The background is a dark navy blue. It features several overlapping, semi-transparent geometric shapes in various colors: bright green, cyan, magenta, orange, and light blue. These shapes are arranged in a way that creates a sense of depth and movement, with some appearing to be layered on top of others. The overall aesthetic is modern and tech-oriented.

# PHP

## Programación Orientada a Objetos

# ¿Qué es POO?

La Programación Orientada a Objetos (POO) es un paradigma de programación que busca que nuestra forma de programar sea más cercana a la forma como nos relacionamos en nuestro día a día.

Los objetos manipulan los datos para la obtención de datos de salida específicos, donde cada objeto ofrece una funcionalidad especial.

The background is a dark navy blue. In the top-left and bottom-left corners, there are overlapping, semi-transparent geometric shapes in shades of cyan, blue, orange, and pink. In the top-right and bottom-right corners, there are similar overlapping shapes in shades of cyan, green, blue, pink, and orange. The shapes are arranged in a way that they appear to be floating or layered, creating a modern, abstract aesthetic.

# 1. Clases

## Definición

Las clases nos permiten agrupar información y acciones que podemos ejecutar con dicha información.

Es un término genérico que, como su nombre lo indica, representa una clase, tipo o conjunto

## Veamos un ejemplo

- > Tenemos la siguiente información: nombre, apellido y fecha de nacimiento.
- > Requerimos de las siguientes acciones:
  - > Mostrar el nombre completo de la persona
  - > Mostrar la edad de la persona

```
<?php
// Como nosotros ya somos expertos en programación estructurada
$nombr3s = "Juan Jose";
$apellido_paterno = "Perez";
$apellido_materno = "Ruiz";
$fecha_nacimiento = "1980-01-20";

// nombre completo
function nombrecompleto($nombr3s, $apellido_paterno, $apellido_materno)
{
    return "$apellido_paterno $apellido_materno $nombr3s";
}

// obtener edad
function edad($fecha_nacimiento)
{
    $edad = 0;
    // cálculo de la edad
    return $edad;
}
```

# Paso 01: Definir nuestra clase

```
<?php  
class Persona{  
  
}
```

## Paso 02: Definir atributos

alcance/scope \$nombre\_atributo = valor;

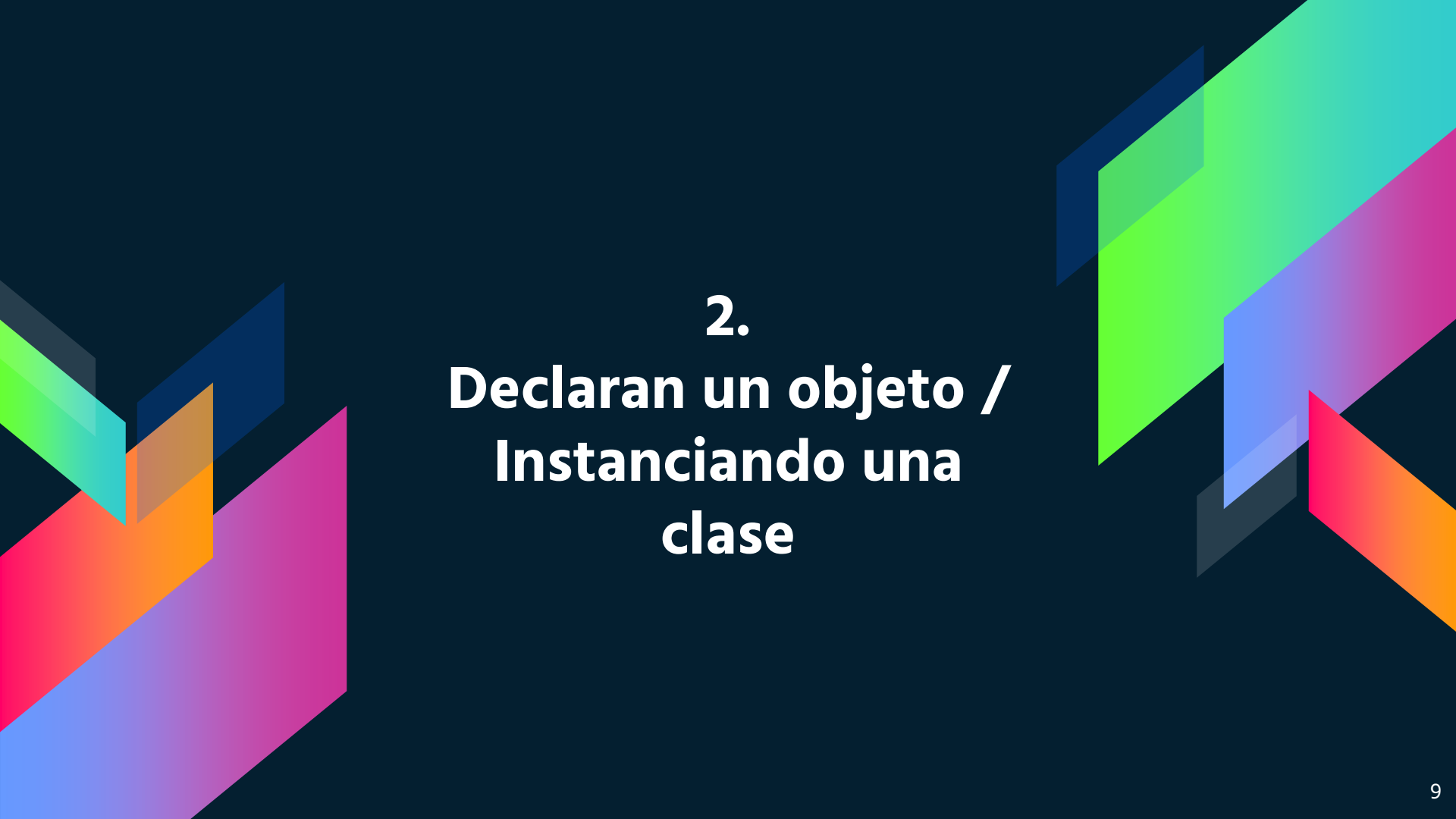
```
<?php
class Persona
{
    public $nombres = "Juan Jose";
    public $apellido_paterno = "Perez";
    public $apellido_materno = "Ruiz";
    private $fecha_nacimiento = "1980-01-20";
}
```

