Algorítmica y Programación

Enero - Mayo 2020

Dr. Iván S. Razo Zapata
(ivan.razo@itam.mx)

ITAM

Programación Orientada a Objetos

Anuncios parroquiales

- Tercer parcial
  - 18 de Mayo
- Examen final
  - 27 de Mayo - 19:00hrs

Recap

- Programación estructurada
  - Ciclos de control
    - If …else
    - For
    - While
  - Funciones

ITAM

¿Qué es la programación orientada a objetos?

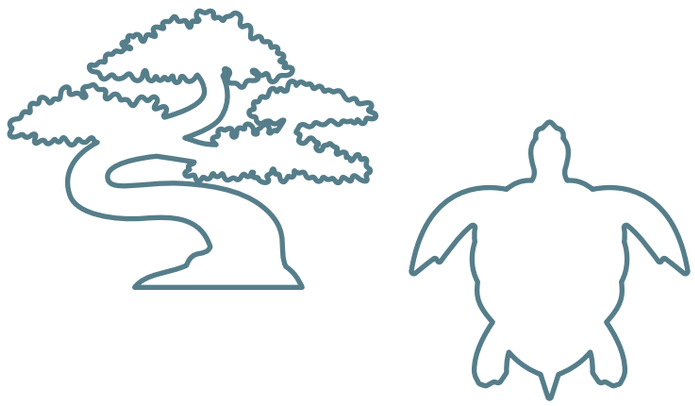
- "es un paradigma de programación en el que los **conceptos** del mundo real **relevantes** para nuestro problema se **modelan** a través de **clases** y **objetos**"

—Python para todos, Raúl González Duque

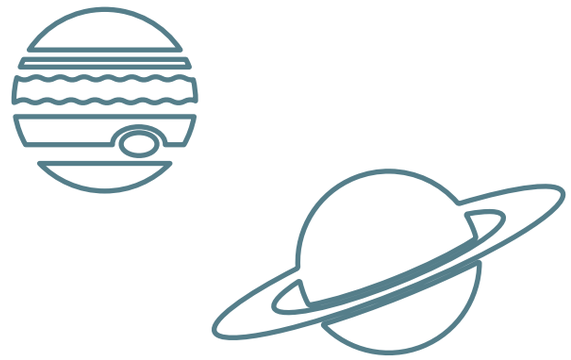
Figuras geométricas

Cuadrado **Circulo**

Ser vivo



Planetas



Conceptos básicos

- **Clase**
  - Una **plantilla genérica** a partir de la cuál se pueden **instanciar** los objetos
  - Define los atributos y métodos predeterminados que tendrán los objetos
- **Objeto**
  - **Instancia** de una clase
  - Corresponden con los objetos reales del mundo que nos rodea, o con objetos internos del programa (del sistema)

Programación orientada a objetos

- Clases definen los atributos y métodos predeterminados que tendrán los objetos

Atributos

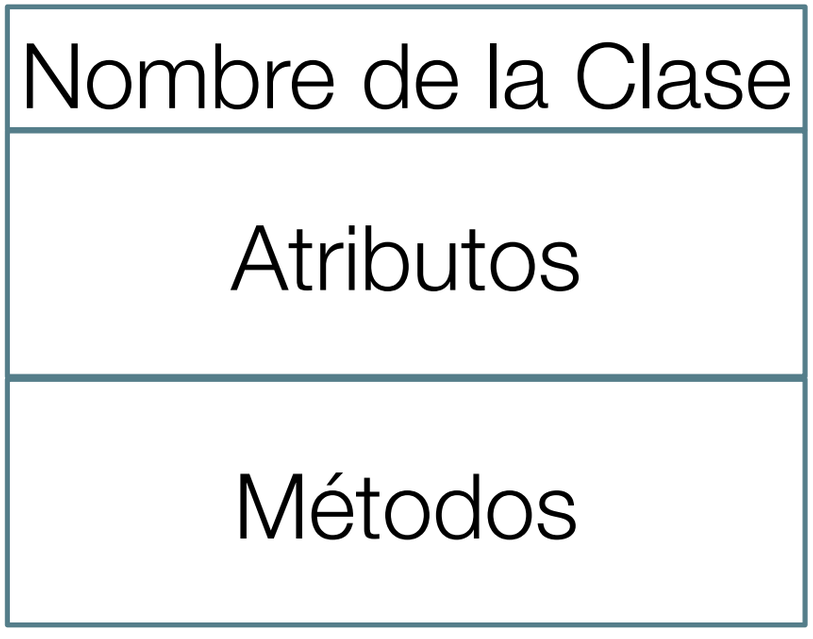
- Base
- Altura

Métodos

- Calcular área
- Calcular perímetro

Programación orientada a objetos

- Clases definen los atributos y métodos predeterminados que tendrán los objetos



Notación **diagramas de clase** UML

UML - Unified Modeling Language

Modelado - a.k.a. definiendo atributos y métodos

- Abstracción de la “realidad”
- Enfoca en describir los elementos **relevantes** para la solución de un problema

