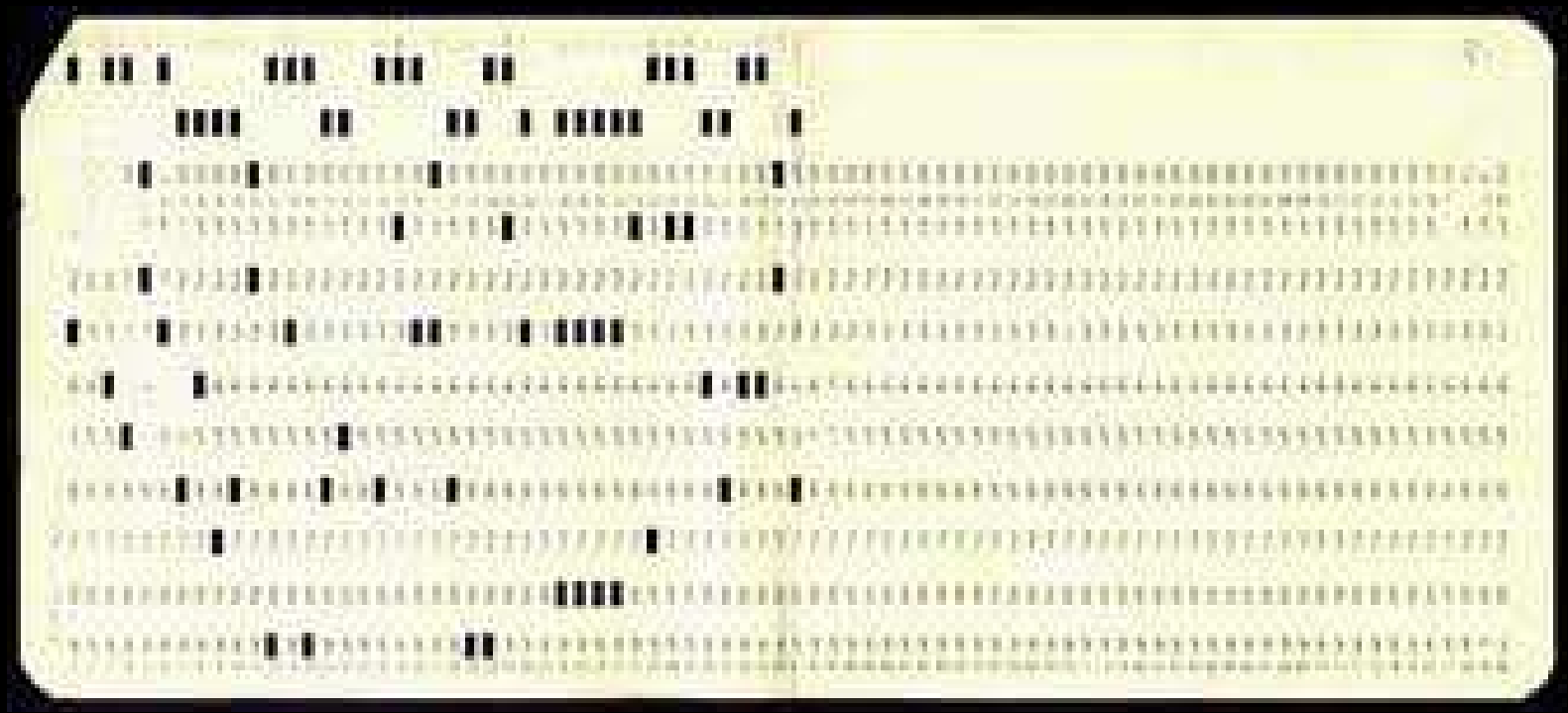
- Dispositivos de salida:
  - Impresoras
  - Pantalla o monitor

- Dispositivos de almacenamiento externo:

  Acceso a través de memoria central

  - Cinta magnética: Acceso secuencial. Copias de seguridad
  - Discos magnéticos: Acceso directo. Discos flexibles y duros
  - Discos ópticos: CD, DVD,
  - Electrónicos: Pendrive, mp3, mp4