## Paso 03: Definir métodos


```
alcance function nombre_funcion($parametros...){  
}
```

```
<?php  
class Persona  
{  
    ...  
    public function nombrecompleto()  
    {  
        return "$this->apellido_paterno $this->apellido_materno $this->nombres";  
    }  
  
    public function edad()  
    {  
        $edad = 0;  
        // cálculo de la edad usando $this->fecha_nacimiento  
        return $edad;  
    }  
}
```




The background is a dark navy blue. It features several overlapping, semi-transparent geometric shapes in various colors: bright green, cyan, magenta, orange, and blue. These shapes are arranged in a way that creates a sense of depth and movement, with some appearing to be layered on top of others.

## 2. Declaran un objeto / Instanciando una clase

A series of overlapping, semi-transparent geometric shapes in shades of blue, green, and yellow, arranged in a diagonal pattern in the top right corner.

**Mientras que la clase representa el concepto genérico, un objeto es la representación específica de una clase. Cuando tenemos una clase podemos declarar uno o más objetos de dicha clase de esta forma:**

A series of overlapping, semi-transparent geometric shapes in shades of blue, green, and yellow, arranged in a diagonal pattern in the bottom left corner.

# Sintaxis


```
$variable_objeto = new Claseainstanciar();
```

```
<?php
```

```
$juan = new Persona();
```

```
$pedro = new Persona();
```

```
...
```





### **3. Accediendo a atributos y métodos**

# Sintaxis

```
<?php  
$juan = new Persona();  
echo $juan->nombres;  
echo $juan->nombrecompleto();  
$pedro = new Persona();
```



## 4. **Método Constructor** **(métodos mágicos)**



**PHP permite declarar métodos constructores para las clases. Aquellas que tengan un método constructor lo invocarán en cada nuevo objeto creado, lo que lo hace idóneo para cualquier inicialización que el objeto pueda necesitar antes de ser usado.**

# Sintaxis

```
<?php
```

```
class Persona
```

```
{
```

```
...
```

```
public function __construct($nom, $ape_pat, $ape_mat, $fecha_nac)
```

```
{
```

```
}
```

```
...
```

```
}
```



# Instanciar clase

```
<?php
```

```
$maria = new Persona("Maria", "Espinoza", "Sanchez", "1980-01-20");  
echo $maria->nombres;  
echo $maria->nombrecompleto();
```

# Ejercicio

## Ejercicio 01:

El array de trabajos de nuestro proyecto convertirlo a un array de objetos, los cuales serán instancias de una clase llamada Trabajo();

```
<?php
$php = new Trabajo(...);
$python = new Trabajo(...);
$devops = new Trabajo(...);
$node = new Trabajo(...);
$vue = new Trabajo(...);
$trabajos = [
    $php,
    $python,
    $devops,
    $node,
    $vue,
];
```