What keeps CEOs awake at night

Modelado

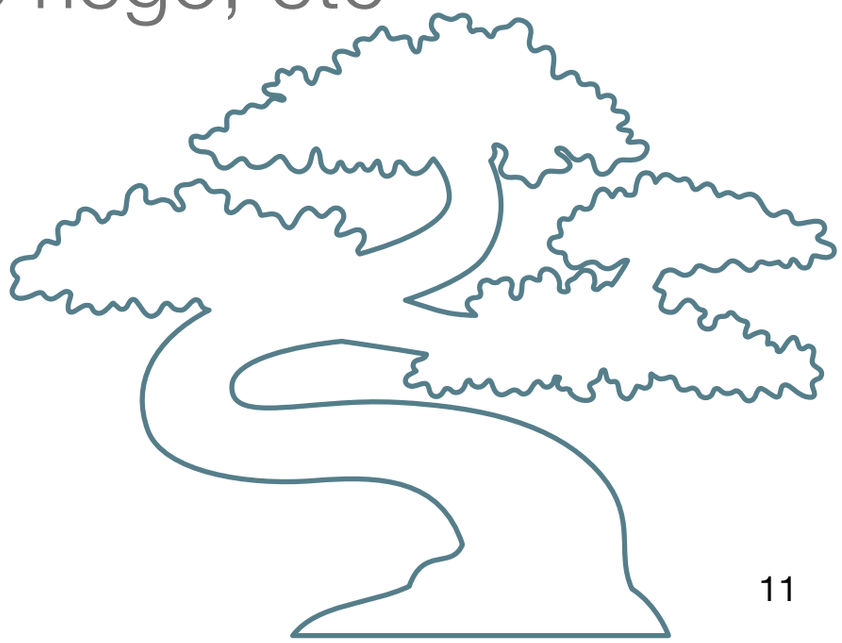
- Ejemplo 1: Eres el CEO de una organización dedicada a monitorear la salud de los arboles en una ciudad. ¿Qué aspectos de los arboles son relevantes para el monitoreo?

- Ejemplo 2: Eres el CEO de una organización dedicada a compra y venta de arboles. ¿Qué aspectos de los arboles son relevantes para la compra y venta?

Modelado

- Ejemplo 1: Eres el CEO de una empresa dedicada a monitorear la salud de las zonas verdes en una ciudad. ¿Qué aspectos de los arboles son relevantes para el monitoreo?

- Familia, altura, año en que fue sembrado, presenta plaga, ubicación (coordenadas geográficas), ubicación en lenguaje natural (parque), frecuencia de riego, etc



Modelado

- Ejemplo 2: Eres el CEO de una empresa dedicada a compra y venta de arboles

- Familia, altura, costo de producción, precio mayorista, precio minorista

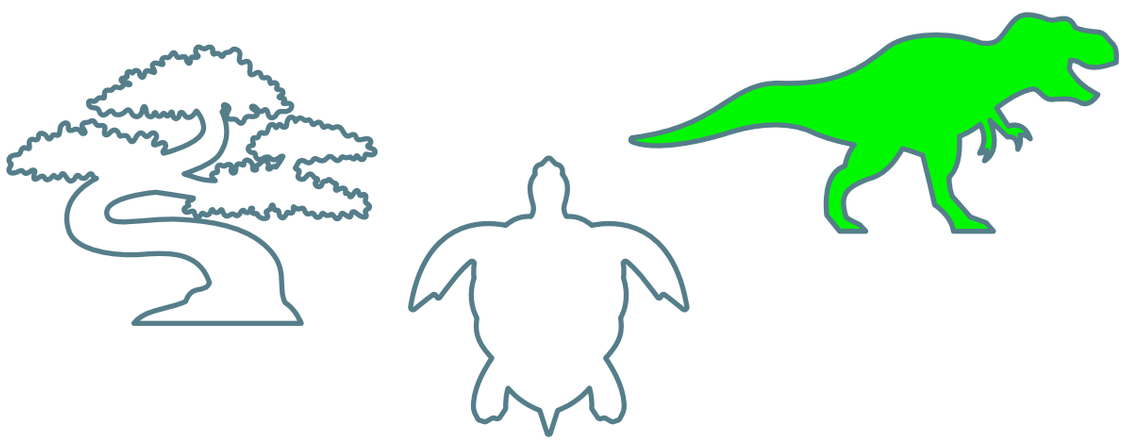
¿Cuál modelo es el correcto?

