ADIMRA
ASOCIACIÓN DE INDUSTRIALES METALÚRGICOS DE LA REPÚBLICA ARGENTINA

Fundamentos de Programación

Slides de apoyo para clases sincrónicas
Paradigmas de Programación

Paradigmas de programación

Paradigma significa modelo o patrón.

Los paradigmas de programación, son esencialmente estilos normalizados para escritura de código, es decir, no estilos personales sino normalizados y documentados, aceptados internacionalmente.

La elección del tipo de paradigma a utilizar, dependerá netamente del tipo de proyecto, y el nivel de complejidad que involucre.

ADIMRA
ASOCIACIÓN DE INDUSTRIALES METALÚRGICOS DE LA REPÚBLICA ARGENTINA

Paradigmas

Por qué existen?

Los paradigmas han acompañando el desarrollo de los distintos lenguajes a través de las décadas, permitiendo cubrir distintos niveles de complejidad y exigencia en los códigos desarrollados.

Mucho **se resume en términos de orden y eficiencia de desarrollo y control de los algoritmos**, a medida que su complejidad crece, y la escritura lineal de instrucciones unas tras otras, simplemente no logra cubrir las necesidades.

ADIMRA
ASOCIACIÓN DE INDUSTRIALES METALÚRGICOS DE LA REPÚBLICA ARGENTINA

Paradigmas

Imperativos
Declarativos

En una primer categorización de estilos, podemos distinguir entre:

- Imperativos
- Declarativos.

Un estilo imperativo, es aquel en el que indicamos en todo momento al sistema lo que debe hacer, paso por paso detallado.

Un estilo declarativo en cambio, solicita indirectamente al sistema distintas acciones, sin detallar como se realizan, es decir, aprovecha llamadas a funciones, métodos y otras alternativas.

ADIMRA
ASOCIACIÓN DE INDUSTRIALES METALÚRGICOS DE LA REPÚBLICA ARGENTINA

Paradigmas

Imperativos
Declarativos

En la programación actual, es muy habitual el estilo declarativo, para simplificar la escritura de código.

El sistema en distintas capas -mediante el uso de librerías y otras ayudas- ejecutará por debajo lo necesario y retornará los resultados.

Ejemplo declarativo clásico, consulta SQL:

SELECT * FROM productos WHERE tipo = 3 (selecconar todos los items de la tabla productos, cuyo tipo sea 3).

ADIMRA
ASOCIACIÓN DE INDUSTRIALES METALÚRGICOS DE LA REPÚBLICA ARGENTINA

Paradigmas

Programación Secuencial o Estructurada

La PE es el estilo de programación con el cual inicia todo programador.

Si el proyecto es sencillo, por lo general se arma de manera “monolítica”, es decir, todas las funcionalidades en un único código, dentro de un único archivo, una instrucción debajo de otra, línea a línea.

Si bien se trata del paradigma más básico, su uso es perfectamente válido en la traducción de algoritmos simples, y el resultado es completamente usable.

Paradigmas

Programación Funcional

Definición

La PF "modula" el código en diferentes partes, utilizando declaraciones de "Funciones".

Una función es un bloque de código identificado bajo un nombre, por fuera del código principal, destinado a ejecutar una tarea específica. Puede o no retornar valores, según se desee.

Este encapsulamiento es muy útil para organizar el código y evitar repeticiones.

ADIMRA
ASOCIACIÓN DE INDUSTRIALES METALÚRGICOS DE LA REPÚBLICA ARGENTINA

Paradigmas

Programación Funcional

Definición

Otro aspecto del encapsulamiento, es que se extiende a la definición de variables. Toda variable definida dentro de una función, es local, y por ende solo disponible en el ambito de esa función.

Cada vez que sea necesario, podrá "llamarse" a la función desde el flujo de código principal. El sistema derivará temporalmente la ejecución hacia la función, procesará las instrucciones listadas en ella y retornará al flujo principal, para continuar con el comando inmediato posterior al llamado.

Paradigmas

Programación Funcional

En Python

En Python, una función puede declararse fácilmente mediante la palabra reservada def:

```
def nombreFuncion():
    primer instrucción
    segunda intrucción
    ...
    enésima instrucción
```

El llamado se efectúa mediante el nombre y paréntesis:

```
nombreFuncion()
```

ADIMRA
ASOCIACIÓN DE INDUSTRIALES METALÚRGICOS DE LA REPÚBLICA ARGENTINA

Paradigmas

Programación Funcional

Llamado simple de función

Encapsulando lo relativo a la impresión por consola en una función, podríamos utilizar:

```
ingreso = 0
resultado = 0

def mostrarCalculoSuperficie():
    resultado = ingreso * ingreso * 3.14
    print("***")
    print(resultado)

# Flujo principal de código
ingreso = int(input("Ingresar radio: ")
mostrarCalculoSuperficie()
```

Paradigmas

Programación Funcional

Llamado con argumentos

En lugar de utilizar variables previamente definidas, podemos pasar argumentos:

```
def mostrarCalculoSuperficie(radio):
    resultado = radio * radio * 3.14
    print("***")
    print(resultado)

# Flujo principal de código
ingreso = int(input("Ingresar radio: ")
mostrarCalculoSuperficie(ingreso)
```

El valor entrado por consola (ingreso) es "pasado" como argumento a la función, que internamente lo maneja bajo el nombre radio.