Tarjeta IBM



- Memoria Central o Interna
  - **BIT** (**BI**nary uni**T**): Unidad básica de información, representada por un condensador. Toma dos estados (0/1, on/off)
  - **BYTE** o palabra: Conjuntos de celdas direccionadas que contienen a su vez un numero fijo de bits, normalmente 8.
  - Capacidad de memoria en múltiplos de bytes: Kbytes, Mbytes, Gbytes.
  - Tipos de memorias:
    - Memoria ROM “Read Only Memory” (BIOS)
      - Solo lectura
      - No volátil
    - Memoria RAM “Random Access Memory”
      - Acceso directo. Lectura y escritura.
      - Guarda programas y datos del usuario. Volátil.
      - Los datos del almacenamiento externo deben pasar a esta memoria para ser procesados.
    - Memoria Caché
      - Memoria de acceso muy rápido entre la RAM y el microprocesador.

- Canales o buses

  Gestión física de las ordenes de entrada y salida, conexiones entre partes del ordenador.

- Unidad Central de Proceso-Procesador-CPU
  - Unidad Aritmético-Lógica → Operaciones aritméticas y lógicas
  - Unidad de Control → Circuitos que coordinan las actividades de la máquina.
    - Reloj: Señales para inicio de ciclo de control.
    - Contador o registro contador: Dirección de siguiente instrucción a ejecutar.
    - Registros de memoria (o de instrucciones): Instrucción a ejecutar
    - Decodificador: Analiza la instrucción

1.3 Sistema Operativo

- Software → Soporte lógico del ordenador. Programas que controlan el hardware.
  - Sistema Operativo
  - Lenguajes de programación y algoritmos. Compiladores
  - Aplicaciones

1.3 Sistema Operativo

- *Conjunto de programas y archivos que posibilitan el acceso del usuario al ordenador, y realizan la gestión de los recursos disponibles del sistema.*
- Actividades comunes a los SO:
  - Control de los datos y de su acceso

    Datos (programas) → Ficheros → Memoria de almacenamiento externo → Copiados a Memoria Central

    SO se encarga de la manipulación de ficheros. Guarda registro: qué ficheros están almacenados, dónde se encuentran en memoria externa, y en sistemas usados por varias personas quién podrá tener acceso a esos ficheros.

- Proveer acceso eficiente a los dispositivos:

  Los SO proveen rutinas para utilizar la mayoría de los dispositivos periféricos.

- Manejo de recursos:

  Los recursos incluyen: área de memoria, dispositivos periféricos y programas.

  Un SO debe permitir acceder a la misma máquina a varias personas al mismo tiempo. Delimitar qué dispositivo, áreas de memoria, etc están siendo usadas y por quién.

  Ej: Una impresora es usada por dos personas al mismo tiempo. Hay que evitar la colisión.

- Control de acceso a la máquina:

  Si mucha gente tiene acceso a una máquina, su uso debe ser reservado para ciertas personas. Código de acceso.

- Estandarización del interface entre la persona y la máquina:

  Aunque dos máquinas tengan diferente diseño y construcción, sus SO deben tener el mismo dialogo con el usuario.

  Ej: Macintosh (Apple Computer) y Windows (IBM)

- Intérprete de comandos :

  Programa que consiste en admitir los comandos solicitados por el usuario, analizarlos y en el caso de que pertenezcan al lenguaje de comandos, ejecutarlos.

Clasificación de los sistemas operativos

- Orientados a comandos: MsDos, UNIX, linux

  Comandos: Órdenes al SO

- Orientados a objetos: Macintosh, Windows
  - Objetos: Ventanas, Barra de menús, Iconos(carpetas, aplicaciones, documentos)
  - Acciones: Seleccionar, abrir, arrastrar