Modelado

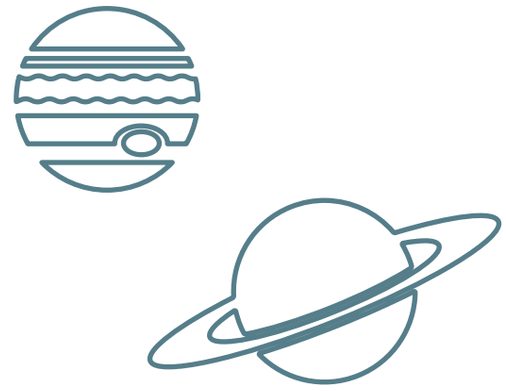
- "... all models are approximations. Essentially, **all models are wrong, but some are useful**. However, the approximate nature of the model must always be borne in mind...." George P. Box

- Usually not written in stone!

Ser vivo



Planetas



Pluton?

Primer clase

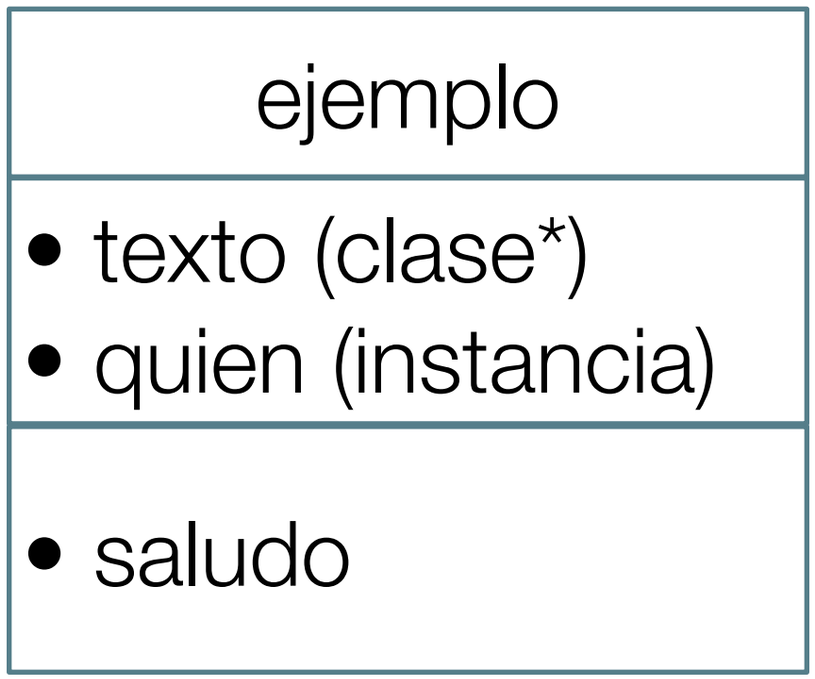
```
class ejemplo:
    texto = "Hola"

    def saludo(self, nuevoTexto):
        self.quien = nuevoTexto
        return self.texto + " " + self.quien
```

- Clase ejemplo
- Atributos:
  - texto (atributo a nivel clase)
  - quien (atributo a nivel objeto/instancia)
- Método
  - saludo

- Atributo a nivel clase
  - Valor compartido por todos los objetos/instancias
- Atributo a nivel objeto/instancia
  - Valor único por objeto

Primer clase

```
class ejemplo:
    texto = "Hola"

    def saludo(self, nuevoTexto):
        self.quien = nuevoTexto
        return self.texto + " " + self.quien
```



Primer clase

```
class ejemplo:
    texto = "Hola"

    def saludo(self, nuevoTexto):
        self.quien = nuevoTexto
        return self.texto + " " + self.quien

objeto1 = ejemplo()
print(objeto1.saludo("Mundo"))

objeto2 = ejemplo()
print(objeto2.saludo("Alice"))

objeto3 = ejemplo()
print(objeto3.saludo("Bob"))
```

ITAM

Primer clase

```
In [8]: objeto1?
Type:           ejemplo
String form: <__main__.ejemplo object at 0x11fd6fac8>
Docstring:   <no docstring>

In [9]: objeto2?
Type:           ejemplo
String form: <__main__.ejemplo object at 0x11fd6f710>
Docstring:   <no docstring>

In [10]: objeto3?
Type:           ejemplo
String form: <__main__.ejemplo object at 0x11fd6f630>
Docstring:   <no docstring>
```

