Universidad Tecnológica de Pereira

Mision TIC 2022

El futuro digital es de todos

MinTIC

Programación Orientada a Objetos

Aplicación Básica en Python

Hechos QUE CONECTAN

Introducción POO Python

Introducción POO Python

- Python nos permite utilizar distintas metodologías (paradigmas) de programación.

- Hemos implementado inicialmente programas utilizando la programación imperativa básica, luego vimos funciones y trabajamos con programación estructurada.

Introducción POO Python

- Prácticamente todos los lenguajes desarrollados en los últimos 25 años implementan la posibilidad de trabajar con POO (Programación Orientada a Objetos)

- El lenguaje Python tiene la característica de permitir programar con las metodologías de objetos, funcional y estructurada.

Relación con Programación Estructurada

- En la Programación Estructurada se declaran variables de diferentes tipos y se actualizan durante todo el algoritmo para alcanzar el objetivo o meta del mismo.

- La responsabilidad de un objeto (por ejemplo automóvil) consiste en realizar las acciones apropiadas y mantener actualizados sus datos internos.

Interacción entre Objetos

- Cuando otra parte del programa (otros objetos) necesitan que el automóvil realice alguna de estas tareas (por ejemplo, arrancar) le envía un mensaje.

- A estos objetos que envían mensajes no les interesa la manera en que el objeto automóvil lleva a cabo sus tareas ni las estructuras de datos que maneja, por ello, están ocultos.

Interacción entre Objetos

- Entonces, un objeto contiene **información pública**, lo que necesitan los otros objetos para interactuar con él.

- **Información privada**, interna, lo que necesita el objeto para operar y que es irrelevante para los otros objetos de la aplicación.

Codificación POO en Python

Ejemplo Codificación

- Implementar una clase que represente un **empleado**.
- Definir como atributos su nombre y su sueldo.
- En el método___init__ (constructor) cargar los atributos por teclado, y luego, en otro método, imprimir sus datos. Implementar un tercer método que imprima un mensaje si debe pagar impuestos (si el sueldo supera a 3000)

Ejemplo Codificación

```python
class Empleado:

    def __init__(self):
        self.nombre=input("Ingrese el nombre del empleado:")
        self.sueldo=float(input("Ingrese el sueldo:"))

    def imprimir(self):
        print("Nombre:",self.nombre)
        print("Sueldo:",self.sueldo)

    def paga_impuestos(self):
        if self.sueldo>3000:
            print("Debe pagar impuestos")
        else:
            print("No paga impuestos")


# bloque principal

empleado1=Empleado()
empleado1.imprimir()
empleado1.paga_impuestos()
```

Observaciones Ejemplo

- Definimos el método___init__donde cargamos por teclado el nombre del empleado y su sueldo.

- Este método se ejecuta inmediatamente luego que se crea un objeto de la clase Empleado.

- Como vemos no llamamos directamente al método___init__sino que se llama automáticamente.

Observaciones Ejemplo

- Los otros dos métodos tienen por objeto mostrar los datos del empleado y mostrar una leyenda si paga impuestos o no.

- Desde el bloque principal donde creamos un objeto de la clase Empleado debemos llamar explícitamente a estos dos métodos:

```
empleado1.imprimir()
empleado1.paga_impuestos()
```

Recordar: Requerimiento Notas

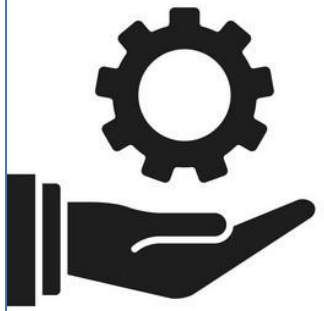
- Un profesor debe calcular el promedio de la nota de quices de sus estudiantes para subirla a la plataforma de notas finales. Sin embargo, el profesor acordó con sus estudiantes que los ayudará eliminando la peor de las 5 notas antes de calcular el promedio que finalmente reportará. Adicionalmente, el profesor se ha dado cuenta qué las notas registradas en su planilla se encuentran en una escala de números enteros de 0 a 100 pero la plataforma está diseñada para recibir el promedio únicamente en la escala estándar de la universidad: de 0 a 5, redondeado a dos decimales.

Especificación Requerimiento

- Escribir una función que reciba como parámetros: una cadena con el código alfanumérico del estudiante y cinco números enteros (nota1, nota2, nota3, nota4, nota5) que representan las notas de los quices del semestre y retorne una cadena de caracteres que le proporciona al profesor la información que desea obtener. La cadena debe tener la siguiente estructura: "El promedio ajustado del estudiante {código} es: {promedio}" dónde, el promedio reportado debe cumplir con las especificaciones mencionadas anteriormente (redondeado a dos decimales, en escala de 0 a 5 y calculado eliminando la peor de las cinco notas del estudiante).