Lenguajes de programación

- Sistema binario (base 2)

$$\begin{array}{cccccc} 1 & 0 & 1 & 1 & 0 & 1 \\ 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \end{array}$$

$$32 + 0 + 8 + 4 + 0 + 1 = 45$$

101101 en binario = 45 en decimal

- Sistema hexadecimal (base 16)

$$\begin{array}{cccc} A(10) & 4 & F(15) & 3 \\ 16^3 & 16^2 & 16^1 & 16^0 \end{array}$$

$$10*4096 + 4*256 + 15*16 + 3*1 = 42227$$

A4F3 en hexadecimal = 42227 en decimal

Usado para indicar direcciones de memoria

Lenguajes de programación

- Clasificación 1
  - **Bajo nivel**: Ligados a un hardware particular. Se traslada fácil a lenguaje máquina (ENSAMBLADOR)
    - Mas complicado
    - Mas rápido
  - **Alto nivel**: Sintaxis más parecida al lenguaje natural. Funcionan en diversas máquinas (PYTHON, JAVA..)
    - Mas sencillo
    - Mas lento

Lenguajes de programación

- Clasificación 2

  - **Intérprete**: Para ejecutar un programa hay que traducirlo del lenguaje de alto nivel al lenguaje máquina instrucción a instrucción (BASIC, PYTHON..)