ITAM

Primer clase

```
In [11]: objeto1.texto
Out[11]: 'Hola'

In [12]: objeto1.quien
Out[12]: 'Mundo'

In [13]: objeto1.saludo()
Traceback (most recent call last):

  File "<ipython-input-13-0fd3ae849b4e>", line 1, in <module>
    objeto1.saludo()

TypeError: saludo() missing 1 required positional argument: 'nuevoTexto'
```

ITAM

Primer clase

```
In [14]: objeto1.texto
Out[14]: 'Hola'

In [15]: objeto1.quien
Out[15]: 'Mundo'

In [16]: objeto2.quien
Out[16]: 'Alice'

In [17]: objeto3.quien
Out[17]: 'Bob'

In [18]: objeto3.saludo("Tom")
Out[18]: 'Hola Tom'

In [19]: objeto3.quien
Out[19]: 'Tom'
```

Segunda clase

Cuerpo celeste	Diámetro ecuatorial	Masa	Radio orbital (promedio, UA).	Periodo orbital (años).	Periodo de rotación (días).
Sol	109	332 950	0	0	25-35
Mercurio	0,382	0,06	0,38	0,241	58,6
Venus	0,949	0,82	0,72	0,615	-243[1]
Tierra	1,00	1,00	1,00	1,00	1,00
Marte	0,53	0,11	1,52	1,88	1,03
Júpiter	11,2	318	5,20	11,86	0,414
Saturno	9,41	95	9,54	29,46	0,426
Urano	3,98	14,6	19,22	84,01	0,718
Neptuno	3,81	17,2	30,06	164,79	0,671

Segunda clase

Segunda clase

```
class planeta:
    # Atributos de clase
    ubicacion   = "Sistema Solar"
    diametroSol = 109
    masaSol     = 332950

    # Constructor
    def __init__(self, texto,valor1,valor2,valor3):
        # Atributos de objeto
        self.nombre   = texto
        self.diametro = valor1
        self.masa     = valor2
        self.radio    = valor3

    def regresaUbicacion(self):
        return self.ubicacion

    def regresaNombre(self):
        return self.nombre

    def relacionSolDiametro(self):
        return self.diametro / type(self).diametroSol

    def relacionSolMasa(self):
        return self.masa / type(self).masaSol

tierra   = planeta("Tierra",1.0,1.0,1.0)
mercurio = planeta("Mercurio",0.382,0.06,0.38)
venus    = planeta("Venus",0.949,0.82,0.72)
```

Accediendo a atributos objeto

Accediendo a atributos de clase y objeto

Segunda clase

```
print("=== Planetas ===")
print(tierra.regresaNombre())
print(tierra.regresaUbicacion())
print(tierra.relacionSolDiametro())
print(tierra.relacionSolMasa())

print(" ")

print(mercurio.regresaNombre())
print(mercurio.regresaUbicacion())
print(mercurio.relacionSolDiametro())
print(mercurio.relacionSolMasa())

print(" ")

print(venus.regresaNombre())
print(venus.regresaUbicacion())
print(venus.relacionSolDiametro())
print(venus.relacionSolMasa())
```

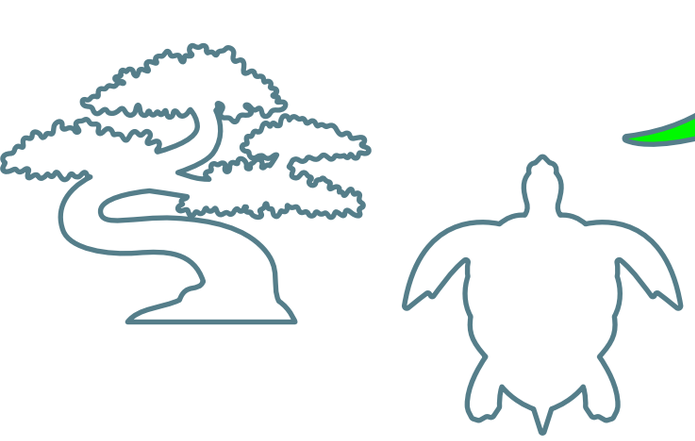
Segunda clase

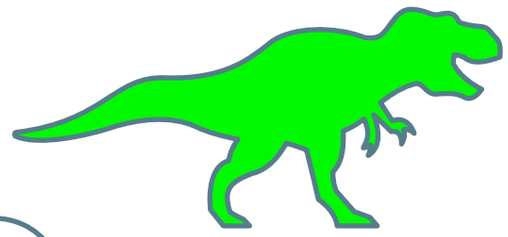
```
In [28]: planeta.diametroSol
Out[28]: 109

In [29]: planeta.diametroSol = 115

In [30]: venus.relacionSolDiametro()
Out[30]: 0.008252173913043477

In [31]: tierra.relacionSolDiametro()
Out[31]: 0.008695652173913044

In [32]: mercurio.relacionSolDiametro()
Out[32]: 0.003321739130434782 5
```