Posible Solución:
Programación Estructurada

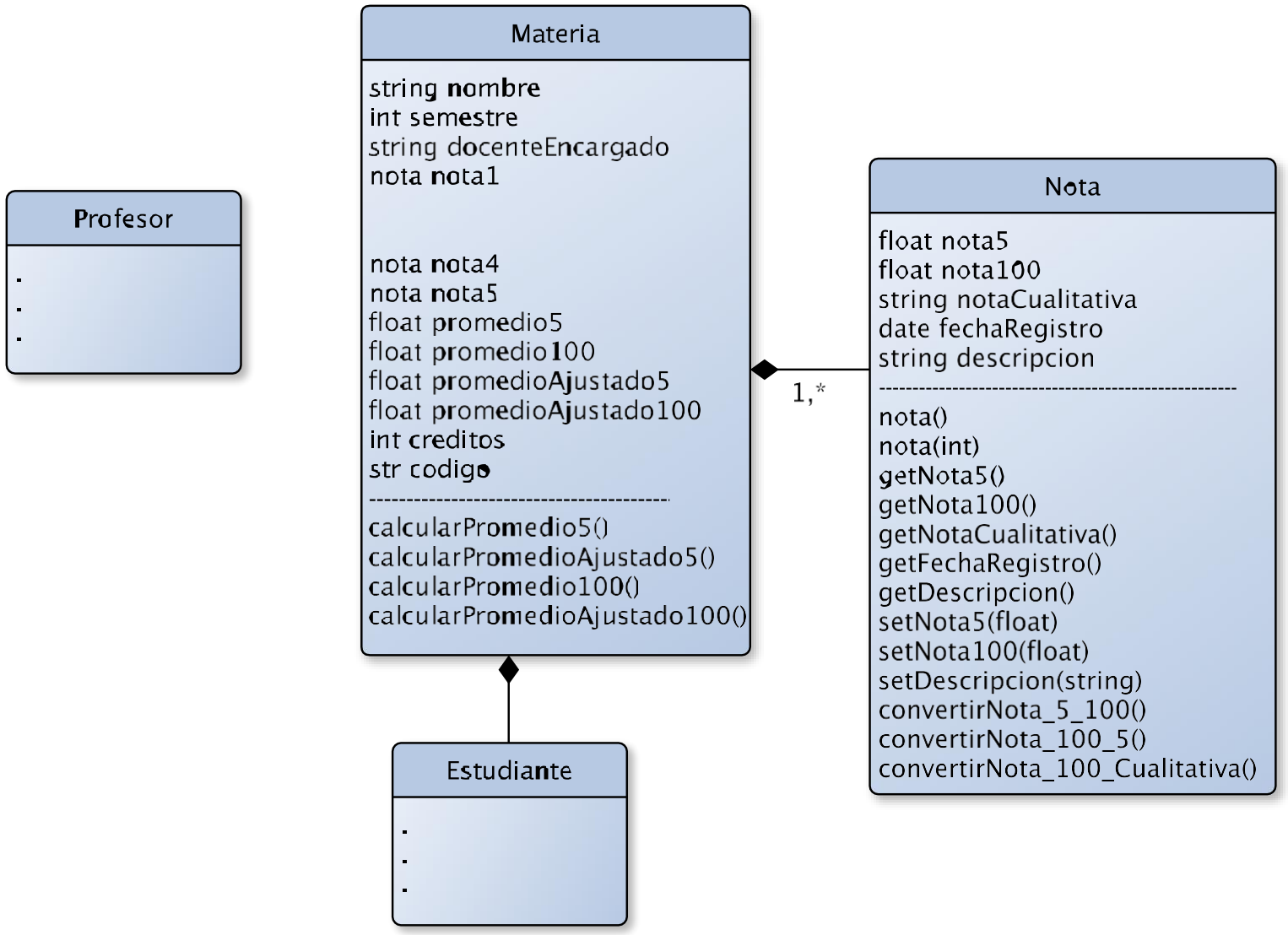
```python
#Solución paradigma imperativo (procedural/estructurado)
def nota_quices(codigo: str, nota1: int, nota2: int, nota3: int, nota4: int, nota5:int) -> str :
    medicion_maxima_promedio = 5
    max_nota_ = 100
    total_nota_sin_menor = 4
    calcular = 0
    if(nota1 < nota2 and nota1 < nota3 and nota1 < nota4 and nota1 < nota5):
        calcular = (nota2+nota3+nota4+nota5)/total_nota_sin_menor
        calcular = (calcular * 5)/100
    elif(nota2 <= nota1 and nota2 <= nota3 and nota2 <= nota4 and nota2 <= nota5):
        calcular = (nota1+nota3+nota4+nota5)/total_nota_sin_menor
        calcular = (calcular * 5)/100
    elif(nota3 <= nota1 and nota3 <= nota2 and nota3 <= nota4 and nota3 <= nota5):
        calcular = (nota2+nota1+nota4+nota5)/total_nota_sin_menor
        calcular = (calcular * 5)/100
    elif(nota4 <= nota1 and nota4 <= nota2 and nota4 <= nota3 and nota4 <= nota5):
        calcular = (nota2+nota3+nota1+nota5)/total_nota_sin_menor
        calcular = (calcular * 5)/100
    elif(nota5 <= nota1 and nota5 <= nota2 and nota5 <= nota3 and nota5 <= nota4):
        calcular = (nota2+nota3+nota4+nota1)/total_nota_sin_menor
        calcular = (calcular * 5)/100
    calcular = round(calcular, 2)
    return f"El promedio ajustado del estudiante {codigo} es: {calcular}"
```