ADIMRA
ASOCIACIÓN DE INDUSTRIALES METALÚRGICOS DE LA REPÚBLICA ARGENTINA

Paradigmas

Programación Funcional

Llamado con argumentos y retorno de valor

Otra opción de organización, es retornar valores desde la función:

```
def mostrarCalculoSuperficie(radio):
    return radio * radio * 3.14

# Flujo principal de código
ingreso = int(input("Ingresar radio: ")
print(mostrarCalculoSuperficie(ingreso))
```

El valor entrado por consola (ingreso) es "pasado" como argumento a la función, que internamente lo procesa y retorna el resultado, el cual en este caso es pasado a su vez a la función print().

ADIMRA
ASOCIACIÓN DE INDUSTRIALES METALÚRGICOS DE LA REPÚBLICA ARGENTINA

Paradigmas

Programación Orientada a Objetos

Definición

Así como la PF utiliza como esencia el encapsulamiento en Funciones, la POO se basa en la definición de Clases. Una Clase puede verse como un molde que define las características de un objeto.

La POO trata de trasladar los conceptos de objetos cotidianos con los cuales estamos habituados a trabajar en el día a día, a objetos de código que permitan expresar de manera más ordenada y eficiente los distintos algoritmos.

ADIMRA
ASOCIACIÓN DE INDUSTRIALES METALÚRGICOS DE LA REPÚBLICA ARGENTINA

Paradigmas

Programación Orientada a Objetos

Definición

De esta manera, al definir una Clase, estamos definiendo una serie de propiedades (atributos) y acciones (métodos) que podrán ser ejecutados sobre ese tipo de objeto.

Cada vez que debamos utilizar un objeto con esas cualidades, crearemos una nueva **instancia**, utilizando como base (molde) la clase oportunamente definida.

A partir de allí, a través del nombre de ese objeto, podremos acceder a los distintos métodos disponibles en la Clase.

Paradigmas

Programación Orientada a Objetos

En Python

En Python, una clase se declara mediante la palabra reservada **class**, esta es una práctica muy standard en todos los lenguajes:

```
class ClaseVehiculo:
  pass
```

Como dijimos, todo lo que coloquemos dentro de la definición de la clase, estará disponible para cada objeto que se **instancie** utilizando como base esa Clase.

ADIMRA
ASOCIACIÓN DE INDUSTRIALES METALÚRGICOS DE LA REPÚBLICA ARGENTINA

Paradigmas

Programación Orientada a Objetos

En Python

En Python, una clase se declara mediante la palabra reservada **class** (práctica muy standard en todos los lenguajes):

```
class Persona:
  pass
```

Como dijimos, todo lo que coloquemos dentro de la definición de la clase, estará disponible para cada objeto que se **instancie** utilizando como base esa Clase.

ADIMRA
ASOCIACIÓN DE INDUSTRIALES METALÚRGICOS DE LA REPÚBLICA ARGENTINA

Paradigmas

Programación Orientada a Objetos

Ejemplo

```
class Persona:
  nombre = ""
  edad = 0
  estatura = 0
  genero = "M"

  def saludo(self):
    print(self.nombre + " dice hola!")
```

Clase Persona, 4 atributos (nombre, edad, estatura y genero), y un método (saludo).

Para instanciar un objeto de tipo Persona:

```
persona1 = Persona()
```

ADIMRA
ASOCIACIÓN DE INDUSTRIALES METALÚRGICOS DE LA REPÚBLICA ARGENTINA

Paradigmas

Programación Orientada a Objetos

Ejemplo

Una vez instanciado el objeto de tipo Persona, **podemos acceder a los atributos y métodos utilizando el nombre y la notación de puntos**, por ejemplo:

```
persona1.nombre = "Pepe"
persona1.saludo()
```

mostrará:

```
Pepe dice hola!
```

ADIMRA
ASOCIACIÓN DE INDUSTRIALES METALÚRGICOS DE LA REPÚBLICA ARGENTINA

Paradigmas

Programación Orientada a Objetos

Ejemplo

Cada vez que se llama a un método, el objeto en sí es pasado como parámetro, utilizando la palabra reservada **self**, por eso se observa ese argumento en saludo().

Si tenemos por ejemplo 2 objetos de tipo Persona (persona1 y persona2):

persona1.saludo() se traduce a:
Persona.saludo(persona1)

y persona2.saludo() a:
Persona.saludo(persona2)

Paradigmas

Programación Orientada a Objetos

Métodos especiales

Dos métodos especiales, propios de el diseño de Clases, son el **Constructor y el Destructor**.

Estos métodos son llamados de forma automática al instanciar y destruir un objeto, y por ende se utilizan para inicializar o liberar recursos según sea necesario.

La palabra reservada __init__, se utiliza para nombrar al Constructor, por lo tanto, cada vez que veamos esta palabra dentro de las declaraciones de una Clase, sabremos que lo contenido allí se ejecutará de manera automática al instanciar.

Paradigmas

Programación Orientada a Objetos

Métodos especiales

Podemos redefinir la clase Persona:

```
class Persona:
    nombre = ""
    edad = 0
    estatura = 0
    genero = "M"

    def __init__(self, nombre, genero):
        self.nombre = nombre
        self.genero = genero

    def saludo(self):
        print(self.nombre + " dice hola!")
```

Paradigmas

Programación Orientada a Objetos

Métodos especiales

Ahora en lugar de instanciar con persona1 = Persona(), utilizamos:
persona1 = Persona("Pepa", "F")

De esta manera, pasamos los argumentos **nombre** y **genero**, que serán usados por el método **__init__** para inicializar el objeto. Si omitimos enviar estos argumentos, el sistema generará un error.

Esta es una primer referencia de organización de la POO, que enriqueceremos más adelante con alternativas como herencia, polimorfismo y otras.