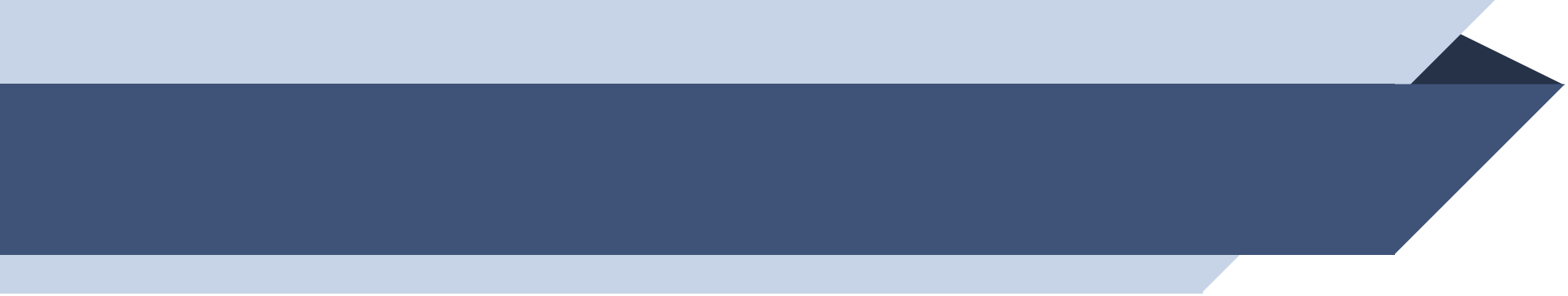


Programación Orientada a Objetos

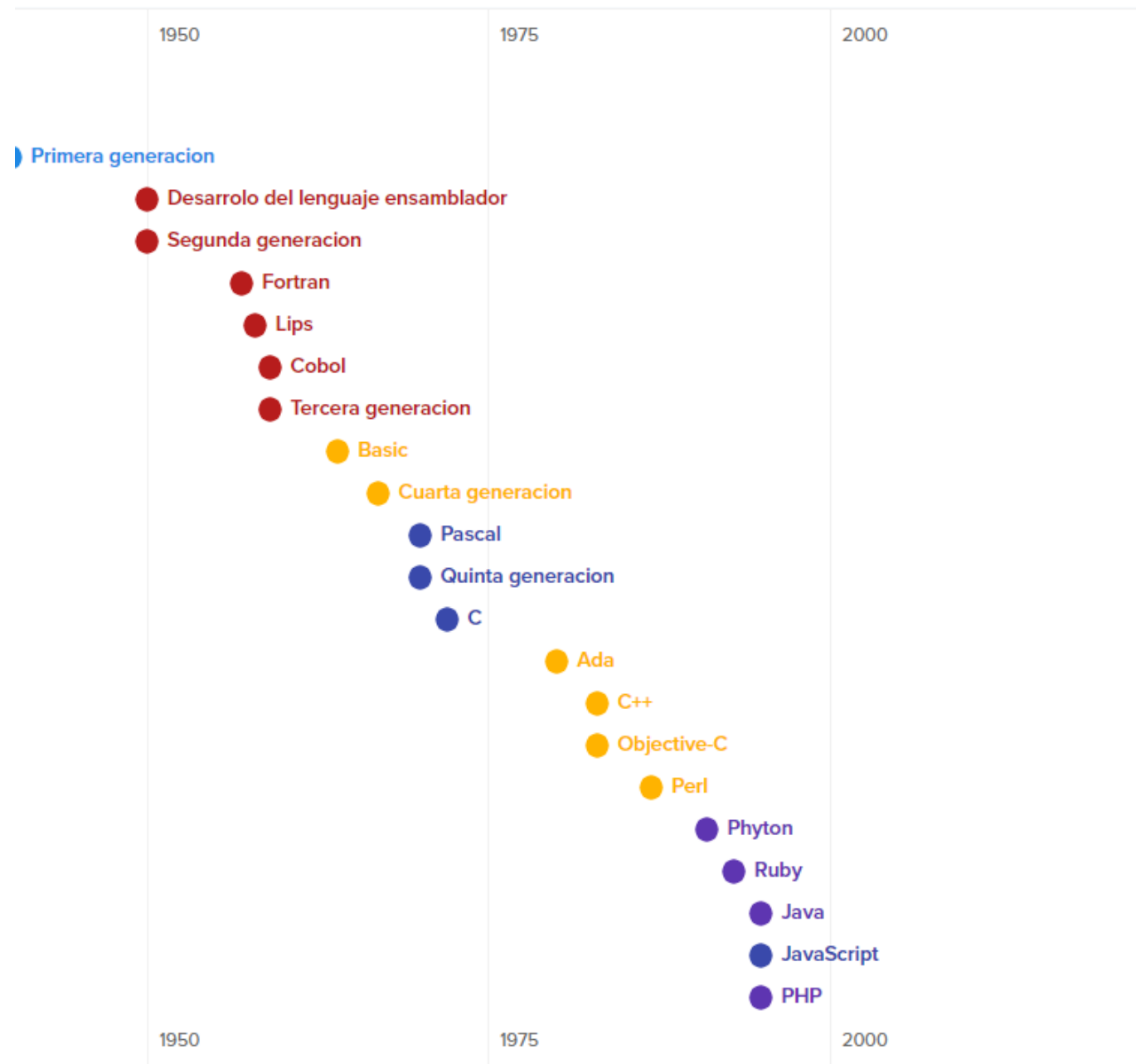
Dra. Lourdes Martínez Villaseñor



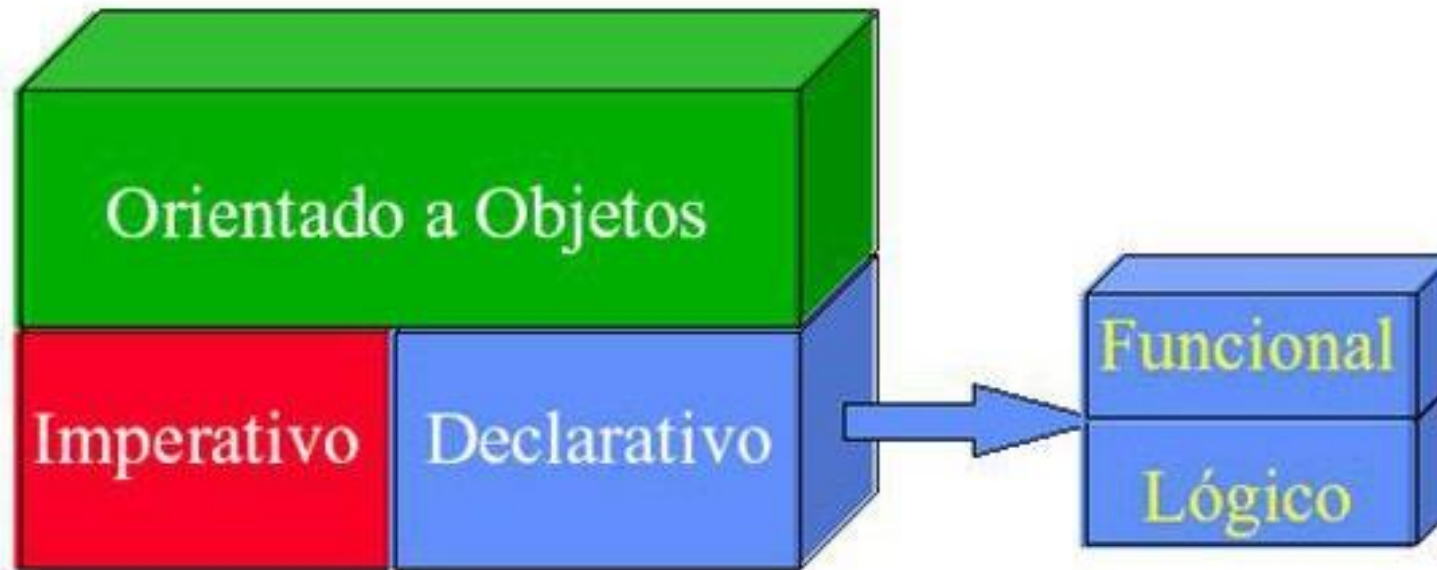
“Much of my work has come from being lazy. I didn't like writing programs, and so, when I was working on the IBM 701, writing programs for computing missile trajectories, I started work on a programming system to make it easier to write programs.”