Mundo del Problema Notas

- El caso de estudio tiene varias entidades relevantes interactuando que pueden integrarse o representarse mejor:

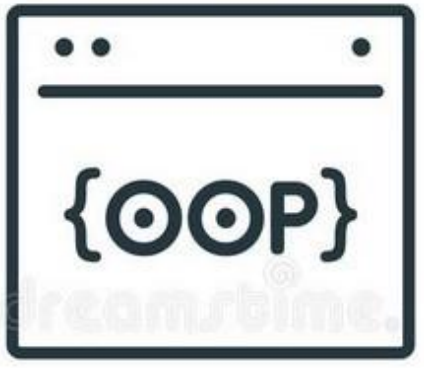
Mundo del Problema Notas



Requerimiento Notas Bajo POO en Python

- A continuación se relaciona el Mundo del Problema con la implementación en Python, la cual se puede descargar de la plataforma iMaster en la presente unidad, o en el siguiente enlace: https://github.com/luismescobarf/clasesCiclo1/tree/master/PromedioNotasPOO

- En dicha implementación se aplican los conceptos de objetos presentados.



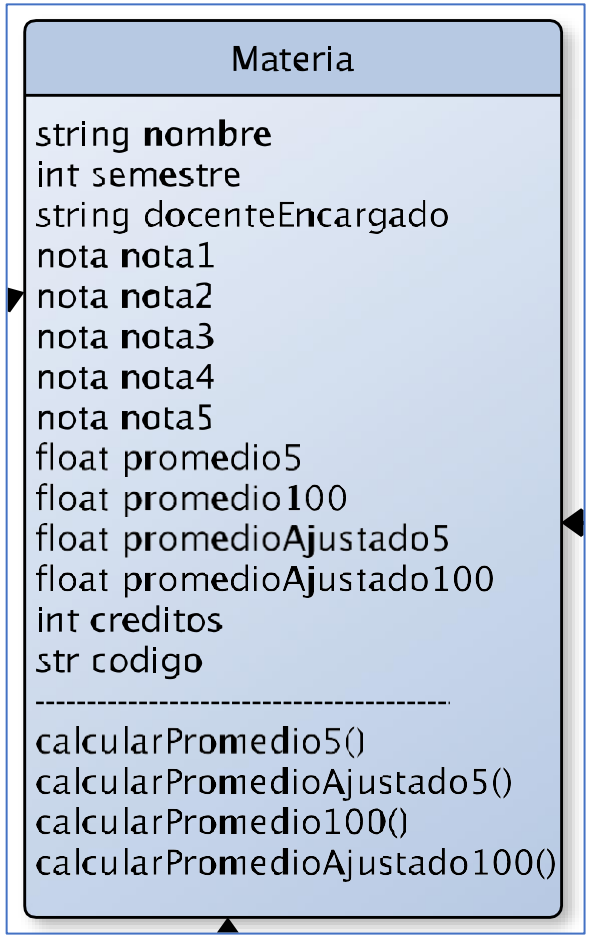
Clase Nota

Nota
float nota5 float nota100 string notaCualitativa date fechaRegistro string descripcion
nota() nota(int) getNota5() getNota100() getNotaCualitativa() getFechaRegistro() getDescripcion() setNota5(float) setNota100(float) setDescripcion(string) convertirNota_5_100() convertirNota_100_5() convertirNota_100_Cualitativa()

```python
##################
#Archivo con la descripción e implementación de la Clase Nota
##################

#Librerías para funcionalidades de la clase
from datetime import datetime

class Nota:

    #Atributos públicos (modificables desde afuera de la clase)
    ######################
    modeloPedagogico = 'Auto-estructurante'

    #Constructor(es)
    ######################
    #Definen los atributos del objeto
    #Se inicializan los atributos
    def __init__(self,nota5=None,nota100=None,descripcion=None):
        self.__nota100 = nota100
        self.__descripcion = descripcion
        self.__fechaRegistro = datetime.now()
        self.__notaCualitativa = None
        if not(nota100 == None):
            #Cualitativa
            if nota100 > 60:
                self.__notaCualitativa = 'Aprobado'
            else:
                self.__notaCualitativa = 'Reprobado'
            #Realizar conversión desde el constructor
            self.__nota5 = (nota100 * 5)/100
        else:
            self.__nota5 = nota5

    #Métodos (comportamiento del objeto)
    ###################################
    def convertirNota_5_100(self):
        pass

    def convertirNota_100_5(self):
        pass
```

```python
    #Getters
    ########
    def getNota5(self):
        return self.__nota5

    def getNota100(self):
        return self.__nota100

    def getNotaCualitativa(self):
        return self.__notaCualitativa

    def getFechaRegistro(self):
        return str(self.__fechaRegistro)

    def getDescripcion(self):
        return self.__descripcion

    def mostrarInfoNota(self):
        print('-------------------------')
        print('Descripción: ', self.__descripcion)
        print('Fecha Registro: ', str(self.__fechaRegistro))
        print('Nota Cualitativa: ', self.__notaCualitativa)
        print('Nota Escala 100: ', self.__nota100)
        print('Nota Escala 5: ', self.__nota5)
        print('-------------------------')