  - **Compilado**: Antes de ejecutar, se traduce completo y almacena (PASCAL, C..). Podemos ejecutarlos tantas veces como queramos

  PYTHON funciona de las dos maneras.

En ensamblador

```
section .data

msg     db "¡Hola Mundo!", 0Ah
len     equ     $ - msg

section .text

global _start

_start:
        mov     eax, 04h
        mov     ebx, 01h
        mov     ecx, msg
        mov     edx, len
        int     80h
        mov     eax, 01h
        mov     ebx, 00h
        int     80h
```

En Python 3.0

```
print('Hola mundo')
```

https://py3.codeskulptor.org/

TEMA 2
PROGRAMACIÓN

TEMA 2. PROGRAMACIÓN

- 2.1 Algoritmos
- 2.2 Tipos de datos simples. Variables
- 2.3 Instrucciones básicas
- 2.4 Estructuras de control
- 2.5 Tipos de datos estructurados: Secuencias
- 2.6 Funciones

2.1 Algoritmos

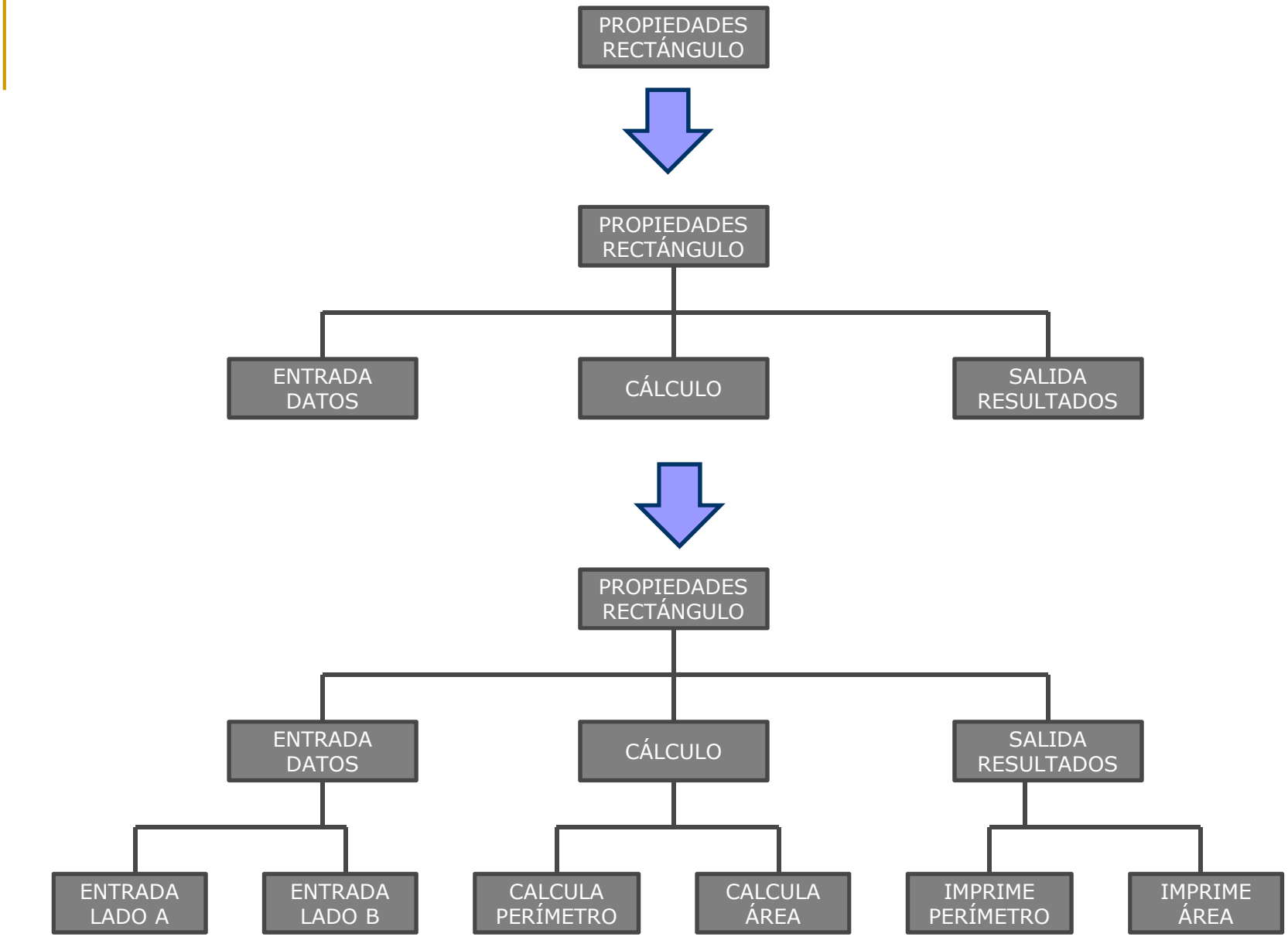
*Un algoritmo es una secuencia finita y ordenada de **acciones primitivas**, entendiendo como tales, aquellas que pueden ser ejecutadas por el procesador*