John Bacus

Evolución de la programación



Paradigmas de programación



Lenguajes representativos

Lógico

Imperativo

```
type
  tDimension = 1..100;
  eMatriz(f,c: tDimension) = array [1..f,1..c] of real;

  tRango = record
    f,c: tDimension value 1;
  end;

  tpMatriz = ^eMatriz;

procedure EscribirMatriz(var m: tpMatriz);
var filas,col : integer;
begin
  for filas := 1 to m^f do begin
    for col := 1 to m^c do
      write(m^[filas,col]:7:2);
      writeln(resultado);
      writeln(resultado)
    end;
  end;
end;
```

Pascal, Basic, C

Funcional

```
(define (factorial x)
  (if (= x 0)
      1
      (* x (factorial (- x 1)))))

(factorial 8)
40320
(factorial 30)
265252859812191058636308480000000
```

Scheme o Haskell, Lisp, Scala

```
padrede('juan', 'maria'). % juan es padre de maria
padrede('pablo', 'juan'). % pablo es padre de juan
padrede('pablo', 'marcela').
padrede('carlos', 'debora').

hijode(A,B) :- padrede(B,A).
abuelode(A,B) :- padrede(A,C), padrede(C,B).
hermanode(A,B) :- padrede(C,A), padrede(C,B), A \== B.

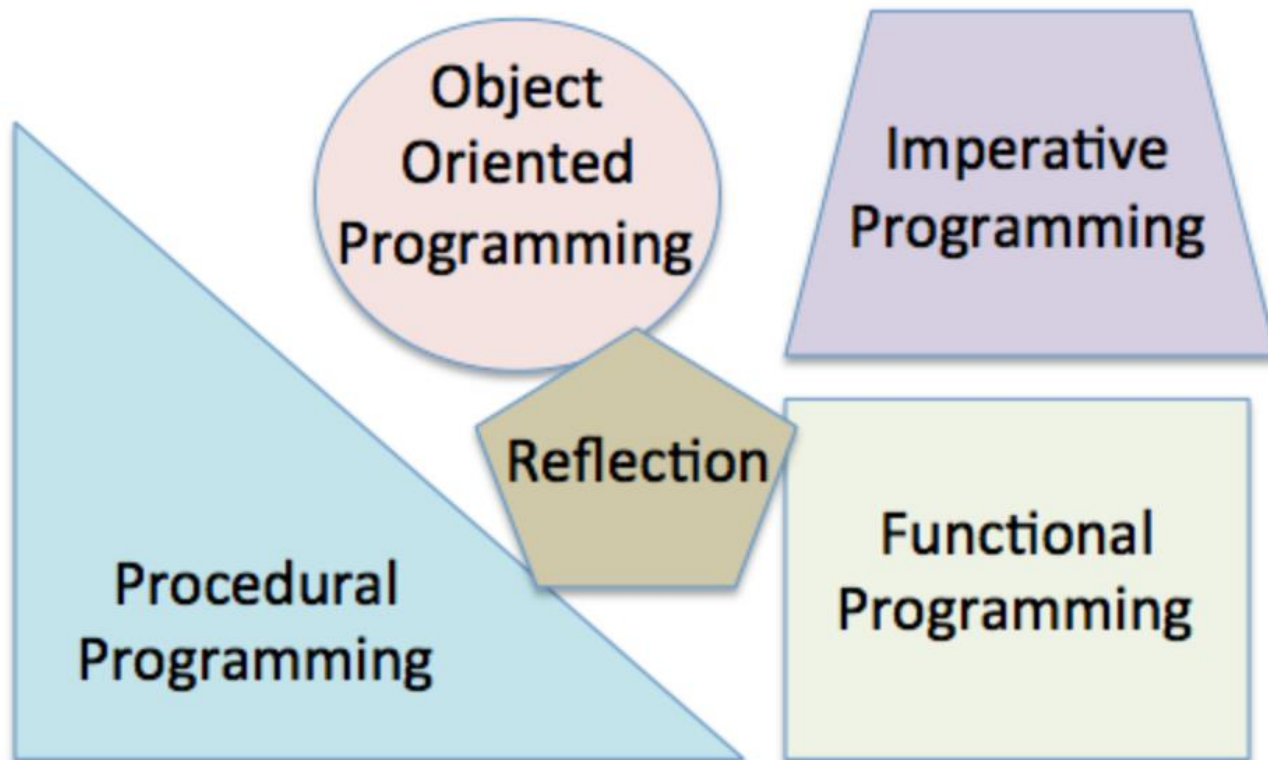
familiarde(A,B) :- padrede(A,B).
familiarde(A,B) :- hijode(A,B).
familiarde(A,B) :- hermanode(A,B).

?- hermanode('juan', 'marcela').
yes
?- hermanode('carlos', 'juan').
no
?- abuelode('pablo', 'maria').
yes
?- abuelode('maria', 'pablo').
no
```