    #Setters
    ########
    def setNota5(self,nota5):
        pass

    def setNota100(self,nota100):
        pass

    def setDescripcion(self,descripcion):
        self.__descripcion = descripcion
```

Clase Materia




```python
##################
#Archivo con la descripción e implementación de la Clase Materia
##################

#Librerías para funcionalidades de la clase
from datetime import datetime

class Materia:

    #Constructor(es)
    ######################
    #Definen los atributos del objeto
    #Se inicializan los atributos
    def __init__(self,nota1=None,nota2=None,nota3=None,nota4=None,nota5=None,nombre=None,creditos=None):
        self.__nombre = nombre
        self.__creditos = creditos
        self.__nota1 = nota1
        self.__nota2 = nota2
        self.__nota3 = nota3
        self.__nota4 = nota4
        self.__nota5 = nota5
        self.__promedio5 = 0
        self.__promedio100 = 0
        self.__promedioAjustado5 = 0
        self.__promedioAjustado100 = 0
        self.__docenteEncargado = ''
        self.__notas = [ nota1, nota2, nota3, nota4 ]
```

Script Requerimiento Notas


```
1  ##################
2  #Aplicación de notas
3  #(Intermediación con el mundo del problema)
4  ##################
5
6  #Importar las clases que representan las entidades del mundo del problema
7  from Nota import Nota
8  from Materia import Materia
9
10 #Sección principal de la aplicación
11 ###################################
12
13 #Instanciar: crear los objetos tipo nota
14 objetoNota1 = Nota(nota100=40,descripcion='Primer Parcial')
15 objetoNota2 = Nota(nota100=50,descripcion='Segundo Parcial')
16 objetoNota3 = Nota(nota100=39,descripcion='Tercer Parcial')
17 objetoNota4 = Nota(nota100=76,descripcion='Cuarto Parcial')
18 objetoNota5 = Nota(nota100=96,descripcion='Quinto Parcial')
19
20 #Instanciar: crear los objetos tipo nota
21 objetoMateria = Materia(nota1=objetoNota1,
22                         nota2=objetoNota2,
23                         nota3=objetoNota3,
24                         nota4=objetoNota4,
25                         nota5=objetoNota5,
26                         nombre="Estructura de Lenguajes",
27                         creditos=3)
28
29 #Realizar varias versiones de promedio para la materia
30 objetoMateria.calcularPromedio5()
31 objetoMateria.calcularPromedio100()
32 objetoMateria.calcularPromedioAjustado5()
33 objetoMateria.calcularPromedioAjustado100()
```

Trabajo Autónomo

- Revisar detenidamente la implementación del caso de estudio, interactuar con este, observar que todas las clases tienen los métodos privados. Realizar actualización de atributos.
- Agregar nuevos métodos (funcionalidades) a las clases.
- Implementar las otras clases identificadas en el Mundo del Problema para apreciar la facilidad con la que se puede escalar un sistema bajo este paradigma.
- Agregar capas de vista (interfaz) y datos como se estudió en la Unidad 5.