- *Tiempo finito*
- *Factible*
- *Preciso*
- *Que nos acerque al objetivo*

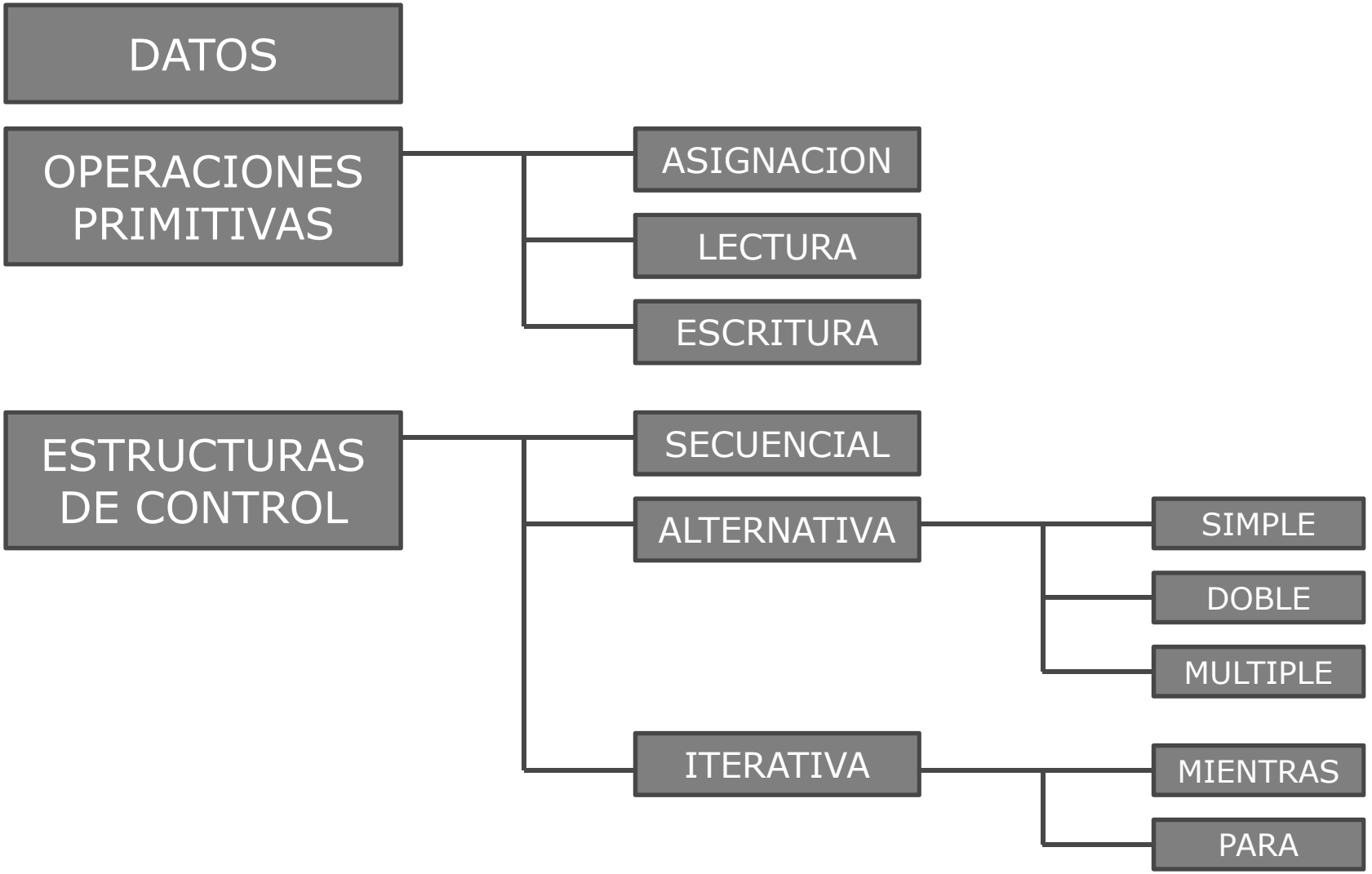
Técnica para el diseño de algoritmos:

Programación estructurada

- *Programa con diseño modular*
- *Cada módulo tiene diseño descendente*
- *Cada módulo se codifica con estructuras de control secuenciales, selectivas e iterativas*


PROPIEDADES RECTÁNGULO
PROPIEDADES RECTÁNGULO
ENTRADA DATOS
CÁLCULO
SALIDA RESULTADOS
PROPIEDADES RECTÁNGULO
ENTRADA DATOS
CÁLCULO
SALIDA RESULTADOS
ENTRADA LADO A
ENTRADA LADO B
CALCULA PERÍMETRO
CALCULA ÁREA
IMPRIME PERÍMETRO
IMPRIME ÁREA

COMPONENTES DE UN PROGRAMA



2.1 Algoritmos
Lenguaje Python

Python fue creado a principios de los noventa por **Guido van Rossum** en los Países Bajos. Es relativamente joven.

Ventajas

- Python es un lenguaje **muy expresivo**, es decir, los programas Python son **muy compactos**. Un programa Python suele ser bastante más corto que su equivalente en lenguajes como C.
- Python llega a ser considerado por muchos un *lenguaje de programación de **muy alto nivel***.
- Python es **muy legible**. La sintaxis de Python es muy elegante y permite la escritura de programas cuya lectura resulta más fácil que si utilizáramos otros lenguajes de programación.

2.1 Algoritmos
Lenguaje Python

Ventajas

- Python ofrece un **entorno interactivo** que facilita la realización de pruebas y ayuda a despejar dudas acerca de ciertas características del lenguaje.
- El entorno de ejecución de Python **detecta** muchos de los **errores** de programación que escapan al control de los compiladores y proporciona información muy rica para detectarlos y corregirlos.
- Python puede usarse como lenguaje imperativo **procedimental** o como lenguaje **orientado a objetos**.
- Posee un rico juego de **estructuras de datos** que se pueden manipular de modo sencillo.
- Es **Interpretado**

2.2 Tipos de datos simples. Variables

Estructura básica de un programa

- **INDICADORES**: Configuran la sintaxis del programa

  Palabras reservadas: and, as, assert, break, class, continue, def, del, elif, else, except, False, finally, for, from, global, if, import, in, is, lambda, nonlocal, None, not, or, pass, raise, return, True, try, with, while y yield.