Prolog, Lisp, Mercury

Paradigmas soportados por Python

Programming paradigms supported by Python



Ventajas de la Programación Orientada a Objetos (POO)

- Es fácil reusar código
- Es fácil dar mantenimiento a objetos
- Las características de POO como herencia y composición son útiles para:
 - ▷ Personalización
 - ▷ Sobrecarga de operadores

Programación Orientada a Objetos

Conceptos

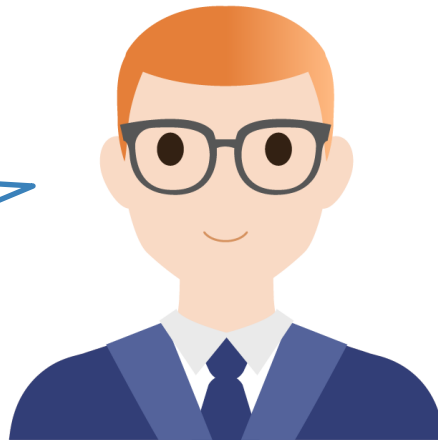
- Clase
- Herencia
- Objeto
- Método
- Evento
- Mensaje
- Atributo
- Estado interno
- Componentes de un objeto
- Representación de un objeto

Características

- Abstracción
- Encapsulamiento
- Principio de ocultación
- Polimorfismo
- Herencia
- Recolección de basura

Conceptos Orientación a Objetos

Objeto



Atributos:

- Nombre
- Género
- saldo
- Hambriento

Métodos:

- Caminar
- Depositar
- Comer
- Estudiar

Conceptos Orientación a Objetos

Clase



Objetos

Para usar clases y objetos

■ Declarar clases

```
#class nombre de la clase :  
  
class Molde:  
    // Campos, propiedades, métodos y eventos
```

■ Crear objetos

```
# nombre del objeto = nombre de la clase ()  
objeto=Molde()
```

Las clases permiten crear estructuras definidas por el usuario que permiten contener información y comportamiento de una entidad.

El objeto es una instancia de una clase. Es una colección de datos (variables) y métodos (funciones) que actúan sobre esos datos. La clase es la plantilla con la que se crea el objeto.

Para usar clases y objetos

```
In [10]: #class crea un objeto clase y le asign un nombre
class Molde:
    'Clase de ejemplo'
    forma='' #las asignaciones dentro de la clase crean atributos
    tam=0
    def imprime(self):
        print('mi primera clase')

objeto=Molde() # crea una instancia de la clase Molde ... un objeto
objeto.imprime()

Molde.__doc__ #nos da la cadena de documentación de esa clase

mi primera clase
```

```
Out[10]: 'Clase de ejemplo'
```

Para usar clases y objetos

```
In [5]: #class crea un objeto clase y le asigna un nombre
class Molde:
    'Clase de ejemplo'
    forma=''
    tam=0
    def imprime(self): # self referencia el objeto instancia que está siendo procesado
        print('forma ',self.forma)
        print('tamaño ',self.tam)

objeto=Molde()
objeto.forma='cuadrado' #los atributos se accesan nombreObjeto.nombreAtributo
objeto.tam=5
objeto.imprime()

forma  cuadrado
tamaño  5
```

Para usar constructor

Inicializador

Es una función que es llamada cada vez que un nuevo objeto de una clase es instanciado

```
In [1]: class Cuenta:
        def __init__(self,s,n):      #inicializador
            self.saldo = s
            self.nombre = n
            self.tipo = 'ahorro'
        def impuesto(self):
            if self.saldo < 5000:
                return 0
            else:
                return self.saldo * .3

ctaJuan=Cuenta(10000,'Juan Perez')
print('Cuenta de ',ctaJuan.nombre)
print("Impuesto a pagar",ctaJuan.impuesto())
```

```
Cuenta de Juan Perez
Impuesto a pagar 3000.0
```

Ejercicios

- Describe con tus propias palabras que es :
 - ▷ Clase
 - ▷ Objeto
 - ▷ Constructor
 - ▷ Atributo
 - ▷ Método
- Escribe una clase de una entidad de tu preferencia (coche, figura, animal etc.) que contenga campos para tres datos, un constructor para la clase y un método que despliegue la información.