

Programación Avanzada

IIC2233 2023-2

Hernán Valdivieso - Daniela Concha - Francisca Ibarra - Joaquín Tagle - Francisca Cattán



Anuncios

Jueves 17 de Agosto 2023

1. Hoy tenemos la primera experiencia.
 2. Recuerden que la modalidad *flipped classroom* implica llegar con el material de la semana estudiado.
 3. Nuevas cápsulas de instalación en <https://www.youtube.com/playlist?list=PLQI-jYMgXtz-siHH3SpdiHBApkQ7z1Q54>
 4. Si no tienen computador, contacten a la coordinación para evaluar alternativa.
-

Anuncios

Jueves 17 de Agosto 2023

Issues del GitHub

1. Lean y cumplan las reglas.
2. No cierren las issues.
3. Usen títulos descriptivos

OOP

- Paradigma de programación
- Interacción entre objetos

Programación Orientada a Objetos

Objeto: Colección de datos que además tiene comportamientos asociados
¿Cómo los representamos en Python? ¡Clases!

Programación Orientada a Objetos

Objeto: Colección de datos que además tiene comportamientos asociados

¿Cómo los representamos en Python? ¡Clases!

```
class Planta:
    def __init__(self, no, res):
        self.nombre = no      # Zapallo
        self.agua = 0
        self.resistencia = res # 50 unidades

    def __str__(self):
        return f'{self.nombre} - {self.agua}/{self.resistencia}'
        # Zapallo - 0/50

    def regar(self, cantidad):
        self.agua += cantidad
        if self.agua >= self.resistencia:
            self.agua = self.resistencia
```

Programación Orientada a Objetos

Objeto: Colección de datos que además tiene comportamientos asociados
¿Cómo los representamos en Python? ¡Clases!



Clase



Instancia

Programación Orientada a Objetos

```
class Planta:
    def __init__(self, no, res):
        self.nombre = no      # Zapallo
        self.agua = 0
        self.resistencia = res # 50 unidades

    def __str__(self):
        return f'{self.nombre} - {self.agua}/{self.resistencia}'
        # Zapallo - 0/50

    def regar(self, cantidad):
        self.agua += cantidad
        if self.agua >= self.resistencia:
            self.agua = self.resistencia
```



Programación Orientada a Objetos

Interacción entre Objetos: Podemos utilizar los métodos y propiedades de un objeto en otro.

```
class Huerto:
    def __init__(self):
        self.plantas = []

    def plantar(self, planta):
        if planta not in self.plantas:
            planta.regar(5)
            self.plantas.append(planta)
```



Properties

- Encapsular atributos del objeto
- Manejar el acceso o modificación de uno o varios atributos

Properties, ¿Cómo funcionan?

```
class Planta:
    def __init__(self, nombre, resistencia):
        self._nombre = nombre
        self.agua = 0
        self._resistencia = resistencia
        self._calidad = 'bueno'

    @property
    def calidad(self):
        return self._calidad

    @calidad.setter
    def calidad(self, nueva_calidad):
        self._calidad = nueva_calidad
        print(f'Parece que ahora soy un {self._nombre} {self._calidad}.')

p = Planta('Zapallo', 50)
p.calidad = 'muy bueno'
```

Parece que ahora soy un Zapallo muy bueno.

Properties, ¿Cómo funcionan?

```
class Planta:
    def __init__(self, nombre, resistencia):
        self._nombre = nombre
        self.agua = 0
        self._resistencia = resistencia
        self._calidad = 'bueno'

    @property
    def nombre(self):
        return self._nombre

    @nombre.setter
    def nombre(self, nuevo_nombre):
        print(f'Soy un {self.nombre} y nunca seré un {nuevo_nombre}.')

p = Planta('Zapallo', 50)
p.nombre = 'Tomate'
```

Soy un Zapallo y nunca seré un Tomate.

Veamos Git y Github

Pasemos a la Experiencia 1

Programación Avanzada

II C2233 2023-2

Hernán Valdivieso - Daniela Concha - Francisca Ibarra - Joaquín Tagle - Francisca Cattán