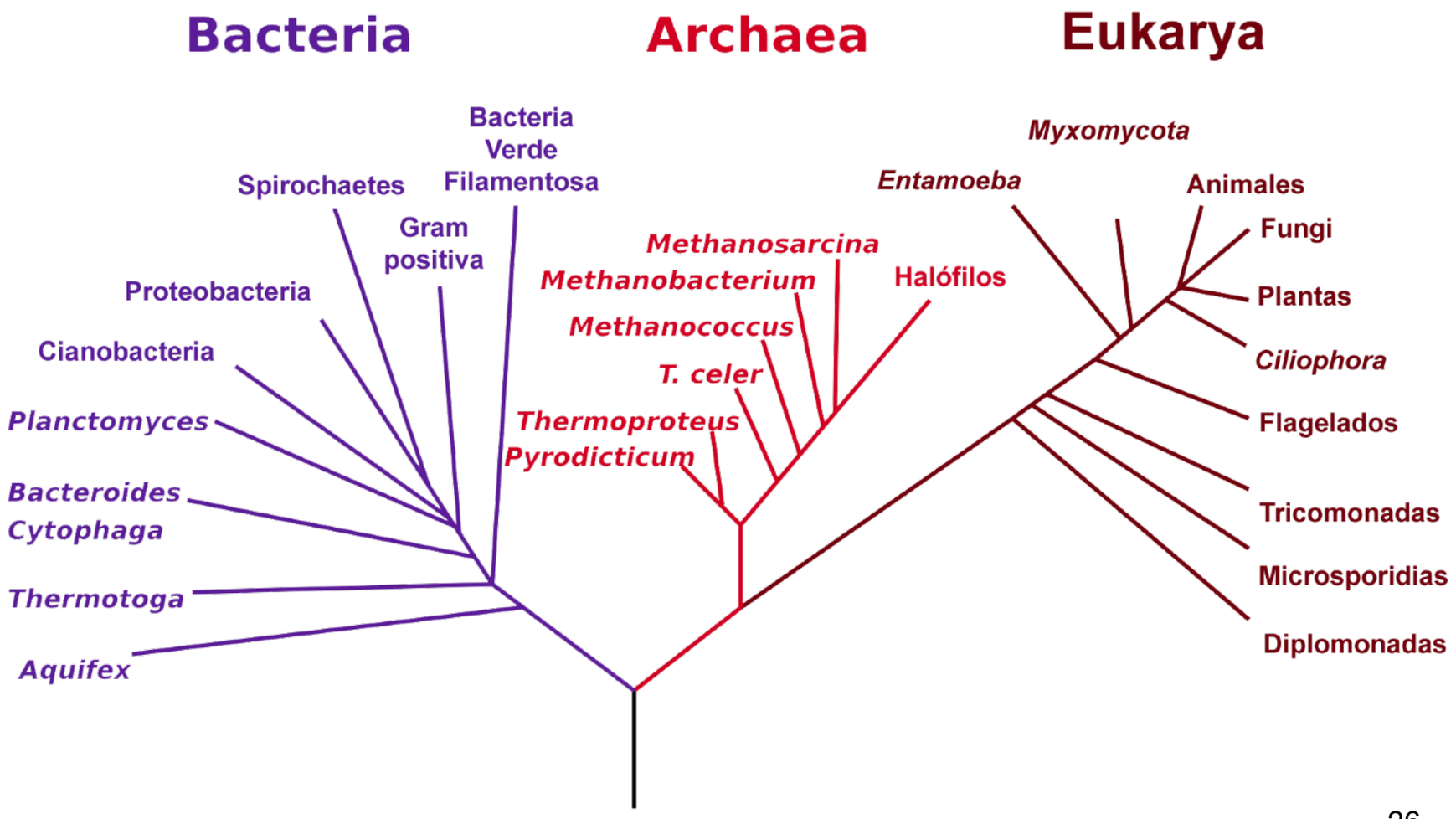
Cambiando valor de atributo de Clase

Representaciones más elaboradas

Ser vivo







Conceptos para la siguiente clase

- Herencia
- Polimorfismo
- Abstracción
- Encapsulamiento

Conceptos para la siguiente clase

- Constructores
- Destructores
- Clase abstracta