- **IDENTIFICADORES**: Designan datos y partes del programa
  - **Identificadores normalizados:** int, print…
  - **Identificadores creados por el usuario:**

  Formados por letras minúsculas, mayúsculas, dígitos y/o el carácter de subrayado, con dos restricciones: que no sea palabra reservada y que el primer carácter no sea un dígito.

2.2 Tipos de datos simples. Variables

Características de los identificadores

- Python **distingue** mayúsculas de minúsculas
- No pueden tener **blancos**
- Formados por **letras, dígitos** y el carácter **subrayado** (_)
- Pueden tener **cualquier longitud** (A ver…)
- Pueden **redefinirse**, pero **no** es conveniente
- Se aconseja que tenga **significado**

SÍ	**NO**
Dia_nacimiento	1virtual
digito	dia 12
NIVEL1	ladoN@
a	numero-plazas

2.2 Tipos de datos simples. Variables

Programas o scripts

Los programas en Python tienen extensión .py

Ejemplos:

perimetro.py

```
1 from math import pi
2 radio = 1
3 perímetro = 2 * pi * radio
4 print(perímetro)
```

esfera.py

```
1 from math import pi
2
3 cadena_leída = input()
4 radio = float(cadena_leída)
5
6 volumen = 4 / 3 * pi * radio ** 3
7
8 print(volumen)
```

2.2 Tipos de datos simples. Variables

- **Tipos simples (Inmutables):**
  - **Enteros *int***, **reales *float*** y **lógicos *boolean***.
  - No existe el tipo carácter.
- **Tipos compuestos inmutables:**
  - **Conjunto congelado *frozenset*** : colección de elementos, potencialmente de distinto tipo (mientras sean inmutables), no repetidos y sin orden entre sí.

    Ejemplo: congelado=frozenset({3,5,6.1})
  - **Cadena *str*** : secuencia de caracteres

    Ejemplo: cadena=’buenos días’
  - **Tupla *tuple***: secuencia de 0, 1 o n elementos, potencialmente de distinto tipo

    Ejemplo: t1=(1,’a’,3,3)

2.2 Tipos de datos simples. Variables

- **Tipos compuestos mutables:**
  - **Conjunto *set*:** colección de elementos (sin orden y no repetidos), potencialmente de distinto tipo simple al que se le pueden añadir nuevos elementos o eliminar existentes.

    Ejemplo: conj1={'infinito',1,0,5,('a',1)}
  - **Lista *list*:** secuencia de elementos, potencialmente de distinto tipo (incluidos mutables), a la que se le puede eliminar elementos, añadir nuevos, y modificar valores individuales.

    Ejemplo: lista=['hola',4, (1,2),{3,4}]
  - **Diccionario *dict*:** colección (conjunto) de pares de clave-valor. La clave es de cualquier tipo inmutable. Los valores pueden ser de cualquier tipo.

    Ejemplo: ingredientes={'tomate':(1,'Kg'), 'pepino':2,'sal':'1 cuchara',
    'aceite':.1, 'oregano':'1 pizca'}

2.2 Tipos de datos simples. Variables

- Tipo entero ***int***

  Tamaño ilimitado.

  - Operadores
    - Binarios de relación: ==, <, >, !=, <=, >=
    - Aritméticos: +, -, *, //, %, **
    - *Casting* (convertir el valor de una variable de un tipo a otro): *int()*

  Ejemplos: 5 // 3 es 1

  5 % 3 es 2

  5 ** 2 es 25

  int(-2.9) es -2

  int('2') es 2

2.2 Tipos de datos simples. Variables

- Tipo real ***float***
  - Python utiliza doble precisión: 64 bits
  - Operadores
    - Relacionales: ==, <, >, !=, <=, >=
    - Aritméticos: +, -, *, /, **
    - Funciones:
      - Predefinidas: *abs*, *round, float…*
      - Definidas en módulos: *sin*, *cos* (módulo *math*)…

