



# 01.1

- > **Introducción a la programación**
- > **Fundamentos**

**¿Qué hace un programa de  
computación?**

**Entrada  
de datos**

**almacenamiento de  
datos**

- Variables
- Bases de datos
- Memoria
- Archivos

**Transformación  
y Procesamiento**

**Salida  
de datos**

- Escribir un archivo csv
- Presentarlo en una aplicación
- Generar un reporte
- Hacer gráficos

# ¿Qué es un compilador?

- Analiza la totalidad del código fuente y lo traduce a lenguaje máquina y produce un archivo compilado cerrado que al ejecutarse, se ejecuta el programa.
- Es el equivalente a un traductor, del lenguaje de programación, a instrucciones de bajo nivel para la computadora
- Si hay algún error en el programa, “no compila”

# Lenguajes compilados

- C
- C++
- Java
- Cobol
- Pascal

**¿Qué es un intérprete?**





# ¿Qué es un intérprete?

- Es un programa que ejecuta directamente el código fuente `_línea por línea_` cuando así es requerido
- Se le llama de alto nivel porque al ejecutar directamente el código fuente, lo ejecutado no está traducido a “lenguaje de bajo nivel”, por ello es más lento
- Los errores en el programa no aparecen a menos que la función que contiene el error sea ejecutada

# Lenguajes interpretados

- Javascript
- Python
- Ruby
- PHP

# Paradigmas de programación

# Programación estructurada

```
#!/usr/bin/python
#encoding=utf-8
import MySQLdb
def leernombres(x):
    mnombres=raw_input('Nombres y apellidos:')
    mnombres=""
    return mnombres
mci=""
mnombres=""
r=""
b=MySQLdb.connect('localhost',user='root',passwd='1234567',db='meivin')
p=b.cursor()
while (r=="s"):
    mci=raw_input('Introduzca código a consultar:')
    #p=b.cursor()
    p.execute('SELECT * FROM usuario');
    g=p.fetchone()
    existe=0
    while (g!=None and existe==0):
        if mci==g[0]:
            print 'Consulta Especifica: Registro existe...'
            r2=raw_input('1 Modificar 2 Borrar 3 continuar')
            if r2=="1":
                mnombres=leernombres()
                p.execute('update usuario set nombres="'+mnombres+'" where ci="'+mci"')
            if r2=="2":
                r3=raw_input('Esta seguro de eliminar s/n:')
                if r3=="s":
                    p.execute('delete from usuario where ci="'+mci"')
                    print 'Registro borrado...'
                else:
                    print 'Eliminación abortada'
                    existe=1
            else:
                g=p.fetchone()
        if existe==0:
            print 'Registro nuevo...'
            r3=raw_input('Desea incluir s/n?')
            if r3=="s":
                mnombres=leernombres()
                mci=""
                p.execute('INSERT INTO usuario values('+"'"+mci+"','"+mnombres+"')')
                print 'Inclusión realizada'
            else:
                print 'Inclusión no realizada...'
            r=raw_input('Desea procesar otro registro s/n:')
    p.close()
```

- Posée un punto de entrada y uno de salida
- El código se ejecuta de arriba a abajo
- Equivalente a un monolito

# Programación modular

```
#include <stdio.h>
#include <iostream.h>
#include <conio.h>
void sumar(int n1,int n2){
int resultado;
resultado = n1 + n2;
cout<<"La suma es igual a: "<<resultado;
}
void restar(int n1,int n2){
int resultado;
resultado = n1 - n2;
cout<<"La resta es igual a: "<<resultado;
}
void multiplicacion(int n1,int n2){
int resultado;
resultado = n1 * n2;
cout<<"La multiplicacion es igual a: "<<resultado;
}
void division(int n1,int n2){
int resultado;
resultado = n1 / n2;
cout<<"La division es igual a: "<<resultado;
}
void main(){
int registro,numero1,numero2,condicion=0;
while(condicion < 1){
cout<<"0.- Sumar";
cout<<"\n1.- Restar";
cout<<"\n2.- Multiplicar";
cout<<"\n3.- Dividir";
cout<<"\nN.- Cualquier numero para salir";
```

- Se basa en dividir un problema grande en problemas más chicos divididos en funciones. Las funciones se reutilizan o se llaman cuando son necesarias.

# Programación orientada a objetos

- Se basa en el concepto de objetos, que pueden ser tipos de datos con “atributos” y código en forma de procedimientos llamados métodos.

```
class User < ApplicationRecord
  extend Enumerize
  include Predicable
  include Abilitable
  include Roleable

  # Include default devise modules. Others available are:
  # :confirmable, :lockable, :timeoutable and :omniauthable
  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :trackable, :async

  ransack_alias :name, :first_name_or_last_name

  build_predicates :type, options: %i[customer inspection_company reloolog

  has_many :orderings, dependent: :destroy
  has_many :inspection_orders, through: :orderings
  has_many :flaggings, foreign_key: 'flagger_id', dependent: :destroy
  has_many :flagged_inspections, through: :flaggings
  has_many :followed_inspections, class_name: 'Inspection',
                                foreign_key: 'follower_id',
                                dependent: :destroy
  has_one_attached :background_check

  validates :email, presence: true, email: true, uniqueness: { case_sensit

  delegate :name, to: :company, prefix: true, allow_nil: true

  before_validation -> { email&.downcase }

  # Filters by user name
  scope :search_by_name, ->(name) {
    ransack(name_cont: name).result
  }

  # Constructs the name with both first_name and last name
  def name
    @name ||= "#{first_name} #{last_name}"
  end
end
```

**Sobre python**

# Guido van Rossum





# Guido van Rossum



# Sobre python



- Inventado en los ochentas, supuestamente en una navidad, en un par de días
- Lenguaje interpretado, orientado a objetos
- Llamado así por Monty Python
- `Import this`

# ¿por qué python?

- Es excelente para comenzar a programar por su simplicidad y legibilidad
- Está entre los lenguajes más demandados en el mercado laboral actual
- Por sus librerías disponibles, es el lenguaje mejor equipado para hacer ciencia de datos y machine learning