  Ejemplo: *abs*(-3) es 3

  float('3.2') es 3.2

2.2 Tipos de datos simples. Variables

- Tipo lógico ***boolean***
  - Un dato de tipo lógico sólo puede presentar uno de dos valores: ***True o False***
  - Operadores
    - Relacionales: ==, <, >, !=, <=, >=
    - Lógicos: **and**, **or**, **not**

Ejemplos: 3 != 5 es True

3 < 5 and 17 < 10 es False

p	q	**not** p	p **and** q	p **or** q
F	**F**	**T**	**F**	**F**
F	**T**	**T**	**F**	**T**
T	**F**	**F**	**F**	**T**
T	**T**	**F**	**T**	**T**

2.2 Tipos de datos simples. Variables

Operación	Operador	Aridad	Asociatividad	Precedencia
Exponenciación	**	Binario	Por la derecha	1
Identidad	+	Unario	—	2
Cambio de signo	-	Unario	—	2
Multiplicación	*	Binario	Por la izquierda	3
División	/	Binario	Por la izquierda	3
División entera	//	Binario	Por la izquierda	3
Módulo (o resto)	%	Binario	Por la izquierda	3
Suma	+	Binario	Por la izquierda	4
Resta	-	Binario	Por la izquierda	4
Igual que	==	Binario	—	5
Distinto de	!=	Binario	—	5
Menor que	<	Binario	—	5
Menor o igual que	<=	Binario	—	5
Mayor que	>	Binario	—	5
Mayor o Igual que	>=	Binario	—	5
Negación	not	Unario	—	6
Conjunción	and	Binario	Por la izquierda	7
Disyunción	or	Binario	Por la izquierda	8

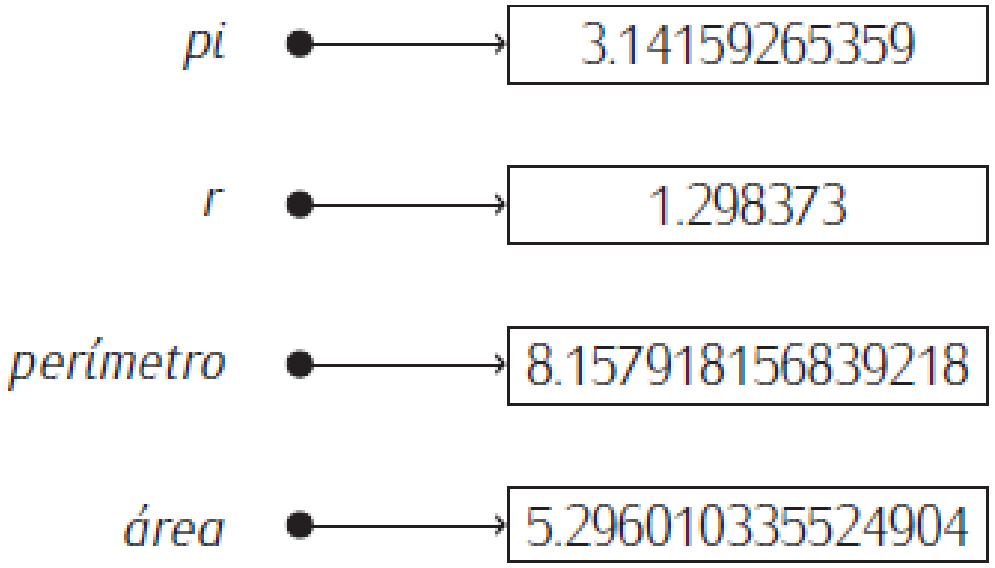
2.2 Tipos de datos simples. Variables

Utilizaremos **variables** en nuestros programas para guardar valores. Estas variables serán de los tipos que hemos descrito.

- El nombre de una variable es su **identificador**: formado por letras minúsculas, mayúsculas, dígitos y/o el carácter de subrayado, no puede empezar por un dígito.
- Este identificador no debe coincidir con ninguna **palabra reservada**.
- Python distingue entre **mayúsculas y minúsculas**, así que nombre, Nombre y NOMBRE son tres identificadores válidos y diferentes.

2.2 Tipos de datos simples. Variables

Ejemplo: Supongamos que deseamos efectuar el cálculo del perímetro y el área de un círculo. La fórmula del perímetro es 2*pi*r, donde r es el radio, y la fórmula del área es pi*r**2. Si el radio es 1.298373 m, tenemos:



2.3 Instrucciones básicas

- **Asignación**

  En Python, la primera operación sobre una variable debe ser la **asignación** de un valor: **variable = expresión**

  Se evalúa la expresión a la derecha del símbolo igual y se guarda el valor resultante en la variable indicada a la izquierda del símbolo igual.

  - Recuerda que comparar (**==**) no es asignar (**=**)

  Asignaciones con operador: +=, -=, *=, /=, %=, //=, **=

  Ejemplos:

```
x = 3
x = x + 1
print(x)
```

```
4
```

```
a = 3
b = 2
a += b * 4
print(a)
```

```
11
```

```
x, y = 27, 12
print(x, y)
```

```
27 12
```

2.3 Instrucciones básicas

- **Lectura**

  Vamos a aprender a hacer que nuestro programa, cuando se ejecute, pida datos que se introduzcan desde teclado: utilizaremos el comando ***input()***

  Esta función detiene la ejecución del programa y espera a que el usuario escriba un texto y pulse la tecla de retorno de carro; en ese momento prosigue la ejecución y la función devuelve una **cadena con el texto** que tecleó el usuario.

  Ejemplo: Tres formas de leer datos para calcular el volumen de una esfera

2.3 Instrucciones básicas

- **Lectura**

A

```
from math import pi
print('Dame el radio: ')
cadenaLeida = input()
radio = float(cadenaLeida)
volumen = 4 / 3 * pi * radio ** 3
print(volumen)
```

B

```
from math import pi
cadenaLeida = input('Dame el radio: ')
radio = float(cadenaLeida)
volumen = 4 / 3 * pi * radio ** 3
print(volumen)
```

C

```
from math import pi
radio = float(input('Dame el radio: '))
volumen = 4 / 3 * pi * radio ** 3
print(volumen)
```

2.3 Instrucciones básicas

- **Escritura**

Vamos a aprender escribir valores en pantalla: ***print(expresión)***

```
from math import pi
print('Programa para calcular el volumen de una esfera')
radio = float(input('Dame el radio en metros: '))
volumen = 4 / 3 * pi * radio ** 3
print('El volumen es', volumen,'metros cúbicos')
print('Gracias por usar nuestro programa')
```

```
Programa para calcular el volumen de una esfera
Dame el radio en metros: 7
El volumen es 1436.7550402417319 metros cúbicos
Gracias por usar nuestro programa
```

Para que no haya retorno de carro: ***print(expresión, end=‘ ‘)***

2.3 Instrucciones básicas

- **Escritura**

  Otros formatos: **%**

```
a = 7
b = 4.5678
c = 'xyz'
print('entero: %d, real: %f, cadena: %s' %(a, b, c))
```

```
entero: 7, real: 4.567800, cadena: xyz
```

```
a = 7
b = 4.5678
c = 'xyz'
print('entero: %6d, real: %5.2f, cadena: %s' %(a, b, c))
```

```
entero:      7, real:  4.57, cadena: xyz
```

2.3 Instrucciones básicas

- **Escritura**

  Otros formatos: ***.format***

```
a = 7
b = 4.5678
c = 'xyz'
print('entero: {}, real: {}, cadena: {}'.format(a, b, c))
```

```
entero: 7, real: 4.5678, cadena: xyz
```

```
print('entero: {2}, real: {0}, cadena: {1}'.format(a, b, c))
```

```
entero: xyz, real: 7, cadena: 4.5678
```

```
print('entero: {:<4}, real: {:4.2f}, real2: {:4.4}'.format(a, b, 34.56554))
```

```
entero: 7   , real: 4.57, real2: 34.57
```